I N T E R E X

San Francisco '89

INTEREX HP Users Conference

September 11-14 1989

MPE, MPE XL, O.A.

PROCEEDINGS

Volume 2

Sponsored by INTEREX

The International Association of Hewlett-Packard Computer Users

INTEREX

The International Association of
Hewlett-Packard Computer Users

Proceedings

of the
1989 INTEREX HP Users Conference
MPE, MPE XL, O.A.
at
San Francisco, California
September 11-14, 1989

Volume 2

Louis Gorewitz, Editor

Does KSAM have an Image Problem?

by
John D. Alleyn-Day
Alleyn Day International
1721 M. L. K. Way, Suite 3
Berkeley CA 94709-2101
415-486-8202

1. INTRODUCTION

Does KSAM really have an image problem? The answer to that is very definitely yes. Look at any application on the HP3000 and you will find lots of IMAGE files and very often not a sign of a KSAM file. Look at the ads for programmers; most of them ask for IMAGE, COBOL and VPLUS; how many ask for KSAM experience? I have heard of companies whose biggest marketing problem was the fact that their system used KSAM and not IMAGE, and therefore many prospective purchasers felt that the system was poorly made.

Is there any good reason for this? In the past, when the HP3000 was a new computer system, KSAM had a tendency to have more bugs and problems than IMAGE; however, although KSAM is now extremely reliable, these prejudices still remain and many people never consider KSAM as an option when it might, in fact, be the best method for the job. Outside the HP world, a database in the style of IMAGE is actually a little oddball. Most PC databases, such as dBase have B-tree key structures similar to KSAM. File systems such as ISAM or VSAM on mainframes are also of this type. KSAM has some very good company.

The purpose of this paper is to look at the advantages and disadvantages of KSAM with the hope that it will encourage a second look at the way in which KSAM can be incorporated into systems on the HP3000. This is not intended to be an exhaustive analysis of what can be done with KSAM, but rather an encouragement to others to come forward with different ways of putting KSAM to use.

2. COMPARISONS

We will start by comparing IMAGE and KSAM, looking at the differences and the ways in which these differences lead to advantages and disadvantages for the two systems. The

biggest difference, namely the key structure, is intrinsic to the database systems. The other differences arise from the particular way that HP chose to put these systems on the HP3000. Note that comparisons are based on using IMAGE without "autodefer".

2.1. Global vs. Local Keys

The most obvious difference between IMAGE and KSAM is a fundamental one in the key system, that I will describe as the difference between global and local keys. The key structure of IMAGE is very localized; the hashing algorithm allows immediate access to master records, but there is no relationship between a given master record and one with a closely allied key. Similarly, in a detail dataset, any given record is tied to its immediate neighbours with the same key by internal pointers. There is no relationship with any other records in the file. In the case of KSAM, the situation is entirely different. Any value of the key is closely tied to the surrounding values, and hence ultimately to every other record in the file.

These differences in the two database structures lead to the most obvious advantages and disadvantages of the two systems. The global nature of the keys is what gives KSAM its advantage in being able to read sequentially by key starting at any arbitrary place in the file, something that is impossible with IMAGE. The global nature of the key structure also leads to a problem: damage to the key file from a system failure will also be global in nature, sometimes requiring the keys to be entirely regenerated. For IMAGE, on the other hand, a system failure will cause only a localized broken chain, with the rest of the dataset being in perfect condition. But IMAGE cannot do keyed sequential access.

2.2. Partial and Overlapping Keys

One of the significant advantages of KSAM is the availability of partial keys. I have seen many IMAGE detail datasets with two and three keys that were only present to allow access by a partial key, say the first 8 bytes and the first 12 bytes of a sixteen-byte key. Furthermore, because IMAGE does not permit keys to overlap, the data has to be repeated three times in the record. KSAM, on the other hand, can achieve the same result with a single sixteen-byte key. Furthermore, if accessed by a partial key, the data will be sorted on the remainder of the key, something that would require more structure and considerable overhead to achieve in an IMAGE dataset.

2.3. Tightness of Control

IMAGE is very tightly controlled whereas control in KSAM is almost non-existent. IMAGE will not allow the programmer to make any mistakes that might interfere with the consistency of the IMAGE database. For example, you cannot update the database without a lock, if a lock is required. Each IMAGE update will force data to be written out to disc whether you like it or not. IMAGE cross-checks everything before executing the simplest command, and has a correspondingly high overhead. On the other hand, it is very easy to mess up a KSAM file if, as a programmer, you don't follow the proper rules; and the rules can get quite complicated at times. However, one can take advantage of this looser structure to gain efficiency impossible with IMAGE.

This extreme tightness of control in IMAGE initially gave rise to considerable difficulty, because the database structure is so rigid that it was impossible to alter it in any way. Fortunately, several third-party vendors have taken care of this problem, and it is now almost as easy to reorganize an IMAGE dataset or change its capacity as it is for a KSAM file.

2.4. Multiple Locking

One of the exceptional features of IMAGE is its locking. In essence, provided that you stay within a single IMAGE database, all locking problems are taken care of for you. In particular, multiple locking is made exceptionally easy; no worries about deadlocking the machine, and no special capabilities needed to do it. On the other hand, if you want to do something a little different, like holding one lock longer than the rest, IMAGE will leave you high and dry.

Another nice feature of IMAGE is the locking by key. However, this is directly connected to the localization of the key structure in IMAGE, rather than the particular embodiment on the HP3000.

KSAM doesn't have key locking, because of the global nature of its key structure. Instead, KSAM files must be locked as a whole. Furthermore, locking more than one KSAM file at a time requires the dreaded MR capability, and thus the possibility of locking up the whole system. I believe the fear of MR is an overreaction, as careful system design and programming can ensure that deadlock is an infrequent occurrence. It should never happen in a production program that has been thoroughly tested.

2.5. Logging

IMAGE also does automatic logging and database recovery for you if you want it. It may not be exactly what you desired, but the ability to just switch it on and off can save a lot of programming time. If you use KSAM files then you have to write your own logging and recovery functions.

3. PERFORMANCE

There are very significant differences in the performance of IMAGE and KSAM. I will discuss mainly the number of disc I/O's required to carry out any particular operation, as this is the most significant performance area. In determining approximate performance, I will assume the KSAM file has three key levels, which is a fairly typical situation.

3.1. Reading

Reading data is one of IMAGE's strong suits. The hashing algorithm generally allows the reading of a master record with a single disc I/O, a feat that KSAM can never equal. KSAM must read down through several levels of the key file before it can retrieve the data needed, so the average keyed read would require 4 disc I/O's, three in the key file and one in the data file.

3.2. Adding and Deleting

Adding and deleting single records tend to be rather similar in both systems. For example, an IMAGE detail set with three paths will usually require around 15 disc I/O's to add a record (see The Image/3000 Handbook, page 125). A KSAM file with three three-level keys would require about 13 disc I/O's, 3 I/O's to read each of 3 keys and 4 to write the data and key records.

However, in many situations, it is common to add several records to a dataset of this type, usually with the same key, such as line items in an invoice. When we add several similar records, KSAM has a distinct advantage, since most of the disc I/O's do not occur until the unlock at the end of the sequence, whereas IMAGE will force the records out to disc for each record. Writing five similarly-keyed records would take 75 I/O's in IMAGE, but probably not more than about 25 in KSAM, depending on the exact content of the keys.

There is a very handy trick that can be used to improve the

performance of deletion in a KSAM file, and this is to use a special flag to indicate that the record is to be ignored. I usually use the "delete word" at the beginning of the file for this purpose. This method requires a small additional amount of programming to check the flag, but the performance improvement is very significant. Although this trick can also be used in IMAGE (see The Image/3000 Handbook, page 135), you must, nevertheless, actually delete the IMAGE record at some later time by, say, a batch program running at night. With KSAM, these records can be taken care of in the reorganization process just as truly deleted records are, with no significant change in performance.

3.3. Updating

There is little difference between updating a non-keyed item in KSAM and IMAGE. It takes one disc I/O in each case, not including the additional I/O's required to read the record originally.

3.4. Changing Key Values

Changing key values, at least for secondary keys, is a place where KSAM really shines. To do this in IMAGE requires a delete followed by a put. Using our previous example of a detail with three keys, changing the value of one key would require a total of 26 disc I/Os. The same operation on the KSAM file requires only 5 disc I/Os. When one considers the possibility of updating several similarly-keyed records at a time during a single lock, the comparative advantage of KSAM in disc I/O can be substantial.

4. EXAMPLES

KSAM thus has certain advantages that should make it the method of choice in many situations. Good examples of this are systems that require a sorted, often alphabetic key and situations with transactions, particularly batched transactions, that go through various stages of processing. These will now be described.

4.1. Alphabetical Sort

As an example of a common use made of KSAM, we will consider a customer file that has as its primary key a customer number. Because this number may sometimes be unknown or unavailable, we want to access the record alphabetically by name. The common way to solve this problem is to use an

IMAGE master dataset with the customer number as its key and to build a KSAM index file containing the customer number as a primary key and the name as a secondary sort key. In this way a name can be looked up on the KSAM file and the data retrieved from the master dataset.

Sometimes the dataset needs more than one key, maybe a zip-code or a telephone number. This necessitates the data being put in an IMAGE detail dataset with several automatic masters and a KSAM index file constructed in a similar manner. The most efficient way of doing this is to use the record number as the primary key so that direct access is made to the record desired.

However, this mixture of KSAM and IMAGE is unnecessary in many cases. Why not put all the data into the KSAM file and dispense with the IMAGE datasets altogether? In the case of the multiple keys, all the secondary keys can be put in the KSAM file. This saves a lot of unnecessary updating when creating or changing the file and unnecessary reads when accessing the data. The uses made of the file should be studied to determine the best method. If it is read frequently by customer number and changed infrequently, then an IMAGE dataset may be valuable, particularly if it is kept as a master. But frequent changes, particularly to key values, militate against an IMAGE dataset.

4.2. Transaction Control

One system design problem that I have seen frequently is the access method to be used for a batch transaction file. Such a file usually has groups of transactions, each transaction represented by a single record. Sometimes, the actual data is kept elsewhere, and the transaction file is used solely as a way of keeping track of the processing. Usually the file contains some kind of status field that indicates the status of the transaction such as "new", "posted", "suspended", etc. depending on the application. The status is used as a key so that processing programs can read this key to determine the records to be processed. However, this key must then be updated to indicate the change in status. This process is usually accomplished with a detail dataset, often with several paths, but I have also seen it done with multiple flat files.

The big problem with using a detail dataset is that changing the status requires a delete and a put, which are both very time-consuming. With several paths through the dataset, the operation becomes particularly inefficient. But the key is needed, for without it there would be no reasonable way of finding the desired transactions.

Using MPE flat files tends to be an even greater problem

than using detail datasets, particularly if additional key data is needed, such as processing date. This often requires concocting complex file-naming conventions to determine which file is which. Processing requires a large number of file opens and closes, and, worse still, this scheme essentially uses the system directory as a key file. Since the directory is single-threaded, this method is likely to have a major impact on the overall system performance.

Using KSAM for the transaction file solves all these problems. If we make the status a secondary key, we can access the data just as easily as we could by using a path through an IMAGE dataset; but we can change it with a minimal amount of disc I/O. Furthermore, we may be able to group our updates, such as all the suspended transactions in a certain batch, thereby making the updating even more efficient.

An added advantage of using KSAM is that the key, as well as containing the status, can contain priority fields. Date is the obvious field to add to the status field, but we can also add a priority based on urgency of processing. Rather than a simple FIFO queue, we can then insert transactions into the queue at any point desired, without any performance deterioration.

5. PROBLEMS

I am assuming that most people have used KSAM at some point in their experience on the HP3000, so I don't plan to go into the simple operations that are readily available in the KSAM manual. There are, however, some points that are not completely obvious.

5.1. Setting up the File

When KSAM deletes a record it keeps track of this by placing high-values in the first two bytes of the data record. The record is not physically deleted until a reorganization is done. This can be very valuable if a record is deleted in error, as it can be retrieved by reading it in special ways using FCOPY or a similar utility. However, to make recovery complete, it is important that there is no needed data in the first two bytes of a KSAM record. For this reason, I always leave these two bytes blank, and I recommend this approach to anyone setting up a KSAM file.

I would also suggest that you always make the primary key unique. It is not absolutely required, but because KSAM

uses this key to decide which record to update, it can be very difficult to be sure that you are updating exactly the right record if the primary key is not unique.

There is an undocumented feature that I have never used, but that other people have used apparently with success. This is the "REUSE" parameter that can be placed on a KSAM file equation, and will cause the space from a deleted record to be reused for new records, in a manner similar to IMAGE. Using this feature will, of course, open you to all the possible dangers of an undocumented feature, particularly that it may change or go away with a new version of the operating system.

5.2. When to Lock

Locking is a major consideration on KSAM files when sharing access, which is almost always the case in an on-line situation. The basic rule is really simple: always lock the file!

This sounds, and is, rather more severe than the KSAM manual would have you believe, but there are good reasons for it. The locking can be relaxed in a few places but only with the proper precautions.

First, always lock when updating the file in any way: add, change or delete. If this is not done, there is a significant probability that the key file will be scrambled and the only rescue from this is a reorganization. If you don't lock and the file is well-used, you can count on being down for reorganization several times a day, so don't take any chances.

How about locking when reading? The KSAM manual will give you the bad news; when doing a keyed sequential read you should lock before the "start" and hold the lock until after the last read. The reason is simple. If you don't, someone else may come along and change the key block that you are using on the disc and you have no way of knowing it. Several records will apparently vanish into thin air. So, if it is important to know that you read all the records (and it usually is) lock the file.

I suppose you are hoping that I will at least let you off for a read-by-key. After all, the KSAM manual doesn't say anything about locking in this case, so by implication it should be unnecessary --- right? Wrong! Sure, most of the time you will get away with it, but every so often it will go wrong.

What is the explanation? Unlike IMAGE, KSAM makes use of the ordinary HP intrinsics without any special protection.

When you ask for a read-by-key you will actually be reading one key for each key record before reading the data record. If you don't have a lock, someone else can change the key file in between these reads and again your record will apparently disappear into thin air. However, there is a little relief; this time, you know about it; you will get a "record not found" error. You may have good reason to be sure that it does exist, in which case you know you have a problem. So, if you don't want to lock on a read-by-key, put in a little extra code. If you can't find the record on the first unlocked read-by-key, lock the file and try again. This way you avoid the deleterious effects associated with locking most of the time, but can retrieve the record infallibly when needed. I am indebted to Lisa Burns for pointing out this particular peculiarity to me.

Actually this situation of losing records during an unlocked read can also arise on an IMAGE detail dataset (see Thoughts and Discourses on HP3000 Software by Eugene Volokh pp. 180-185). The reason that it occurs less frequently is directly related to the locality of the keys in an IMAGE dataset. A put to an IMAGE dataset will affect only other records with the same key; a write to a KSAM file can affect the keys to any of the other records on the whole file.

5.3. Using KSAM with Powerhouse

Be very careful. Also, be aware that my comments on Powerhouse apply to Version 5.01F and may not all be true of Version 5.06. Powerhouse really hasn't come to terms with the realities of using KSAM properly. For example, although it claims to always lock when reading sequentially, in fact this is not true. The standard "find" procedure for a KSAM file doesn't lock the file when doing a keyed sequential read. However, this is not overly serious as one can always override it with a custom procedure that does lock the file.

Be careful if you open a file twice. I ran into a nasty problem doing a "while retrieving" on a KSAM file and updating in the loop via an alias of the same file. The lock is applied to the first open, but not to the second, and the key file can be scrambled in this way.

Generally speaking, simple operations seem to work well. Reading and updating single files have given me no problems, but complex processing will be liable to get you into trouble.

One unexpected, and very unfortunate side-effect of the Powerhouse system is that Quick will force a disc write of a KSAM record each time it is placed in the buffer, regardless of whether the file is kept locked or not. This invalidates some of my previous considerations about the efficiency of

KSAM.

Powerhouse will not give you the record number of an IMAGE detail record, nor will it allow you to do a direct read on a detail dataset, so the scheme described above for using a KSAM file as an index on a detail will not work. Probably the best workaround is to use the KSAM file alone instead of a detail and a KSAM file.

One very peculiar quirk of Powerhouse is that Quick doesn't seem to understand about changing secondary keys, and does a delete and an add in this situation. QTP, however, does it right and will change a secondary key with an update. This again, affects my previous statements on the efficiency of using KSAM.

5.4. Fast Reorganization

One of the major drawbacks to using KSAM is the danger of a system or power failure and the consequential damage to the file. A KSAM file must be recovered after a power failure. In fact, the operating system will not allow the file to be opened until a recovery procedure has been followed. In bad cases, a complete reorganization is necessary, and this can be very, very time-consuming. As an example, I am familiar with a KSAM file with 250,000 records, each about 30 bytes long, and with three keys, that takes over 5 hours to reorganize. I leave it to your imagination as to how long your own files might take.

What is the problem with reorganizing? Just as the original method for increasing the size of an IMAGE dataset was to do an unload and a reload, so the official method for reorganizing a KSAM dataset is to use FCOPY. As with IMAGE, there is third party software available that will transform the reorganization process by using far more efficient methods. I have tried out one of them, and experienced a five times improvement in reorganization time. This changes entirely the prospect of recovering easily from failures.

6. SUMMARY AND CONCLUSIONS

KSAM has an undeserved bad image as a database system, for historical reasons. It has capabilities that do not exist in IMAGE, such as partial and overlapping keys, and can perform some functions more efficiently than IMAGE, particularly updating key values. IMAGE is more efficient when reading data, has more extensive locking capabilities and readily available logging facilities.

KSAM is absolutely irreplaceable if a system needs a file with a sorted key, such as an alphabetical index. It can be combined with an IMAGE database or used on its own.

KSAM is invaluable in situations in which a key value must be changed during an update, such as the control of transactions. KSAM will also allow the construction of sophisticated queueing systems that go beyond simple FIFO queues.

In general KSAM should always be considered as a possibility when designing a system, as it may well lead to a better solution, either on its own or combined with other database systems such as IMAGE.

ABBREVIATED DEVELOPMENT METHODOLOGY

J. B. Watterson
Computer Data Systems, Inc.
20010 Century Blvd.
Germantown, MD 20874
301-353-2179

SYSTEM DEVELOPMENT METHODOLOGIES

Applications systems fall into three broad categories: Information Systems (IS), Automated Office Support Systems (AOSS), and Analytic Systems (AS). An IS (often referred to as a Management Information System or MIS) is usually developed to automate a single large function, such as payroll process. An AOSS application usually automates an office procedure. An Analytic System is developed in quick response to a one-time and/or highly specialized user requirement (such as a response to the president of the company or a government agency inquiry).

We utilize two system development methodologies specifically tailored to the above system applications. These are the Comprehensive Development Methodology (CDM) for use with IS applications, and the Abbreviated Development Methodology (ADM) for use with AOSS and AS. Each methodology defines a disciplined development process, from requirements through system delivery.

Prior to the expenditure of significant resources on a prospective development effort, a selection checklist procedure is followed to help determine which methodology should be utilized. This determination leads to the identification of the system approach to use as either CDM or ADM.

Since the CDM is the classical development approach, this presentation focuses on the selection checklist and the ADM methodology. The ADM approach includes the activities followed to develop an operational system during the development life cycle.

CHOOSING A METHODOLOGY

Applying the Checklist

To help with the decision between a CDM development versus an ADM development, the Development Methodology Selection Checklist is applied. The Checklist is a tool to help the developer decide which one of the two system development methodologies to use for a particular development project. Please note, the selection of a methodology does not predispose an application for implementation on an Hewlett-Packard (HP) type mini-machine, a mainframe, or a microcomputer.

Using the Checklist requires knowledge of certain basic system requirements information. This information can be obtained in one meeting with your system

users. You should not rely exclusively on the Checklist to make the decision on the development methodology. Remember, the Checklist is a quantitative tool, the results of which must be factored in with other important information and guidance. The bottom line is the processes used to develop the end products need to be highly professional, standardized, timely, and cost-effective.

How to Use the Checklist

Prior to beginning a full-scale Requirements Definition stage, the project leader and the technical manager should evaluate the application, using the Development Methodology Selection Checklist. See figure 1 for an example. The evaluation should be based on knowledge of a few basic system requirements and future plans for the system.

The Checklist consists of eight ratings criteria. Each criterion should be scored using the descriptions given. If in doubt about which of two (or more) scores to assign to a criterion, the highest should be used. As an example, using criterion number 3 (Total number of individual users using workstations), if the system will be used by only one user initially, but six might use the system in the near future, the higher score of '4' should be assigned.

Total the eight selected scores. Then, compare the total score to the Methodology Selection Table, which follows the Checklist. Select the appropriate development methodology, CDM or ADM, based on the following:

- o The results of the selection table.
- o Other important criteria relevant to the particular project.
- o Corporate ADP guidelines.

Applying the Methodology

Once the selection of a development methodology is made for a particular application, the developer should continue in accordance with that methodology. Avoid using "hybrid" or "special situation" methodologies. It is very important to be consistent in the application of each methodology because it contributes toward consistent high-quality user products.

THE ADM SYSTEM DEVELOPMENT LIFE CYCLE

ADM is generally for smaller development efforts. It implies a more simplified approach to development, and requires the use of documentation guidelines discussed later in this paper. In view of generally limited scope and size of the effort, you want to apply limited resources to the development cycle.

Criterion Number	Title	Description	Scores
1	Application criticality to the end user mission	Low criticality	2
		Medium criticality	5
		High criticality	8
2	Resources required for development	0 to 160 hours	2
		161 to 500 hours	6
		More than 500 hours	16
3	Total number of individual users using workstations	1	1
		2-4	2
		5-10	3
		More than 10	4
4	Requirements for concurrent access to shared data	None	1
		Query/reporting	4
		Data update	8
5	Frequency of application system utilization of all users	One time only	2
		Occasionally once monthly	4
		Once daily	6
		Multiple sessions daily	8
6	Communications network complexity	Stand-alone processing	1
		Between no more than two processors at one time	4
		Concurrent data communications among more than one processor	8
7	Required completion date	Within 1 month	1
		Within 6 months	3
		Willing to wait	8
8	Future software support requirements	None expected	1
		Minor modifications/ enhancements	3
		Major ongoing support and maintenance	8
	TOTAL SCORE		

METHODOLOGY SELECTION TABLE	Score	Appropriate Methodology
	11-42	Abbreviated Development Methodology (ADM)
	38-68	Comprehensive Development Methodology (CDM)

DEVELOPMENT METHODOLOGY SELECTION CHECKLIST
Figure 1

Each ADM application system development should consist of the following activities:

1. Systems Analysis

   - *Requirements Definition*. One or more meetings to discuss, determine, and analyze user requirements.

   - *System Design*. A proposed design is developed. In addition, an estimate of the time for completion and an activity schedule to complete is prepared. The requirements, design, and completion schedule are provided as the analysis design study.

2. Draft System

   - *Programming*. Coding of the system is completed as an initial draft system from the approved analysis design.

   - *Testing*. The initial draft system is tested to a level consistent with the complexity of the system design.

   - *Demonstration of Draft System*. The tested draft system is then provided and functionality demonstrated to the end user.

   - *System Documentation*. At the time of the demonstration to the user, a draft Application Reference (User) Manual is provided for their review and use.

3. Operational System

   - *System Modification*. Following a reasonable period of user testing of the draft system, modifications to the system are identified. Any changes necessary to complete functionality of the system, consistent with the original requirements, are programmed and tested as the final system.

   - *System Installation*. The final system is then installed in the end user environment and demonstrated as the operational system.

   - *User Training*. Appropriate user training is conducted consistent with the scope of the original user audience and the complexity of the system.

   - *System Documentation*. Following review of the draft system and its documentation, any appropriate revisions are prepared to complete the final Application Reference Manual.

4. System Certification

- o *Certification*. Acceptance of a system is undertaken only upon your specific ADP policies and procedures.

The CDM/ADM life cycle comparison chart is shown on figure 2.

ADM REQUIREMENTS ANALYSIS

The purpose of requirements analysis is to define user requirements so that these may be translated into hardware and software solutions. The process necessary to perform requirements analysis involves two phases: *research* and *analysis*.

Research

The research phase defines the purpose of the proposed system in terms of the user's existing situation, what the user wants to accomplish, the current limitations, and what is needed to satisfy the user's requirements. This process is performed in three tasks. They are:

1. Task Definition and Initiation. This step defines the purpose, objectives, and expected benefits of the new system. Specific roles and responsibilities of the users are identified, and the system environment and communications needs are determined. The required time frame for implementation of the system is determined in order to prepare a primary task activity schedule.

2. Information Collection. You need to first develop an interview questionnaire to direct your questions with the user organization. A sample questionnaire is shown in figure 3. You need to address their current method for handling the application, obtain existing documentation, and determine the need for enhancements needed for a new system to meet their requirements. Other information to obtain includes performance criteria, input, key fields, output, and interface requirements. In addition, you need to learn the user organization, clarify unfamiliar terms and procedures, and of course obtain copies of existing reports, original input and update source documents, and an explanation of each.

3. Information Validation. Compile the user interview finding, and then meet with the user again to review, correct, and validate the interview summary. Complete additional research, and obtain final concurrence on the interview summary from the user. This is very important as it becomes your baseline to proceed to the analysis phase.

Since much of the research phase involves interviewing the users. Scale the user interviews to the size of the effort; i. e., they should be long enough

Life Cycle Step	Comprehensive Development Methodology (CDM) Work Products	Abbreviated Development Methodology (ADM) Work Products
Feasibility Study/ Cost/Benefit	Feasibility Study (*)	
Analysis	Requirements Specification	ADM Systems Analysis (Requirements, Design, and Estimates for Completion)
Systems Analysis and Design	System/Subsystem Specification Program Specification Database Specification (*)	
Programming	System Test Plan Programming, Unit Testing, Integration Testing and System Testing	Draft System (Includes: Programming, Testing, Demonstration of Draft System, and ADM Application Reference Document)
Documentation	System Reference Manual (Draft) System Maintenance Directory (Draft)	
Training	Training Guide (*) User Training	
Implementation	Initial System Operation	Operational System (Includes Demonstration and Installation of Operational System, User Training, and System Documentation)
System Acceptance and Turnover	Acceptance Test Plan System Reference Manual (Final) System Maintenance Directory (Final)	Certified System (*)

(*) = Work Product sometimes not required or combined with another

CDM/ADM LIFE CYCLE COMPARISON
Figure 2

SYSTEMS ANALYSIS QUESTIONNAIRE

GENERAL INFORMATION

1. System name:_____

2. Office:_____

3. Who has overall responsibility?_____

4. Are there any critical ("drop dead") dates for implementation (Y/N)?_____ If yes, list date(s) and reason(s).
_____

5. What is the desired implementation date:_____

6. How critical is the system to the organization's operation?_____
_____
_____

7. Describe the current system:_____
_____
_____
_____
_____
_____

8. What hardware is currently used?_____
_____

9. What software is currently used?_____
_____

10. What are the criticisms of the current system?_____
_____
_____

11. Number of system users:_____ How many require training? _____

SAMPLE QUESTIONNAIRE
Figure 3

12. Location(s) of users:_____

13. Is this a stand-alone system (Y/N)? _____ If no, does it interface with another system (Y/N)? _____ If yes, list system. _____
_____
Is concurrent access required?_____
Can updating be done in batch mode?_____
Is there a requirement to limit access to the system,_____;
or, to applications or elements of the system?_____

14. Who is responsible for: Data Entry?_____ Organization: _____
Calculations? _____ Organization: _____
Report Preparation?_____ Organization: _____

15. Input frequency:_____

16. What specific algorithms are required for the application?_____
_____
_____

17. Importance of data currency?_____

18. What is the system output?_____

19. How timely must the output be?_____

20. Who uses the output?_____
_____
_____

21. How and when is the output used?_____

22. What will be the performance criteria for the new system?_____
_____
_____

23. What security or privacy issues need to be addressed?_____

24. Are backup and recovery procedures necessary?_____

25. Other comments:

SAMPLE QUESTIONNAIRE
Figure 3 (continued)

to insure complete understanding of the project, yet brief enough so as not to burden the user.

Analysis

The purpose of the analysis phase is to examine the current methods or processes, and the accompanying documentation, to determine what can be done to automate or enhance them cost-effectively by implementing a new system.

This is accomplished by examining all pertinent information gathered and materials collected during the research phase. Then proceed to analyze the existing and required methods, procedures, and information flows in light of acceptable and supported automated software tools. Define hardware and software requirements, data storage requirements, and system interfaces to determine how the new system will be integrated with existing systems and procedures.

ADM DESIGN SPECIFICATION GUIDELINES

The purpose of the design specification phase is to propose a system design that meets the requirements specified by the user. The design specification phase involves five tasks. They are preparation of design alternative, system level design, detailed design, documentation, and review.

Design Alternatives

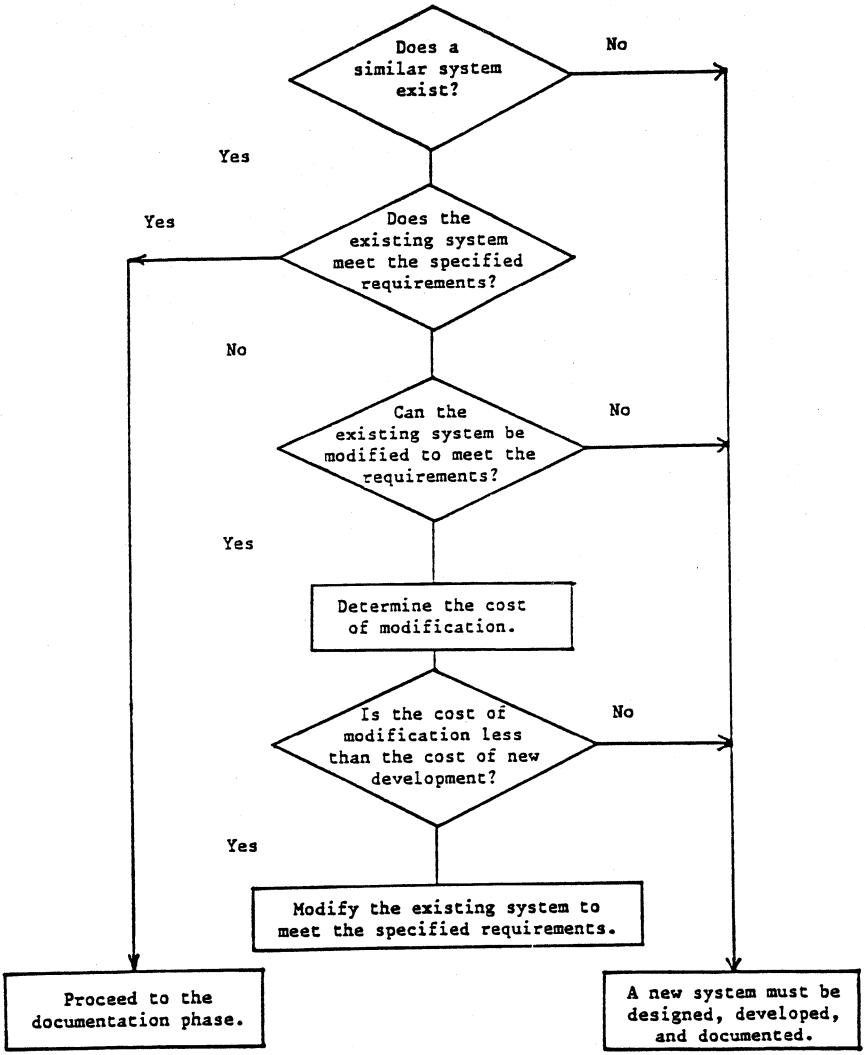
The purpose of this step is to select a design alternative from one of the following possibilities:

- Identifying an existing system that meets the specified requirements.
- Identify an existing system which may be modified to meet the user requirements. Possible sources may have been identified in the previous option.
- Design and develop a new system.

This procedure is illustrated by the flowchart depicted in figure 4.

Let's assume, for the purpose of describing the ADM methodology, a new system is to be developed. The design will proceed through the following three tasks:

- System Level Design.
- Detailed Design.
- ADM Systems Analysis Documentation.



ALTERNATIVE SELECTION FLOWCHART
Figure 4

System Level Design

This step specifies the major components or modules to be used in the proposed system. Reevaluate the objectives, update the requirements, and be certain the objectives are quantifiable. We will use the statement of objectives later to ensure the user requirements have been met. Then identify the hardware configuration and software needed to develop and implement the application.

The next step is to design the overall system structure. Look at the big picture first. We do this by using the following design techniques:

- Use a top-down strategy to identify and describe the major system modules and interfaces.
- Design the most important modules and interfaces first.
- Model the problem solution using appropriate analysis techniques.
- Define each module and the relationship between modules.
- Establish the information flow between components.

Detailed Design

In this step, we completely specify the design of each of the system modules. This includes the specification of file structures, the design of input and output formats, edits, and the interfaces to the user and to other systems. We use the following techniques in developing our detailed design under the ADM methodology:

- Design the details of each system module.
- Use a structured design to minimize portability and maintainability of the system.
- Establish a minispecification of each component. The actual coding of the system should follow the specification.
- Identify or design the source documents to the system.
- Design the formats of all system inputs and outputs.
- Design all menus, screens and reports.
- Design the user interfaces; e. g., detailed procedures for data entry, queries, and report generation.
- Design edit criteria required for data input by the user.
- Design database and file structures.

- o Design any interfaces to other systems.
- o Design any connections to data communications.
- o Design measures to implement security requirements, if necessary.

ADM Systems Analysis Documentation

Requirements analysis and design specifications for an ADM application are contained in one document called the "ADM Applications Systems Analysis" document. The use of structure diagrams, flowcharts, and similar graphic media as replacement for text is encouraged. We all must understand that requirements and design may change over the course of the development effort. This is especially true in a prototyping environment. Make sure you document these changes and obtain approval prior to implementation of any changes.

The sections required in the ADM Application Systems Analysis document are:

Section 1: Executive Summary - a short (one-page) summary to include descriptions of the project objective, user environment, major requirements, security issues, current procedures or system, and recommendations.

Section 2: Requirements Analysis

General Information - background information

Existing Methods and Procedures - a brief description of how the user currently performs the operation being analyzed. You should highlight shortcomings of the present system, and indicate why the development effort is being proposed.

Requirements - specific functional and data requirements of a recommended alternative or enhanced system. These will include specific interface, reporting, data validation, and data volume requirements. Inputs and outputs must be clearly and completely defined. State the data communications requirements, and user interface requirements such as whether the system will be menu or command driven, whether help messages or tutorials will be included, etc. Define processing options (add, change, delete, etc.).

Note: Section 2 must completely define all user requirements. It doesn't recommend a design to meet these requirements; that is the purpose of section 3.

Section 3: General Design Alternatives - compare and contrast solution options with each other. Identify and justify one of these conclusions, based on that review:

- o An existing system can be modified to meet current user requirement.
- o The cost of modifying an existing system is greater than the cost of developing a new application.
- o No similar system is available.

Describe the detailed design of the system and include:

- o Hardware and System Software Specifications - describe the environment needed to meet the user requirements.
- o Software Development Specifications - describe the system architecture and, for each major component, the processing to be performed and the input/output data flow.
- o Data Specifications - describe file (database) structures, input and output data descriptions, sources of data, and validations to be performed. Include an estimate of storage requirements.
- o Security - describe steps needed to implement defined security requirements; e. g., sign on, data access, etc.
- o Communication Specifications - clearly state the data communications design solution, if needed.

Section 4: Development Plan - an estimate of staff and other resources needed for project completion. Include time for documentation, training, and testing by the developer and user. If specific hardware or software is needed for application development, discuss these requirements in this plan. Specify delivery date of the application.

ADM APPLICATION REFERENCE DOCUMENTATION

The ADM operational system documentation includes both user operating instructions and programmer maintenance information. These are contained in a document called the "ADM Application Reference Manual".

The manual contains three major components:

- o System Abstract - designed to provide introductory and summary information about the system.
- o Operating Procedures - describes how the operator uses the system.
- o System Maintenance Information - for use in subsequent application maintenance support.

The information required within each of these components is as follows:

Section 1: Abstract - a brief (must be contained on one page) summary of the system.

Section 2: Operating Procedures - step-by-step procedures which allow the user to "walkthrough" the operation of the system. The level of documentation should be geared to the expected user of the system, as determined in the analysis phase. If differing levels of users are expected, documentation must be geared towards the least experienced. Specific information to be included as appropriate includes:

- o Hardware Operating Procedures.
- o Software Operating Procedures.
- o Security Procedures.
- o Backup Procedures.
- o Restore Procedures.
- o Report Samples.
- o Error Messages and Codes. (This may not be needed if system messages are clearly explained, and if online help messages are available).
- o Data Entry Codes. (This may not be included if codes are well known to all system users, or if codes are explained and defined in an online prompt or help message).

Section 3: System Maintenance Information - information provided for use by programming support staff. This section should include the following:

- o Installation Procedures.
- o Interfaces.

- o Program Flow Description - (preferably graphically).
- o Data File and Database Definitions.
- o Macro Definitions - of any spreadsheet or other macros.
- o Programs - listing by program name.

QUALITY REVIEW AND CONTROL

With both the Abbreviated and Comprehensive Development Methodologies, there are three levels of internal quality review and control: peer, task leader, and technical manager reviews. All three levels use the methodology and other standards to measure product quality. Peer review is performed by conducting structured walkthroughs for every major functional component of a development product. Task leaders provide a second level of quality review to ensure compliance with technical standards and with their day-to-day direction of the task. A technical manager reviews and approves all products prior to delivery to ensure quality and compliance with overall policy and standards. System certification represents a formal review and acceptance of the system by independent certifiers and the users.

CONCLUSIONS

The Abbreviated Development Methodology is a disciplined development process. ADM requires a less rigid approach, and fewer deliverables than the Comprehensive Development Methodology. It can be used on small information systems, office automation, and analytic systems efforts. The identification of a system as an IS, an AOSS or AS does not always indicate whether the system uses a mainframe, mini or micro computer; or imply that you should use the ADM or CDM methodology. The Development Methodology Selection Checklist will assist you in choosing the appropriate methodology. The use of prototyping methodology works well in the implementation phase of ADM.

With a generally limited scope and size of ADM applications, it is appropriate to apply limited development effort. Yet, ADM still provides the users with the benefits of structure and standardization.

Are You Missing the Boat? Prototyping Revisited

**D. Knight, M. Michalak, B. Bliesner, C. Abbott,
J. Dixon, K. Eriksen, M. Fish, E. Yoneoka
Boeing Computer Services
Box 24346 M/S 6H-66
Seattle, WA 98124**

Abstract

With expenditures on computing resources growing faster than other segments of companies' operating costs, increasing productivity in application development becomes all the more critical. Many have turned to Application Prototyping as a methodology for increasing programmer productivity and shortening application development time over traditional life cycle methods. There is a tendency to accept this methodology as static. However, in the past few years the computer industry has undergone tremendous change. New hardware and software products have appeared at a phenomenal rate. The concept of application prototyping needs to be updated to reflect these industry changes. "Prototyping Revisited" takes a practical look at the subject and describes the new tools, new technologies and industry trends on this continuously evolving methodology. This paper reviews past influences and puts prototyping in perspective for today's thinking. Now, more than ever, prototyping has a significant role in the data processing industry.

Introduction

'Prototyping', 'CASE', 'Expert Systems', '4GL', 'Neural Networks',.... The list of acronyms and 'buzzwords' associated with current computer applications technology is long. It can be confusing. At times it is beneficial to take some time to see where we've come from, to look around at where we are, and to see where we may go from here.

In 1986 Boeing Computer Services - Aerospace Support formed a team to study the concept of Application Prototyping on the HP3000. The team's goal was to research methods of increasing programmer

productivity and shortening application development time. To accomplish this goal the team began to investigate productivity tools, project control methodologies and the attitudes and philosophies of users, programmers, and their management. The team, representing more than 100 years of combined programming experience, came to grips with the litany of computerese which swirls around the world of Application Prototyping. They all knew a bit from a byte, Pascal from Fortran, but did they know the difference between CAD and CASE, a 4GL and an SQL? Books and articles were read; definitions pinned down. Eventually a sense and an understanding of this method of computing began to emerge. (Most of those who had worked in the Mini/Micro environment for any length of time discovered that they had been doing at least some aspects of application prototyping all along - they just didn't know it.)

This paper presents the findings of that team. It defines the Application Prototyping methodology, and then describes the types of tools currently available for Application Prototyping (here, among other things, is where the difference between a 4GL and an SQL will be revealed). Next it explores existing methodologies. Finally, areas of concern and future directions are discussed.

In this paper several software products are mentioned as examples of a particular type of tool for Application Prototyping. Mention of these products does not constitute endorsement by the authors or the Boeing Company. Lack of mention of a product does not indicate that it would not be suitable as a tool for Application Prototyping.

Overview/Definition

To get a sense of what Application Prototyping is all about, imagine a programmer working alone on a program which will require some kind of data entry and some kind of output based on that data. The programmer knows what must be accomplished. His[1] preparation prior to code and test probably includes flowcharts, data requirement lists, etc. After the first successful run of the program, he will view the screen or printout critically, looking at such things as accuracy, consistency, ease of use and presentation. Unless it is an extremely simple program, something was probably overlooked, or the format of the presentation isn't quite right. So he modifies the program until he is finally satisfied with it. Once everything is to his satisfaction, the application continues to run, supplying new or changed input and/or

fresh output whenever it is needed. And, if the application is used for any length of time, he will inevitably discover that at least one requirement, either for input or output, will change. After an analysis, the changes are incorporated into the existing program. Hopefully, our programmer had 'documented' the original program with internal comments and was able to find the flowcharts, etc. which aided in the program's creation. Testing and implementation take place and the program runs until a change in requirements comes along or until the need for it disappears.

Several steps in the development process for this program have occurred here:

- Requirements were established.
- Data requirements lists, flowcharts, etc. were developed.
- Program code was written and tested.
- Program flow and data entry routines were critiqued.
- Output accuracy and format were critically analyzed.
- The program was put into use.
- Modifications were made as requirements changed.

These steps are essentially the same for any software application, no matter what methods are used to generate a final product. The difference between Application Prototyping and more traditional methods of software development is the level of involvement of users in the development process. Traditionally, the end user doesn't get to use an application until it has been formally released to him, which may be months or years after the need for the application was established. The application meets the requirements, but in reality may not function as the user actually envisioned. Then too, lack of interaction with the end user can result in miscommunications. Errors present in initial phases have gone undetected. In traditional life cycle, the correction of these errors becomes more expensive as you move further into the life cycle of the application. In contrast to the traditional approach, in Application Prototyping only fundamental goals and objectives are determined. A working model of the application is quickly built with an advanced programming language (4GL). An iterative process then takes place between the analyst and the end user in which revisions and enhancements are implemented. This rapid interaction between analyst and end user is made possible by new technologies. A control prototyping methodology is necessary to complement these new technologies.

Application Prototyping is defined as follows for this paper:

> Application Prototyping is a methodology using tools and techniques to design, develop, and implement a computer application. As user requirements change or technologies improve these steps are repeated through the life of the application. This is accomplished through:

- Increased user involvement in the incremental development of applications.
- Improved availability of easily used tools.
- Enhanced analyst role while de-emphasizing the programmer role.
- Consolidated life cycle methodology.

The benefits of this approach are substantial. Since the end user is provided with a working model early on, he has the opportunity to clear up misconceptions and because the end user participates in the development process, the end product is more likely to meet with acceptance. The time from conception to initial implementation is shortened and this approach allows smooth transition to improved technologies. Also, selecting an appropriate 4GL for the application can yield cost savings, effectively trading expensive labor for intelligent software.

Prototyping Tools

Our programmer's shop, typical of many other data processing shops, has a large backlog of requests. In addition, his customer has become discouraged with the results of the last system that was implemented. It was months before the system was delivered, and then although it met requirements it really didn't fulfill the customer's needs. Now, another new system is to be developed. The programmer, reading about new tools on the market to increase productivity, is ready to incorporate some of them in his shop. But what do all the terms mean, and which tools will truly improve productivity? A review of some of these tools will help answer these questions.

4GL (4th Generation Language):

This is the name given to any new programming software that will provide a several fold increase productivity over 3rd generation

languages, such as FORTRAN, COBOL, PASCAL, etc. These products range from a product that does one piece of the application to a full application development tool set. A full development tool set usually consists of a screen builder, report writer, database and database interface, and application development language. The screen builder allows the user to quickly create input screens and menus for the application. The database and database interface allows for the storage of data and easy access to that data. The data structures created in the database are easily modifiable for changes and enhancements. Because relational databases are easy to use and easily modifiable, data storage is frequently relational in 4GLs . The report generator provides a tool for quickly producing reports. The language used to build the application can be non-procedural, procedural, or object-oriented. It provides the user with the capability of building either simple or complex applications.

Examples of full tool sets: Oracle, RELATE/3000, Today, Speedware

Code Generators:

These are software which are capable of generating source code. The source code is then available for the application programmer for maintenance, if desired, although more often the generator is used again. Some generators have limitations in the complexity of programming, so it becomes necessary to modify the generated code in complicated applications. However, once modified, the code can no longer be maintained by the code generator. Some code generators have hooks in them to allow execution of programs or routines written in other programming languages.

Examples of code generators: Protos, Artessa

Report Writers:

These are software which are capable of generating reports on specified output devices in formats determined by the package or by programmer specification. Report writers vary in complexity. Different levels of report writers are available ranging from simple ad hoc reports designed by end users, to more difficult reporting formats requiring programmer support. Many report writers designed for end users are menu-driven, where the user can pick from a list of elements, and get the report either on the terminal, or on a printer in batch mode

or while they wait. Many 4GLs have reporting capabilities incorporated in the package. Some of these are Speedware, Powerhouse, and Relate/3000.

Examples of End User Report Writers: Data Express, Inform
Example of Programmer Report Writers: Business Report Writer

Debugging:

This is a feature of a language that helps debug a program. Some 4GLs have built in debugging capabilities which highlight errors as the code is being written. Online debugging features allow developers to modify screen or code without terminating the program. Some, such as Transact, have debugging routines that can be called when problems are encountered during testing.

Examples of 4GLs with built in debugging: Flexible/3000, Relate/3000, Transact

Application Generator:

These are software products which will create the basic program structure automatically, because the functions will normally be similar from one system to another. Filling in values on the menu will define basic transactions, input/output fields and files, and interrelationships between them.

Examples: Insight, Today, Speedware

Screen Builders:

These products contain easy, automated means of creating basic screens from dictionaries/encyclopedias. These are the ideal for rapid prototyping demonstrations for the end user.

Examples of products that have screen builders: Flexible/3000, Today, Powerhouse, Speedware

Database Utilities:

Good database manager utilities allow easy and rapid changes to database design as requirements are obtained from the user during the

prototyping process. Selection of the appropriate database manager is based on the type of access required. Some 4GL products that have full function database managers are dBASE, HPSQL/V and Speedware. Other tools are single function such as Adager or DBGENERAL which modify database structure, or IMACS*LYNX which manipulates data dictionaries. Some tools that enhance performance for database retrieval include OMNIDEX, SUPRTOOL, and SORTMATE

Automated Documentation Tools:

One of the main characteristics of a 4GL is a centralized dictionary for data element and file information. This becomes the core of the system documentation. Some 4GLs have generators that print data in a document format from the dictionary, and from internally maintained files and tables.

Examples of 4GLs with self-documentation feature: Speedware's Documenter, Today

Design Tools:

These tools automate the tasks of system planning, requirements analysis and design. Computer Aided Software Engineering (CASE) products like Excelerator from Index Technology provide front-end design tools for 4GLs. Dictionaries/encyclopedias control the integrity of data element definitions across software packages. Encyclopedias extend the control of the dictionary by containing diagrams of data flow or entity relationships of the system. Software can then be generated straight from the encyclopedia.

Example of utilization of an encyclopedia: SYBASE

Expert Systems:

This is one of the fields in artificial intelligence. This is software designed for a specific task where the knowledge of the expert has been built into the software. There are many software products which will help the knowledge engineer develop an expert system. These range from languages like Prolog and Lisp to expert system shells. With Prolog and Lisp, the developer must start from scratch and build everything that is needed. The shell provides the developer with an environment for ease of developing an expert system.

Examples of Expert System Shells: AION, M.1., Personal Consultant + (PC+), Knowledge Craft

Platforms (Distributed Processing):

To improve the cost effectiveness of the programmer/analyst the use of tools that allow easy porting of software from one machine to another should be used. The computing power on the desk (PCs) needs to be more efficiently used. There are tools that generate software on a PC that can be uploaded to either a mini or large scale system and then run by the end users.

Examples of products executing on multiple platforms: Today, Relate/3000, System Z, Speedware, Oracle, Powerhouse

Terminal emulation software is available for a PC that will emulate a terminal that is connected to a host computer.

Examples: Reflection 1, 2, 3, 4, 5

Prototyping Methodology

Our programmer, excited at the prospect of using all of these new tools, plunges into the task of developing the application. He creates a prototype and demonstrates it to the users. The users are not happy; they suggest several dozen minor and major changes that they need implemented. No problem; the programmer sits down and, using no formal methodology, immediately makes the changes that they have requested. These changes are not exactly what the users wanted either; so they detail further changes. Again, no problem; he puts his wonderful new tools to work and once again makes all of the changes that the users have asked for. The users are a little happier with the application, but there are still several changes that need to be done. This goes on for several iterations. After about the fifth or sixth iteration, the programmer is becoming a little confused. He is no longer sure of the functionality of the system. So many changes have been incorporated that he no longer understands exactly what some parts of this system do. In short, he has created a monster application and has very little documentation to take him through exactly what is going on, and the users are still surrounding him clamoring for yet still more changes. He realizes that in order to regain control over the application,

some kind of prototyping methodology has to be implemented. But what are his choices? After researching many formal approaches that have worked for other people, he decided to concentrate on two methods, Application Development Guidelines (ADG) and Evolutionary Development Methodology (EDM).

ADG was developed in response to customers' needs by the COGNOS company. The steps in ADG's methodology are as follows:

Problem Analysis
Requirements Definition
Data Analysis
Iterative Development
System Completion
Implementation

The purpose of the problem analysis section is to determine the scope of the project, give an overview, state the objectives, define the problem, and to make preliminary plans. The purpose of the requirements definition section is to define requirements, analyze the business functions, define a conceptual solution, design a rough draft of menus and screens, make general plans, and to communicate the approach taken.

The third section is data analysis. The overall objective of this section is to produce a logical definition of the data. There are two different approaches that can be taken to accomplish this. They are entity modelling and data normalization. For prototyping purposes, entity modelling is probably the best way to go.

The fourth section, iterative development, is when the prototyping comes in. Armed with the requirements definition and the data definition, development begins. The data base is created, the system environments are set up, the prototype is written, and user reviews are scheduled. Here the functions are demonstrated. Requested changes are documented, implemented, and tested. The new version is then demonstrated again and the cycle is repeated. This continues on until the users are satisfied with the product or until a predetermined number of iterations have been done. This ushers in the system completion section, where the remaining functions are built, system interfaces and documentation are done, consistency is ensured, the

complete system is tested and is prepared for production. The implementation section is where the system goes into production.

Note: The preceding discussion is based in large part on a presentation given by Marc Praly of COGNOS on July 20, 1987.[2]

The other development methodology EDM is supported by EDM Management Systems. EDM consists of five phases. They are as follows:

Management Study
Technical Review
Feasibility Project
Implementation Project
System Review

The first two phases are done only once, at the beginning of the project. The last three phases are done per iteration.

The purpose of the first phase, the management study phase, is to determine whether or not it is in the customer's best interest to computerize the application. Ideally, this would be done by the customer, since they would have the best handle on what direction they want to take. This whole phase is essentially dedicated to the question, "Does an EDP project look like the way to go?". Outputs from this phase include a definition of goals and objectives, an analysis of the current situation, and a prioritized list of business requirements to be automated.

The purpose of the second phase, the technical review phase, is to determine what the system will cost, and a rough idea of what it will look like. This is done by a systems analyst in response to the output from the management study phase. Outputs from this phase would include a verification of the strategies and assumptions as given in the management study, a conceptual definition of the system, a rough draft of a release strategy, a description of the implementation environment, a rough data analysis and model, and a cost/benefit analysis. At this point the customer's management will either approve or cancel further development.

As said before, these next three phases are done for each iteration. The purpose of the third phase, the feasibility project phase, is to give the customer's management enough information to determine whether or

not to continue on and to implement the proposed changes. Note that if management refuses to authorize these changes, it does not cancel the whole system; only that the proposed changes that were going to be implemented in the current iteration are not to be done. The outputs from the feasibility projects are a definition of exactly what will be done during this iteration and how it will be done, a complete analysis of costs and benefits that pertains to this iteration, a detailed data model of affected areas, a shell prototype system demonstrating the viability of the proposed changes, and the customer's evaluation of the prototype.

The fourth phase is the implementation project. The purpose of this phase is to deliver a working system as defined by the feasibility project and to implement it. Along with this is customer acceptance and training. It is within this phase that the concept of 'Stages' is utilized. Essentially, Stages are used to chop the overall objective of the iteration into bite-sized chunks. The Stages are developed independently, but all must work before the implementation project is completed. The outputs from the implementation project are a final data model for the affected areas, accepted software and documentation, converted data, trained users, and a final review. The final review verifies that the current iteration fits in with both previous and planned iterations.

The last phase is the system review phase. There are two objectives to this phase: to review the success of the iteration and to apply what has been learned to the future. This serves to keep the overall goals of the customer in sight and on track. The outputs from this phase are a review of work done and a release strategy defining future activities.

Note: The bulk of the EDM discussion is based on the document "Evolutionary Development Methodology: An Overview", by Dave Wattling, DWM Computer Systems Ltd. and Douglas P. Kelly, Newton Data Inc.[3]

Potential Concerns

Our programmer was given a list of concerns from his managers to address before being given a green light for assembling his new tool box.

- Has the software vendor thoroughly tested the product before marketing?

- Does the product perform as advertised or are some of the features vaporware?
- Software updates occur at a phenomenal rate. Should purchase be postponed until all the newest features have been incorporated into the product?
- Are his current applications suitable candidates for a 4GL?
- Without experience in the prototyping tools, how does he evaluate which ones are right for his shop?
- Will the new software be compatible with existing software and data base structures? What conversions will be necessary?
- Can our computer system handle the increased overhead usually generated by a 4GL?
- Will our end user be willing to commit adequate time and effort to ensure success of the project?
- Will it be portable to all the platforms currently utilized and how will it mesh with future hardware configuration plans?
- Will performance become an issue if an increase in business places new loads on the system?
- No one in his shop has experience with a 4GL, let alone a whole tool box of new products. New products and a new methodology will move these programmers out of their comfort zone. How long before he can realistically expect training to be complete and the whole shop using the new tools with some proficiency?

In order to answer these and other concerns, our programmer needs to become knowledgeable in a whole new arena of terms and products. This will require manager support, budget, time, interest and hard work. Trade journals can provide background information and a familiarity with the subject. Vendor shows and seminars provide additional information and a review of available products. Considering his shop's requirements, two or three candidates can be selected for indepth investigation. Preliminary questions as to length of time in business, product support, training, etc. need to be addressed. If possible a formal training class for the product could be attended. A talk with someone who has already invested in the product, might yield some candid answers for these concerns. Someone who is already using the product might provide some insight into issues of performance. The evaluation steps need to be a carefully thought out process, involving all the personnel concerned with the project. Only after a complete evaluation can the best possible decision be made for the business.

Data Processing Industry Future

There are a number of general areas within our industry which have shown major advances in the last several years, and we predict will play a significant role in the future. They are Artificial Intelligence, Computer-aided Software Engineering (CASE) technologies, the "Workstation", and Neural Networks. A number of other areas which we will call trends could develop into significant contributors, and will be covered later in this report.

Artificial Intelligence

Artificial Intelligence (AI) is an application technique that more closely resembles the human mind and the way it works. AI uses a number of inventive methods of organizing and accessing data. Several areas of AI have been identified:

- expert systems
- natural language understanding
- perception
- robotics
- machine learning
- planning
- theorem proving
- symbolic mathematics
- game playing

A promising area for the user community, and the most developed at this time is the expert system. The purpose in building an expert system is to replicate expertise and/or to combine expertise for a specific application area. This would result in an increase in productivity by:

- avoiding delays
- distributing expertise to remote sites
- making expertise available to less experienced personnel
- preserving corporate knowledge
- increasing consistency of decisions
- handling routine reasoning and bookkeeping
- leaving an "audit trail".

The medical system MYCIN is a good example of an expert system. It was developed at Stanford in the 1959 - early 70's time frame. Its task is to diagnosis and suggest therapy in certain cases of infectious blood diseases. It is a rule-based system paradigm using backward chaining.

Another expert system example is "Prospector" which is a geological prospecting tool. The knowledge is in semantic nets. The savings from using this program alone would have paid for all early AI research.

Several languages (LISP, PROLOG, etc.), have been developed especially for AI to take advantage of the AI concepts, however any language can be used to apply these techniques.

Until recently the AI application community has been restricted by machine size and speed. With the advent of very powerful desk top machines, AI is starting to find its way into almost every PC product. The techniques will become more and more prevalent but will most likely take the form of the application product, and will not be identified as AI technology. Expect to be a user of AI technology and not be aware of it.

CASE Technology

Computer-aided software engineering (known as CASE) has become data processing's newest buzzword. Computer-aided Software Engineering is the automation of software development which focuses on the whole software productivity problem instead of just implementation solutions. CASE technology has been defined as a combination of software tools and methodologies. Software productivity problems are attacked at both ends of the life cycle by automating many analysis and design tasks, and combining them with program implementation and maintenance. CASE products include some grouping of the following tools: diagramming tools for system design, screen painters and report writers for prototyping, dictionaries, data base management systems, and accompanying reporting facilities for storing, reporting, and querying system information, automated consistency checking against system specifications, and code generators to automatically generate executable code from system specifications. In some instances CASE analysis and design tools serve as front end processors to 4GLs.

For now, actual perception of CASE technology seems to be inconsistent and somewhat confusing. And perhaps because the products are so new

to the marketplace, it seems to be an arena filled with vendor hype and vaporware. No full CASE product exists today. Present day CASE products range from the simple generation of bubble charts to sophisticated products which automate requirements analysis through creation of a physical model of a large system, including screens, reports, etc. However, major vendors have brought CASE products to market and future CASE products will automate as much of the analyst's and programmer's job as possible. CASE products may someday add some artificial intelligence, such as an embedded expert system which helps an analyst find his way. CASE seems to be an area which shows real promise as an aid to application development.

Some examples of CASE technology products are:

- The Yourdon Analyst/Design Toolkit from Yourdon, Inc. - provides an integrated data dictionary for error and consistency checking throughout the analysis and design phases.

- ProKit*Workbench from McDonnell Douglas - fully automates the application of structure techniques of planning, analysis and design phases.

- The Design Machine from Ken Orr and Associate, Inc - generates the deliverables of system requirements and analysis.

- DESIGNAID from Nastec Corporation - automates structured analysis/design techniques.

Most of the CASE tools on the market place today support various phases of the Life Cycle Methodology. CASE tools to support more rapid development methodologies are expected to emerge, but we are not sure of the form or timing of these products.

Workstation.

We define a workstation as a microcomputer on an employee's desk which has at least the capability to do word processing and spreadsheet processing and is connected into a "Workgroup" which allows access to other computers in the network (could be other PCs, or minis, or mainframes).

During the last few years more and more employees are getting microcomputers on their desks. In many cases these are replacing dumb terminals and their prime usage is to access another computer (mini or large scale). This computer power on the desk opens up new methods of doing normal office operations. Microcomputer applications are increasing in usage and importance within the company. The most common usage today is in word processors and spreadsheet processing.

As more powerful microcomputers become available, and as we become more proficient in using this local computer power, it will play an increasing role in the DP industry. The AI techniques discussed above and the CASE tools will have a prime focus on the micros and not on the more traditional minis or large mainframes. These machines will be included in the workgroup which may include a complete range of processors networked together. The possibilities of these networks are just now beginning to be realized. Electronic mail is finding its way into the work stations as well as some object-oriented programming techniques.

Neural Networks.

This is the general area of applying biological memory techniques to networks of silicon memories. This provides a better replica of the human brain than we have seen before. Recent breakthroughs in the technology have made this technique very promising. It is expected to provide considerable speed as well as vast memory storage with rapid retrieval. The technology is still in the research labs, but looks to hold such promise that it is mentioned in this report.

Trends

What does the future hold as seen by the prototyping team? We see this as a very exciting and rapidly advancing time for the data processing industry. The items mentioned are the best guess at this time of what's coming in the future. We look at this section as being very speculative. We will address these trends in several areas, future hardware, future software, where is DP going, and the end user.

Future Hardware

There is a trend to place large amounts of computing power on the desktop. This will have a major impact on large scale and mini

computer usage as the applications migrate to the desk top computers. Jean-Louis Gassee, Apple Computer's Senior Vice President of Research and Development, has a dream of placing a 200 mip PC on every desk .[4] He is not sure that he will see it in his lifetime, but he is sure it will happen. Does this mean that large scale and mini computers are going the way of the dinosaurs? It may mean that machine size and speed will no longer be a factor, but how the machine(s) are used will be.

Networking is beginning to mature. The hardware is available to virtually connect any computer to any other computer. We expect this function of our business to continue maturing and will become a major cost factor in the future. This connectivity will make electronic mail a true reality. The desk top machine will have the phone built in and have FAX capability.

Disk and memory continue to become less expensive. As this trend continues all the known restrictions on memory and storage space will disappear. This, along with networking and larger desk processors will make distributed processing a household word, not the dream it is today. Expect to see desk top machines with 16MB memory and many gigabytes of storage commonplace within the next few years.

Hardware is being placed in office environments instead of a computer machine room with special environment requirements. As this trend continues expect to see the operations tasks normally delegated to data center personnel turned over to the user community as is the hardware itself.

Hardware technology has been progressing much faster than the software industry can keep up with it, and the user and DP communities cannot keep up with the software changes which are much slower than the hardware changes. As users and DP begin to use these new hardware technologies expect major changes in the way of doing business. Micros and Minis are blending into one machine which looks more like a micro than it does a mini of the 1970s.

Future Software

Expect CASE tools to eliminate the job of the system analysts as we know them today. Coding will be a thing of the past. The CASE tools will allow an analyst working with the user to build and maintain a system with much less effort than is currently being spent. Code

generators will be built into products and not even discussed. Since labor is more expensive than hardware, functionality will be the focus, not efficiency of hardware resources. The CASE tools may allow the software designers to keep up with the hardware designers.

Look for an increase of user friendly software. Many of the software tasks now done by the system analyst will be taken over by the end users. End user reporting is the most common item in this area today although expect it to extend into system design and maintenance.

Expect to see the life of a software system to be shortened considerably. This is caused by a number of factors:

- New tools make the initial investment much smaller.
- Rewriting the software may be more cost effective than maintaining it.
- Rewriting the system to make it run on new inexpensive hardware may be cost effective.

AI techniques will be used in all software. The CASE tools will generate software which begins to think and adapt itself.

Object-oriented Programming is just emerging. Expect to see more packages such as Hypercard on the Mac which use Hypertext technologies, and allow end users to graphically build their own applications.

DP Trends In Business Applications

Systems will be able to be designed very rapidly. Does this mean that full time DP Analysts are no longer needed? Some people think so, so expect to see a migration of DP expertise drifting to the user communities in part-time DP positions. Others feel that as it becomes easier and less expensive to generate computer systems, the user community will demand many more systems than they are now doing, thereby increasing the need for DP analysts. In either case, we all agree that the way DP is doing business today is going to change.

The DP role will become more specialized. The DP analyst will determine network handling and provide consultation on the project

design and development. The end users will do nearly all of their own reporting requirements, and most of their machine operations and simple changes to the system.

What does this mean to the analyst of today and their DP organizations? Should they be looking for a job in the user community? It appears that programmers may need to either move to the user community or to develop their design skills. Analysts who will move into either the network areas or the consultation areas must become more skilled on the tools available and how to apply these tools to make them more productive. Communication skills will become more important both written and verbal. During a transition period old systems will need to be maintained and old data centers need to be staffed until both of those areas can be phased out to the new methodologies. Plans need to be made on how to make this transition most efficiently.

Data centers as we know them today may only be needed in extremely large applications. Most of the mini data centers will be replaced by user installed and run machines in their own areas.

What should the DP analyst of today be doing? We suggest the following:

- Keep abreast of what is happening and react in concurrence with your personal goals in the Company/Industry.

- Get as much training as possible in new technologies.

- Work on your communication skills.

- Learn as much as possible about your customer's way of doing business. This knowledge can give you an edge.

End User Trends In Business Applications

As the generation of applications and reports become easier, it is possible for the non-DP professionals to develop their own applications and reports. Many users in our shops are currently using ad-hoc report writers to do their own reports. In the Apple community, many users are developing their own applications and reports using Hypercard.

Training vs Education

We do a very good job of providing training on specific tools but do a very poor job of educating our DP community or our end users of the trends identified here. More emphasis must be placed on education to allow all computer users to more fully use the computer potential in their normal daily jobs. The tools must be made easier and more friendly so they do not get in the way of getting the job done. (i.e. We have had enough of stick shifts, we now need automatics.)

In Summary

Competition to increase productivity will force change. The utilization of new technologies and methodologies to achieve improvement in software application development becomes all important. Preparation and education are the keys. The better we understand what the new tools and hardware can do for us, the better we can make them work for us. Management needs to be committed to an early selection and acquisition of new products and to the training of their programmer analysts not only in the use of these tools, but also in the effective use of a controlled prototyping methodology. Establishment of a closer working relationship with the end user becomes a necessity. Application Prototyping dictates the early and continual involvement of the end user in the development process. Development time will be shortened and the resulting software will have a better chance of user acceptance.

---

[1] Ooops, now we've done it. Well, for ease of writing as well as reading, all gender references in this paper will be masculine. The reader is politely asked to mentally switch gender gears as he/she finds necessary.

[2] Marc Praly, Sales Representative, COGNOS Corporation
Training presentation at Boeing July 20, 1987

[3] Dave Wattling, President of DMW Computer Systems Ltd.
EVOLUTIONARY DEVELOPMENT METHODOLOGY: An Overview.
May 1986

[4] Statement from the Keynote Presentation at MACWORLD Expo in Boston on August 13, 1988.

PRODUCTS LIST

Adager	registered trademark of Adager De Guatemala, S.A.
AION	registered trademark of AION Corporation
Artessa	product of RAET Software Products
Business Report Writer	registered trademark of Hewlett-Packard Co.
Data Express	trademark of Imacs Systems Corporation
dBASE	registered trademark of Ashton-Tate Corporation
DBGENERAL	product of Bradmark Computer Systems Inc.
DESIGNAID	registered trademark of Nastec Corporation
DESIGN MACHINE	product of Ken Orr and Associate, Inc.
Evolutionary Development Methodology	registered trademark of EDM Management Systems Inc.
Excelerator	registered trademark of Index Technology Corporation
Flexible/3000	product of Sages-America
HPSQL/V	registered trademark of Hewlett-Packard Co.
Hypercard	registered trademark of Apple Computer, Inc.
IMACS*LYNX	registered trademark of Imacs Systems Corporation
Inform/3000	registered trademark of Hewlett-Parkard Co.
Insight	product of Computing Capabilities Corporation
Knowledge Craft	registered trademark of Carnegie Group Inc.
M.1.	product of Teknowledge Inc.
OMNIDEX	product of Dynamic Information System Corporation
ORACLE	registered trademark of Oracle Corporation
Personal Consultant +	registered trademark of Texas Instruments Incorporated
Powerhouse	registered trademark of Cognos Incorporated
PROKIT*WORKBENCH	registered trademark of McDonnell Douglas Information Systems
Protos	registered trademark of Protos Software Company
Reflection	registered trademark of Walker Richer & Quinn, Inc.
RELATE/3000	registered trademark of Computer Representative, Inc.
SORTMATE	product of Pacific Coast Building Products, Inc.
Speedware	product of Infocentre
SUPRTOOL	product of Robelle Consulting Ltd.
SYBASE	registered trademark of Sybase, Inc.
System Z	product of Zortec, Inc.
TODAY	product of bbj Computer Services, Inc.
Transact 3000	product of Hewlett-Parkard Co.
YOURDON ANALYST/DESIGN TOOLKIT	product of Yourdon, Inc.

DOCUMENT MANAGEMENT SYSTEM FOR THE UNSOPHISTICATED USER

Diane W. Harman
Boeing Computer Services
P.O. Box 24346, MS 6R-37
Seattle, WA 98124-0346

Telephone: (206) 234-7116

INTRODUCTION

The Boeing Electronics On-Line Command Media System (CMS) is a document management system that provides the tools necessary for retrieving a variety of on-line information both quickly and easily. By using built-in search programs, a user can retrieve company policies, procedures, instructions and directives. Selected media can be displayed to the screen or be sent to a local printer. The media selected can be in form of the procedure itself, or a list of media that was qualified by the search.

There are two separate components of CMS: a User System and an Administrator System.

The User System consists of:

1. Search and Find function which allows the user to search document text and cataloging information using document number, document types, partial titles, and document text as criteria. Searches can be independent or can be nested.

2. The latest newsletter which can be read on-line or printed by the user.

3. A menu to locate the nearest hardcopy station.

4. A method of requesting lists of documents that pertain to specific government audit questions.

The Administrative System contains:

1. Data entry and inquiry screens.

2. Menus to automatically generate administrative reports and hardcopy indices for the Command Media libraries.

BACKGROUND

A new Boeing Electronics plant was due to open in Corinth, Texas in the first quarter of 1988. Seattle area Boeing Electronics personnel were asked to provide this factory with a set of existing company policies and procedures (Command Media). The manager of Command Media Administration wanted to explore the possibility of using an on-line computerized document management system, so Boeing Computer Services was called in to assess the project.

A requirement of this system was that it be designed for a Hewlett Packard (HP) Series 70 computer. In order for a system of this type to be successful, it would have to be designed so that an unsophisticated user could search for information with little or no training. The administrators of the Command Media were sophisticated computer users who were already using a PC database that enabled them to track documents by number, to keep cross reference lists and to produce reports.. This functionality needed to be included in the new system. Thus, the two major modules needed to be designed with different levels of user interface to provide the search user with ease of use and the administrative user with flexibility.

Two problems were approached in parallel by the Boeing Computer Services analysts. The first was the problem of what software should be used on the HP to develop and run the system. The second problem was how to get the documents into the HP computer to be indexed and searched.

HP CONSIDERATIONS

The database to be used for CMS was IMAGE/3000. This was dictated by available software and the knowledge of the assigned analyst. OMNIDEX was selected for the document management system because it provided:

1. The capability that was needed for document cataloging, management and administration, as well as the ability to be extensively tailored for this particular application.

2. An interface to Powerhouse 4th generation development language that would provide quick prototyping capability. This was the methodology selected to produce this type of user friendly system.

3. Low cost alternative compared to more highly specialized packages.

4. Speed of retrieval for document searches.

5. Quick retrieval and design flexibility that could be used by multiple projects.

6. A valuable asset to the prototyping method, by allowing "What if?" testing of the database. It allowed keys and access paths to be moved on and off the database to see what would be most effective for this application. This was a feature that was to become more valuable to us, the further we proceeded into the project.

Drawbacks to the selection of OMNIDEX were considered to be:

1. Lack of proximity search
2. Lack of synonym list
3. Inability to limit search to areas of text in this application

DOCUMENT INPUT

Document input was approached by analyzing word processor packages to see which ones gave the most satisfactory conversions to and from ASCII files. Since the input to the system was to be done by contract labor, a PC word processor that is commonly used in the industry was required.

Attempts were made to scan hardcopies, but the analysis proved that re-typing the documents was cheaper than the cleanup process from the scan of diversely created and formatted documents, and it resulted in consistently formatted output. Some documents were obtained electronically and were just uploaded into the HP from PCs using Reflections scripts. These same scripts were used to upload the converted word processor files.

DESIGN METHODOLOGY

A team approach to prototyping was the methodology of choice. This meant that analysts were trained to understand the administrative procedures involved and administrators were trained to understand the development software. Team members developed a preliminary design,and then attended OMNIDEX and Cognos Powerhouse classes together. The process that ensued from this point was referred to by Takeuchi and Nonaka in The New Product Development Game: Harvard Business Review; Jan-Feb, 1986 as 'Scrum'. Picture a Rugby scrum, where the ball is kicked into action while the various members are wrestling for position. The ball is then picked up by a team member, moved down the field and then passed on to someone else. All members are progressing in a pattern toward the goal at the same time. This gives a general idea of the methodology used by the CMS team.

A trial database was designed. OMNIDEX gave us the capability to add, delete and change keys so that we could design the optimum database for our needs. Menu screens were developed in a skeleton of the proposed system and demonstrated to the user for review. Changes suggested by the user were incorporated and other screens and reports were tied in as designed. The screens were tested at each stage by people who were not familiar with computers.

In this method, there is continual feedback amongst the administrators and the developers, and repeated acceptance testing of the system.

IMPLEMENTATION

Once the methodology, software, and document input choices had been approved by the customer, a parallel implementation of the project took place. As a result of the planning that went into the design and organization of the project, the product was quickly delivered in a phased rollout.

The customers for this project were computer oriented, and were able to participate in the overall design of the database, so as the customers were gaining computer knowledge of the computer side of the system, the analysts were able to gain insight into the administrative side of the customer application. As soon as an overall database design was agreed upon, the analyst moved to complete the details of the design and write the Powerhouse schema. Documentation was developed by the analysts in parallel to the design and programming.

While the analysts were proceeding with the database development, the administrators were overseeing the organization of the documents and the process of typing the documents into a word processor. By the time the design phases were completed, a number of documents were waiting to be uploaded to the HP. An analyst developed a menu driven PC script that allowed entire disks of files to be renamed and uploaded into specified directories on the HP. A test search screen to experiment with the core function of the system was put in place first and default data entry screens were brought up within DATADEX, a component of OMNIDEX. The advantages of this approach allowed everyone involved to work with the keyword search program and to become familiar with it, and data entry could proceed in parallel with screen and report development. Thus, the administrators could be actually using the prototyping stages to input and validate the actual data that was to be used in production.

The actual process of setting up a document management system requires that the document files be moved into the HP, that the corresponding cataloging information (including a field that contains the file name) be entered and that the file itself be keyworded into the database. In this system, the entire text of the file, excluding certain insignificant words, was keyworded for retrieval.

DESCRIPTION OF SYSTEM

The heart of CMS is the Search and Find capability. The documents are stored on-line as ASCII files in an HP3000, Series 70. The files average five hardcopy pages in length. Cataloging information, such as: the filename where the document is stored, pertinent dates, titles, status and cross referencing information was entered into an IMAGE database enhanced with OMNIDEX. The entire text of the documents, with the exception of words entered into an exclusion list, is keyworded for retrieval.

The user interface was developed using Powerhouse Quick and Quiz. One of the primary customer requirements was that even the most unsophisticated user could use the search facility with little or no training. All development was geared towards producing a "user seductive" system that was fast and forgiving. The developers were determined that the user should have to have no previous computer knowledge, other than directions for logging into the system. The entire system is Function Key driven with incorporated Help documents displayed by means of OMNIDEX's ODXVIEW intrinsic.

The completed document management system integrates text files, an IMAGE database, OMNIDEX enhancements, Powerhouse applications, a Reflections interface for loading PC files to the HP, and an optional HPDesk interface for the user.

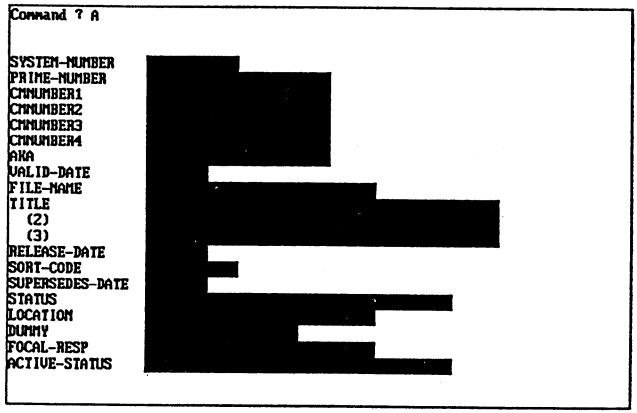
GENERIC SYSTEM

In order to provide a measure of the system that was developed, a description of the generic document management application will be given. A generic system, using an IMAGE database, OMNIDEX intrinsics and the OMNIDEX utility, DATADEX, can configure quick prototype default screens with minimal customization. The basic default screen for the User Search and Find (customized to only show pertinent fields) is shown below:

```
DS76003.2.02   Datadex/3000     Data set 4: PRIME
Command ? 0
               ** Omnidex subsystem.  Type E to exit **

PRIME-NUMBER?
TITLE?
STATUS?
RELEASE-DATE?
FOCAL-RESP?
Document keywords?
```

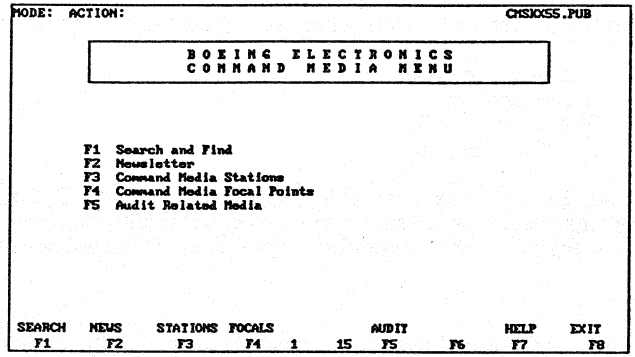
The default data entry screen for the primary master dataset looks like this:

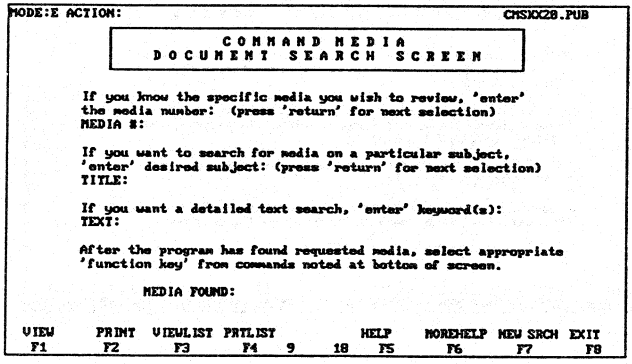


Entry into linked fields in other datasets must be done via "Jumps" and the data re-entered. There are no validation procedures, or specialized update procedures. The user must have knowledge of Boolean operators, wildcards and DATADEX commands.

CUSTOMIZED SYSTEM

The customized system developed with Powerhouse gives the User the following set of screens:





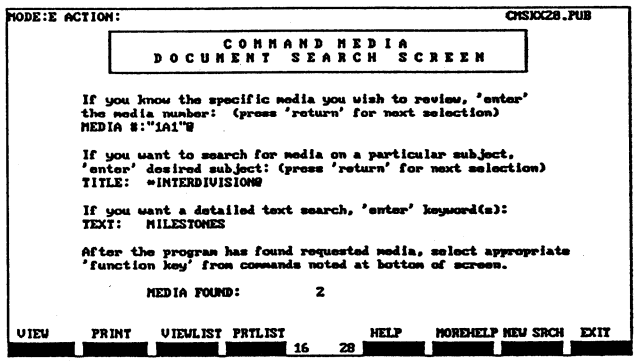
Note that the system is totally Function Key driven. The Function Keys provide a menu of choices for the novice user. The Boolean operators are replaced with English logic words. The user is carefully walked through the Search and Find procedure, as well as subsequent screens. Help screens are MPE flat files displayed with ODXVIEW.

By calling up the Search and Find function, a document can be located by:

1. Document number
2. Document type
3. Words in the title
d. Words in the text

Wild cards, nested searches and multiple keywords can be used. The programmer fed wild card characters, nested search indicators, and English word replacements for Boolean operators where appropriate, so that the user did not need to know them.

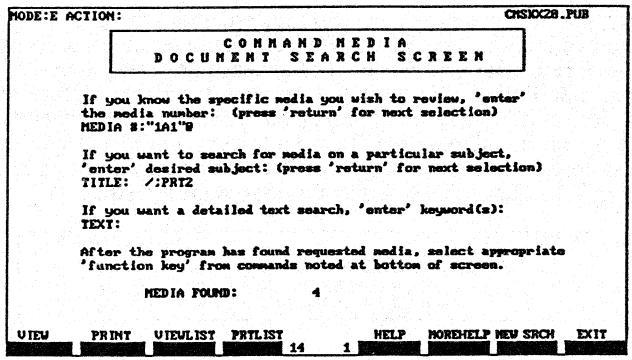
If a document number is entered in the first search field, and then a word in the title and/or a word in the text are also entered as criteria, then an asterisk is automatically fed to the program to indicate a nested search.



After a document(s) is located, the user can:

1. View a list of qualifying documents.
2. Print a list of qualifying documents on a slave printer.
3. View the document(s).
4. Print the document(s) on a slave printer.

These searches can be intimated without the user knowing to return to the Action field by feeding a /;action into the Function Key definition.



When the user requests a list, the screen code creates a report listing the documents found. This report is written to a file and then the file is viewed with ODX'VIEW. This allows the user to move around within the list, rather than just scroll through it.

An interesting example of some of the customization of the search function was the fact that an identification number could appear in one of five fields in the database. The user should be able to enter one of several sections of these numbers to look for a match, and it was immaterial which of the numbers matched the criterion. The identification number fields consisted of a division acronym [1:4], a document type [6:8], and a document number [9:24],[10:24].

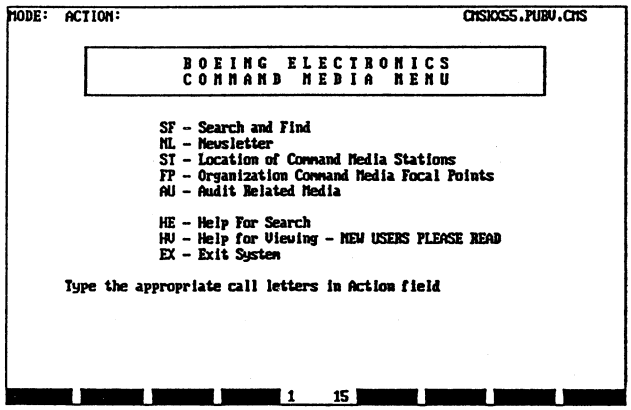
OMNIDEX allows grouping of fields to be included in a single search operation, so the fields were redefined a number of times and all redefinitions for all five identification fields were grouped so that any meaningful combination of identifiers could be used as criteria.

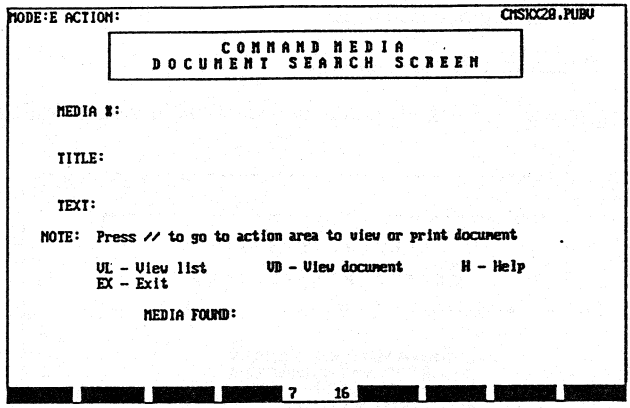
With another screen, the employee can locate command media pertinent to a particular task and can either view or print a list of the documents.

ACCESSIBILITY

The Command Media program has been set up so that HPDesk users can run it within their HPDesk accounts. This can provide access control to the system.

Surprisingly, the only complaint that was received about the User system after it was put into production, was that non-HP and "dumb terminal" users could not access it. As a result, an alternative set of screens was designed that removed function key capabilities and substituted screen instructions. Unfortunately, this system requires a little more user training, but extended the availability of the system to most of the organization. In order to permit our Ethernet VAX users to access the system, a file server on the DECNET was hardwired to a MUX on the HP network. This allows ANSI-terminal access to the system.





ADMINISTRATION

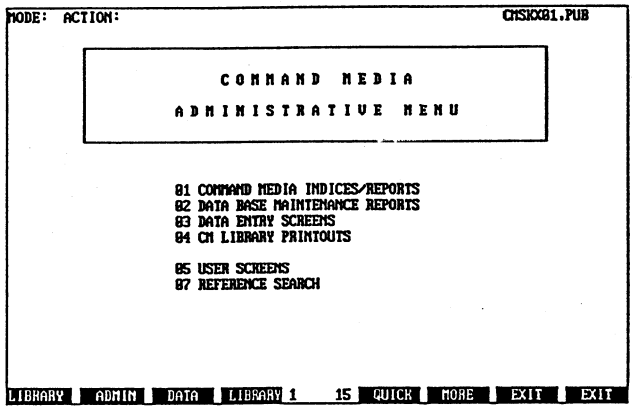
The Command Media administrators have a rigorous, auditable procedure to follow in managing the Command Media documents and library. They must be able to do such things as:

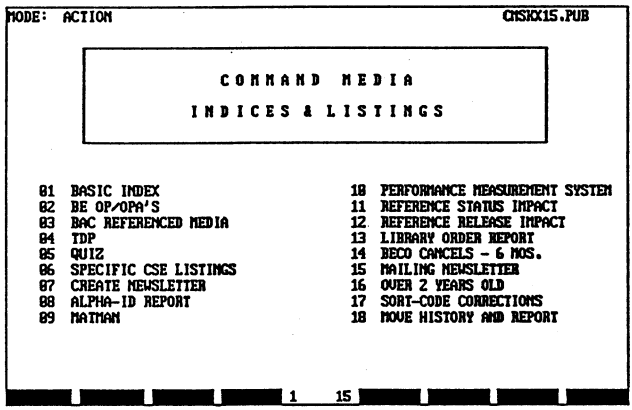
1. Maintain cataloging information on documents.
2. Produce indices of documents for the various books found throughout the division.
3. Produce reports that indicate status of documents.
4. Identify focal points for documents.
5. Identify which documents answer specific government audit questions.
6. Be able to cross reference documents that refer to each other, so that changes on one document can be coordinated by with related documents.
7. Report on documents by originating division.
8. Keyword new document.
9. Enter information into all datasets.

10. Check database capacities.

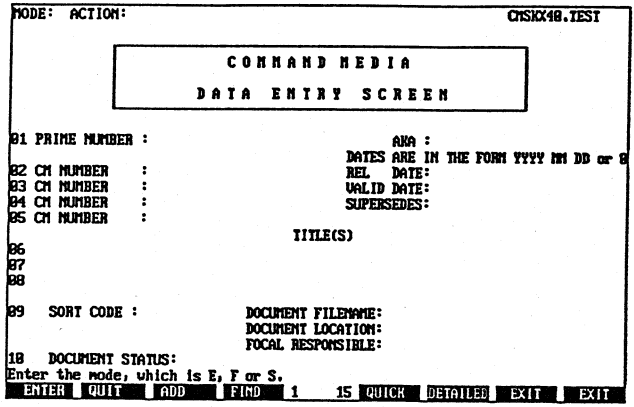
11. Maintain a history database.

The following screens show the types of administrative functions available:





The principle data entry screen is shown below. As documents are entered, the system reference number is stored in a file. Every evening a stream job is run that looks to see if there are new entries in that file. If there are, a "ghost" QUICK screen is run that keywords the text referenced by the system number.



Indices and other reports are generated by selection from QUICK menus. Cross reference reports are available so that the administrators can determine which documents are referenced by an altered document, and which documents are referenced on other documents. This capability has been cited as leading our industry in the ability to meet compliance audits.

SUMMARY

In conclusion, the Command Media System has been immensely successful. It has been in production for a year with very few problems. The system is referenced frequently by the user community, an average of fifty inquiries per day with no noticeable effect on response time.

The administrators have continued to refine the screens and reports as their needs evolved. The result has been a high level of automation and satisfaction with the system.

The system has received commendations as a well, designed, cost effective project that is truly usable by the unsophisticated user. The team members all received Special Achievement Awards for their effort.

I would like to express my admiration for the fine work of Joseph Fuller (programmer/magician), Marilyn Shelton (administrator/analyst), Sharon Batten (administrator), Sean O'Sullivan (analyst) and Susan Vann (administrator/analyst). The secret to the success of the project was that it really was a lot of fun. Thanks and good luck to you all.

Roles of Management in 4GL Application Development

Leigh Solland
Cognos Corporation
2301 E. Lamar Blvd., Suite 250
Arlington, TX 76006
(817) 649-1944

Abstract

This paper explores the roles of non-data-processing management during the development of computerized commercial application systems, from three perspectives:

1. Senior Management: scope approval, financial funding, quick policy decisions, resolution of interdepartmental issues and moral support;

2. User Department Management: detailed involvement in the iterative development process, including requirements analysis, reviews of prototype and subsequent iterations, documentation and implementation.

3. Project Management: keeping the project on track, estimating, and coordinating the efforts of the project team.

The paper expands on several ideas from Cognos' Application Development Guidelines, with specific reference to the formation and effective use of the Project Review Committee and the User Committee/Lead Team.

It is based upon personal experience in many application development projects.

What is 4GL Application Development?

Since the first computer pioneers decided to automate their payroll and billing systems, an enormous effort has gone into making the automation process more efficient, more accurate and more predictable. After four decades of commercial computing, we are still making the transition from "black art programming" to a more scientific, professional form of application development.

There have been many advances along the way. Some have been the direct result of research in Computer Science, some have been logical derivations from advances in computer or communications equipment, and some have been "field developed" by the masses of working programmers and analysts.

Sometimes advances in software have driven other developments, or certain incremental improvements have made theoretical ideas practical.

One such advance was the arrival of Fourth Generation Languages (4GL's) at the beginning of the 1980's.

4GL's, when properly used, reduce the time and amount of programming spectacularly. This in turn reduces the elapsed time for a given project, and reduces concurrently the number of people involved, both of which add even more to the overall efficiency of the effort. Well-designed 4GL-based systems inherently perform and utilize machine resources very efficiently.

The reduced programming effort has exposed application design and implementation as the weak links in the development cycle. This has led to a resurgence of interest in software project management, and to a brand-new market in so-called Computer Aided Software Engineering (CASE) tools.

At Cognos, we have learned in a very practical way that development with 4GL's is not the same as development with older methods and tools. The time gained by reducing the programming effort can best be used in improving the requirements definition and system analysis process, data design is far more important than most of us suspected, and the users of the system are an immensely important and useful resource in successful development and implementation of an application.

User Involvement

A key tenet of the modern kind of application development is that "The System Belongs To The Users!"

What does this mean?

Each completed system (not necessarily the hardware or software, but the conceptual system) is the property of the user departments, and has an "Owner of Record", a manager or other executive outside the MIS department.

The accuracy of data is the responsibility of the users. If it is wrong, it is up to them to fix it.

The application is managed and operated by the users. While professional computer operators may be responsible for tapes, printers, backups and so on, users make the decisions about what processes are run, when, and in what sequence.

Development is based on user requirements first and foremost. The MIS professionals involved in development bring their knowledge of Computer Science to the process to work out how best to implement those user requirements. They are also responsible for taking the more global view of the organization, considering integration constraints, multiple user access, shared data, and so on.

Quick, Accurate Development

Three generally unrelated trends have come together to bring a unique concept to 4GL development: swift development of a system based on an even swifter functional prototype.

1. Engineering has long used prototypes and models (have you ever flown in a plane which was not tested in scale model form in a wind tunnel?). However, software engineering has not in the past taken advantage of this. One may speculate that this is because the software system itself is supposed to be the model (of the real-world processing), or simply because the programming languages made it sufficiently difficult to develop the system that there was not time to build a prototype first. A more cynical observer might feel that we have been

building the prototypes all along; we just haven't been throwing them away and building the production model!

2. Measurement in software development has borne out what many people instinctively understand, that developers working on simpler, smaller projects are much more efficient than those involved in larger, more complex efforts. As Frederick Brooks says in The Mythical Man-Month, "adding manpower to a late project only makes it later."

3. A major feature of 4GL's is that they are able to make a lot of decisions where specifications are incomplete or missing, using "default logic". This is ideal for the creation of a "quick and dirty" prototype, with the general functionality of the final system but without the myriad of details. (This may be contrasted with traditional development, where it was difficult to have anything until one had everything, even though it has been attempted with structured programming and other techniques.) Code generators have made this process even faster and more effortless.

With the first functional prototype working, the detailed requirements may be worked out on a live model, with the active participation of the users.

The programming is simplified not only by the use of the high level 4GL language, but also by deliberate analysis and modularization with the 4GL in mind.

To achieve all this, a focused effort on the part of all involved parties is required.

The Project Concept

Many MIS organizations are missing out on the use of project management to keep their efforts rolling in an agreed direction. They have historically just made their "best effort" at getting things done as soon as possible.

Working in a consulting organization, we have found that the project orientation greatly helps in maintaining focus, managing change and minimizing distractions and interruptions.

The first, smallest and hardest step in instituting a project-oriented environment is the establishment of a formal scope,

or charter, for each piece of work to be done. This prevents wild goose chases and keeps the project within achievable limits. It also provides a means for getting senior management involved in these projects, about which more later.

A project-oriented philosophy will help to keep each project dealing with business objectives, not the latest desires of the MIS people. This should be stressed constantly.

Activities will be planned, not just ad hoc. This will help to ensure that people are aware of and prepared for meetings, and that sufficient time is built in at the beginning for analysis, testing, data conversion, documentation and the myriad of things to be done for implementation.

Scheduling of resources and establishment of clearly defined deliverables and milestones are the critical tasks in the implementation of the project concept. A project schedule helps the users to schedule their own staff and other changes, and sets realistic expectations as to what can be achieved by when. However, it takes time and effort to do the estimating and scheduling up front, and this must be supported by management.

Of course, no plan is worthwhile unless it is used. Managing to the plan requires discipline and training, but can be very effective in keeping goals in sight. With very creative people such as software developers, there is always a tendency to add "just one more feature" or try "just one more idea". The project plan provides a way for them and their managers to know when there are no "just one more"'s.

Information Engineering

The current buzzword for computer systems analysis is "Information Engineering". This recognizes that the information provided by the system is the goal, and not the system itself.

Information Engineering is the analysis of business information requirements, with the aim of developing a data structure and process to support those requirements.

Multiple viewpoints must be considered, with the operational, integrated systems of today. For instance, an order entry clerk, a sales representative, a sales manager, and an ac-

counts receivable collector have quite different views of what a sales order looks like. In this instance, the seven blind men who described the elephant in the old story must all be considered correct!

Older commercial applications systems analysis concentrated on modeling the processes of the operation. They then would attempt to write programs to execute these models. This was particularly effective in the early batch systems, but proved less useful with the advent of on-line, interactive systems where humans were once again involved. In this older method, data structure and design was secondary to the processes, and was basically a way to buffer data between processes.

As commercial systems grew up, it became apparent that data should be shared between systems and programs, to avoid redundant maintenance effort, ensure data integrity and consistency, and minimize computer storage requirements.

It then became apparent that the data of interest to a business has a relatively long life, compared to the processes that manage it. For instance, even if order processing changes for an enterprise, it is still interested in orders, customers, products and so on. In addition, even if a particular customer comes and goes, the necessity to maintain data about customers survives.

This all has led to the realization that the data model is crucial to the successful implementation of systems to serve the whole organization. The Entity/Relationship (E/R) model developed by Peter Chen is a convenient way of looking at the overall structure. Within each entity or association, normalization to 3rd normal form enhances efficiency and identifies any structural weaknesses and anomalies not shown by the E/R model.

From this model, the data may be implemented in relational data bases, network data bases, indexed files or some combination of these. It is optimized by design for general access, and may be further optimized by a data base administrator for specific access.

The data model alone is not enough for complex systems which do include "process", although it is sufficient for "file cabinet" applications. It is important to develop a model of the processes for both current and desired operations.

This may be done with any of a host of techniques developed for batch systems.

The process model has the additional benefit of clearly identifying interfaces between sections of the operation or the enterprise. These may be used to modularize development where possible, making each project smaller and thereby reducing its overhead.

Out of this exercise should come a data dictionary based on the data model, a structure chart based on the identified viewpoints, and a process model which can be used for subsystem and program identification.

Iterative Prototyping

It is desirable to generate the functional prototype as soon as possible, so that the developers and users of the system may use the working prototype as a focal point for discussion. Therefore, the first iteration is produced using the best code generation facility available.

As noted before, if the best code generator available is a programmer writing COBOL code, the prototype will probably not be produced quickly enough to be effective. It is important to show results quickly, to keep everyone motivated and interested in the project. It is also important to get the system built before requirements or users change significantly.

Code generation capabilities vary in 4GL languages, but they typically base their activities on the data model. This implies a somewhat simplified, though not inaccurate, view of the world as a place where all processing consists of table management and reporting.

Deliberate enhancements are made to this original prototype of the system, with user review and input at each step. At no time is there an effort to build the whole system at once, but rather to put it together a layer at a time.

For instance, the second round will ensure that data validation is complete, so that only acceptable data will be allowed into the system. The third round will deal with the additional viewpoints into the system, adding cosmetic features as required to satisfy each representative type of user. Next,

control, security and any "batch process" requirements are addressed.

After the iterations are complete and the users are satisfied, the system must still be completed before it can be implemented. This involves such "non-user" features as audit trails, backup/restore procedures, and maintenance documentation. User "how-to" documentation may be produced at this time by the user organization, in parallel with the developers' completion activity, because the "look and feel" of the system is frozen.

User training should also be carried out at this time. If the system involves new hardware, it should be in place by now (The installation of hardware, communications equipment, phone lines and so on can proceed in parallel with the software development effort.)

Finally, the system is implemented. This may mean batch or manual data conversion, parallel operations, starting with pilot sites, or any number of activities, all of which must be planned and scheduled. It may also mean having new forms on hand, counting inventory stock, or other matters completely out of the realm of "data processing".

What is Required from Management?

We have learned over the years that for MIS, EDP or whatever the computer department is called to attempt to do this all alone is disastrous.

Management support is needed, for moral and financial support, for requirements and detailed information about the application itself, and for general assistance in managing relatively large efforts.

Senior Management

Senior management has historically not taken a particularly active interest in the computerization of their organization, particularly the business portions.

This is rather surprising, considering the impact of the computer systems on the organization, their cost, and the potential for competitive advantage with good information.

It may possibly be attributable to a certain discomfort with the technical aspects of computer systems, or perhaps to a missing layer of management between senior management and MIS management which could translate between their two languages of work.

In any case, senior management involvement will increase the probability of success. The larger the project, and the more organizational divisions it cuts across, the higher the level of management which must be involved.

The first responsibility of senior management is approval of the scope for each project. Where does this fit in to the overall business plan? What is it expected to do for the organization? The business plan may be in a binder, on a computerized enterprise model, or on the back of an envelope in the CEO's pocket, but every organization has one.

Funding is always required, and the user manager may or may not be able to command the necessary resources. Projects need not only cash, but people, equipment, communications, extra data storage, tools, software, desks, pens and so on.

The analytical process is liable to uncover gaps in policy. It is the responsibility of senior management to make policy decisions promptly, so as not to slow the project. The alternative is to let the programmers create the policy by the way they write the programs!

Anomalies may be discovered, as well, in the way issues are viewed by various departments. A very important role for senior management is to resolve these interdepartmental issues for the benefit of the organization as a whole.

Finally, senior management can use its "ceremonial" nature to lend a lot of moral support. Publishing a picture of the President, Project Manager, and User Department Manager together in the company newsletter, for instance, can smooth out a lot of rough spots in a hurry. Typically, senior managers have some talent for leadership and motivation, and this can be lent to a project with tremendous results.

User Department Management

User managers generally have the most to gain and the most to lose in the implementation of a new system. They also have to incur the greatest cost in supplying requirements, checking for validity and implementing the new system.

The user manager is responsible for identifying the benefits of the new system (or improvement to the old). While the MIS manager may be able to estimate development costs, the costs of implementation are again in the realm of the user manager (or managers, in a complex system). This cost/benefit analysis should be reviewed periodically during the project, and especially at the end.

The user department is responsible for providing detailed requirements to the development team. For the user manager, this means supplying personal time to the developers, as well as making available staff representing each viewpoint to be considered. These will typically be the most creative, productive, knowledgeable staff members in each functional area, which will impact production even further.

Not only are these people required for the initial requirements analysis interview process, but also for periodic reviews of the prototype system and subsequent iterations.

Users are responsible for documenting how to use their own parts of the system, although MIS may be able to provide standard forms to accomplish this. In some large organizations, there may even be professional writers available, but in any case the original material should be produced by the users, in their own wording and style.

Training should also be carried out by the users themselves. Ideally, each person representing a particular "viewpoint" will be familiar with the system from the iterative development process, and will in turn develop and carry out training for others in the same viewpoint.

As mentioned earlier, the users should take ownership of their own systems, and responsibility for the quality of the data. This means that, although the developers can create some test data, and are responsible for any mass conversion of data for a new system, the users have two duties. They can be reviewing and cleaning up any old data even before the new system is implemented, and they should provide representative test data to the development team.

Finally, the user manager must provide both the resources and management for the implementation process. By this time, the technical part of development is completed and it is time to integrate the new system into the everyday life of the organization.

Project Management

The MIS group may or may not be able to supply the project management required, depending on the size of the organization and the level of people.

This should not prevent the organization from enjoying the benefits of formal project management, if this skill set is available elsewhere in the organization.

The advantage of a project manager from outside MIS is that he may be more likely to consider non-data-processing needs such as personnel moves, new buildings, phone lines and the like than someone with a data processing orientation.

The advantage of a project manager from MIS is that he is in

a position to more realistically evaluate the developers' technical problems and needs, and to quickly estimate the impact of a requested change on the development process.

Whether the project manager is from MIS, the user department or elsewhere, or is even a consultant, the use of formal project management will reduce wasted time and interruptions, keep the technical people on track, and manage changes requested by an excited user community.

How Can this be Implemented?

There are some very practical steps which can be taken to make these ideas a reality.

Naturally, every change has its own disruptions, and everything matures over time, so these ideas will provide more and more benefit as time passes.

Adopt Project Philosophy

It is every bit as important to start thinking "Project" on small tasks as it is to see the 2-year MRP implementation as a project. Otherwise, until people are comfortable with the project concept, work will be fragmented even more than usual because "this is just a little fix" means that it does not need to be dealt with as a project.

The MIS organization has the biggest job, and often the most to learn.

To start with, scope everything. Require cost/benefit analysis on everything. Prioritize based on return to the overall organization and refuse to do work without a clear benefit (senior management should decide, in the event of a conflict). And DELIVER!

Assign project managers to everything MIS is working on, to give even the more junior members of the development staff a feeling of "ownership" and some experience with project control.

Estimate and plan everything, without exception.

Manage the work to the plan. If things aren't getting done, improve the plan. This will, over time, improve the planning as planners learn and get better at their job.

Always conduct project post-mortems. Review the original scope and business objectives. Look at the cost/benefit analysis and justification, and try to measure the resulting benefits. Add up the costs of the project, and get better at estimating for next time. This applies to user managers, as well as MIS developers.

Obviously, this process will be more or less formal, depending on the size of the project.

Establish Project Review Committee

A Project Review Committee should be implemented or formalized. This is the mechanism for interface with senior management.

It must include sufficient organizational power that quick, binding decisions can be made. The more power, the better.

It should not include MIS management, although they may have an advisory role.

It should be small enough to get some work done. A group of larger than five is too hard to focus, and three seems to work well.

This body controls the scope and funding of the project. It approves the original scope, and any subsequent changes in scope.

It also gives approval to proceed at designated milestones. Typically, these are: completion of the initial problem analysis, completion of detailed requirements analysis, and completion of system development. Finally, the Project Review Committee is involved in the project post-mortem, at some designated time after implementation. Therefore, it will deal with a given project four or five times if no problems are encountered.

The Project Review Committee should be a permanent organization, meeting as required but no less frequently than every month in a typical organization.

User Committee or Lead Team

The name of the formal user organization depends on the style of the organization. More conservative thinkers appear to like the term "User Committee", while "Lead Team" is preferred by more radical thinkers.

Whatever the name, this organization ideally represents all the viewpoints affected by a project. This may range from departmental managers to truck drivers to telemarketers. It may include members of senior management, if they are directly affected by the system. It should not be restricted to management or office staff, but should include "white-collar", "blue-collar", "pink-collar" and "no-collar" workers as necessary.

Because of the necessity for user involvement in the iterative prototyping process, this group will meet formally at least eight to ten times during a non-trivial project.

Management of the user department or departments may even decide to provide full-time user resources to the development team. This can be invaluable, as all the subtle decisions are made with the immediate knowledge and input of the users. A person in this role needs to be knowledgeable, so a new hire or a junior clerk is not particularly useful.

The User Committee/Lead Team provides a two-way conduit for information between the affected parts of the organization and the development team.

Not only do they provide requirements (and checks on meeting those) to the developers, but they should also provide regular information to their "constituents" on the progress of the project. They are also responsible for "how-to" documentation and training in their functional area.

These are temporary positions, and the organization should be constituted only for the life of the project. It may be convenient to reconvene it from time to time, say every six months, to review the system in production and discuss changes and enhancements, but it may be "overkill".

Summary

1. Adopt a "Project" philosophy.
2. Establish a Project Review Committee.
3. Set up User Committees/Lead Teams.
4. Plan everything.
5. Talk to each other.
6. Listen to each other.

Managing Your Software Life Cycle:
Program Library Management, Change Control, and Release Management

Betsy Leight
OPERATIONS CONTROL SYSTEMS
560 San Antonio Road
Palo Alto, CA 94306
(415)493-4122

INTRODUCTION

Today's HP 3000 professionals are becoming increasingly concerned about software and data integrity. And as a result, implementing a reliable method of tightening control over the changes to systems and applications has become a prime objective of many shops.

An isolated change may initially seem harmless, but the ripple effect upon interrelated systems can often be devastating. As the volume of code used in a shop becomes greater, the effect of these change can become staggering. With an industry average cost of $50 per line, the software investment of even a small site can be valued well into the millions of dollars.

Despite the enormous value of today's software applications, much of it is vulnerable to inadvertent errors. The reason: MIS departments often use tedious manual procedures to track program changes, and these procedures are seldom if ever followed faithfully throughout the organization. Without an improved approach to monitoring how and why changes are made, a company's entire computer and software asset are at serious risk.

First, consider the case of our firm, Operations Control Systems. With over 4000 products installed throughout the world, our customer support staff receives requests for dozens of modifications every month. At the same time as these maintenance and enhancement requests are coming in, our R&D team is also under pressure to develop new products. Shipping a new release represents a major project for almost every department within the company. Without an efficient means to manage changes, maintenance work would forever delay progress on new development. Through the task of managing our own internal software maintenance and release management process, we at OCS developed much of our initial expertise in the area of change management.

Through feedback from our customers, we started to refine these techniques and make them available to the general HP user. When we first surveyed our customer base in 1986, we found that the application maintenance backlog was not unique to software developers. Some customers who relied entirely on vendor-supplied software with seemingly minor modifications ran as many as 6 months to 2 years behind schedule on their maintenance work. Thus, while many non-data-processing executives think that their MIS staff is focusing on new applications, the reality may be that MIS is bogged down in maintenance and change related activities.

This highlights a second major need for change management. Change management improves the ability of the MIS department to react to user demands for program changes.

THE NEED FOR CHANGE CONTROL

In order to understand the change control problem and how to solve it, let's explore the nature of the problem in an uncontrolled environment. The problem stems from the fact that in many HP 3000 shops, programmers can access production source directly. This can have several consequences.

First, consider an environment where multiple programmers make simultaneous changes to source code files. In this situation, the last programmer to update each file overwrites the changes made by other programmers. The result is frustration and wasted time synchronizing and retesting all of the changes.

Second, a careless programmer might inadvertently destroy the source code of a critical application. If a production error occurs and the source code cannot be restored easily, production could be delayed for hours or even days. The original author may not be available. The program may even have to be rewritten from an old copy, wasting development and test time and possibly reintroducing old bugs or creating new ones.

Third, the development procedures found in many shops eventually result in discrepancies between source and object code files. That is, the master (or "official") source files may not recompile into the current production object files. Since it is almost impossible to recreate source from object code, days might be spent searching for the correct version, recompiling it and validating it against the current production code. And, with all of this effort, there will be no guarantee that the source will exactly recreate the current production code.

These situations point out a fundamental rule of good change control: Programmers should never have unrestricted access to production files!

At first glance, these problems may appear obvious, but the fact is that many HP 3000 data centers continually react to the problems they cause, rather than introduce standards and controls to avoid them. After all, a data center is there to support its users. This charter forces management to focus resources on meeting daily user requests, distributing reports, managing hardware and completing batch production, rather than on implementing long-term solutions to problems that are not well understood. Although many shops can run error free for months, an innocent looking error can snowball into a major catastrophe.

If this is the orientation of your shop, you might want to consider this fact: the efficiency and accuracy of your work depends upon the integrity of your software assets.

Is your data center a candidate for better change control? Try taking the following test.

What software systems do you have?
What programs comprise these systems?
How often do these programs change?
Who is responsible for the changes?
Where and why were the last changes made?
Where are the appropriate backup copies?
Which user requested the last set of changes?
What is the status of a specific change request?

If you can't answer all these questions readily, your shop probably needs a better approach to managing changes.

Proper change control provides immediate answers to these critical questions. The result is more than just visibility and auditability. It is more than data integrity and improved maintenance efficiency. A good change control system ensures that every program change is authorized and controlled, making it possible to quickly restore systems when necessary. Change control techniques also make it possible to monitor changes made to software from the earliest stages of development through the testing phase and all the way through the move to production process.

Change control also improves the Quality Assurance process. It allows operations personnel to test new systems, moving software from development to test, or from test to production. And, all of this will be done with complete audit trails that allow the identification of each entity promoted and tested.

Another important aspect of change control is provision for management inquiry in areas such as: the status of a change, the current version of a file, the change history of a file, and the impact of a proposed change. Supplying this information can often be a tremendous benefit to managing the software life cycle.

RECOMMENDED CONTROLS FOR ALL SHOPS

In order to achieve the benefits of change control, a shop should address each of the following five major categories:

Control through stages: Keeping control over the movement of software through the development and maintenance process.

Change history: Providing a complete history of every change requested and made to the source code.

Control over source and procedures: Maintaining control over users authorized to access source code and the processes that render source into executable form.

Security: Retaining control over the personnel who are authorized to make program changes.

Inventory: Maintaining an accurate inventory of all the programs, files and other components of each system.

A more detailed checklist identifying good change control practices appears at the end of this article.

With these items in mind, lets look at the costs and benefits of manual change control techniques.

MANUAL CHANGE CONTROL

HP 3000 users have developed a variety of creative procedures for controlling changes in their environments.

A common strategy requires programmers to FCOPY source code from the production location into a development location. This strategy can be implemented at three levels.

1. The development area may be nothing more than a set of individual groups where files reside. In this case, each programmer copies all of the necessary files into his own group where he makes the appropriate changes. There is no standardized testing environment.
2. Separate account structures may be maintained for production and development. Often a separate account structure is maintained for testing as well, to duplicate the production account structure. Thus, each programmer can be confident that testing is conducted in an environment that closely resembles production. Establishing separate account structures on a single computer often produces adequate results.
3. A separate development computer may be used, thus eliminating the possibility of direct access to master files located on the production computer. Although this approach is ideal, it is not possible without a separate development computer.

Unless stringent controls exist, however, this strategy does not guarantee that only one individual can check out a particular file at a time. And, since programmers are not restricted from accessing production accounts, there is no way to audit their access.

Once the development is completed, many shops allow the same programmer to test their own work and simply overlay the original production files with the new version of code. Since such a procedure seldom represents adequate control, it is widely suggested that a formal Quality Assurance (Q/A) process be used.

There are several ways to initiate such a process, depending upon the resources that are available. Smaller companies may have programmers Q/A test their colleagues' development efforts. In these cases, tests are usually performed in the development account.

Larger organizations may dedicate one or more individuals whose sole function is testing. Here, a separate Q/A test account is generally set up to reflect the production account structure.

In this case it is vital that the developer is finished, all claims to the code are relinquished. After it has been moved to Q/A for testing, the one copy of the code will exist only in Q/A. If the Q/A analyst locates an error, the code will be returned to the original developer for revision. It is now Q/A's turn to relinquish all claim to the code by moving it back to the development location and purging it from Q/A. If simultaneous copies were to exist in both locations, last-minute changes in development might not be included in the Q/A

version. Thus, the final production version would not be accurate. It is surprising how often this obvious safeguard is overlooked.

At the conclusion of the Q/A phase, another step will often exist. A higher level manager will perform a final approval on the development code to certify that all standards have been met and all tests have been satisfactorily completed.

Following final approval, the finished code is ready to be moved into the production location. Since the new files will be copied into production with the same names as the original files, the original files must first be backed up to tape to avoid loss of the earlier version. Next code must be recompiled to assure an exact match between source and object. Lastly, JCL and other files must be updated.

This update process is tedious, time-consuming and error prone, especially when large numbers of files are involved. Nevertheless, precise standards cannot be eliminated at this final stage or the shop will run the risk of serious inconsistencies.

Based on the previous discussion, it is possible to state several general rules for good change control:

First, establish separate accounts for production (master files), development and, if possible, test. Test accounts should be mirror copies of production. This allows files to retain their original FILE.GROUP names as they are moved from test to production and makes it possible to visualize the link between a developing program and its production/master version.

Second, when files are "checked out" from the production location to development, a copy should be made. The original source should never be destroyed. However, when development is complete, files should be moved to the test location, thus eliminating synchronization problems with old versions.

Third, when Q/A has approved all changes, a project leader or manager should verify that adequate and accurate test procedures have been followed. Only at this point should code be moved into the production library. If system loads are heavy, such updates could occur in batch and should be initiated by operations or a production Librarian. The timing of the release into production should be coordinated with the user's schedules, with the release of other modules and with documentation changes, etc.

Prior to updating the library, the original production copy should be verified and stored. Without this step it is much more difficult to return to a prior version in the case of a problem.

As a general rule, we recommend that the authority to release files to production be restricted.

AUTOMATED CHANGE CONTROL

Although this article has described the minimum requirements necessary to achieve reliable change control, implementing these controls places a considerable burden on everyone involved in the change process ...unless the process is automated.

Although an automated change control system will require some initial planning, once it is operational, the burden on the MIS staff will actually decrease. The process would be as follows:

The first step in implementing an automated change control system is to define your development environment. Decide what files you want to control, what types of file movements will be performed, what approvals will be required at which stage, and who will be authorized to perform or authorize each type of file movement. This will depend on such things as:

The size of your development staff
The levels of management and job responsibilities in your development group
The size, complexity, and volatility of your applications
The amount of packaged software used by your firm
The physical hardware configuration of your computers
The degree of control which you and your auditors agree is appropriate.

The following examples show four typical development environments, presented in order of increasing complexity.

As you read through them however, keep in mind that these are only intended to be representative examples; each shop is different. A good automated change control system must be configurable to meet the specific needs of the organization.

EXAMPLE 1 - BASIC DEVELOPMENT ENVIRONMENT

The basic development environment is a small shop with a single computer and a staff consisting of two programmers, one day-shift operator, and a shirtsleeves" manager. The development control objectives are basic, but critical:

* They need to know where the current production versions of all source, object, and job files are,
* They need to be sure that all changes are made to copies of those files in a separate "test" location,
* They need to assure that all changes are approved by the manager before they are put into production, and
* They need to do this without further burdening their staff who is already working long hours.

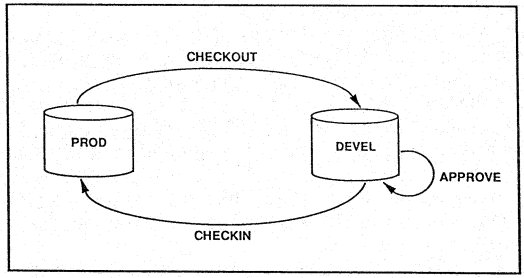
The first step here, as for most installations, is to identify all of the production source, object, and job files. These files need to be grouped together and secured. Through an automated change control

system, this can be accomplished by creating customized filesets which represent whole groups of files. This eliminates the need to change the operating system account/group structure or move any of the files around.

The second step is to define the file movement policy, or steps, that will be allowed in the shop. In this example, we have three basic file movement steps:

1. A CHECKOUT procedure, which copies source, object, or job files from the master location into a development location,
2. An APPROVE step, which allows the manager to stamp a changed file with their approval, and
3. A CHECKIN step, which allows the programmer to move a group of files back into the master library when they have been approved. This step also makes an automatic backup copy of the old master file. In addition, the backup copy is compressed to a small fraction of its original size to save disc space.

This process is illustrated below:



To maintain the security of the master library using MPE alone, the CHECKOUT and CHECKIN steps would have to be performed by a manager, an operator or an additional employee. The approval step would have to be documented on paper. With an automated change control system, the CHECKOUT step can be defined to make test copies of the master files even though the master library is secured. The automated system can keep track of these copies and can prevent multiple programmers from checking out copies of the same file at the same time. The APPROVE step can be defined to mark the file(s) as approved, and can be restricted so that only the manager can use it. Furthermore, the CHECKIN step can be defined to allow the programmer to push his changed, approved files from his test environment into the master library without having to log on to the master account, and automatically archive the old versions.

EXAMPLE 2 - SEPARATE Q/A FUNCTION

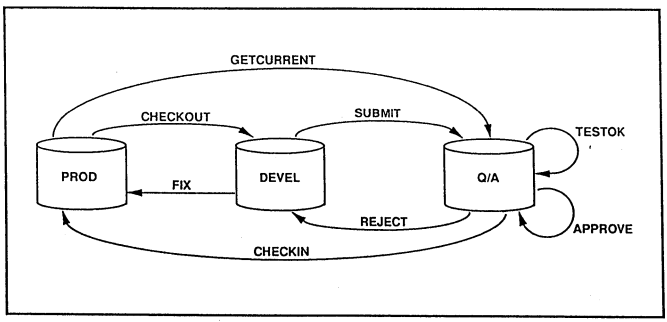
This example illustrates a medium-sized shop with a tightly controlled development process. The MIS organization consists of a manager, six programmers, two operators, and a full-time quality assurance staff of two. Due to the critical nature of their

applications, this shop insists that their Q/A staff perform full unit and system testing on every change. They also require management approval of the testing procedure before any new or changed software can be put into production. They do, however, need to allow their programming staff to make "quick fixes" on an emergency basis, thus bypassing the delays that would normally accompany the Q/A process. It is especially critical that a complete and reliable audit trail be maintained for these quick fixes.

We can define this environment as follows:

1. A CHECKOUT step as described in Example 1. This allows the programmers to easily make development copies of the secured master files, and assures that only one copy at a time is modified (other concurrent copies can be made with read-only access).
2. A SUBMIT step, to allow programmers to move their modified source, object and job streams into the Q/A area.
3. A GETCURRENT step to bring read-only copies from the master library directly into the Q/A area. This allows the Q/A staff to perform an integrated test using the current production versions of programs that are not currently being changed.
4. Since not all changes will pass Q/A, we need a way to get the files back into the development location. We define a REJECT step which will be performed by the Q/A staff when a program fails testing.
5. Q/A will signify that a change has passed its system tests with a new step called TESTOK. This will help the manager keep track of the status of work on various programs and is a positive indicator that testing is complete.
6. We still need an APPROVE step, but it is defined to operate on files in the Q/A location. The APPROVE step will be performed only after they have been TESTOK'd.
7. The CHECKIN step now moves files directly from Q/A to the master library, once they have been APPROVEd.
8. Finally, we define a FIX step, which moves files directly from the development area to the master library.

This development environment is illustrated below:



Here again, the automated system facilitates file movement, emergency fixes, and management approval without burdening their staff. The CHECKOUT, SUBMIT, REJECT, and CHECKIN steps are defined to operate on specific groups of files, so users do not need to fully qualify file references. Wild-card references may also be used to move groups of files at once. These steps can be defined to automatically purge the files from their original location after copying to eliminate any need for additional housekeeping efforts.

Because an audit trail is maintained for each step, no extra effort is required to control use of the emergency FIX capability - the manager gains complete visibility by simply listing all uses of the FIX step.

EXAMPLE 3 - NETWORKING, SOFTWARE DISTRIBUTION, SOURCE/OBJECT SYNCHRONIZATION

The primary difference between this example and the two previous ones is that there are separate computers for production and development. In addition, the company's applications run on multiple remote computers. As in previous examples, the master library contains the production source, object, and job files, but in this case, the object and job files are actually executed on the various remote computers. This adds the important control objective of assuring that all of the remote sites are running the correct versions of the code. To this we will add the audit requirement of source-object synchronization: we must assure that the object files in the master library (and on the remote computers) were in fact generated from the source files in the master library.

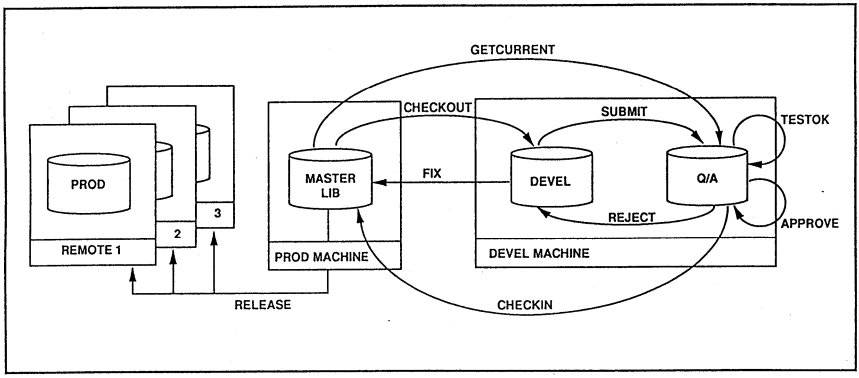
The rules and file movement steps for this environment are the same as for Example #2, with the following exceptions:

1. A new step, called RELEASE, is defined to distribute groups of object and job files to the remote computers. This step will copy the files specified to all of the remote computers, and produce an audit trail to verify that the copies were successfully transferred.

2. The CHECKIN and FIX steps will be modified to move only source and job files into the master library, and to automatically stream compile jobs to generate the object files from the new source. Since the new source has already been compiled to test the changes, this step is redundant, but it is an effective way of assuring that the source and object files always match and are synchronized.

3. While the remainder of the steps function as they did in Example #2, the automated system makes the multi-computer environment transparent to the development staff. Now CHECKOUT, GETCURRENT, CHECKIN, AND FIX all operate between the production and development computers without requiring extra steps or commands.

The resulting environment is illustrated below:



EXAMPLE 4 - PACKAGED SOFTWARE, ADVANCED VERSION CONTROL

In this example, the company uses third-party application software and receives periodic releases of the software from the vendor. Their own programmers have also customized portions of the software in-house. They have developed custom reporting and other extensions to the software. Whenever this shop receives a new release from the vendor, the new software is placed into its own separate account. There it is integrated with existing software by the programming staff, tested by the Q/A group and eventually put into production in the same manner as internally developed software.

Since the software is run on a large number of remote computers, the company releases newly approved production software in stages. First, a small number of remote users conduct a "beta" test for a limited period. When this stage is satisfactorily completed, the software is released to the remainder of the sites. Because of this procedure, the company needs to maintain two or more versions of the software simultaneously, and, in emergency situations, make changes to a prior version without interfering with work performed on the new version.

The rules and file movement steps defined for this environment are the same as Example #3 except for the following:

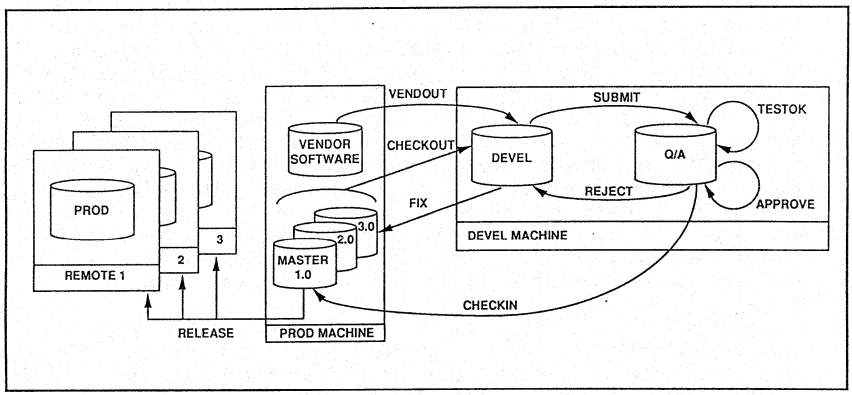
1. A new step, VENDOUT, has been defined for programmers to check out source and job files directly from the vendor software location. The step will prevent the programmer from accidentally getting an old version from a different location.

2. The CHECKOUT step is redefined to select the latest version of the software from the master library. This is accomplished by searching the "new release" location first. If the file is not found there, the "old release" location is searched. This assures that the programmers do not inadvertently use old versions of the software.

3. The CHECKIN and FIX steps are redefined to move files into the "new release" location.

4. The RELEASE step now operates on the version locations much as CHECKOUT does. When specific files are released, the automated system searches for the most recent version, and finds it even if it has not been changed for several versions. If a "general release is made, only the files that have changed are distributed. This reduces tape and disc storage requirements as well as network distribution costs by eliminating unnecessary file transfers.

The resulting environment is illustrated below:



It is important to note that this rather complicated development environment can be precisely controlled with minimal effort through the careful choice of predefined file movement steps. Searching through two or more locations for the most current version can be performed automatically by the automated change control system. This enables the shop to maintain the integrity of prior versions while building a new version . . . without replicating files that have not changed. It also assures that programmers always get the most current version of any file.

SUMMARY

HP 3000 data centers have built a substantial asset in their software and data files. Yet, without effective change control, it is virtually impossible to guarantee the integrity of modifications to this valuable asset.

Furthermore, industry surveys show that most firms have a substantial software maintenance backlog. Without efficient means to manage changes, this backlog severely hampers the progress of new development efforts.

As a result, it is becoming quite common for HP 3000 shops to establish formal procedures for software change control.

Although many shops initially attempt to implement change control through a manual system, these systems have met with limited success due to the heavy burden they place on the MIS staff and the inherent unreliability of the manual approach. Because automated change control systems provide an effective, reliable solution while reducing the burden on the MIS staff, they have recently become the standard. These automated systems improve development and maintenance efficiency and insure that all program changes are properly authorized, documented and tracked. Automated systems provide management with quick access to the status of changes, the change history of a file and the impact of a proposed change. They provide programmers with an environment that allows them to concentrate on programming rather than on searching for the correct versions of files and documenting their usage.

Safeguarding the integrity of the software asset while paving the way for more efficient productive development and maintenance, is one of the most compelling challenges facing HP 3000 professionals today. Automated change control systems can provide the tools to meet this challenge.

CHANGE CONTROL CHECKLIST

Change control procedures for computer programs should be established and followed. The intent of these controls is to prevent unauthorized, inaccurate, and unreliable program changes from being incorporated into the live production environment. Both scheduled and emergency changes must be appropriately controlled to maintain the ongoing integrity of software.

You can use the following techniques to ensure that proper controls are being maintained over your program changes:

* Develop and adhere to formally approved written standards for all program changes.

* Define and enforce procedures detailing who can initiate and who can authorize program change requests.

* Describe and track the nature and reasons for proposed changes.

* Enforce testing and acceptance procedures for all program changes including emergency changes.

* Test all program changes under normal operating conditions.

* Involve users in preparing test data and reviewing test results.

* Investigate and correct all errors before transferring code to production.

* Certify that all test results demonstrate adequate protection from fraud, waste, and misuse of the program.

* Document all program changes and update appropriate documentation as changes are made.

* Log all completed changes as well as those changes in progress.

* Utilize formal systems to report all changes to users and managers.

* Enforce a checkout-checkin procedure that prevents a file from being simultaneously modified by more than one programmer.

* Develop procedures to analyze whether other systems are affected by new program modification.

* Retain and secure original source code until changes have been processed, tested and updated.

* Limit the frequency of program changes, except for emergency cases.

* Notify both the user and EDP project manager of emergency changes.

DESIGNING USER INQUIRY SCREENS FOR KEYED ACCESS

PAUL EDWARDS
BRADMARK COMPUTER SYSTEMS, INC.
1506 ESTATES WAY
CARROLLTON TX 75006
(214) 242-6660

DESIGNING USER INQUIRY SCREENS FOR KEYED ACCESS

ABSTRACT

The design of user interface inquiry screens is complex. There are many factors to be considered during the design phase. With the use of IMAGE or KSAM, the design is focused around a single key value to be used for the inquiry.

With the increased use of keyed access systems, the time devoted to design of the user interface is increased. These systems provide partial key, generic key, keyword, and relational access. This causes an additional work load on the systems analyst to define the real access needs of a user that does not understand these access systems or his own requirements.

An in-depth discussion of these access methods will be provided and practical examples of inquiry screen styles will be presented.

DESIGNING USER INQUIRY SCREENS FOR KEYWORD ACCESS

The design of user inquiry screens has been done much the same way since the HP/3000 environment of IMAGE and VPLUS has existed. The traditional access methods will be reviewed, access key types will be defined, the different types of access available will be discussed, application design requirements will be outlined, and the screen design options will be described.

TRADITIONAL ACCESS. IMAGE and KSAM are two MPE subsystems used by most applications software developers to provide random and serial access to data. There are several disadvantages in both systems.

IMAGE allows only one key in each master dataset and that key has to be unique. There is no generic or indexed sequential key retrieval. To provide additional keys in a record, a system of automatic/manual masters and details must be implemented. This adds additional structures to a database that can degrade performance and complicate the program access of the database. Sorted chains are used to retrieve data records sorted by other fields. The updating of these chains can cause serious degradation in performance of the application program. IMAGE doesn't provide any relational access capability.

KSAM provides generic key and indexed sequential retrieval of data records. There are performance considerations in the use of KSAM with a large number of users. It doesn't provide the logging and security features of IMAGE. Key file corruption is possible with power/system failures. Key files must be checked for integrity after these events occur. Many people combine IMAGE and KSAM file structures to overcome the limitations of IMAGE. This causes a more complex program environment with the possibility of loss of data integrity and causes more disk I/O.

Besides the performance considerations, the serial access of IMAGE and KSAM files have several disadvantages. The serial access of a master dataset can consume considerable time because all records are read whether they contain data or not. A detail dataset has a delete chain that is used to add records in the empty spaces left by deleted records. If the delete chain is empty, the records are added to the end of the dataset. The result is that a serial read of the dataset will retrieve records in no particular order. Even if a chained read is used, the records are returned in order by their placement on the chain. A sorted chain could be used to ensure a certain sequence, but with the associated performance overhead. KSAM records are stored in chronological order. Access of these records can be by any of the keys which is sorted in order by the key or by chronological order.

KEY DEFINITIONS. The types of keys available are simple, concatenated, grouped, and keyword. Custom and independent indexing can be done, as required. These types of keys provide different ways to access the data records. A B-Tree structure is used to store the keys and pointers.

The simple key is usually a single value in a field. This is the type of field normally used in KSAM or IMAGE. An example of this type of key would a customer number.

Several simple keys can be combined together to form a concatenated key. The order in which they are combined together are significant when retrieval is done by a keyed access system. Alphanumeric sorting order is left to right. An example of this type of key is an invoice number combined with the invoice line item number.

Multiple fields that are all to be searched at retrieval time are called grouped fields. They can be simple or keyword fields. Two address lines are an example of grouped fields.

A keyword key is created for each individual word in a field that is separated by spaces, slashes, or dashes. To save disk space, a keyword exclude file is available to enter common words that don't need to be keyworded.

Custom keys are used when additional intelligence is required to create the key. A special procedure is used to prepare the key. Data type conversion, reformatting dates, upshifting, embedded keys, and stripping unneeded characters are examples of key preparation.

Independent indexing is used for other than IMAGE databases. This includes MPE flat or word-processing files. The key datasets only contain the key values and any additional values required to be passed to the calling program.

TYPE OF ACCESS. The types of access to records in a keyed system are serial, partial key, generic key, relational, and boolean.

Serial access is in sorted sequential order. Access of the indexes by traversing the B-Tree will return the entries in ascending or descending order.

Partial key retrieval used when only part of a value is significant. A wildcard character is used to fill the places that are to be ignored in the search. An example would be a search value HEW??TT that would retrieve any entry starting with HEW followed by any two characters followed by TT.

Generic search is used to find entries beginning with certain characters. A wildcard character is used to signify that all following characters are not significant. BRAD@

will find all entries starting with BRAD with zero or more characters following.

Relational access is very powerful. It can provide access with multiple values across multiple fields, datasets, and databases using dynamically-joined indices. A common field in two datasets provide the link between records in each dataset. If a common field is not available, then a projection is made using a common field in a third dataset.

Boolean expressions can provide logical choices to be made qualifying entries. The AND, OR, and AND NOT operators are be used for making multiple criteria retrievals. A method of reduction retrieval can be used against a set of retrieved entries to reduce the number of entries displayed. This is achieved by making multiple DBFINDs against the same set of data records and changing the criteria each time.

APPLICATION DESIGN. Application design tasks are more complex with keyed systems due to the virtually unlimited ways to access the data. Two areas that need to have special consideration are intimate knowledge of the data and in-depth analysis of the retrieval requirements.

Knowledge of the data includes the normal field sizes and field data type descriptions with several other requirements concerning the contents of each field. The type of key, as defined above, must be determined for each keyed field. Any concatenations or grouping of fields must be specified. The length of the key is important so as not to use too much disk space and to provide enough length to make the key as unique as required for inquiry. How the data is represented in the field will determine patterns for partial lookups. For the keyword fields, a minimum number of characters per word will determine which words are keyed. The size of each word and the average number of words in a keyworded field will affect the B-Tree structure.

The retrieval requirements are more complex than with IMAGE or KSAM. The type of access required for each field or key must be selected as defined above. Since boolean relationships are available between different keys, these relationships must be defined. A thorough knowledge of the user's manual systems must be obtained as with traditional systems. In addition, the user's knowledge of his data retrieval requirements and education level is very important. Since any field in an IMAGE database is now accessible, the user could very quickly be overcome by the complexity of the resulting screen design. The KISS (Keep It Simple, Stupid) principle should be followed.

SCREEN DESIGN. A sample database with sample screens for a simple customer invoice system is illustrated in the copy of the slides to follow. A traditional approach will be shown with the addition of a keyed system design added.

DESIGNING
USER INQUIRY SCREENS
FOR KEYED ACCESS

TRADITIONAL ACCESS

- IMAGE
  SORTED CHAINS
  AUTOMATIC MASTERS
- KSAM
- SERIAL ACCESS DISADVANTAGES

KEY DEFINITIONS

- SIMPLE
- CONCATENATED
- GROUPED
- KEYWORD
- CUSTOM
- INDEPENDENT

TYPE OF ACCESS

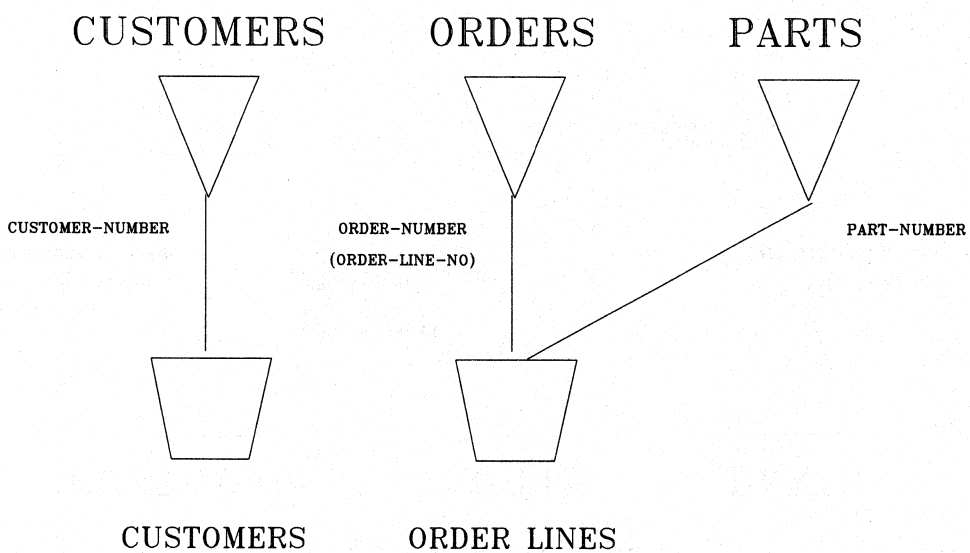
- SERIAL
- PARTIAL
- GENERIC
- RELATIONAL
- BOOLEAN

APPLICATION DESIGN

- KNOW YOUR DATA
  TYPE OF KEYS
  KEY LENGTHS
  FIELD CONTENTS
- ANALYZE RETRIEVAL REQUIREMENTS
  TYPE OF ACCESS
  BOOLEAN RELATIONSHIPS
  KNOW YOUR USER

CUSTOMER ORDER DATABASE

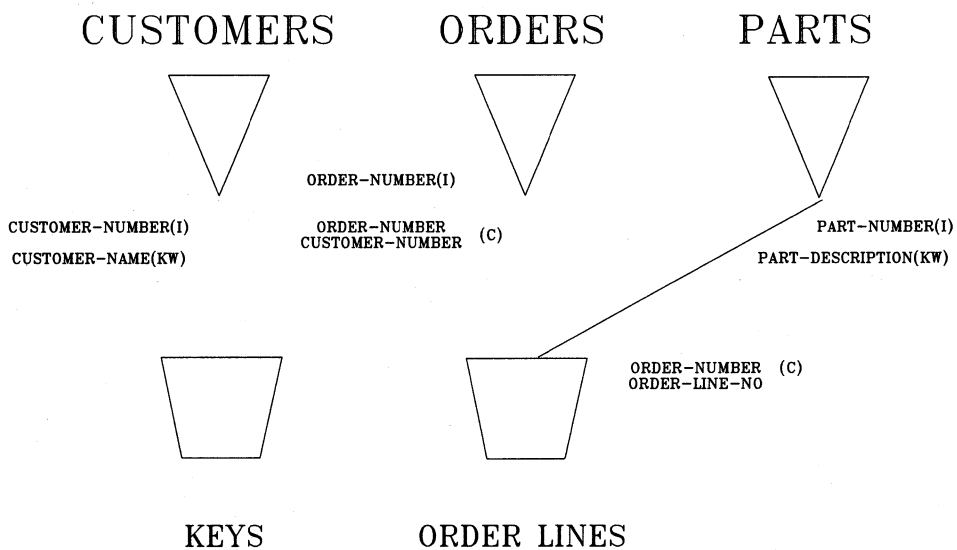
TRADITIONAL ACCESS



5133-11

CUSTOMER ORDER DATABASE

KEYED ACCESS



5133-12

CUSTOMER ORDER SCREEN

TRADITIONAL ACCESS

CUST.NO [_____] NAME [_____]
ADDR [_____]
ADDR [_____]
CITY [_____] ST [__] ZIP [_____]

ORDER NUMBER	ORDER AMOUNT	SHIP DATE
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]

5133-13

CUSTOMER ORDER SCREEN

KEYED ACCESS

CUSTOMER: NAME [_____]
ADDR [_____]
ADDR [_____]
CITY [_____] ST [__] ZIP [_____]

ORDER NUMBER	ORDER AMOUNT	SHIP DATE
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]
[_____]	[_____]	[_____]

5133-14

DESIGNING BUSINESS CONTROLS INTO INFORMATION SYSTEMS

J E F F G I B B S

HEWLETT-PACKARD

3000 HANOVER STREET

PALO ALTO, CA

GENERAL APPROACH

Any viable approach to designing controls into information systems needs to be based on considerations of cost-benefit of the various individual controls. Most controls have a cost, either a direct monetary cost or else a cost in terms of reduced functionality. Thus, it is important to identify the purpose and objective of each control proposed. It is not the case that "any control is a good control!" But rather each control should have a justification in terms of the overall control environment.

One approach to assessing the importance of individual controls is to examine the exposures present in a system under development and then to ensure that adequate controls are included to cover each exposure. Exposures in this context are things which can go wrong with a system. These exposures are different for each system under development and they need to be identified early in the project, preferably during the feasibility stage. Once these exposures are identified, they can serve to guide the remaining controls analysis. Although there are many different types of exposures, some of the main catagories are as follows:

Inadequate Data Integrity: The system will not process data in a manner consistent with the intent of the users.

Inadequate Data Security: The system can be accessed by unauthorized individuals and/or confidential data may be compromised or stolen.

Inadequate Availability: The system will not provide adequate service levels to its users, due to inadequate performance and/or unreliability.

Inability to Recover: The system cannot be recovered satisfactorily in the event of a failure.

Non-Compliance with Company Policy: The system performs in a manner inconsistent with company directives.

Each of these main types of exposures may have a number of contributing factors. Once again, these detailed exposures should be identified early in the development project.

One approach to ensuring the identified exposures are suitably controlled by the software which is developed is to utilize some sort of exposure matrix. One useful approach is to list the exposures which have been identified down the left hand column of a spreadsheet (see Exhibit I). Once this is accomplished, the next step is to list across the top all the possible controls which could be brought to bear to address the exposure. This list of possible controls can be developed in conjunction with the development team and should ideally be reflected in the design documentation for the project.

EXHIBIT I

Application_____

Exposure Matrix

Exposures	Mandatory Controls	EVAL	Discretionary Controls		EVAL
	Automated		Automated	Manual	
Integrity Exposures					
Security Exposures					
Availability Exposures					
Recovery Exposures					

The next step is to perform a cell-by-cell analysis of the matrix to determine how well each identified exposure is likely to be covered by the proposed controls. In this regard, the matrix should serve as a worksheet during the entire life of the development project. During the investigation stage, each cell where a control should impact an exposure might merely be identified with an empty circle. Once the system specifications and/or other design documents are released, the cell can be tentatively filled in with an indication regarding how the particular control should address the exposure. Finally, in the prerelease phase, each cell can be updated and "written in ink" to represent how the system actually developed uses the specific control to address the identified exposure.

The constant objective throughout this process is to continue to assess the degree to which each identified exposure is likely to be covered by the various controls in the system under development. (An example of this type of analysis is included as Exhibit II.) Finally, the outcome of this analysis could serve as the basis for periodic control reviews of the project or as the basis for discussions with the project team regarding areas where additional controls are needed. This type of analysis can also be useful in identifying redundant controls which might be eliminated without adversely effecting system integrity.

EXHIBIT II
XYZ CORPORATION
APPLICATION SYSTEM CONTROL MATRIX

EXPOSURE	CONTROL AREAS: DATABASE DESIGN	AUTOMATED INPUT VALIDATION	TRANSACTION LOGGING	TRANSMISSION ACKNLG. MESSAGES	OUTPUT REPORTS	PROBLEM AREAS
GENERAL						
1. Theft or loss of significant dollar amounts may occur without being detected.			Daily printing of this log should minimize the potential of this occurrence.	Acknowledgement messages may reduce the possibility of this occurrence.		
2. System may not be in conformance with corporate policy.		Validations are based on corporate policy.			Output reports should contain management information in support of corporate policy.	
3. System may have errors in program logic and/or calculations.					Output reports should be useful in detecting program errors.	X
4. System may be inauditable.			Transaction log should limit this exposure.	Acknowledgement messages will reduce the extent of this exposure.	Output reports should support auditability.	
INPUT						
5. Unreasonable or inconsistent source data values may not be detected.	System does not provide a data dictionary which would reduce data inconsistency.	No range checks or reasonableness checks will be performed.			System does not provide exception reports.	X
6. Incomplete, erroneous, or poorly formatted transaction data records may be treated as complete and accurate.		Extensive validations are provided which should minimize the possibility of this occurrence.				
7. Reentry of erroneous transactions may bypass normal validations.						X
8. Transactions may be lost before they are used to update master files.			Transaction counts should help minimize the potential of this occurrence.	Acknowledgement messages should minimize the possibility of this occurrence.	Daily transaction log should provide transaction counts.	
9. Intersystem transactions may not be correctly posted.	Database design does not have necessary chaining to assure appropriate posting to various application systems.	No intersystem validations are performed.	Log should minimize the potential of occurrence for CFIS transactions only.		No intersystem transaction log reports are provided.	X
10. The system may be accessed in an inappropriate and/or unauthorized manner.	Database design does not include appropriate controls at the record/field level.		Transaction log should report these occurrences after the fact.	Acknowledgement messages may reduce the possibility of this occurrence.	System does not provide a log of unauthorized access attempts.	X

APPLICABILITY TO PROTOTYPE DEVELOPMENT

While this general approach has been demonstrated to be quite useful in development projects utilizing traditional, linear progression methodologies, its use in the review of projects utilizing prototype approaches requires some modification. Prototype methodologies typically involve the development of a working model to assist in the definition of user requirements. However, increasingly, attempts are being made to extend the use of prototypes to cover all aspects of system design, including the definition of detailed, or internal, design specifications. The difficulties which this presents for the review of design specifications are as follows:

1. Prototypes are inherently unsuited for this extended application in that they do not lend themselves well to the specification of the sorts of detailed design specifications which are necessary to successfully complete the construction of the software. Rather, prototypes are more useful in defining such aspects as basic functionality and user interfaces.

2. Because the use of prototypes essentially involves an iterative process, it is quite common to find that the design for the project is not "frozen" until very late in the project, sometimes only a month or so before the system is ready for field testing. This does not give the design team adequate time to respond to any control issues identified during the review process.

There is also a related tendency to utilize a 4th generation language-based prototype as the basis for the final system without adequate technical evalua-

tion to determine if system performance will be adequate when written in this often less efficient code. Thus, prototype development projects may present special system availability exposures.

To encourage the development team to provide for adequate design specifications, it is suggested the controls matrix discussed above be utilized to identify system design features which require further detailing. The design features which may require further detailing are those which will be critical to ensuring each identified exposure is appropriately controlled. (The testing of these critical controls should also be a primary objective of the the test plan developed for the system.) To identify these critical controls, the developer should read across each exposure line to assess the degree to which specific controls will be relied upon to adequately address the exposure.

The documentation of these features need not be an onerous task, but rather may merely involve the development of several design memoranda covering these issues. This sort of documentation will also be indispensable in being able to properly assess the degree to which the system's controls are likely to address the identified exposures.

CONCLUSION

The use of controls matrices can provide a quick and easy way to ensure that the controls designed into information systems are adequate to cover the identified exposures without presenting excessive costs in terms of functionality or service levels. They can be particularly useful in a

prototyping environment to enable the developers to identify key design aspects which need to be further documented in a traditional manner.

Application Testing -- Doing It Right!

Lisa Burns
Hewlett-Packard
Corporate Offices
P. O. Box 10301
Palo Alto, CA 94303

How many of us have had the following experience: you are working on a new application, due to be delivered in a week! Because of the tight schedule, all you are able to do is get a clean compile on some of the modules, and run a few test cases through the others. As a result, you go into the live installation knowing that much of your code has never been run, and certainly that you have never run the complete system together. Sure enough, the conversion process to install your new system falls over several times, and the project team is up all night trying to fix it. Finally, the database is loaded, but the online system has several major problems, causing program aborts and unpredictable data record contents...

You know the rest. Your test site personnel are angry, the users are disappointed in your system and don't trust it, and the project team is demoralized. What could you have done differently to prevent this kind of occurrence? Software inspections or walk-thrus would have helped you catch some of these problems and better management support of adequate schedules would have given you more time. But perhaps the most important thing you could have done was to TEST that conversion job, and TEST that online module. In this article, I'll discuss the levels of testing performed in our shop, and give you some advice about setting them up in your environment. Finally, I'll talk about how project teams in our area have performed testing and the results we have achieved.

We do three, and sometimes four different levels of software testing in our MIS department. We call the first level of testing unit testing. This is the lowest level of testing and tests a single, newly-developed module or single change to an existing program. The purpose of unit

testing is to verify that the module being tested performs according to its requirements. In our shop, unit test plans follow a standard testing process, and are documented so that they will be easily repeatable if additional changes are made, if defects are discovered, or if the tests are repeated as part of the next level of testing, system testing.

System testing is begun when all coding and unit testing is completed for a new project or update to an existing project. This level of testing integrates all modules to be released to users. System testing also includes testing of all system interfaces. All on-line and batch functions are tested in an environment as close as possible to that of production.

For some of our projects, an intermediate level of testing between unit and system test may be performed. This level of testing, called integration testing, incorporates the interface between two or more new modules in a system.

The final test of a system, of course, is when your users get their hands on it. We call this test an Alpha test. Sometimes, especially for new systems, the first phase of an Alpha test consists of a parallel run, where new software is installed on a separate machine or account and run at the same time and with the same data as the live system. The results of the two systems are then compared.

Other systems may go to their Alpha test directly into production. The users then simply do their job functions using the new software, carefully monitoring the results.

"So," you say, "how do I go about doing these various levels of testing?" The first thing to do is sit back and design the process you will use for the tests. Let's think about unit testing first.

As you prepare to re-vamp the process of testing individual modules in your shop, you should consider the following questions:

1) Who should perform the testing?

2) What tools will be used?

3) Which conditions and cases will be covered?

First, you will need to decide who should perform the testing. Some teams have the programmer who wrote the module test it. This is certainly the easiest approach, but may lead to missed test cases. You might consider having another programmer familiar with the system, but who did not write the code perform the test. Alternately, user representatives or users themselves may execute tests. This is especially good for ensuring that your software is easy to use. Users may not be the best choice for running a sophisticated conversion job, however.

Next, you should decide what tools you will use in your testing. A good set of tools might consist of a test plan outline document, problem tracking log, file compare utilities and test file naming and tracking conventions. A written test plan ensures that no cases are forgotten, and since it is documented, it can be repeated when a module is changed. The problem tracking log ensures that all bugs are noted and fixed before the system is released. The file compare utility allows easy checking of database and flat file contents against known good results. Finally, by organizing your test data files, source code and object code files by group and filename, you will easily be able to ensure that you are testing and ultimately installing good code.

The question of which conditions and cases should be covered is perhaps the most important question for unit testing. You may decide, for critical areas of your system, that all of your code should be tested, and that you will track path execution coverage using a Path Flow Analysis (PFA) tool. For other areas, you may decide that production data cases plus a erroneous conditions constructed by your programmers will be adequate to catch a majority of defects. These decisions should be made based on your knowledge of risky areas of your code, those likely to have problems or those peforming important functions (pricing, paycheck calculation, production date assignment, etc.) Obviously, the more tests you run, the more resources you will need to invest. This is a decision you will have to reach with your project management. Remember, however, that it is much better for YOU to catch and fix a problem that for your USER to find it, so the more testing the better.

When choosing the particular test cases themselves, pay particular attention to boundary cases, those close to and past the edge of valid limits for data. For example, if your progam accepts from 1 to 10 values, you will want to test no values, 1 value, 10 values and 11 values as boundary cases. In addition to boundary cases, choose valid as well as invalid data elements. For example, you might want to test "F" as a valid value and "X" as an invalid value for a Male/Female edit on a gender field.

To test all possible conditions within your program is usually impossible or impractical. If you are concerned about methods for choosing more test cases than the examples above, you may wish to consult testing methodology books. However, one method which is usually recommended is Error Guessing. This involves having experienced folks on your project team try to predict typical program failure modes and adding these as test cases. In other words, have your gurus try to break the code! These test cases, along with the ones cited above, will complete your test plan.

The answers to the questions above will allow you design an effective unit test process. Now it's time to think about system testing. In addition to the three questions we answered for unit testing, we need to pay particular attention the the system test setup. Your system test process should include an installation step, where all necessary files are placed into the test environment. Ideally, this step should

include the same procedures as you will use when you move your new software onto your users' machine during the Alpha test. Remember that you should include all interfaces and external databases that you will need for the test. Your goal should be to get as close as possible to your users' environment. The more data files and interface modules you can steal production copies of, the better off you will be when it comes time to Alpha test.

Now you know how to go about testing in your particular environment, and it's time to write a unit test plan. Your unit test plan should include:

1) A test objective

2) Step-by-step procedures for running the test

3) Expected results for each step

4) Area to record actual results for each step

A sample test plan with each of these is show in Figure 1.

The person executing this test should record the results of each step in the "ACTUAL RESULTS" section. If the actual results differ from those expected, a problem log should be completed. A sample is included in Figure 2. Problems identified on this log sheet should be corrected and the test plan should then be re-executed. This process is repeated until all expected results are obtained.

The process of writing and executing unit tests should be completed for each module in a new system, or for each change in an update to an existing system. Once unit testing is completed, it is time to conduct a system test. The system test plan should include the following:

1) A description of all modules in the system in the case of a new system, or a description of each change in the case of an update

2) Steps to follow in assembling the test environment

3) Objective for each step

4) Steps to follow in the complete test

5) Expected results for each step

6) Area to record actual results of each step

Excerpts from a system test plan are shown in Figure 3. During the system test, you may repeat some or all of the unit test plans developed earlier. At this point, especially if they were run by the original programmer at unit test time, you may wish to have a second test person execute the unit tests. In addition to the unit tests, be sure to push data through your entire data flow, from data entry to

reporting and through each of your interfaces. Test all major functions within your system. For system updates, you will want to ensure that your new enhancements and defect corrections did not create new bugs!

For some testing, you may need to include personnel from other project teams. This will be especially important for interface testing. Be sure these programmers know that they will be involved and how much time they will need to allot for this effort. Explain what data you will provide to them, and what kind of data or reports you need back from them. Will they run this data through their own test system? Will they desk-check the file? How soon will you need the results? Be sure they know what is expected and when.

As you execute the system test, follow the error logging described above to track defects and their correction if actual results do not match those expected. File naming and release procedures are especially critical during system test. You will want to be sure that the corrected modules moved into the system test environment are also the ones you take with you to the user machine!

Once your system test is completed or as it is in progress, you may want to sit down with your users and discuss how to conduct the Alpha test. For a new system, users may want to conduct a parallel. For a system with a lot of data entry, this can be a big time commitment. Decide how data will be entered into both the live and the test systems, and how the data will be compared. If you are going live immediately, discuss how to test each module or change, and how results will be recorded. Also, establish clear communication paths with your users in case a problem is discovered. Who on the development team should be contacted? How quickly can a fix be expected? How long will the test last? How will user signoff be obtained? These questions should be answered up front, so that no surprises are encountered come Alpha.

In our shop, we have found the steps listed above to work very well in producing excellent Alpha test results and very happy users.

For an update to a new system, testing takes between 20% and 30% of our construction time, and is done as coding for a particular change or enhancement is completed. This means that unit tests are being written, inspected, and executed throughout our typical six month development cycle for a system update. The last two months of development for an update is spent in testing. First, unit tests are completed, then the last 4-6 weeks are spent in system testing. No code changes, except those required to correct defects found in system testing, are made in those last 4-6 weeks. Finally, we take our code to the user site, and it is installed and run there for about one month before being signed off.

For a new system, unit tests are done as modules are completed, with some integration testing done as modules are linked together. Because of this extra level of testing, percent of time spent in testing may be

slightly higher for a brand new system, between 25% and 35% of the total development time. Then, just as for an update, a system test is run for 4-8 weeks. Finally, user testing is completed at an Alpha site, with a new system typically running parallel to production for a few weeks before going live at the Alpha site.

After the Alpha test is completed, our project team meets to discuss what could have been done better in our development and testing process. The recommendations from this meeting will be used as we begin our next project. Through these debriefing sessions, we have found the following points to be especially important when planning and executing tests:

1) Ensure that your test environment matches the one you will find at your user site.

Problems are frequently caused by testing with incorrect versions of interface files, interfacing software or utility programs during unit and system testing. These problems can be very frustrating to debug. Be sure that your test files are correct. If you can, make copies of the files you need from your users' machine. Allow enough time in your testing schedule to set up an accurate environment on your test machine or in your test account.

2) Ensure that unexpected conditions are covered.

What will your program do if it encounters no input at all? if it encounters too many records? if it cannot obtain access to an interface file? These are cases to watch out for and to be sure to include in your unit and system test plans.

3) Be sure that test cases match reality.

In addition to watching for boundary conditions, be sure that your test cases are an accurate reflection of how your program will be used in production. Test using data taken from your user site if possible. For an online system, you may wish to have users try out your programs using data they bring from their daily work.

4) Allow enough time.

Be sure the programmers on your project have enough time built into the schedule to perform adequate unit and system testing. Also, if you require assistance from programmers from other projects or departments, be sure that they understand your testing schedule and can support it.

The time we invest in testing, although significant, pays off. We have found that by carefully writing and executing test plans, we have greatly reduced the number of problems encountered at the user site. For our project, we typically find under five defects during user

testing. None of these are typically serious defects, and can be corrected quickly. After the Alpha test period, we rarely encounter defects in production.

I hope that the information in this article allows you to implement improved testing procedures in your own shop. I have worked on projects where little or no testing was done, and the results usually matched the situation I described at the start of this article! I have also been fortunate enough to work on projects where the programmers took a disciplined, structured approach to unit and system testing, similar to the one described above. I can tell you that the testing effort pays off ten-fold in bug-free code, smooth installations, and very happy users. I plan to keep testing!

Figure 1

TEST PLAN FOR ORDER NUMBER TABLE CONVERSION JOB
Module tested: Conversion job
Release affected: G.00.00
November 27, 1987

A. OBJECTIVE

The purpose of this test is to ensure that order numbers in the OMSTBL database are properly converted to the new order number format. The three fields affected are LAST-USED-NUM, BEGINNING-NUM, and END-ORD-NUM in the ORDER-NUM-TABLE dataset. The current order numbers are stored with a leading zero. The zero should be at the end of the order number.

B. EXECUTION STEPS

1. Record what is currently in the ORDER-NUM-TABLE.

PROCEDURE:

Use QUERY to go into the OMSTBL database. ORDER-NUM-TABLE = dataset. Do a FIND ALL ENTITY-CD-SF. Report all entries found to a line printer.

EXPECTED RESULTS:

Listing is produced of current entries in ORDER-NUM-TABLE.

ACTUAL RESULTS:

2. Convert ORDER-NUM-TABLE entries.

PROCEDURE:

Stream CABCONVJ and check the $STDLIST to ensure that all steps completed successfully.

EXPECTED RESULTS:

Job completes successfully.

ACTUAL RESULTS:

3. Verify that the new ORDER-NUM-TABLE entries are correct.

PROCEDURE:

Use QUERY to go into the OMSTBL database, ORDER-NUM-TABLE dataset. Do a FIND ALL ENTITY-CD-SF and report all entries to a line printer.

EXPECTED RESULTS:

Listing of current entries is produced.

ACTUAL RESULTS:

PROCEDURE:

Compare this listing to the one created in step 1.

EXPECTED RESULTS:

All Fields should be exactly the same except LAST-USED-NUM, BEGINNING-NUM, and END-ORD-NUM. In the first listing, these three fields should have a zero in the first position, and then five numbers after it. In the second listing, those same five numbers should be the first five characters, followed by a zero in the sixth position.

ACTUAL RESULTS:

Figure 2

PROBLEM LOG FOR ORDER SYSTEM RELEASE G.00.00

Date: MARCH 15, 1987 Severity (H,M,L): M

Person: LISA BURNS Priority (H,M,L): H

Screen or job affected: FIND CUSTOMER SCREEN

What happened:

CUSTOMER LOOKUP SEARCH ROUTINES DO NOT WORK. A KSAM ERROR "BLANK ARGUMENT" IS PRODUCED. PROBLEM IS RELATED TO USING CUSTOMER INTERFACE VERSION B.03. WORKS CORRECTLY WITH VERSIONS B.04, B.05. WITH B.03, NO KEY IS BEING MOVED TO THE KSAM BUFFER.

What should have happened:

SEARCH SHOULD WORK FOR ALL VERSIONS OF CUSTOMER INTERFACE.

Where should this error have been detected:

ERROR SHOULD HAVE BEEN CAUGHT DURING INSPECTION OF UNIT TEST PLAN.

PROBLEM RESOLUTION

Date: March 17, 1987

Person: Lisa Burns

Description of Resolution:

PATCH INSTALLED ON 3/17. MODIFICATIONS TO OMS2470. FUTURE TESTS SHOULD INCLUDE ALL VERSIONS OF CUSTOMER INTERFACE FILE.

Figure 3

SYSTEM TEST PLAN -- ORDER SYSTEM UPDATE G.00.00

A. Changes to be tested:

SR#	Description
12345	Renumber order numbers in ORDER-NUM-TABLE.
50789	Add new edit to check order type.
67542	Show product shipments in-transit on status screen.

B. Assembling test environment:

1. Set up MKTING account on test machine.

2. Set up ORDERD, ORDERP, ORDERJ groups in MKTING account. Follow security specified in standards manual.

3. Ensure availability of the following interface software:

CUSTOMERS	Version B.03. Load offices 2417, 2403.
PRODUCT	Version A.04. Load Product lines 99, D3, 58.
CENTRAL	Load test environment matching 3/15 production version.
MPE	Version G.A2.B0 (U - B MIT)

4. Load previous version of order system (F.03.00).

5. Following installation procedures, load new version of order system (G.00.00).

C. Testing steps.

1. Test new order edit.

PROCEDURE:

Follow unit test plan 50789.

EXPECTED RESULTS:

See unit test plan.

ACTUAL RESULTS:

2. Ensure current reports still run.

PROCEDURE:

Run JOB5, JOB6, JOB7.

EXPECTED RESULTS:

Reports are produced.

ACTUAL RESULTS:

PROCEDURE:

Use QUERY to dump STATUS dataset contents. FIND ALL STATUS NE SPACES; REPORT ALL. Compare to reports produced by JOB5, JOB6, JOB7.

EXPECTED RESULTS:

Reports match database contents.

ACTUAL RESULTS:

A version of this paper first appeared in HP Professional Magazine.

Production Reporting

Using

Business Report Writer

by Jeff Hecker

Apogee

4632 West Frankfort Drive
Rockville, MD 20853
(301) 460-0021

The Production Environment

The production environment is often two very different environments at different times during the day. During "business" hours, the production environment is usually on-line environment, with a large number of interactive users, making many varied requests of the system. During off hours, or at night, the production environment becomes a batch environment, with strings of database updates, consolidations, and reports.

During the day, system resources are required to provide the quick reponse time needed for interactive applications. A typical example is a telephone operator who is taking customer orders, or answering some inquiry. In any case, fast system response is necessary to make efficient use of the operator's time.

Programs used in this environment are designed to do very small amounts of work at a time. Their goal is to consume as few system resources as possible, to get in and get out very quickly. Such programs are generaly not well suited for the batch environment.

Overnight, many computer systems become very different animals altogether. Once all of the on-line terminal operators go home, the system shifts gears into batch mode. Overnight batch jobs are characterized by the volume of data which they process. For example, the on-line account balance program accesses a single account and displays the balance. The overnight account balance report accesses all of the accounts, sorts them, calculates aging periods, groups them by regions, sorts them again, subtracts quotas, and prints it all out in a couple of hundred pages. The goals of the overnight batch programs are to be finished before the morning shift comes in. A "batch" job may consume as many resources as is necessary to finish in the minimum amount of wall time. After all, if an overnight batch job uses 3, 600 CPU seconds per hour, who cares?

HP Business Report Writer

HP Business Report Writer (BRW) is Hewlett-Packard's latest MPE report writer product. BRW includes options and features which make it a powerful and flexible program for business applications.

Like some other report writers, BRW can access multiple data base files at once. Each may be opened in any IMAGE access mode, including if necessary, exclusive. BRW may also access MPE "flat" files and KSAM indexed files.

Surprisingly, HP BRW cannot access HP SQL data base files. This is surprising because of BRW's relational model operation. In the documentation, the help screens, and in the manual, the standard relational model is adhered to. Intermediate files are reffered to as "tables". Relating records from one file (or data set) to another is reffered to as "joining", not "linking".

Unlike some report writers, BRW makes no distinction between key (or indexed) fields, and ordinary fields. Again, this follows the relational model. Any field (or combination of fields) may be joined to any other fields of any other table.

Input data which is joined is stored in a temporary disk file. These files are saveable as MPE disk files. This is similar to saving Cognos "subfiles" in order to use them over and over. For example, a complicated report may access several data files. A summary version of this report may also be desired. Rather than accessing each of the original data files twice, the intermediate file used in the first report can be saved, and used by the summary report as well.

BRW reports select data from input files based on either "hard coded" selection criteria, or user entered parameters. Parameters may be entered at report invokation time. A V/3000 form is used by BRW as a fill-in-the-blank interface. When creating a BRW report, parameters may be declared to

be optional or required. Optional parameters may be left blank, required parameters may not. Enough parameters are available for the most sophisticated reports.

On the output side, BRW uses the sort-break model to specify output lines. BRW can, if necessary, output headers and/or footers for the report, for each page, and for every sort group. Of course, output for every record is available.

BRW includes facilities for conditionally supressing output lines. For example, a report could be written which would supress output for customers who have not ordered anything in the past year. This is sometimes possible with other report writers by using apropriate selections from the input files. However, unless there is some flag in the input file which the report writer can select on, then input filtering will not always work. Consider a report to print accounts with a higher balance this month than last month. All accounts must be processed, then the ones which do not meet the conditions can be supressed.

Individual lines may also be supressed. This allows two entirely different outputs to be used in the same report. Consider a report which lists various international monetary amounts printed. With BRW a separate line is defined for each different currency, and then all but the correct one is supressed. The same thing can be done for page headers and sort group headers and footers. This way, different currencies can have different currency symbols, different column headers, different column widths, etc.

Ordinary BRW Operation

All BRW reports operate in the same basic manner. The input files are scanned, selected and joined into a "final" intermediate table. From this table, output lines are generated. Whether multiple input files are used, or a single input file is used, BRW's processing is the same.

Unless otherwise specified, BRW accesses all input files sequentially. This "default" operation is foolproof. All files, regardless of their type may be sequentially accessed successfully. After all, BRW has no way to know which keys, or which indexes will yield a useful result.

Sequential access also allows BRW to "join" files which contain no indexing or keys of any kind, an MPE flat file for example.

Joining files which have no indexes is a time consuming task, but it can be done. Consider joining file "A" and file "B" on fields which have no keys, and which are not indexed. First, file A is read (sequentially) into a temporary file. The temporary file is sorted on the field to be joined. Then file B is similarly read and sorted.

The two sorted temporary files are read and compared by their common field. Those which match are "joined" and written to the intermediate table. Those which do not match are skipped. This is BRW's SORT/MERGE join.

Of course, BRW can access indexed or keyed files as well. IMAGE data base files and KSAM files are accessed by key if the files are joined by a keyed field and the programmer directs BRW to use keyed or chained reads. Even though BRW knows which files and which fields are keyed, BRW will not use keyed reads unless specifically directed to.

Improving BRW Performance

There are several opportunities for improving the performance of BRW reports. Some are relatively obvious and trivial, but some are more subtle.

One of the simplest and most obvious improvements which can be made, is to access input files in the correct order. Consider a report which generates an account statement for a customer. The data for such a report is usually contained in an IMAGE database. A typical organization might be an account master file, joined to a transaction detail file. Remember that, by default, BRW will access both files sequentially and use a SORT/MERGE to join them. In this case, it is much more efficient to scan the account master file for the correct account, then read the transaction detail file with a chained read.

This is a good improvement for an on-line version of this report, where an operator might request it during the day. Since it does not read the entire detail file, it should run quickly and not interfere with the on-line environment. However, what about the batch version of this report? Many situations require a quarterly, or monthly batch run of statements for all customers. For this case, reading the master file and chained read of the detail is not the most efficient method.

Batch versions of reports typically access most (if not all) of the data base. In this case, chained read is much less efficient than a sequential access. When reading an entire IMAGE detail data file, the additional time taken during the additional disk I/O more than overshadows any saving in sort time. In this case, BRW's SORT/MERGE is the better processing choice.

There have been a great number of papers and seminars on IMAGE data base accessing. A common threshold for using chained read (as opposed to sequential access) is 10 or 20% of the data set. If expecting to access over this amount, then sequential access may be more efficient. This is a severe oversimplification of the subject. Other factors to consider are the total number of records in the file; whether or not the file has been re-organized (using DB GENERAL, ADAGER, or an UNLOAD/RELOAD); and if an IMAGE performance product is being used. These considerations merit far more detailed discussion than can be provided here, and will not be pursued further.

Sometimes, a batch report must be run during the day, because of backlog, or errors, or some unforseen emergency. This simple improvement (having two different versions of the same report, one for on-line, one for batch with two different access methods) can save many hours of system time.

As a rule, BRW attempts to filter out data records which do not match selection criteria as soon as possible. Consider the previous customer statement example. If records were not selected early, then all of the intermediate files would contain the entire data base. Such large files take longer to create, longer to process, longer to sort, and use more disk space to boot. The single biggest reason for limiting data selection as early as possible is sorting. Big temporary files are not really important. And scanning a record which doesn't meet selection criteria doesn't really take very much time. However, sorting is one of those operations which always seems to take longer than it should. One of the reasons (in BRW reports) is that data is being selected and stored in temporary files which needn't be there.

For these reasons, BRW attempts to select data as soon as possible. However, BRW cannot always tell when "as soon as possible" is. When a single input file is used, BRW has no trouble determining when a data selection should be used. In more sophisticated reports, using multiple input files, and multiple intermediate files, BRW cannot always tell when to apply selections to input files.

BRW uses a concept of "projected items" when joining two or more tables. Often when two tables are joined, they are joined on fields which have the same name, such as `ACCOUNT-NUMBER`. When two tables have the same name, BRW chooses only one of them to be written into the intermediate file. The field which is copied into the intermediate file has been "projected". This, in itself, is not a problem because both fields from both files contain the same data.

BRW's selection process only processes projected items. Again, this is reasonable; only projected items will be in the temporary files, and therefore are the only fields which require

processing. Consider the master detail joining typical in IMAGE data base files. Often, it is the detail file information which is actually desired. The programmer sets (or allows BRW to set) the detail file fields to be the projected items into the temporary file. But now consider BRW's processing of such a report.

First, BRW accesses a master file record and begins a chained read into the joined detail file. For each detail record, the selection process will examine the ACCOUNT-NUMBER, and determine if it meets the selection criteria. Suppose that it does not. BRW will reject the record, as it should, and go on to the next record in the chain.

Clearly if the first record in the chain does not meet selection criteria on ACCOUNT-NUMBER, then none of the remaining records will be selected either. However, BRW will read them all, scan them all, and reject them all. BRW will then go on to the next master recond and begin another chained read.

The solution is to force BRW to select (and reject) records from the master file, even though it's fields are not the projected items. There are two ways to accomplish this. The first is to make the fields of the master file the projected items. The advantage is that it is simple to change the projected items for a table. The disadvantage, is that it is not documentable within the report itself. When defining an intermediate table, BRW will fill in the projected items based on the order of the input files. BRW assumes that the first table accessed should have its fields projected. A separate screen is used to explicitly select which fields from which files are projected. Whenever a programmer alters the table significantly, BRW will create a new set of projected items. There is no way to leave a "comment" in the BRW source to tell future programmers which fields of which files should be projected.

A second way to force BRW to apply selection criteria to fields which are not projcted items, is to reference them explicitly in the selection criteria. For BRW to apply selection criteria as early as possible, the selection must be entered generically. For example, specifying a selection such as

```
ACCOUNT-NUMBER = 20853
```

allows BRW to apply this selection on any input file from which the field ACCOUNT-NUMBER is projected. In order to select from a file from which ACCOUNT-NUMBER is not projected, an explicit specification, such as

```
ACCOUNT-MASTER.ACCOUNT-NUMBER = 20853
```

could be used.

The advantage here is that BRW does not overwrite selection specifications when the report is modified. The selection criteria are explicit, and to a degree, self-documenting, since future programmers will see this selection. The disadvantage of this method is that BRW does not overwrite selection specifications when the report is modified. If changes in the report require a change in the selection criteria, BRW will not make them automatically. A programmer must review the various selections to be certain that they are still valid.

Another selection which can aide BRW efficiency is the elimination of useless data. For example, consider an accounting application which reports current balances. In many cases, account balances are zero. In some cases, most account balances are zero. Unless there is a need for some kind of average balance, the zero balance accounts are just taking up space. Unless such accounts are specificly excluded by selection, such as

```
ACCOUNT-BALANCE <> 0.0
```

then BRW will process it, sort it, sub-total it, store it into an intermediate file, and print it out. As

a rule, BRW can detect a zero value during selection much faster than it can process it (including disk I/O, sorting, etc.)

BRW allows defined values. A defined value is one which does not exist in one of BRW's input files. For example, the amount of a salesman's commission is not stored in the sales data base. It is calculated based on a percentage of the total sales. A COMMISSION-AMOUNT calculated item might be defined as

```
SALE-AMOUNT * 0.025
```

for a 2.5% commission rate. BRW contains two different types of defined values. They are "table calculated items", and "layout calculated items". They are similar. Their only difference is when they are used.

Table calculated items are actually stored into intermediate tables as are data items. In fact, data items are indistinguishable from table calculated items. Selection criteria may be based on table calculated items; sorts and breaks may be performed on table calculated items; and table calculated items may be used to join to another file. Table calculated items may be defined for any intermediate table.

Layout calculated items are processed only while reading the "final" intermediate table. The same mathematical operations may used to define layout calculated items, but they cannot be used for selection, sorting, or joining.

The choice between table calculated items and layout calculated items is based on how it is used. If the item is required for selection, sorting, or joining, then it must be a table calculated item, defined along with the table it is being used with. Otherwise a layout calculated item should be used.

Consider the COMMISSION-AMOUNT example. If a table calculated item were used, BRW would calculate the value of COMMISSION-AMOUNT for every record and store it into the intermediate table. This in itself is not the problem. After all, the COMMISSION-AMOUNT is needed for the report, it has to be calculated sometime.

The problem is that BRW has written the value into the temporary work file which holds the intermediate table. This increases (albeit slightly) the size of the temporary disk file. But more importantly, it increases the sort time for the file. If is not necessary to sort on COMMISSION-AMOUNT then it is simply taking up space which could be used by the sort process. Remember that the time it takes to sort a file varies as the size of the file and the size of the data records.

COMMISSION-AMOUNT can be defined as a layout calculated item. It will be calculated during BRW's final pass. It may still be printed and sub-totaled, just as any other data item or calculated item, but it will not take up any space in any temporary files.

Expanding BRW Usefulness

HP BRW can actually be used in applications which are not strictly reporting applications. When invoking a "report" by name, BRW searches for a file with that name. That file may be a report execution file (the output of the BRW compiler, an REXEC file), or it may be a report job file.

A BRW job file is a specially formatted ASCII file which contains some BRW parameter specifications, followed by MPE batch job JCL. A job file can be created by an ordinary text editor, or manipulated as any other HP3000 text file would. BRW scans the job file, displaying the parameters on the parameter entry screen. Once specified, the parameters are inserted into the JCL portion of the job file, and the job is :STREAMed.

A BRW job file can contain any number of MPE commands, options, parameters, programs, etc. In fact, a BRW report job need not ever execute a BRW report. Once users are used to the BRW parameter screen, it can be used to launch any kind of job at all. The BRW parameter screen allows a crude sort of parameter input editing, such as numerics, integers, and dates must be in the correct format. The BRW parameter screen also allows built-in defaults, deferred scheduling capability, and output file specifications.

The BRW parameter screen, in combination with a report job file, can be used to automaticaly switch between multiple versions of the same report, based on input parameters. This relieves the operator of having to memorize several similarly named reports, and which one is appropriate. Within the report job file, MPE `:IF` statements can be used to branch around statements and select the correct options, parameters, and file equations for a given set of parameters. Such a job file can even select reports or file equations based on the time of day, or day of the week, by using combinations of BRW parameters and pre-defined MPE JCWs.

BRW Limitations

As with most computer software, BRW has its disadvantages. There are many trade offs in the world, and the designers of BRW had to make some. None of BRWs limitations are very severe, but they must be considered.

As with most report writers, BRW can create only one output print file per run. The intermediate output file capability is intended to overcome this limitation. In real life, there are many cases where very similar versions the same report are run at the same time, such as month end reports and month end summaries. Both select the same data. Both sort the same items. Both sub-total the same fields. But one doesn't print out all of the detail lines. Using intermediate output files, a month end job could be created which runs BRW three times, once to select the data and create the intermediate file, once to print the intermediate file with the details, and once to print a summary version.

BRW uses a configuration file for many of its run time operational parameters. One of these is the maximum size of a temporary work file. When installing BRW the value (in records) can be set to any value. A small value risks an end of file error on a large report, but a large value wastes lots of disk space for small reports and reports which use many temporary tables. It is possible to have two different BRW configuration files, one with a large limit, and one with a small; and use MPE `:FILE` commands within the report job file to use the correct one. There is only one value per report, however. If the temporary file size is set to 1,000,000 records, and one of the intermediate tables only selects a few records, the rest will be wasted.

BRW requires a specially formatted dictionary file in order to define and compile a report. This special dictionary, (an `RDIC` file) can be created in one of two ways. First, a conversion program is available which will convert a DICTIONARY/3000 (not SYSTEM DICTIONARY) into an RDIC file. The other method requires a special program which converts an HP application dictionary into an RDIC file. (Application dictionaries are used with some of HP's business application packages, such as MM/3000)

The BRW report definition is a V/3000 based process. This makes some parts of report definition a simple fill in the blanks operation. However, it makes simple reports, and minor modifications very tedious, as many many many screens are required to make even the simplest of spelling corrections. There are rumours of utilities in the works, and programs from the INTEREX CSL, but they are of limited use so far.

BUILDING EFFICIENT TRANSACTION MANAGEMENT SYSTEMS USING MPE SPECIAL CAPABILITIES

George Hall
HBO & Company
Southern Division
301 Perimeter Center North
Atlanta, Georgia 30346
(404)393-6000

A little less than ten years ago software development was underway on a full financial system for medium size hospitals. This was to be the first vendor supplied system of its type to run on one of those new mini-computers in the hospital itself. The system was installed on an HP 3000. This amazing machine had a multi-processing operating system, 512k of main memory and would soon have 120mb disc drives. State of the art software technology was used also. An early version of the VIEW forms management system and the IMAGE data base management system were used to complement COBOL. The system was installed at the first customer site and, as is often the case with new technologies performance was unacceptable. During the next few months a transaction management system was written to squeeze the most out the limited resources available by eliminating many process creations, file opens, virtual memory swaps and other resource intensive operations. While fast CPU's and peripherals, and large memory configurations have allowed us to outgrow these problems in many cases, there is still often competitive advantage and cost savings in fitting your software on a smaller hardware configuration. This is particularly true for software vendors who can leverage the time involved in developing and supporting such a system over many installations. This paper will explain the basic design of this transaction management system and how MPE special capabilities were used to accomplish the design. A conceptual understanding of process handling, no wait I/O, and extra data segment manipulation will be assumed.

The entire transaction management system consists of one large process tree (see exibit 1) which is executed from the system console. User terminals are driven as slave devices by the user processes (UTIC##) called Functional Controllers. The Functional Controllers, numbered 1/13 in our case, are logical groupings of user applications bound together as subprograms into one process. They are sons of a driver process (UTIC00) which manages users switching from one Functional Controller to another. UTIC00 is a son of a master controller (UTICON) which is run from the system console. UTICON accepts, parses and in some cases executes commands, and controlls the process tree. UTIFAS which is a son of UTIC00 is a log on processor. All of these processes

communicate with each other by way of a shared extra data segment called the Communications Extra Date Segment (CX'DS). The user applications are written in COBOL using V/3000, Image and KSAM. All of the controller processes are written in SPL. The system has not been migrated to MPE/XL. The next section of the paper will deal with each of these processes in more detail.

UTICON

UTICON is the master controller program which is run from the system console. It accepts commands entered on the console, parses them and attempts to execute them. Each command is matched against a table of internal commands. If a match is not found the command is passed to MPE via the COMMAND intrinsic. Some of the internal commands are:

- ACTIVATE ldev [ALL] - This command takes a single device, a list of devices, or the keyword "ALL" and prepares the terminal for user log on to the process tree. This command also issues a "REFUSE" command against the device to prevent MPE sessions from being established on the terminal when it is DEACTIVATED (see below).

- DEACTIVATE ldev [ALL] [;NOW] - This command removes a device from the system. If the ";NOW" keyword is used the user will be removed immediately. Otherwise they will be removed the next time they change screens or when their View screen times out. The users keyboard is locked after they are removed.

- UNLOCK ldev - This command will issue an "ACCEPT" command against the terminal and unlock the keyboard. This will allow the terminal to be used as a session device.

- DISPLAY - This command displays information on all ACTIVATED terminals. The device number, the initials of the user logged on to the device, the name of the screen they are using and the time this screen was entered is presented.

UTIWHY

Rather than having UTICON leave a read pending against the console all day for command input, the designers decided to have it suspend when it is finished processing a command and activate UTIWHY. UTIWHY is about ten lines of code long and all it does is arm a control-y trap and pause until the operator presses control-y. When control-y is pressed UTICON is activated and UTIWHY suspends. The console is then prompted by UTICON for command input.

UTIC00

UTIC00 creates and manages the Functional Controllers (UTIC##). When the system is first started several copies of each Functional Controller are created and suspended, and their PIN's, controller ID numbers, and current status are stored in internal tables (CTLR'PIN, CTLR'NMBR, CLTR'STAT). UTIC00 then suspends in a SON wait. When it is activated it scans the table of valid terminals (T'TAB) in the CX'DS to see if any one has requested a new Functional Controller and/or has exited the one they were using. If one is being exited it's CTLR'STAT entry is flagged as available. If a new one is requested, CTLR'NMBR and CTLR'STAT tables are checked to see if a copy of the requested Controller already exists and is available. If a copy is available the user and the process are "matched up" (more detail on this later), otherwise a new copy is created. This ability to share user processes and to reuse them as many times as necessary greatly reduces the amount of process creation being done on the system. UTIC00 executes in the "BS" queue.

UTIFAS

UTIFAS is a log on processor that operates in a timed loop. It suspends for .5 seconds then performs it's loop again. The first thing it does each time through the loop is to check the CX'DS to see if UTICON has flagged any terminals to be activated. For each device that has been flagged as such since the last loop the program calls GETPRIVMODE, opens the terminal device for NO-WAIT I/O then calls GETUSERMODE. The PIN of the terminal file is stored in a table in the program. The FOPEN is the only line of code executed in PM. A NO-WAIT I/O read is issued against the newly opened terminal file then IODONTWAIT is called to see if any I/O has completed for any terminal file. If any I/O completed and function key 1 was read, the user at that terminal will be logged on. The T'TAB entry in the CX'DS for that device is set to indicate to UTIC00 that a log on has been requested by this terminal. UTIC00 is then activated and UTIFAS suspends for .5 seconds and does it all again. UTIFAS operates in the BS queue.

FUNCTIONAL CONTROLLERS (UTIC##)

The Functional Controllers are logical groupings of user applications. These COBOL applications are compiled as subprograms into a single process with a generic SPL outer block. The idea is to minimize the amount of switching between user processes. For example all of the inventory applications may be in Functional Controller #2. In addition each process contains all of the logic necessary to select another Functional Controller and an application within it. This requires the user to make their selection and commit themselves before any process switching takes place. The user terminals are opened for I/O by specifying the "termfile" option in the VOPENTERM call. The name entered in "termfile" is directed to the desired terminal with a :FILE command issued by the SPL outer block. When a Functional Controller is activated by UTIC00 it first checks the CX'DS to see which device has requested it. This number is then used in the :FILE statement which redirects "termfile" to the specified device. It then checks the CX'DS to see which of the application subprograms was requested and enters this subprogram. If a user wishes to change Functional Controllers they press F1, which triggers the programs to back up a screen, until they are presented the Functional Controller selection screen. After they select a new Functional Controller they are presented with a menu of its applications. At this point they still have not left the initial Functional Controller since the code to do all of this is in all Functional Controllers in the generic SPL outer block. After they select an application in the new Functional Controller, its number and the name of the selected application are noted in the CX'DS, the current Functional Controller is flagged as inactive, the terminal is closed by the VCLOSETERM intrinsec, UTIC00 is activated and the Functional Controller suspends.

UTIC99

UTIC99 is a special case of a Functional Controller. It is the generic SPL mainline which is common to all Functional Controllers, compiled without any application subprograms. Like all Functional Controllers it contains selection screens for all Functional Controllers. Each user is placed in UTIC99 when they first log on to select their initial Functional Controller. Since everyone logs on at the same time several copies will be needed. This is a very small program with a very small stack. Using it for initial selections instead of using a full Functional Controller will cause less memory trashing during those first few minutes when the system is brought up. It also will use less virtual memory after most users are logged on and fewer copies will be needed.

CX'DS

Exibit 2 is the copy file which defines the CX'DS. The T'TAB is the heart of the system. There is one entry in it for each valid terminal device. The T'TAB entries which are pertinent to this discussion are defined below:

word	definition
0	STATUS - Current status of this terminal. Values we will use are NOT'ON'SYSTEM, WAITING, RUNNING, and SWITCHING.
1	COMMAND - The action to be taken. Values we will use are NONE, DEACTIVATE, ABORT.
3	LOGICAL DEVICE NUMBER - The ldev # of the terminal this entry defines. They are not necessarily in order by device number.
4	UTIC00 TABLE INDEX. - This entry contains an index into UTIC00's tables which will point to the Functional Controller assigned to this device.
5	CONTROLLER NUMBER - The Functional Controller currently associated with or requested by this device.
6	TRANSACTION NAME - The name of the screen this terminal is accessing or has requested.
15	TIME OF ENTRY - The time the user entered this screen.
17	USER INITIALS - The initials of the user accessing this terminal.

At this point we will step through a user "ACTIVATION" and log on making special note of the CX'DS and it's role.

- The operator enters "ACTIVATE 25" at the console. UTICON accepts this command and places WAITING in the STATUS field of the T'TAB entry for device 25.

- UTIFAS wakes up from it's regular half second PAUSE and scans all of the T'TAB entries looking for one with WAITING in the STATUS field. Device 25 qualifies and it is FOPEN'ed in PM and a No-Wait read is issued against the terminal file. UTIFAS checks to see if any previous read's have completed then sleeps.

- A user presses f1 on device 25 to initiate a log on. The next time UTIFAS wakes up, after it scans the T'TAB for new activations, it calls the IODONTWAIT intrinsic to see if any I/O has completed. I/O has completed on device 25. UTIFAS checks to see that it was an f1 that was read.

- UTIFAS places SWITCHING in the status field of this device's T'TAB entry and places 99 in its CONTROLLER NUMBER field to indicate which Functional Controller (UTIC99 in this case) is needed. UTIC00 is then activated.

- UTIC00 scans the T'TAB looking for someone that needs service. The T'TAB entry for device 25 indicates that it is switching to UTIC99. UTIC00 has three internal tables which define and track the available Functional Controllers. CTLR'NMBR, CTLR'PIN, and CTLR'STATUS are indexed by the same value (ex. CTLR'NMBR(01), CTLR'PIN(01) and CTLR'STATUS(01) all refer to the same Functional Controller). For each '99' in CTLR'NMBR, UTIC00 checks CTLR'STATUS to see if it is in use. If all copies were in use a new one would be created and added to the three tables. After an available copy is found, STATUS is changed to RUNNING in the T'TAB and UTIC00 TABLE INDEX is set to the index for the copy of UTIC99 in use. UTIC99 is activated and the index into the T'TAB entry for the device it is to open is passed in the PARM.

- The Functional Controller (UTIC99 in this case) checks the T'TAB entry (indexed by the value of PARM) to get the device number it is to access. It is used to issue a FILE statement for the formal file name opened by the VOPENTERM intrinsic. Screens which will prompt the user for Functional Controller desired, the specific application desired, passwords and initials are presented and the data is read. These are placed in the CONTROLLER NUMBER, TRANSACTION NAME and USER INITIALS fields in the T'TAB. The STATUS field is changed to SWITCHING and UTIC00 is activated.

- UTIC00 performs the same process again. The Functional Controller we have just left is flagged as available and a copy of the newly requested one is located.

This system was implemented before message files were available on MPE. The system could be built nearly as efficiently and more simply (conceptually at least) by adding a process as a table server and having it provide the same information to the other processes as the CX'DS does via message files.

DISADVANTAGES

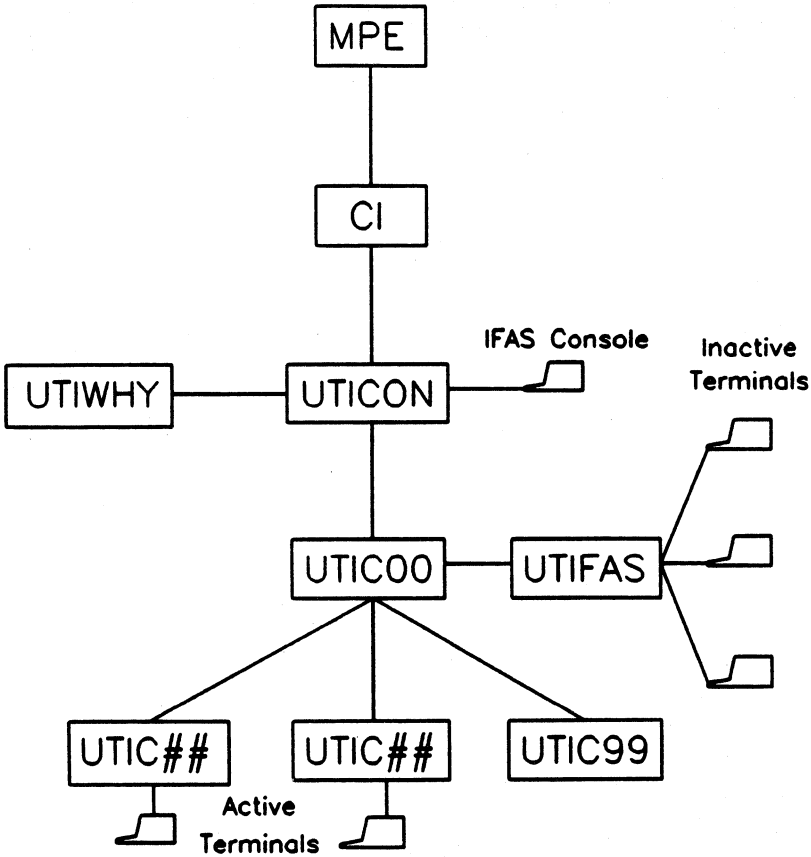
As with any system there are disadvantages also.

- At least one highly specialized person is needed to support and enhance this system. This may be difficult for a small shop.

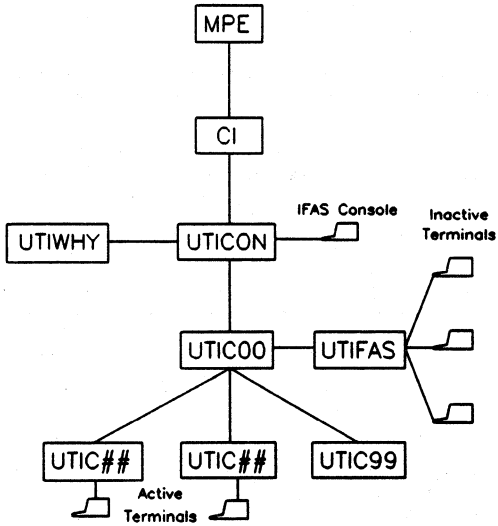
- HP and third party software usually assumes that each user will have their own session. This will occasionally be an inconvenience. For example, we wanted to use a third party logging tool for audit purposes. Since all users are part of the console session all changes to the data bases were recorded as being made by the console.

- It is difficult to use interpretive languages in this environment. We were asked to determine if we could use POWERHOUSE in our process tree. A prototype system was programmed and tested. It worked, but we had to trap the WHO and XCONTRAP intrinsics in a group level SL and change how they worked to do it. Vendors of these languages are usually not very forthcoming with information on the internals of their software, requiring lots of reverse engineering to figure these things out. Any language or tool that gives you a USL when you are finished should be easy to incorporate.

EXIBIT 1


MPE
CI
IFAS Console
Inactive Terminals
UTIWHY
UTICON
UTIC00
UTIFAS
UTIC##
UTIC##
UTIC99
Active Terminals

```
$PAGE "********** CX'DS COPY **********"
<< ************************************************************ >>
<< *                                                          * >>
<< *                      - I - F - A - S -                   * >>
<< *                                                          * >>
<< * Interprocess Communication Data Segment Element Definitions * >>
<< *                                                          * >>
<< *                          (CX'DS)                         * >>
<< * -------------------------------------------------------- * >>
<< * !                                                      ! * >>
<< * !  ---------  ---------  ---------  ---------  ------- ! * >>
<< * !  : F'TAB :  : L'TAB :  : T'TAB :  : X'TAB :  : V'TAB : ! * >>
<< * !  ---------  ---------  ---------  ---------  ------- ! * >>
<< * !                                                      ! * >>
<< * -------------------------------------------------------- * >>
<< *                                                          * >>
<< *  F'TAB (fixed, global info)                              * >>
<< *                                                          * >>
<< *    word   definition                                     * >>
<< *    ----   ----------                                     * >>
<< *       0   pointer to start of L'TAB entries              * >>
<< *       1   pointer to start of T'TAB entries              * >>
<< *       2   pointer to start of X'TAB entries              * >>
<< *       3   pointer to start of V'TAB entries              * >>
<< *       4   pointer to start of FLAG words                 * >>
<< *       5   reserved                                       * >>
<< *       .     .                                            * >>
<< *      19   reserved                                       * >>
<< *                                                          * >>
<< *  L'TAB                                                   * >>
<< *                                                          * >>
<< *    the list of Security Defined logical devices; those   * >>
<< *    which may be accessed by IFAS.                        * >>
<< *                                                          * >>
<< *  T'TAB (active process info)      V'TAB (release-update #) * >>
<< *                                                          * >>
<< *    word   definition                word   definition    * >>
<< *    ----   ----------                ----   ----------    * >>
<< *       0   status                      0   UTIC00        * >>
<< *       1   command                     1   UTIC01        * >>
<< *       2   flag                        2   UTIC02        * >>
<< *       3   logical device number       .     .           * >>
<< *       4   UTIC00 table index          .     .           * >>
<< *       5   Controller number           .     .           * >>
<< *       6   Transaction name           11   UTIC11        * >>
<< *       9   Transaction index          12   reserved      * >>
<< *      10   security parameter          .     .           * >>
<< *      11   previous Menu Form name     .     .           * >>
<< *      15   time of entry               .     .           * >>
<< *      17   user initials              18   UTIFAS        * >>
<< *                                      19   UTIC99        * >>
<< *                                                          * >>
<< *  X'TAB                                                   * >>
<< *                                                          * >>
<< *    each entry is an index into the L'TAB list; one per   * >>
<< *    each process created through the Menu Controller.     * >>
<< *                                                          * >>
<< ************************************************************ >>
```



UTICON

SUMMARY: Creates the process tree and processes commands.

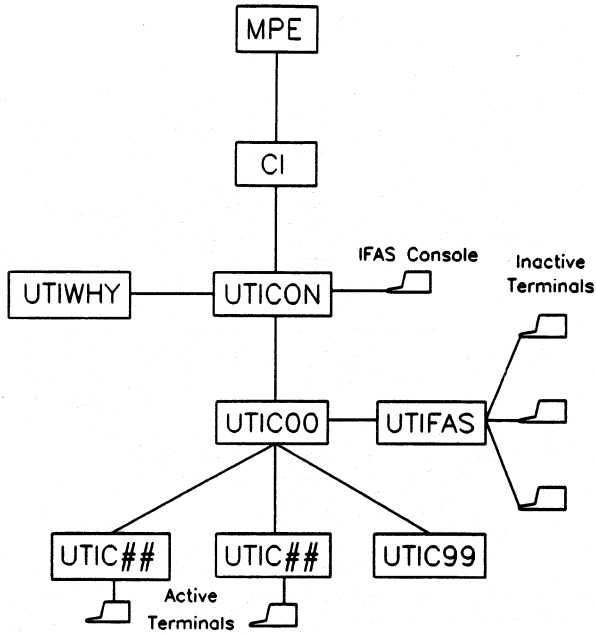
1. UTICON is run by the 'IFAS' UDC.

2. UTICON creates the UTIWHY process immediately and the UTIC00 process when the first IFAS user terminal is activated. UTICON will suspend in a SON wait as the other processes are being created.

3. Some special processes may be created when UTICON is first run on hospital systems which have purchased additional modules. These processes may involve IPC and/or RJE. These additional modules are not widely used now.

4. Executes MPE or IFAS commands. UTICON does an FWRITE of a ":" to the terminal and then does an FREAD for 80 characters. It then executes the command (either MPE or IFAS). When the command is completed, UTICON does an FWRITE of "Ready" to the IFAS master console, activates UTIWHY (which has been in a FATHER wait) and goes into a SON wait. The first user terminal activation creates UTIC00, updates the extra data segment table with the pertinent information, then suspends. Subsequent user terminal activations update the extra data segment table then suspend.

5. Operates in the C queue.



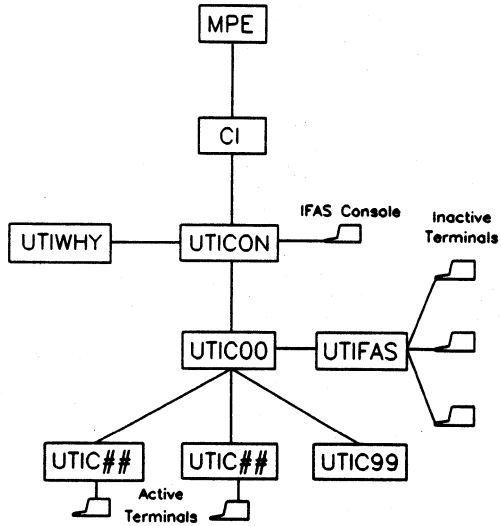
UTIWHY

SUMMARY: Continually monitors for a Control Y at the IFAS console.

1. Spends most of its time in a loop which consists of a PAUSE. It wakes up, checks for a Control Y, then pauses again. It will be in a timed wait most of the time.

2. When UTIWHY senses a Control Y, it wakes its father, UTICON, and goes into a FATHER wait.

3. UTIWHY operates in the C Queue.



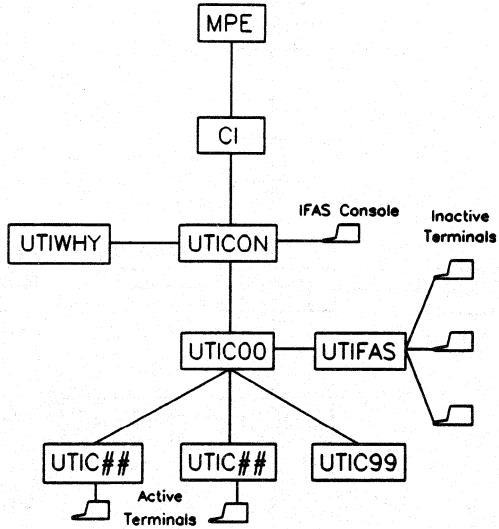
UTICOO

SUMMARY: Processes the switching of user terminals from one transaction processor (Functional Controller) to another.

1. Whether activated by UTIFAS because a user is ready to enter his/her first transaction processor, or activated by a transaction processor because the user wants to switch to an alternate transaction processor, the UTICOO switching process is the same. UTICOO looks for an available transaction processor of the type requested by checking the extra data segment table. If none is available, ie. none is in the suspended state, he creates one. UTICOO then updates the extra data segment table with the PIN for the process and other information needed for IFAS interprocess communication, activates the process and passes it the index into the extra data segment table associated with that logical device. UTICOO then goes into a SON wait.

2. Spends most of its time in the wait state (FATHER and SON). When UTICOO is activated, he scans the extra data segment looking for the action that needs to be taken.

3. Operates in the linear queue.



UTIFAS

SUMMARY: Processes the "log on" of IFAS user terminals

1. For each terminal which was placed on the "Activate" list in the extra data segment table by UTICON:

   a) Opens each terminal for NOWAIT I/O. GETPRIVMODE is called immediately before the FOPEN and GETUSERMODE immediately after.

   b) Writes "The IFAS System is Ready" to the terminal (NOWAIT).

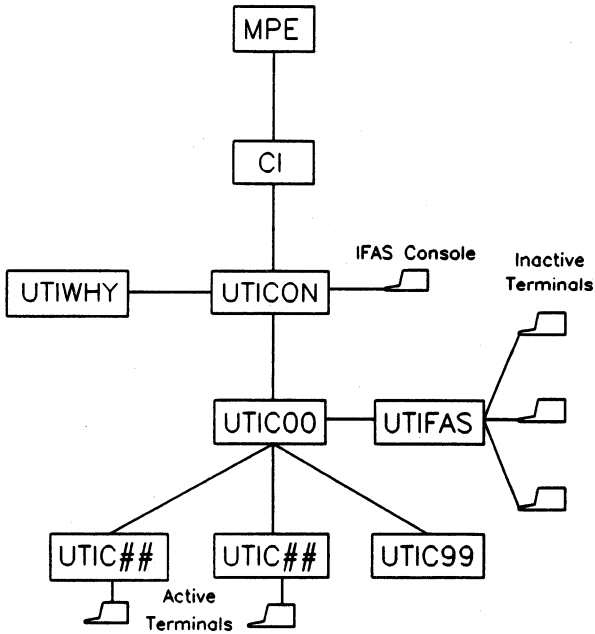
   c) Posts a two character read to the terminal (NOWAIT), waiting for the user to press F1.

   d) Does periodic IODONTWAIT's to check for completed reads.

2. Calls the DELAY intrinsic instead of the PAUSE intrinsic, so this process will most often show as IMPEDED in a dump. The GETPRIVMODE is immediately before the DELAY and the GETUSERMODE is immediately after.

3. If an FREAD has completed, UTIFAS updates the extra data segment table with the LDEV number and activates the UTIC00 process.

4. Operates in the linear queue.

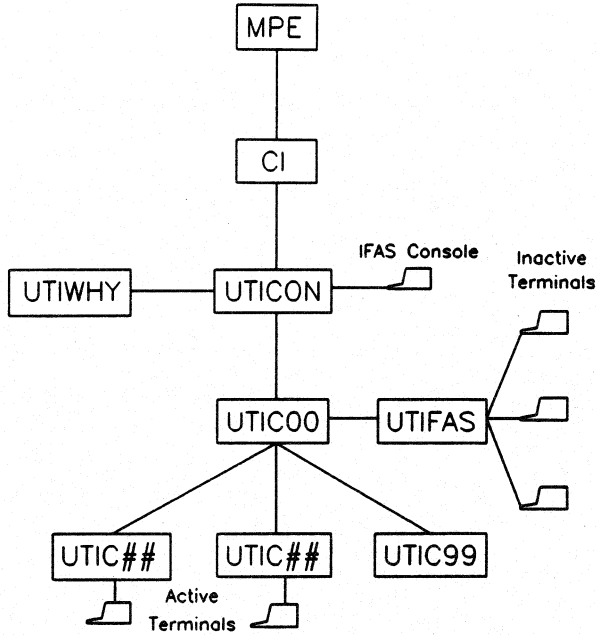


TRANSACTION CONTROLLERS (UTIC##)

SUMMARY: Contain the user applications in logical groupings. Called Functional Controllers.

1. SPL outer block with COBOLII, VPLUS and IMAGE modules compiled in as subprograms. The controllers are functionally modular so process switching is kept to a minimum. All terminal I/O is conventional (mostly through VPLUS screens).

2. When a transaction is requested that is not contained in the current module, the controller updates the proper entries in the extra data segment table and activates UTIC00, suspending himself in a FATHER wait.



UTIC99

SUMMARY: Special case of a transaction processor. Displays the initial selection screens.

1. UTIC99 is a transaction processor compiled without any of the user application programs. UTIC99 is the transaction processor that UTIC00 associates with a user terminal when that terminal is first passed to it from UTIFAS (the user "logs on" by pressing F1). The user makes a selection from the menu and is passed to the chosen transaction processor. Each transaction processor contains a copy of these menus, so it is never necessary to come back to UTIC99 until the user needs to log on again.

PROGRAMMING FOR OPERATIONAL EFFICIENCY

by

Robert G. Boynton

Warn Industries

13270 S.E. Pheasant Court

Milwaukie, OR 97222

(503) 659-8750

INTRODUCTION

Two primary issues of operational concern are disc space usage and run-time efficiency. Systems and Programming efforts to support operations in these areas have centered on several traditional areas.

In the area of disc space usage, the Systems and Programming effort is an attempt to project the ultimate size of a file or data set, and to set those sizes at the time of turnover for production. As Systems and Programming attempts to retain only the data that is needed, usually they can do little to improve disc space usage.

However, Systems and Programming can better address the issue of efficient disc space usage at the time of development. Also, with a minimal amount of programming effort, can probably experience some major disc space savings for existing systems in a short period of time.

In the area of run-time efficiency, the traditional Systems and Programming endeavor has usually been limited to pre-production functions. Before production implementation, Systems and Programming makes an effort to fine tune the code to maximize the run-time efficiency. But once the file, data base, and programs are implemented, it usually takes a major disc space or run-time problem for the Systems and Programming staff to re-evaluate those items already in production. Requested modifications to production systems do not cause this re-evaluation unless there is a significant difference between the "production" and "new" versions.

However, Systems and Programming can help improve run-time efficiency, both at the time of implementation, and in on-going efforts to fine tune the system. Although this involves Systems and Programming in non-traditional tasks, their expertise is needed in this effort.

This paper proposes some possible ways for Systems and Programming personnel to aid Operations in the above areas. In order to use these techniques, it is necessary to re-think some things we have learned regarding dynamic disc allocation and data base design. The techniques as taught by Hewlett-Packard and others are essentially correct, but can cause some major inefficiencies in the areas described above. The following will describe some alternative techniques in detail.

DISC SPACE USAGE

From a disc space usage point of view, the worst thing we can design into a data base is a manual master. In order to keep the migrating secondaries within acceptable limits, we must "size" the capacity of the data set so that the projected entries into the data set only represent a relatively small percentage of the capacity. While there are some utilities available that determine the percentages for a given data set, the standard recommendation is forty to sixty percent.

Because the capacity established for a data set pre-allocates the disc space, it is easy to see that a data set with a large number of entries is a poor user of disc space. This is particularly true for those data sets that have a large entry length. Compounding this is that for a manual master that has a large entry length, when a migrating secondary is encountered, the likelihood that the entry is in the accessed block is reduced.

There is an alternative that requires some data base redesign and some minor program changes. Simply put, it is to intelligently eliminate most of your manual masters.

By making a manual master into a detail data set (tied to an automatic master for the key) it is now possible to "size" the detail data set to reflect the current entry count with the projected growth. By then "sizing" the automatic master to the recommended forty to sixty percent entry count to capacity ratio, you still get the reduction in migrating secondaries and get a significant savings in disc space. By efficient blocking of the automatic master, you can insure that, even when a migrating secondary is encountered, the desired entry will be in the block most of the time.

This is followed up by changing the program access to do "DBFIND" and "DBGET" for the affected data sets, instead of just the "DBGET". It may also be necessary, in some program constructions, to validate that there are not any entries for the key value before adding an entry. Although it is not common, some programs are designed to evaluate the returned errors on masters to show that a duplicate entry was attempted. Quite obviously, this will not work when the data base is redesigned.

Admittedly, there is some additional data base overhead. The data base controller will have to access both the automatic master and the detail data set on outputs and directed reads. If this is a concern, testing the impact and comparing it against the savings in disc space will help you reach a determination for a given data set.

You do not want to do this for every manual master you have in the system. If the manual master has a small entry count and entry length, the savings, if any, is not worth the effort. A good technique is to set an in-house minimum savings standard (i.e., 100 sectors) and if the change will not save at least that much, don't mess with it.

An added benefit to this structure is that you no longer need to develop "cross reference" data sets to tie key item information from various manual masters together. As the former manual master is now a detail data set, it is possible to carry the other key items associated to the data item in the same data set. Often, the reduction in accessing the "cross reference" data set will significantly reduce the impact of the overhead caused by changing the master data set to a detail data set. Admittedly, this particular change to an existing data base may cause some significant rewriting of existing programs. If this is the case, you will have to evaluate whether the change is practical in your environment.

In short, by rethinking data base design, both existing and new, and using the idiosyncrasies of the Hewlett-Packard Image data base structure, it is possible to develop data bases that are much more efficient in disc space usage.

RUN-TIME EFFICIENCY

Beyond writing, or modifying, programs to better utilize the systems resources, not much thought is usually given to other aspects of run-time efficiency.

The biggest inroads can be made in the areas of batch processing. This area usually consists of processing large amounts of data in a single program, or series of programs. The processes are much more susceptible to being hardware bound, particularly disc I/O, than online processing that is usually controlled by human interaction.

Although some of this technique is applicable to files as well as data bases, the major benefit can be obtained in the area of data bases. It is possible to extrapolate these techniques to files, if this is an issue in your environment.

In order to improve run-time efficiency, it will be necessary to partially abandon the "dynamic" allocation of data to disc drives for data bases used in heavy batch processes. By "spreading" the data sets onto specific disc drives, you will remove disc contention between data sets in the same application. You then insure that the read/write heads of a disc drive dancing back and forth across any single disc in order to process the application.

By using the actual number of disc drives you have to work with, start determining which data sets should reside on which drives. I suggest that you not count your system drive (drive 0 - ldev 1) any more than necessary, because this drive is already being heavily used by the system itself.

The first data sets that you will want to keep on different disc drives are, quite obviously, detail data sets from their associated masters. This can be done by the operations staff.

Next the Systems and Programming staff will have to evaluate the programs that run in batch mode and separate those data sets that are being accessed in the same program cycle from one another. This will entail analyzing the individual programs. You must see which data sets are used in the program and when they are accessed. It will then be possible to determine those data sets that will be in contention. These data sets should be placed on separate disc drives.

Lastly, if several batch processes run at the same time on a regular basis, you will want to "spread" the affected data sets from all these processes.

Once the spreads have been determined, it will be necessary for Operations to physically move the data sets and to maintain the spreads.

Because no facility will have enough disc drives to develop the ideal spread, it will be necessary to make compromises as you go along. This then makes the spreading of data sets a "best guess" technique. It will be necessary to re-evaluate the spread on a regular basis in order to "fine tune" the spreads to best use your resources. Because of the need for evaluation of the actual programs, it is necessary for Systems and Programming to be an integral part of this process.

The moving of the data sets can be accomplished by using several data base management and MPE extension utilities that most every shop has one or more of. In order to maintain the spread, it will be necessary to reload the data from backups and use the option(s) that will place the files on the same drives as they originated from.

If you are going to reload your system to more evenly spread the data, you should load the affected data sets first, placing them on their specified disc drives. Then you can reload everything else and allow the system to dynamically spread these files.

If you include critical files in this technique, it will be possible to modify the procedure, at file creation time, to create the file on a specified disc drive.

By partially abandoning the use of dynamic allocation of disc space, you will be able to make some significant gains in run-time efficiency for your batch processes.

CONCLUSION

By re-evaluating some of the data base design techniques and dynamic allocation facilities of the Hewlett-Packard environment, it is possible to garner some major gains in disc space usage and run-time efficiency.

The changes will involve the Systems and Programming staff in areas that have been, traditionally, the concerns of the Operations department. By developing the "team approach" to these issues, particularly the data base spread technique, the impact will not be too significant in either area and will allow for some interaction that will benefit both sides.

Remember, just because the system does certain things for you, it may not be the most efficient use of your resources. You should spend some time evaluating manual alternatives to improve your systems. The benefits can easily outweigh the cost of the extra effort.

IMPLEMENTING SOFTWARE ENGINEERING

Stephen L. Day
James River Corporation
110 S. Seventh Street
Richmond, Virginia 23219

Introduction

Software Engineering is an old idea with new initiative. In every article dealing with the current and future development of software, the author manages to work in the buzzwords "CASE" or "JAD" or "Structured Analysis" somewhere in the article. As the tools and techniques become more integrated and the practices become more accepted within development organizations, Software Engineering techniques will become part of the application developer's toolkit.

But Software Engineering is not a tool, it is a culture. Within an organization, the implementation of a Software Engineering program can cause radical change in the way developers work with user organizations, the way software projects are managed, the way estimates are developed, and the way analysts work within their own groups. This has the potential of causing a development organization to rebel against the changes. This paper will examine the animal that is called Software Engineering, assess the potential issues in implementing the process, and put forth an outline of how to proceed with introducing Software Engineering practices into your own organization.

Defining Software Engineering

Software Engineering is commonly confused with many of its parts. When the term is mentioned to many Information Systems professionals, they immediately equate the term CASE, or Computer Aided Software Engineering as being the whole world of SE. Or, they will point to a Systems Life Cycle as being Software Engineering, or use the Yourdon Structured Design as an example. All of these things are components of Software Engineering, but none of them alone are capable of bringing successes in developing software. Software Engineering is defined (by me) as:

> A collection of interlocking technologies that embrace methods, tools, and procedures to achieve quality software and productivity gains of the practitioner.

Let's take a look at each component of Software Engineering and its goals. The procedures used to develop software within an organization comprise the Systems Development Life Cycle. This may or may not be formally stated within your organization, and if there is not a documented SDLC to follow, you as a practitioner probably follow a different life cycle than your peer on the other side of your office wall. The Systems Development Life Cycle defines the process which is followed in bringing a software project from initiation through implementation. A formal plan should detail the different phases of the life cycle, appropriate checkpoint and review milestones, end-user acceptance procedures, and the end-product, or deliverable, for each life cycle phase. The SDLC is designed to be flexible enough to allow a project manager to make judgements on whether or not a phase or deliverable is appropriate, but should also mandate a baseline set of documents.

For purposes of discussion here, we will use a fairly simple Systems Development Life Cycle as our procedures for developing software. It is comprised of six stages:

The requirements stage is where user needs are translated into formal documented goals of the system. This is also where the boundaries for the system interfaces are determined. Analysis is the translation of the system goals into logical system functions and components, while the design activities turn the logical functions into physical system units. Coding is the translation of the physical design into language the computer understands, and testing involves the checking of each design unit, assembly, subsystem, and the entire system against the original requirements definition. Finally, implementation is the turnover of the system to the original requestors to use in their normal work.

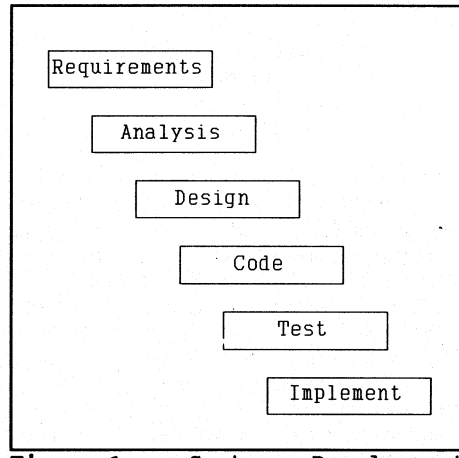


Figure 1 - Systems Development Life Cycle.

While the Life Cycle determines the overall plan for a system, the methodology determines what is done within each phase of the Life Cycle. Each phase of the SDLC has different deliverables, and each deliverable will have one or more techniques associated with its development. Joint Analysis and Design, or JAD, is a technique where applications designers and applications requestors jointly develop the external architecture of the application, which may be defined by the methodology to be a Data Flow Diagram, Entity-Relationship Diagram, or process narrative. During this process, the Requirements Specification will be developed, as well as accomplishing a large portion of the Analysis phase of our SDLC.

For tools to be effective, they should directly support the activity dictated by a methodology. Software Engineering utilizes a family of tools that support

* Analysis & Design
* Testing
* Prototyping
* Reverse Engineering
* Project Management
* Change Management
* Code Generation
* Desk-Top Publishing

which are all of the functions required in the production of the design, programs, and documentation for a system. Software Engineering tools enable several of the methodologies which have been proposed since the mid-Seventies, but have been practically impossible to follow. For example, Ed Yourdon's text on Structured Analysis and Systems Design was written in 1978 and was hailed as the future of systems development. However, anyone who has practiced Yourdon's teachings on paper with a pencil better have a LARGE eraser! Although the concepts were as valid then as they are now, the tools to enable the technology are still being fully developed. Tools are available that will enable the designer to automate portions of the life cycle, but the tools are not yet capable of the degree of integration necessary to fully utilize them. Many tools support more than one methodology, and several support more than one phase in the Systems Life Cycle.

The goals of Software Engineering are to improve the quality of the software developed and to increase the productivity of the practitioner. Software quality is improved through better requirements definition and earlier problem resolution, as well as through the re-use of system modules. The productivity of the designer improves through the use of the tools and through the same re-use of code. Other benefits are the increased project management capabilities by using a defined life cycle and in better software estimating, which allows the developer to deliver projects on time and on budget.

Why is Implementation an Issue?

If the benefits of Software Engineering are so great, why is there any debate on whether or not (or how to) implement this technology? The biggest dangers are in the cultural changes necessary to embrace Software Engineering. The movement from systems development "the old way" to new techniques involves changing familiar habits and disrupts the status quo. As with any change, it must be carefully managed to nurture the people willing to accept change and to protect the process from those who will attempt to torpedo the process.

Another issue in the implementation of Software Engineering is becoming "enamored with the technology." The toolset available to the applications designer is very sexy and very powerful, but the toolset alone will not gain the benefits possible through engineering - usually, the tools used in this manner will allow the designer to generate bad systems which look "good" to the users and their management. Eventually, this leads to the failure of the first few projects where Software Engineering techniques are used, and will significantly retard the implementation process.

How To Implement

Although there is no single correct way to implement Software Engineering, the following is a method that has been used in several organizations. The method can and should be adapted to your own organization's culture, or there are other methods that have been published over the past years. The major point is to manage implementation of Software Engineering as you would a systems project for any of your users: assess the current environment, decide on the criteria against which the project will be judged, select the appropriate solution from a set of alternatives, install the solution, and evaluate its effectiveness.

The Assessment Phase:

The first phase of the implementation cycle involves two activities: evaluating the current organization as to its capability to embrace Software Engineering techniques, and surveying the marketplace to see what tools and methodologies exist. The organization can be polled, interviewed, or otherwise questioned to ascertain two points, what is the current knowledge base about tools, techniques, and methods, and what are the strengths and weaknesses in the current development process. This will define whether there are any pockets of knowledge to build upon, how much training will need to be done before the evaluation cycle can begin, and what areas of the current methodology need to be strengthened or modified.

The marketplace survey will serve to educate the assessment team as to what is available from vendors. This investigation should focus first on methodology (if the organizational assessment turned up methodological problems), and second on the tools and techniques which support the methodology. The methodology survey can encompass a product search, such as Method/1, Stradis, or others, or it may be more theoretical, such as looking at Information Engineering, Structured Analysis, or similar. In the theoretical search, you will be more likely to develop your own methodology from several sources and adapt them to your own organization, where the commercially available methodologies will more likely be accepted in their entirety and your organization will make the transformation.

In researching tools and techniques, a general survey will turn up probably 30 or more options which fit with the general development strategy. The tools should be classified according to their area of support in the life cycle or methodology, their alliance with other vendors/strategic partners, and their ability to interface or integrate with tools from other vendors. Since the CASE tools are still developing, flexibility and integratability is very important in the long-term viability of the product. From this list, you will be able to assess the degree of fit between your requirements and the CASE tool.

Develop Criteria:

After you have made the determination of what methodological changes will be required and have a census of the available options in tools and techniques, a set of criteria for the selection of the methodology and toolset should be developed. Initially, the current methodology study is used to define any changes to the methodology to be made. The strengths of the current development practices should be built upon and the weaknesses targeted for modification.

The systems development process should be evaluated for improving the techniques after the methodology improvements have been targeted. This also includes the tools which are needed to produce the documents dictated by the methodology. Although tools exist which can automate or assist in all components of the life cycle, it may not be cost-effective or managerially feasible to implement all of the tools available. Identify the areas where the payback is greatest in either quality improvement or productivity enhancement, and concentrate on these areas.

From the methodology changes and technique improvements identified, a set of short-term and long-term goals can be identified for the implementation of Software Engineering. The short-term goals can be specific timetables for implementing the improvements or training your staff, while the long-term benefits are related to quality improvement, application availability, and improved application development productivity. These goals should be related to the opportunities discovered in the assessment, and should be realistic in terms of the technology available to implement it.

Selecting the Alternative:

To select an appropriate solution, two major factors come into play: the goal set, and the available resources. Many different methods can be used to choose an alternative from the selection set: quantitative methods such as a Kepner-Tregoe analysis, group consensus from the team, or just a "gut feel" on which solution best fits with your technical problems and cultural environment. This is the place where you must be able to state the technical case for productivity improvement and software quality in business terms. For each alternative chosen, it should have a basis for selection in how it relates to the goal set and its business justification as to how it will help accomplish the overall business objectives of the organization.

Once the alternative is selected, the idea must be funded by your organization. This step may be done either before or after the implementation plan is done, but there are some advantages to seeking funding after drafting the plan. If the plan is developed, management can see the timing of both the dollars outflow and implementation benefits which will be realized. It also sets their expectations on what will be accomplished in the short and long term.

If a cost benefit analysis is required for approval, the absence of measures of productivity and quality for your organization will make this task very difficult. Several authors on Software Engineering advocate everything from a loose approach to the cost-benefit to some very rigorous and finite models. Select the path that gives the most complete picture that you can comfortably explain.

Installing the Solution:

The first step in the installation cycle is to develop the implementation plan. This plan should include a schedule of training, an acquisition plan for whatever portions of the solution that will be purchased, a construction plan for those pieces which will be built, and a rollout plan for all the pieces of the Software Engineering solution. The training schedule details what training is available, what training is required for all personnel, and what training is appropriate for

specialists in certain areas. For example, the Data Base Administrator may require special training in data normalization, while someone in a Lead Analyst role may need training in Business Area Analysis. The training can be any combination of seminar, formal classroom, video, computer-based, or text-based. Also, you may be able to take advantage of using "training the trainer" concepts, where a core group of people acquire the skills in enough depth to transfer the technology to the rest of the organization.

Another very effective way to accomplish the technology transfer is through the use of consultants. This may be done through their assistance in the implementation cycle, in their help on pilot projects, or through their designing a full training program. The use of consultants should be identified in both the training plan and the rollout plan.

The acquisition and construction plans contain the same information as would be in their counterparts in an ordinary software project. They include resource plans for installation of hardware and software, manpower plans for development, and the timetable for all events.

The rollout plan should detail the coordination of the training, acquisition and construction plans, as well as the timing of the initiation of using the methods. As with any new technology, there will exist a certain amount of frustration on the part of the practitioner when the techniques are used the first time. This is because many of the things he used to do are no longer applicable, and must be unlearned and replaced with new practices. Here, a facilitator can be invaluable in the first one or two projects.

Evaluation:

As with any project, monitor the performance of the pilot and the whole implementation and compare them with the results expected from the goal set. In most instances the results achieved will meet or exceed the anticipated results. In some cases, there will be side-effects which may or may not fit with the goals. Here, the solution may need to be modified and re-implemented to make the result fit more closely with the goals.

Critical Success Factors

As with most projects, there are certain elements which must either exist or be created for Software Engineering to flourish. The most important of these is a commitment on the part of management to implement Software Engineering within the organization. This is not an inexpensive endeavor, with estimates of from $25,000 to $75,000 per applications developer in expenditures over a five year period.

In implementing the tools, a sound methodology support platform is essential. Even though you may not think that the methodology is important, surveys have shown that of those who fail in implementing Software Engineering, over 40% believe that a major contributing factor was the lack of a strong methodology. The triad of tools, technique, and methods is a three-legged stool, and the omission of one will cause the whole thing to turn over.

Finally, adequate training is most important. Implementing Software Engineering is not like learning a new programming language. It is modifying the behavior of the applications designer at the base of his process, and the change is not one that can be self-taught. Classroom training, seminars, and on-the-job technology transfer from consultants are necessary for the core group of people who will become the engineering experts within your organization.

Summary

Software Engineering has the potential to transform your organization. This transformation must be carefully planned, however, or the transformation may not be a desirable one. The implementation of Software Engineering techniques involves changing your corporate culture. It requires the support of your management, your team, and your clients. With attention to its evolution and careful management of the change, your organization can achieve the quality and productivity enhancements that the technology offers.

Bibliography

"Characteristics and Functions of Software Engineering Environments,"
DATAPRO, July 1986.

"Implementing the Promise,"
Datamation, February 1, 1989, Michael Lucas Gibson.

"Software Engineering and CASE: A Guide for Installing New Technology,"
DEADLINE, March 1989, Roger S. Pressman.

"Making Software Engineering Happen,"
Roger S. Pressman, Prentice Hall, 1988.

Developing large generic systems using 4GL and Case

by
Håkan Davidsson
Teamco Utveckling AB
Box 352
S-931 24 Skellefteå
Sweden
Tel: +46 910-85170

An analyzing of the technic that are used when developing new systems, gives on hand that there are much to do, both in the methods used and in the technics. Several misstakes can be avoided if other methods and modern technic is used in the right way.

Teamco Utveckling (utveckling is the swedish word for R&D) is the result of using new technics and methods in developing big generic systems. One of the major targets with this company is to avoid conventionell problems in the developing process, as much as possible. Several changes, compared with usual developing technics, have been made during the selection of projectmembers, projecthandling, tools and methods that are used.

With this speech, I shall try to inform you about how we developing our systems in this new way.

Problems

In general speak we can isolate the problems in four sections. This four sections are Cost- and timeestimation, Bad quality, Bad documentation and the specific problems with big generic systems.

Bad cost- and timeestimating

This is the most known problem. Studies has shown that 95-98% of all system developing has problem with either cost- or timeestimating. It's very common that a developing process is in need of more money ore more time to be finished.

Lack of quality

The second major problem is that a new software product usualy contains a lot of bugs and errors. Big money can be saved if these errors can be eliminated in an earlier stage of the project, since it's very timeconsuming to find the errors when the product is nearly finished or, even worsh, when the customer has start to use the product. Some time even functions are of bad

quality, such as missing parts of functions or functions that the customer wish to have solved in an other way than what's present in the system.

Bad documentation

The third major problem is the documentation. It's very common that the documentation for minicomputer software is of worse quality than documentation for a PC-package. Software written for minicomputers is usualy of a more sofisticated art and includes functions that are more complicated and therefore need a documentation of a better quality than the PC-packages or, at least of the same quality.

Standard software packages

The last major problem is the result of writing standard packages. These systems must be very general in functions, to suit many types of customers, and therefore are very complicated. A result of this is that these kind of systems are very hard to test out in a succesful way. They are also very often parameterdriven, wich result in bad performance when executing the systems. Another thing is that you can't build software to suite every customer, it's imopossible. Almost every customer need some kind of customizing before the software will suite the needs.

How to avoid delivery problems?

This is one of the most dificult problems to solve. It includes all the faces of the developement process, from the product idea to the release. Following can be done to reduce the problems:

Isolated R&D

To keep the systems engineer and the programmers as effectiv as possible, you must avoid to involve then in other problems such as support, customizing and maintenance of older products. Every minute their time are consumed by work outside the project, is very expensiv for the projekt. The hardest problems to handle is errorhandling and support for other products. That kind of work is impossible to make schedules for and is creating a lot af setup time for the mind, every time you have to change to this kind of work and then back to the project. Normally you are deeply involved in the developement project, and it's very timeconsuming to work with something else for a while and than go back to the project and in to the problem you try to solve.

Well-founded timeestimations

When you do the estimation for the first time, usualy to get a frame for the costs for making a decision to starting the project, you can be shore that this estimation is wrong. How much and in which way you don't know, but you can be shore it's wrong.

The reason for this is that you try to make an estimation for a project you don't know the details for. The details is unknown until you start to work with them, why you can't estimate the

resources at this time. Another reason for the estimation to be wrong, is that the estimation often are made up of a special kind of people, the people who work with solving problems. This kind of people have an enormous ability to make things much easier than they really are. They only see just a part of the problem, for wich they have a solution, but they never see the hole problem and a solution for that.

To be able to make good timeestimations you must turn the developing process backwords. You only make a ruff estimation at the beginning, just for the decisionmakers, but you have to explain that this is not a correct estimation, just based on unknown facts. Than you have to get the people to understand that they must made up another decision before the project starts up. Shall the projectgroup have the releasetime as a target or shall they follow the costlimit. This is a very important decision to make. You can't fulfil both of them, you must concentrate on one of them.

After that, you must get a second decision. What's the contents of the projects, i.e. what functions shall be included in the final system. When this description, the **project description**, is ready the work with the project could start.

As a startup-procedure, a timeestimation shall be made up for a description of the functions in the system, based on the project description. Each major function of the system shall be described in an overview level and contain information so you can fully understand how the function is operating in the system. All this work is done using a CASE-tool for having a total overview of the project and the information processes within the project. The CASE-tool shall follow the project in every step, until it's time to write the source-code. Then you also have a good material for making a timeestimation for a detail description of each function. This should explain how to use the functions, in all the different ways, and what's happening in the system when you use each function. You can say that it's a guideline for every and each detail part of the system. When you have done this, you have plenty of material for doing a new timeestimation, in this step for making the detail specification. This step also include screendesigns and reportdesigns, using a 4GL's prototyping abilities. When this is done, a new timeestimation is making up for the next step, which include the rest of the documentation. Then, all the documentation for the system shall be ready, including Functiondescriptions (both overview and in detail), Userguides, Screen- and Reportdescriptions and other part of the complete documentation for this package, without writing a single codeline. You have now, if you are using a 4GL, consumed about 60% of the total projecttime. The rest of the project will include writing the code and testing it.

This method makes it possible to create a very good estimation for the process of writing the source-code. You have now a very good material for doing this in a correct manner. The next big step in the project, is to make a good timeestimation for writing the source-code.

The big advantage for using this method is that it's possible to do realistic estimation for every step in the project until you have finished the step before and you have also created a possibility to stop or continue the project between every step, depending of the result of the latest step. This

method also give high priority to the documentation, sinse the documentation is a part of the complete process and you can't follow this method without writing the documentation since it's a basic document för the next step. It also give the opportunity to validate the functions in the system, in detail, before you put all your money in the project and creating the source-code.

Project-managing tool

Another thing that's importent, to do it possible to delivery the system in time, is a good **project-managing tool**. It's nearly impossible to have a complete control over the developement process without this kind of tool. Today there is a lot of good PC-based tools, who fulfil the demands you can ask for as beeing a god project-managing tool. One of the demands is to be able to optimize the resources involved in the project. Another demand is to get indications about late activities in the project, as soon as possible, to be able to take steps to minimize the effects of a late single activity.

In our company, Teamco Utveckling, we are using two tools, MicroSoft's TimeLine for the smaller projects and PMW, for the bigger projects. The reason is that PMW, the most advanced tool of these two, take a lot of time to work with, which is not so good for a smaller project.

Another reason for using a project-managing tool, is to serve the managing people with "easy to understand" information. The border, for instance, have a lot of interest in big cost-consuming projects and you can, with this kind of tool, currently give them information of the project-status.

Stop increasing projects

Another very common problem for beeing late with the project, is that the contents of the project increases all the time. Under a projects lifetime, there is a lot of new ideas coming up, which the project members like to involve in the final product to make it better. If you don't stop that, the project will increase all the time and it's impossible to match the release date and this will usually end up whith a project that's much bigger than was estimated.

It's much better to save these new ideas (at least most of them) to a second version of the system. It's the only way to be able to release the system on schedule.

How to avoid bad quality?

There is a lot of money in every developement project, which is spent on unnecessary work. The kind of work I mean is correcting errors that never should exist, correcting functions with important parts that should have been a part of the system from the beginning and to add on functions that the user can't affort. These kind of problem, as the problem with late delivery time, is one of the hardest problems to solve.

Make a good overview for the final product

To do this, a CASE-tool is necessary. It's the only way to get a complete control over all the functions in the system and over all the information these functions need for working properly. Usualy a number of systems engineer work out different parts of the system and when you finaly put these parts together, you have missed some importent part for several functions. It's very timeconsuming and expensive to add on missed parts of functions at this time. With a CASE-tool, you have a complete control over the information that every function need and from were the information should be taken.

Create good methods for testing

Try to find the errors in the source-code as early as possible. It will cost a minimum of time to correct an error just after it has been created but it will cost a lot of time to correct an error at the end of the project or, even worse, after the system has been installed for customers.

Build up testdatabases and testroutines that are permanent and always make a complete test irrespective of how small the correction in the programs is. When you detect a new error, you also know that you have missed some parts in the testroutines, testmaterials or testdatabases. Add on material for these new situation to the old material to be shure of that you also testing these part the next time you make a test.

It's also important that the systems engineer do a complete test of the system before you release it. They usualy know how every and each functions should work, much better than the programmer. Another thing that made it possible to find errors before you release the system, is to install the system in two or more alfa and/or beta test sites.

Use the right projectmembers

To avoid that functions within the system is missed or of a bad quality, it's important to use people with the right skill to constructing the system. Within Teamco Utveckling we use a lot of people from "the real world" instead of systems engineer. That's people who have worked in the situation where the new system is going to be used, such as product managers, planning managers and so on. That's the only people who know all the details for the functions and how to use the functions when running the system. Our experience is, that this people make a much better system than a professional systems engineer. The final system is beeing of a much higher quality in each function and the system is more user friendly to work with.

Create the system in functional units

Instead of making the system part by part, it's better to make the system from the functions. Describe each function in detail and when you create the source code, do it from the function level, i.e. implement the specific function in every part of the system where it should be handled. The CASE-tool will help you to find all the parts of the system that is involved for every specific

funtion. Otherwise it's a big risk that you exclude some vital part of the system when using this function and this is very expensive to correct after the system has been released.

How to avoid bad documentation?

There are a few major reasons for the bad quality of the documentation. The most common problem is that the system release is late and that it's no time for doing a good documentation. To handle the problems with documentation, a few simple steps can correct it.

Lack of time

As mentioned earlier, the most common reason is that the system usualy is late and there is no time or money for making a good documentation. The solution for this problem is to follow the instructions earlier about how to avoid problems with the time-estimating.

Use the right people

Reading documentation written by a technician is usualy not as easy as it would be. The technician (usualy the programmer or the systems engineer) don't talk the same lanuage as the enduser. You have to think of, that the enduser maby not have any knowledge about computer systems, just knowing how he's part of the work should be made.

Therefore, it's very important that the documentation must be written by the kind of person mentioned in the part about using the right projectmembers. They must have big knowledge about the customers situation to be able to write a useful documentation. You can't use "computertalk" for explaining complex parts of functions within a system.

It's also important to use a technical writer, not for writing the documentation, the knowledge for that is in the projectgroup, but for making up the layout and for correcting the use of the language. No complicated language should be used. Try to explain things as simple as possible and try to explain the system in a correct way, function by function, not part by part.

Good documentation is equal to quality

Have you think of that the documentation is the only part of the system you can really tuch? Thats true! You can't tuch the rest of the system, except the magnetic tape. If you create documentation of high quality, you also increase the quality of the system, seen with the customers eyes. It's the feeling of quality you pass on from the documentation and it's a simple way of increasing the quality of the complete package, to make the documentation as good as possible.

How to avoid the standard packages problem?

The reason for making standard software packages, is to make as much money as possible on a single investment. But, you create problems at the same time. The systems includes a very big and complicated source-code, which make it difficult to testing out all the functions. Usualy the systems are parameterdriven for a maximum of flexibility, which makes the system very heavy for the computer, i.e. you get bad performance. It's also difficult to implement customer unique functions and to still be able to use standard maintenance for these customers.

How about using 4GL's and making up the system using prototyping? Well, for small systems it will be good, and for big systems used by a few customer. But, for big systems used by many customer it's a problem. You can't build big generic systems using prototyping. It's impossible to handle the maintenance in a acceptable manner. A standard system whitout maintenance is not a real standard system and to avoid these kind of problems, there is some new ideas that we have used in developing a new system generation for the Scandinavian Teamco group, which can give you ideas and tips about how to solve the same problem in your part of the business.

Make the source-code in smaller units

To just make the source-code in smaller parts don't solve any problem, but if you do it in a special way it will solve many problems.

Each module of the source-code should only contain the smallest selectable function of the system. Reading the pricelist will usualy give the right portions of the size of these modules. In general it's much smaller modules than you are used to create, but this method makes it simple (if you are using a 4GL, of cource) to select among already created modules, mixed them together and create system parts. These also makes it possible to create a customer unique system from your standard system, but only containing standard modules, i.e. you can still support the customer with full standard maintenance for the system and can therefor collect the maintenance fee.

This method also make it easier to test the system, since you can test every single module (function) isolated. To do this, you must specify the interface between the actual module and the other modules that are communicating with this, in detail. But, if you do that, you have also created a very good basis for, later on, building customer unique functions and mixed them with the standard system, just concentrate doing the actual interfaces with the other modules correct.

Change the source-code instead of using parameters

The target for us, when we started up our company, was to be able to generate customer unique systems from a standard package. To do this, we have to invent a new process for handling this, called SCOP. The SCOP-process (SCOP i the shortname of Speedware Customizing and Optimizing Package) using the small source code modules and, within the installation process,

pick up, for each customer, from the module stock, the actual modules and mixed them into a customer unique standard system. Then, SCOP can include customized code, for every customer who has who has the need for that, and mixed the customized code with the rest of the system and SCOP are still able to maintain the standard parts of the system.

Let the customer customize the system

Very common is that the customer wish to implement customized parts within the standard system. To do that in the usual way, make it impossible to maintain the standard parts of the system, becuse when you release a new version of a standard part, the customized part are overwritten. But, with SCOP we can handle this problem.

SCOP also contains a module for the customer, where the customer can change a lot of things, like screen-layouts (move fields, delete fields, add fields, making frames etc.), the menustructure and all the texts within the system. Every change made to the system is saved in SCOP-librarys, for making it possible to install a new release of the system and to have all these changes made automatically in the new release.

As a final step in the SCOP process, the source-code (in our case the Speedware specfile) is analyzed and opimized for deleting all unnecessary code for actual customer. This end up with a standard system that has been customized by the customer and with optimized code (only the code that actual customer is going to use is left in the system). Even the databases is optimized as a result of the SCOP-process.

Conclusion

Yes, it's possible to make much better systems, with high quality in both functions and documentation. But, there is no secrets or miracle involved in this. The CASE-tools can't do it alone and neither can a 4GL. It's the complete developement process who make the differens. A good process can prevent bad quality but can't create good quality. That you must do yourself.

Good Luck!

The Case for Information Engineering . . . or

IE or Not to IE, That Is The Question

The CASE for Information Engineering, or IE or Not to IE, ...That Is The Question

Mitchell Kleiman
Ernst & Young
One Sansome Street
San Francisco, CA 94104
(415)951-3000

ABSTRACT

Information Engineering is an enterprise-wide approach to planning, defining, designing, and constructing information systems. This approach emphasizes business analysis, a systems building discipline, and a data structure analyzed independently of applications. In addition, Information Engineering focuses on developing a comprehensive knowledge base and on automation (from design to code generation).

This paper will provide an overview of the four major phases of the Information Engineering process: Information Systems Planning, Business Area Analysis, Systems Design, and Construction. The role of tools in the Information Engineering approach and the critical need for a methodology will be discussed.

Several situations where Information Engineering has been applied will be presented, providing a look at some basic factors that often determine the success or failure of development projects.

WEINBERG'S LAW:

If builders built buildings the way programmers wrote programs, the first woodpecker that came along would destroy civilization.

"Reprinted from Robelle's Encyclopedia of the 3000 - Telephone (604) 888-3666 to obtain a copy."

WHAT IS INFORMATION ENGINEERING?

JAMES MARTIN DEFINES IT AS FOLLOWS:

****
*

- An interlocking set of formal techniques in which enterprise models, data models, and process models are built up in a comprehensive knowledge base and are used to create and maintain data processing systems.

- These techniques depend on highly automated tools. The result is systems built with strong end-user participation, which solve local problems, but which have horizontal integration so that they work together, and vertical integration so that data processing systems are anchored into the top management goals and strategic plans of the enterprise.

- An enterprise-wide set of automated disciplines for getting the right information to the right people at the right time.

THE KEY CHARACTERISTICS OF INFORMATION ENGINEERING

- Strong emphasis on Data Sharing
  Data and structure are analyzed independently from the applications that use them. Data models are used to define business data, satisfy enterprise-wide data requirements, and promote data sharing.

- User-Driven Approach
  Users are involved throughout the development process and interact with graphics models and prototypes.

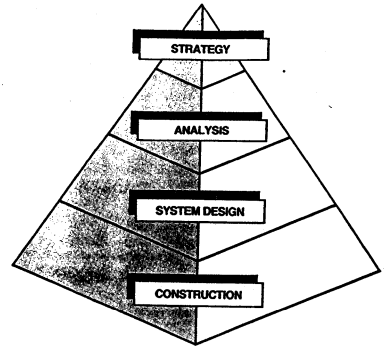
- Business Analysis Focus
  IE is concerned with getting the planning, analysis, and design correct so that application code can be generated automatically.

- Systems Building as an Engineering Discipline
  IE is based on formal techniques and the use of automated tools adds rigor to the development process when they provide for consistency and correctness checking.

- Emphasis on Automation
  Tools are used to store and coordinate the information in a knowledge base for use in all stages of development. In addition, tools can link the design automation to code generators to accelerate the construction phase.

- Graphical Presentation
  Diagrams are used as the primary vehicle for expressing data and activity models, to make communication easier and faster.

- Development of a Comprehensive Knowledge Base
  An important objective of Information Engineering is to organize and store the significant amount of data about the enterprise, organizational and structural details, models, strategies, and plans. Then this information can be shared by the entire enterprise and enhance the effective integration of applications.



INFORMATION ENGINEERING PHASES

I. Information Strategy Planning

This is the initial phase and is concerned with top management goals and critical success factor of the business. A high-level model of the enterprise and its data, functions, and information needs is produced.

II. Business Area Analysis

In this phase the IE professional is concerned with what processes are needed to run a particular business area, how these processes interrelate, and what data are needed. The enterprise model is studied to identify and set the boundaries for a focused area analysis project. The purpose of this phase is to create a logical information model, which then provides an architecture for designing integrated information systems across the business area.

III. Systems Design

During this phase, the requirements revealed in the Business Area Analysis phase are transformed into detailed application system specifications, with heavy user involvement. In this phase the IE professional is concerned with how selected processes in the business area are implemented in procedures and how those procedures work.

IV. Construction

In this phase, the systems solutions defined in the design phase are implemented. Where practical, fourth-generation languages, automated code generators, and end-user tools are to be employed. In addition, in this phase the organization develops procedures for operating the system and users are trained.

Information Engineering:

Environment
Each phase needs all three.

Without Methodology

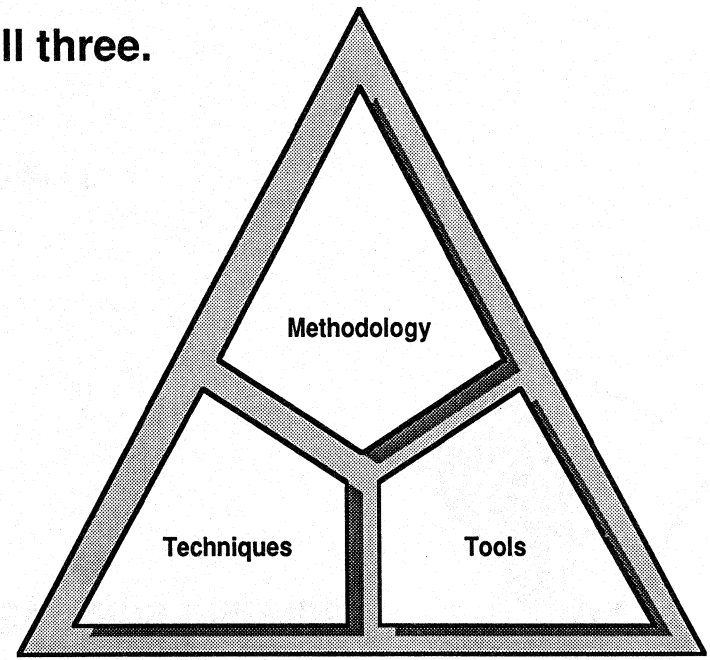
- ... No Structure
- ... No Objectives
- ... No Focus

Without Tools

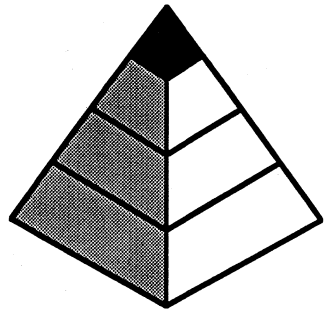
- ... Paper Intensive
- ... No Validation
- ... No Maintenance

Without Integration

- ... Poor Tool Fit
- ... Poor Techniques Support
- ... Poor Methodology Coverage



Strategic Information Systems Planning



Objectives:

Collect high-level information about:

- **the enterprise**
- **its business segments**
- **their functions**

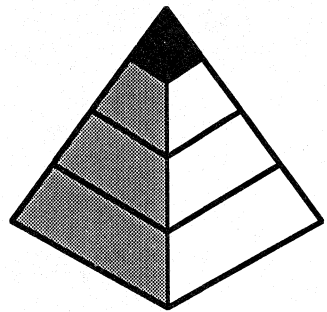
Tasks:

- **Identify Major Classes of Data**
- **Assess Existing System, Technologies & Services**
- **Prioritize Information Needs**

Diagrams:

- **Entity Diagrams**
- **Decomposition Diagrams**
- **Property Matrices**
- **Association Matrices**

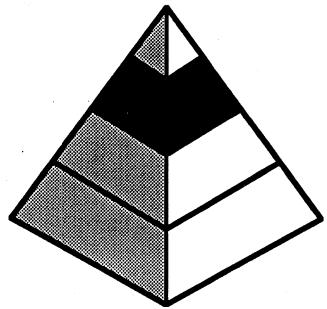
Strategic Information Systems Planning



Deliverables:

- Strategy Description
- Goals and CSFs
- Information Needs
- Organization Models
- Current System Assessment
- Enterprise Model
- Data Architecture
- System Architecture
- Technology Architecture
- Management Infrastructure
- I/S Strategy
- Project Definitions
- Strategic Information Systems Plan

Business Area Analysis



Objectives:

- Develop a logical data & process model of the business area
- Respond to Information Needs identified in Planning Phase

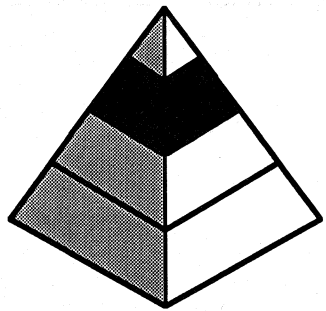
Diagrams Used

- Entity Diagram
- Decomposition Diagram
- Data Flow Diagram
- Process Action Diagram

Deliverables

- Business Requirements
- Business Area Data Model
- Business Area Process Model

Business Area Analysis



Encyclopedia

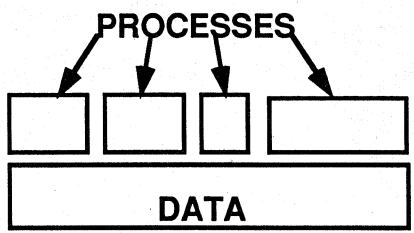
- Each object added & refined enriches the content of the knowledge base.

The Important Word is Business

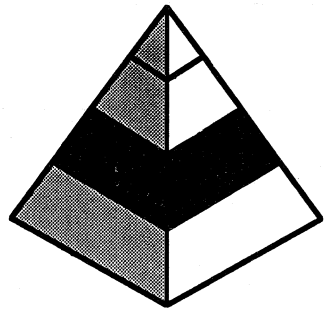
- Done separately for each business area
- More than one may be done simultanously
- Does not design systems, rather attempts to understand & model the data & processes necessary to run the business area

Data vs Process

- Processes change frequently
- Data Models are relatively stable within a business area



System Design



Objectives

- Development of Physical Design Elements ***Based on*** Logical Model
- Some components can be automatically derived but not all

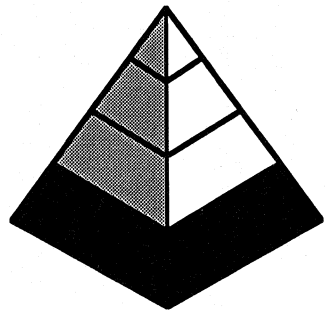
Diagrams Used

- Module Structure Chart
- Data Structure Chart
- Module Action Diagram
- User Interface Layout

Deliverables

- External Design
- Database Design
- Internal Design
- Technology Strategy
- Migration and Conversion Plan

Construction



Objectives

The production of complete, tested business application systems using

- code generation
- 4th generation languages
- decision support tools.

Deliverables

- Coded, Tested, and Installed System
- Implemented Databases
- User and Operational Documentation

BENEFITS OF INFORMATION ENGINEERING:

- Direct link to corporate strategic goals helps senior management more effectively use information technology.
- Identifies how to get the right information to the right people at the right time.
- The coupling of analysis and design automation to code generation yields high Information Technology productivity.
- The engineered solutions of IE are easy to modify at the model level and can be regenerated so that maintenance problems are reduced.
- Integrated applications across the enterprise by managing the sources and uses of shared data.

TITLE: Transformational Programming

AUTHOR: Larry Van Sickle

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5330

SYSTEMS DEVELOPMENT UTILIZING POWERHOUSE

BY
MARK P. SHIRMAN
INNOVATIVE INFORMATION SYSTEMS INC. (IISI)
63 NAHATAN STREET
NORWOOD, MA. 02062
(617) 769-7511

This paper will discuss development projects that rely heavily on fourth generation tools, specifically Powerhouse by Cognos. I am part of an organization that provides consulting services in HP environments, so I have no vested sales interest in the products. I will try to speak from the experiences we have had in this development environment. Additionally, I will address how the classic systems development life-cycle is altered when using 4GL solutions and what are some of the gains and pitfalls with these type of projects. Lastly, I will try to communicate some of the management and project control related issues that sometimes accompany development in 4GLs.

The cornerstone of the application life cycle is the Information Planning step. It is at this time that new and existing technologies are studied and preliminary budgets are laid out. The current systems are reviewed and each project or potential project is addressed in a way that allows resources to be effectively managed. User requirements are typically developed in this stage of the cycle, preferably with a heavy emphasis on business functionality and organizational objectives.

It is my belief that even with the utilization of Powerhouse, this stage should remain relatively unchanged. It will always be of paramount importance to develop systems that are designed to support the basic business operations and the short, medium and long-term business objectives. I have always been of the opinion that regardless of the development environment, that this stage is often overlooked and not given the attention that is necessary. A good Information Plan can help shape management's expectations as to the types of systems that will be developed or enhanced. Since 4GL based solutions are sold to organizations espousing a variety of panaceas from ease of maintenance to speed of development, it is essential that these expectations be brought in line

and tempered with what management expects from the total systems environment. All the positive aspects of Powerhouse development can be brought to bear in the creation of an Information Plan, some of these being:

- Systems stressing business functionality are more easily developed because iterative or prototyping techniques
- Systems can be better and more easily documented
- Systems have a better chance of being delivered on time and on budget
- Control or management checkpoints can still be built into development projects
- Ease of support and maintenance

The second stage of the life cycle is the Design stage. This is where most of the user requirements are laid out and placed into the context of a data base design and their related application programs. In any type of development project this is probably the place that the system's success is often determined. The advantages to the completion of a comprehensive design stage have been widely documented. This may be the phase of development that has been changing the most over the past few years. It is also the stage that is probably most effected by utilizing Powerhouse.

One of the most important tasks undertaken during the design phase of development is that of data base and data dictionary (Qschemac) design. Because Powerhouse can sometimes be CPU intensive, data base design is at least as critical for this type of development as with any in a more traditional environment. Data base normalization techniques should be strictly adhered to. If additional data base utilities are available, such as OMNIDEX, then it is important to consider how these will affect your more classical IMAGE design. We have found that OMNIDEX is an excellent compliment to Powerhouse. The additional functionality that it can add to your Powerhouse applications make it great fit for many situations. However, you need to rethink how such a tool will affect both your data base design and your application programming and this is best accomplished during careful planning in the design stage. Data sets that were originally thought to be IMAGE Details may become Masters in order to accomodate the functionality of the programs as well as adhere to the interfacing of OMNIDEX to Powerhouse. Additionally, because of some of the extra things that a product like OMNIDEX can bring to an application, some programs or sorts may in fact not be necessary. This point

emphasizes the importance of completely identifying your entire development environment as early as possible. Lastly, it may be an opportune time during this design stage to identify any additional utilities that may be necessary to insure the adequate working of the proposed applications. An example of this may be integrating a utility such as Suprtool from Robelle, which can be used to front end large QUIZ reports and programs. This will alleviate having to retrofit application programs and job streams out of necessity at a later stage in the system's development.

It is often during the design stage of development that the individual project's scope and objectives is defined. As with any other type of development, this is absolutely critical so that the project(s) can be successful in meeting organizational business objectives. So that project management can be appropriately set up for all the tasks that are going to come up, it is extremely important that the scope be clearly defined for the Powerhouse development project. This is especially critical because the programming should probably involve more user input than with 3GL projects. If the scope is not clearly defined, the project will keep evolving in such a manner that would be a potential waste of time and money and could negate the productivity gains associated with using a 4GL such as Powerhouse.

One of the beauties of utilizing Powerhouse is that it requires far less application code than with traditional 3GLs. In fact, when using Powerhouse it is usually advisable to keep the amount of procedural code to a minimum and take advantage of some of the efficiencies already built into the language. Because of this, it isn't really necessary to create elaborate structure charts or flow charts to be used for programming. We have found that a basic design packet consisting of: a comprehensive functional narrative, an Input/Output chart, file layouts and descriptions, and any necessary cross-referenced report or screen layouts provides sufficient information for both the programmer and the user/analysts to work off of for application development.

CASE tools have significantly changed the way developers have begun to look at design tasks. A couple of the things that CASE provides an organization is ease of maintenance, comprehensive design documentation, and a structured/ standardized approach to systems development.
With the proper planning, Powerhouse can be utilized to achieve many of the basic things that CASE can offer, often at a fraction of the cost. This may be especially true for

smaller shops that may not have the resources to develop using CASE tools. I do not mean however that Powerhouse is always an adequate substitution for CASE products, but it can provide a cost efficient solution.

With many 4GL packages documentation can be done as the data dictionary and data base is created, and in fact some of the packages currently available will document the system as you create it, ie. Cognos' Architect product. Detail design tasks are often not necessary since it is easy to create screen and report layouts by some simple development and data manipulation in Powerhouse. Ease of maintenance is something that is inherent in most Powerhouse development. Whether this ease of maintenance rivals CASE tools is subject to discussion, however it is true that overall system modification and maintenance is usually easier in a Powerhouse environment.

The last thing that needs to be addressed during this phase of development is to evaluate whether or not each individual application is a good candidate for 4GL development. The best way to utilize Powerhouse is to know when not to use them. The sooner that these programs are identified the better. There are certain types of applications that are simply not a good fit for Powerhouse. This would include heavy table or array maintenance and very intensive "heads down" transaction processing.

The third stage of systems development is the Installation phase. This phase usually consists of detail design, programming, testing(unit and system level), conversion, and implementation. It is another area that utilizing Powerhouse can make a profound impact if handled carefully.

Because it is easy to get the basic workings of the screens or reports up quickly, it makes it possible to get the user community more involved in the development effort. This is also aided in part by the relative ease that aesthetic changes can be made. Getting the users more involved with the actual programming can be a double-edged sword. This is where it becomes important that the project's objectives are clearly laid out so as to avoid "scope creep".

The process we have used in developing/programming with 4GL tools is often referred to as iterative development. It starts with the basic design documents as developed during the Design phase. As programming commences the key user personnel get involved as soon as the basic systems start to

take shape. The little details of the application's can then be worked out elbow to elbow so that the user's have an excellent idea of how the program's will flow earlier than they typically would in a 3GL based system. It is therefore incredibly important that the key user(s) be trained in the operational commands that run in Powerhouse, which are often different than they have ever seen before. The advantage of getting the users involved earlier in the process allows them to think they are closer to the technical developments as well as allowing them to take some pride of ownership in the overall development process. The speed of the iterative process also serves as a motivation to the user community that is often used to systems typically being developed in a vacuum. This will help sell the system to the organization as a whole.

The programs should not be coded in random order. The key data maintenance programs and the key transaction based programs should be started first. This will allow the test data environment to be set up early to facilitate better user involvement for training and assistance in the iterative development effort. The reports and inquiry screens as well as any non-essential utilities should be left to last.

There are some pitfalls to this type of development in the programming phase. There is usually a tendency to get the first 90% of the application completed on time on budget and sometimes even way ahead of schedule. But oh that last 10%, it can really kill you. It often contains the logic that pushes Powerhouse to the max or may be the concepts that are most difficult for the programmer to grasp. It is the last 10% that can make the difference between a successful project and a project that is over budget and beyond time expectations. One of the advantages of using a 4GL is that you don't always have to have technical geniuses doing all your programming to be successful. What we have noticed is since these programmers made such rapid progress at first, it can be very distressing, even depressing to an individual to toil over the last bits and details. Therefore the last 10% of each application and the unit testing must be closely monitored and quality assured.

We don't believe that just because you may be using productivity tools such as Powerhouse, testing should take any different tact than in traditional development environments. I spend a lot of time lecturing to various groups on the finer points of unit and system testing. We employ the same type of rigid testing discipline when

programming in Powerhouse, however I thin it may be easier to employ. Powerhouse has a batch input process(QKECHO) that allows you to simulate multiple users as well as enter in the same test data many different times. They allow you to capture all potential input data in a flat file and then play your application against that data many times. This takes a lot of the tedium out of entering test transactions and makes it easier to repeat test cycles many times over. Additionally, with QUIZ, it is easier to create a series of ad-hoc reports that can be used to test before and after test results.

The conversion process can also be aided by utilizing Powerhouse. Routines utilizing QTP can really assist by converting large amounts of data with a minimum amount of programming. The beauty of using this type of approach is that although it may sacrifice a little bit of processing efficiency, it will usually insure that IMAGE rules and element definitions are adhered to. By removing the amount of time it takes to create conversion processes it frees up more time to concentrate on the actual application development. Since these are programs that typically will only be used once or twice, it shouldn't matter too much that they may not be as efficient as COBOL or 3GL based programs.

The final phase of classical development life cycle is support. If the systems are appropriately designed and documented(granted no small feat), then this is where Powerhouse can really excel. Maintenance is typically easier because there is less volume of code. Changes can be made with a little less effort, especially those that relate to aesthetic issues. The Architect will allow developers to keep their documentation up to speed as changes are made. This products allow for cross-referencing of files as well as facilitate some sort of modification history. Keeping up on the support tasks and documentation will make it easier to migrate to new releases of the software as well as new architectures ie., RISC, when the time comes.

By utilizing Powerhouse for systems development a lot of gains can be realized and time can be shaved from the overall process. Many of the phases of the classical systems development life-cycle become melded together, without sacrificing quality or audit/control trails. There are pitfalls and these need to be built into both your project management and the politics of each project. Management may expect things that aren't always in the best interest of the

overall system(so what's new) and programmers may be more apt to utilize some of the inherent shortcuts available in most 4GL's. Careful planning and adherence to strict design standard can make a Powerhouse project a successful and rewarding situation for all concerned.

ASSESSING AND IMPLEMENTING AN END USER REPORT WRITER

Written by Alan Munday
Presented by Rusty Babcock
Infocentre Corporation
5674 Stoneridge Drive, Suite 218
Pleasanton, California 94566
415/460-8220

When a company has a small number of staff in the Data Processing department, relative to users, and end users constantly requiring new reports, there will inevitably be a backlog of outstanding requests. This means that the users will have to wait an unacceptably long time for the information that they need causing frustration for the user and a siege mentality on the part of the computer programmer.

The DP staff during a heavy period of development, may not have the time or people to donate to the task of writing the report programs that the users require. This is not providing the service that a DP department should.

The way to avoid this problem of course, is to let the users write their own reports. Doesn't this open up a can of worms? Well potentially, yes it does. there are numerous problems that may arise in the minds of both the user and the Data Processing Manager as to the practicality of end user reporting. Let us consider the possible points of view of each of these individuals beginning in this first part with the user.

Time

The end user has a workload that currently does not include writing reports. Indeed, it may not include any computer usage at all. The first problem therefore is one of time. Time to learn the report writer; time perhaps, to learn how to use a computer terminal, to learn computer fundamentals, to learn the concepts of data storage and database structure and then to remember it all. So far the user has not allocated the time to actually write the reports themselves. This is quite a learning process for a person not employed by the DP department. The implementation process then must see to it that the learning period is short. This can be achieved by ensuring that the software is sufficiently easy to learn and that the primary users of it already understand the fundamentals of the HP3000 (terminal use and logging on at least).

It may seem that the cart is preceding the horse when speaking of implementation at this stage but looking at how the user will most likely respond to the software will help in the selection process.

User Programmers?

Once the learning period has passed, the user has to know how long it is going to take to actually write a report and this begs an important question. Suppose that the user continues as usual and submits a request for a report to the DP department. A programmer will eventually write a report program which the user will then use. It therefore follows that now the user has to learn a new skill; that of computer programming, right? Strictly speaking, yes they will. Or at least a computer program will be the result of their actions. Without getting into a corner regarding what constitutes a computer program, the user should be presented with software that by a selection of choices on the user's part and very little coding as such (if any) a computer program can be generated. There are companies where the users have been allowed to use a report writing programming language and some of these users have become quite adept at writing the kind of reports that they need. On the whole though, this is not a satisfactory solution since only a small number of users will bother with this kind of product because of the amount of syntax involved, both in the learning of it and in the likelihood of omitting or mistyping something.

Report Requirements

As an extension to this, we have to look at the type of reports that will be written. What does the user want in a report? Is it likely that the report will always be a listing of data held in files or is there a need for calculations to be performed. Will the user be creating programs that print invoices, cheques, labels and other "special stationary" output and will there be a requirement for data to be downloaded to a micro for input into a micro software package. Will the users require that the output be used to create graphs and if so, how easy is it to interface with a suitable graphics package.

The reality of the situation is that it depends on the company, the individual departments within the company, the amount of rein that the DP department and department heads allow and the overall departmental or company requirements. The very first requirement has to be that the user should be

able to write a report quickly on existing data. This is usually the most common report and is the basis of all other types of report. In implementation, it is amazing how pleased the users can be when they realize that this basic ability is within their control. This is the starting off point for all users and having grasped the means of creating a basic report, they can then build on this to include totals, calculations, prompting, selection criteria, control breaks and all the things that provide the users with a meaningful report.

User Evaluation

It is important at this point to refer to the software selection process. To evaluate the case by which users can grasp the basic functions of reporting, a user has to be involved in the selection procedure and should be allowed to "test drive" the software, preferably at the time the salesman installs the demo version on the computer. This allows the salesman to quickly provide the answers to any questions that the user may have to help start off the process in as smooth a way as possible and also give the user the confidence to progress once the basic report it written.

If a user is provided with the software, a manual and the ability to log on without the benefit of having someone with experience of the software at hand, the seeds of frustration and possible rejection of the software may be sown. Of course, On-line Help and Documentation has to be an integral part of the package but what the user requires is the initial confidence to write that first report and the assurance that they won't "break" anything or erase any data.

Implementation

On implementation, some users may be reluctant to use the software for one reason or another. It may be that they haven't had the benefit of the "easy" route into its usage and will therefore not have the confidence to begin alone. This is to be avoided and the way it can be avoided is this. The user selected to be the first "test driver" of the software will typically be one who is reasonably computer literate, who has a positive attitude towards computers in general and knows the kind of reports that they want to write. This user has been involved in the salesman's demo and from that point on is the "key" user and primary contact with the DP department.

Any problems that this user has with the software should be addressed by a "key" contact in the DP department. The key user is the one who takes the software and "runs with it" putting it through it's paces and reporting any problems to the key DP contact.

Once the key user has produced enough reports to feel at ease with the software, they would then help another user to begin using it beginning from the basic report creation and building confidence on several "listing" reports. This next user should also be reasonably experienced in computer/terminal usage. This process should continue until all the users requiring the facility to write their own reports have been trained by their colleagues.

What is to be avoided is the situation where the end users are told that they can all begin using the report writer at exactly the same time after having had a brief run through by a DP person. This will cause headaches for the users who need to know how to use the software for their own particular reports then go to the DP dept. who will be deluged with queries all at once and the benefits of the report writer are, for the moment, lost. This situation will create a bad attitude towards the software from almost everybody and there may be a reluctance to use it in the future.

User Acceptance

There will be some users who take to report writing easier than others. The slower ones would usually be those with little computer experience and the idea of being taught by another user is especially attractive since they may feel more in common with another user rather than a "computer person".

Some users need special attention in using the report writer. There have been occasions when a user has approached someone in the DP department and asked if a report could be written for them using the installed end user report writer. The response should be, "no, but I'll sit beside you while you write it and help you if you get stuck". Once this approach is taken, the user will know that help is available if required. The point though, is that the user acquires the confidence to write the next report or the one after that. The DP involvement should decrease as user confidence increases.

There will be some users, of course, who will refuse to use it at all for a variety of reasons that may have little to do with the actual software itself. This will occur regardless of whether the other users are using the report writer or not. They still should be able to get the report they need by the old method of the submission of a request to the DP department. They now have the choice of waiting for the long turnaround time that this incurs or by doing it themselves and getting the report done sooner. The practice of going to another user and getting them to do it for them should be actively discouraged.

Tuition

There is no question that the best kind of introduction to a product is a presentation by the software company. This should occur once the software has been purchased and once the "key" user and DP contact have used the product enough to feel comfortable with it. The tuition should consist of an initial overview of the product's concepts, a guided tour through the product starting with the "simple" type of report and progressing into more advanced concepts and the time to allow the attendees to get some practical experience using some test data. The implementation begins after this, user by user as prescribed earlier. The instructor's role is to explain what can be achieved by the product and how the user may find their way around it to create the type of report they need.

This instruction is important to show the potential of the product. If this was omitted and user learning relied exclusively on the "key user" method of teaching, certain features may be overlooked if the "teacher" had no previous need to use them. Everyone down the chain of tuition would then find that the knowledge that they may require has not been shown to them and the next person up the chain, also lacks that knowledge. Tuition before the process begins will show everyone what is available so there is less chance of features being universally overlooked or forgotten.

Problems

A final point to finish the user aspect of end user report writing. There will be times when a user has a problem and cannot find a way to achieve the desired result. To whom does the user turn? The user should first of all go to the other users for the simple reason that they are the ones who use it most so they may have encountered the problem previously and solved it. The "key user" may have encountered this before or if not, the DP contact. If this is found to be a previously unknown problem, then the software companies Help line should be called. Initially, the user should be the one who calls because the problem can then be explained first hand. There may be times though when the Help line person needs to speak to the DP contact. The reasons why this may happen will be apparent in the next part when we look at the DP Manager's point of view.

The Data Processing Side

When an end user report writer is to be introduced into a company, it is potentially a DP Manager's nightmare. The idea of many users writing their own reports and thus reducing the backlog of requests is the major benefit for the DP section but this is balanced out by the performance problems that may easily occur. The assessment of the report writer by the DP department has to focus on the safeguards (or lack of them) that the software contains so that its introduction and use is not detrimental to other computer tasks.

Another factor that the DP "contact" has to consider is one of introducing a programming tool to people who have little or no computer experience. What training will be required from the DP section, how much do you have to explain to the user about computer concepts albeit simple ones in order for them to begin to use the product and how much involvement is required on the part of the DP section. This should all be discussed on assessment of the product and the software representative has to give an idea of the implementation process on presentation of the product.

The main criteria to look for in the product should be; will it be easy to use for the end use and will it do what the end user wants? As mentioned in part one, an end user with a good attitude and who is quite comfortable with computers should be involved as soon as possible in the assessment of the

product. Obviously, on first look, there will be some purely

DP considerations that the software representative and the DP contact will want to talk about, but after this, the user should be brought in.

DP Preparation

Before we look at performance considerations, lets see what is required by the DP contact by way of preparation for the users to use the report writer. For the duration of implementation, one DP member of staff should be responsible for the report writer to make sure that the users are able to begin to use it in the smoothest way possible. This should not be a permanent dedicated assignment, but the users will need support from someone who knows computers initially. A report writer will not usually be the kind of software that can be put on to the computer and immediately given to the users. The DP contact will have to have a healthy appreciation of the process involved and what the ramifications might be. This person should be one who knows the HP3000 well, who knows IMAGE well and who knows the applications used by the company. This is because the users know the kind of data they need and it is up to the DP contact to make sure they can access it, but using their own terminology.

This DP contact has to be able to see both sides of the fence and interpret computer concepts to the users when necessary. The more explaining that has to be done though, the less likely the users are to accept the report writer. The software has to be easy to use enough to require minimal explanations of computer concepts. What is required of the software and what is required of the DP person so that the user can get on with report writing?

The user has to be told how to log on, how to use a keyboard, how to run the software and the concept of files. This can be done by the DP person providing they can refrain from using computer terminology. This is a sound base on which the user can work. Given this information, the user can concentrate on learning the product. In part one, a distinction was made between a report writing programming language and an end user report writer. The latter we said, should be one where the user can select options to write the report and code as little as possible if at all. Let us look at a problem that the end user would normally be presented with, but which the software should, with the help of the DP contact, have a solution for. This is one of accessing the data itself.

Getting at the Data

Data can be held in an IMAGE database, a KSAM file or an MPE file, all of which the report writer software should recognize. Looking at an example, let's suppose that we have a database called ORDER which as a read password of READ and a customer master dataset (which is a manual master) in it called CUST-MAST. This dataset has a customer number or CUST-NO which is the key, a customer name field called CUST-NAME and an address array of 4 lines called CUST-ADD. It also has a date field called DATE where the date is stored in YYYYMMDD format and a credit limit numeric field called CRED-LIM. The user needs to write a customer master listing, which anyone can understand, using all of these fields. What does the user need to know to access this data? Clearly, all of the above. This is a lot to absorb for a user with little computer experience. what is required so that the user can access this data easily?

The user should not need to worry about the database name and password. The only concern should be knowing which file holds the relevant data. But what about security? Surely, the authorized users need to produce a password of some kind? The report writer software should already have built in security and the user doesn't need to know the IMAGE password (as they don't on the usual programs that they run).

Supposing there are a number of databases on the computer and the user needs information from all of them. As before, it's the files that count, not the database so providing the user can uniquely identify the information by the file name, there is no problem. Ah, but supposing in two databases there is a dataset called GLMISC-EX and the user knows the file name but not the database name. Far be it for me to question whether it was a good idea to name them the same in the first place but how can the user select the right one. Well, the answer is of course, change the names so that the user can now see that the two files are different and while doing this, make sure that the files are given names that are meaningful to the users.

But this can't be done can it? What about the logistics, the hassle involved, all the other programs that would need to be changed and you call this a benefit?

Supposing the report writer itself provided the ability to present the files to the users with unique and easily identifiable names and thus negate the need to change the actual names of the datasets. This is a much more acceptable situation. Going back to the customer master example, although CUST-MAST is a reasonably descriptive name for programmers, it's definitely a "computerese" name for end users so maybe a different name should be used such as CUSTOMERS or CUSTOMER FILE or CUSTOMER INFORMATION (note no hypens). This would guide the user to the correct file simply because the name describes the contents.

Having chosen the file to report on, the user must decide on which information is to be selected from the file. Again, we run into a case of "programmer friendly", "user unfriendly" item names. So what happens now? How about repeating the exercise and changing the item names this time so that CUST-NO becomes CUSTOMER NUMBER, CUST-NAME becomes CUSTOMER NAME, DATE becomes DATE ADDED, CR-LIM becomes CREDIT LIMIT and CUST-ADD? CUST-ADD, as described is an array. End users don't recognize arrays. The other items are single pieces of information but an array holds a number of pieces. Each piece has to be separated out and appear no different in storage to the previously defined items. Therefore, instead of changing CUST-ADD to CUSTOMER ADDRESS LINE 1 and CUSTOMER ADDRESS LINE 2, it can be changed to CUSTOMER STREET and CUSTOMER TOWN. The end user does not now have to distinguish between array and non array items. They are all individual pieces of information.

The software should provide this ability so that users do not have to decipher the name or guess at the information held in a file or item. Obviously, this may be time consuming for the person who has to change the names but it need not be painful. As I mentioned before, the choice of DP person should be one who knows the applications and therefore should know or be able to find out, the data that the application uses. Providing the dataset and item names are changed when each user's data requirements are assessed and not changed initially database by database, it need not be quite so tedious a task. If all file and item names are changed at one go, it might be a very long day.

As you can see, this method of data selection is very much one of "pick the file, choose the items, create a report". This is the simplest route for the user to access the data that is required and providing the file "names" and item "names" are there to be scanned by the user. There shouldn't be an inordinately large learning curve for the user to write basic reports. All well and good if a basic report based on one file is all that is required. Suppose though, information is required from several data sets to be put into a single report. What is required of the software and of the DP contact in setting this up?

Multifile Access

The software must have the ability initially to allow reports to be written on more than one file. For instance, you might want to produce a listing of the day's orders taken by the Order Entry department and the report requires some information from the Customer Master file. Should the user have to know that the information comes from the Order Header file, the Order Detail file and the Customer Master file? Not only that, should the user be required to know the relationships between these files and therefore a sound knowledge of IMAGE? An understanding of Keys, Chains and Paths should not be mandatory for a user to write reports.

The DP contact should be able to provide the information that the user needs and in conjunction with the software, allow the user, very simply, to access information from related files AS THOUGH THEY WERE ACCESSING ONE FILE. This is the simplest method for users to retrieve data from more than one file. The relationships should be defined in the software by the DP person beforehand. This might be termed the "second level" of preparation for the users after initially setting up the individual files. Again, this should be approached, user by user, rather than trying to anticipate all users needs at one go.

Security

A potential problem that the DP Manager or department heads may foresee is that of security. If users can access data outside of the programs written for them, what stops them from accessing any data they like, especially if one of the files that they can choose to write a report on is called PAYROLL

INFORMATION. As we said earlier, all the files would have been given recognizable names so that the information is easily identifiable. First of all, lets see how to separate and uniquely identify one user from another.

The most obvious method that MPE provides is the user log on ID. If Sally in Personnel logs on as SALLYPER and Fred in Marketing logs on as FRED.MKT there is an easy distinction. Even if there is another Sally in the Personnel department, the use of initials or log ons such as SALLYB.PER as opposed to SALLYM.PER could be used to distinguish between them, both, of course, with their own user passwords. The report writing software must be able to distinguish the users from each other by the same method. If this is so, the security has to be such that the files should be available either to one user, or the other or, when necessary both. This means that when SALLYM from Personnel wants to write a report, the files that will be presented to her will be the ones that the DP contact has specifically assigned to her or her department or any other group of users that she is a member of. Lets look at an example.

On the computer there are a number of files; CUSTOMER INFORMATION, ORDER HEADER INFORMATION, ORDER DETAIL INFORMATION, SALARIED PAYROLL INFORMATION and PERSONNEL INFORMATION, SALLYB and SALLYM from Personnel are allowed to write reports only on SALARIED PAYROLL INFORMATION and PERSONNEL INFORMATION. DAVE from the Order Processing Department can access CUSTOMER INFORMATION, ORDER HEADER and ORDER DETAIL INFORMATION but not the Payroll/Personnel files. When Dave wants to write a report and lists the files that he can access, he should only be able to see the files assigned to him, not the personnel files. He may, in addition, have a file grouping of all three files that he can access as one file (Multifile access).

An extension to this should be that although the two Sallys can access both of the files allocated to them, only SALLYB as personnel manager should be able to see the field in the SALARIED PAYROLL INFORMATION file that contains the employees salary. This should not appear to SALLYM at all. Taking this a step further, suppose we allowed SALLYM access to the salary field, SALLYB as department head may choose that SALLYM may have access to the salary field only when the figure in that field is less than 15000. This should be permissible in the report writing software and would be set up when the DP contact is bringing the personnel department into the report writing environment.

Performance

Let's now look at the performance considerations. Assuming that the report writer software is not a system "hog", what should a DP department be wary of?

The user wants to write a report on a file. The file happens to be 200,000 records in length. The DP manager's point of view would probably be "we don't want users clobbering the system with long online serial reads at 10:30 in the morning". This is a reasonable comment. The users shouldn't be able to do this at 10:30 in the morning. How about 10:30 at night though when there are no online users around. Why not run the report then? Who's going to run the report then though? The user, of course, given that it's run in batch with a start time on it. Why would the user run it in batch rather than online if they think that with online they would get the report almost immediately? Because they should not have the option of online or batch when reporting on a large file or when the report is likely to take a long time.. The DP department should be able through the software to control each users report writing environment and allow or disallow the ability to run reports at certain times, especially if the reports are large ones.

Control

Given this ability, the DP manager should feel much happier in having an end user report writer installed on the computer. The ability to write and run reports by the user should occur in a controlled environment, controlled by the DP department. The software should be able to work out whether the file is large, whether the report will take a long time and therefore whether running the report immediately is permissible. This would allow users running smaller, faster reports to run them when they require them and not have to wait.

Can you see what has happened here? the idea of a user-programming tool is changing into one of a controlled environment where the report writing process is initially defined and continually controlled by the department managing the computer's resources so that users can write the reports they require and run them INSIDE PREDEFINED BOUNDARIES. They are not let loose to wreak havoc on system performance. They are allowed to work within the DP department's control but not under its constant vigilance. The boundaries are initially set and once set, the DP presence should be required less and less.

There should be no fear in implementing an end user report writer providing the Dp department holds the reins. Eternal DP involvement is to be avoided if the report writer is to be a success though correct preparation is essential in creating the proper environment. The ability to define the information requirements of each user and give the report writing ability to them, user by user is the easiest and least troublesome method of implementation and I can't recommend this strongly enough. Report programming backlog can be substantially reduced by DIY end users given the proper tool. Be sure you choose wisely.

QUIZ systems functions and midnight follies

By Asher F. Tunik - Computer Task Group Inc. (CTG)
One Commerce Center, Suite 1000
12th and Orange Streets
Wilmington, DE 19801

This presentation is prepared for those individuals who are familiar with QUIZ, and generated some, not very complex, reports in the last six to twelve months. Discussion will include use of some of QUIZ functions.

Let us consider a scenario:

It is 2:00 am. You had a great day at work, wonderful evening, your favorite sports team won all the games this season. Are you a happy person. I bet you are!... Let us add a little twist to the situation. Phone rings... You slowly open your eyes and glance at the lighted dial on your night stand... Then very slowly you turn on the light and with the angrily-roaring yawn you answer the phone. You friendly night operator on another end of the phone line, tells you cheerfully that usual reports did not print today for some reason. It is his time to go home! Hurray, Adios, Chiao, Ta-Ta... Are you a happy person at this moment?

This particular example is a part of this presentation just to illustrate the point that grief and happiness should be measured in the same units of measure (whatever those units are).

Cursing the day that your parents went into the delivery room X number of years ago, you get into your VW and drive to work. The moment you see the standard list from the job that was supposed to generate those reports, you realize that job which normally runs at 6 pm, did not log on until 00:02 am and relying heavily on the system date, instead of picking up yesterdays date for the daily processing it picked up today's date. It is 3:00 am, there was no processing done and therefore there are no invoices. You modify your job to pick up the correct date and go home.

However, being determined dataprocessor, you cannot leave your job stream the way it is now. So what are you going to do? You can try to handle the situation procedurally. Tell the operators that this particular job must run at 6 pm. In return they will ask you, what are they to do if machine goes down for some reason or other...

Well folks, it does not get any better, does it?!

So you decide to write a small routine, using any language of your choice, that will take the system's date and make a determination what date to use for the processing, output that date into the flat file and do the daily processing using that date from the file that you created.

When I was facing with that task, I decided to use QUIZ. Quiz is a fourth generation language from COGNOS. Deciding factor in my selecting QUIZ was ability to handle date calculations fairly easily.

I figured out that if report job runs at any time between 5 pm and 5 am during week (Monday, Tuesday, Wednesday, Thursday and Friday nights) my job is running at night after day's processing and I should process this day's transactions. On another hand if my job runs between 5 am and 5 pm, there were probably some problems with the system or some reports did not come out accurately, and therefore I need to process transactions from the day before (On Monday day before is Friday, because usually there is no work done on Saturdays or Sundays). If my job runs on Saturday or Sunday, it probably needs to process transactions that happened on Friday.

Next thing I asked myself was: How can I find out what day of the week and what date it is now. Then I remembered that HP setup system-wide JCW's for that purpose. JCWs HPYEAR, HPMONTH and HPDATE are the JCWs that will hold integer values for the year, month and date. So if HPYEAR = 88, HPMONTH = 09 and HPDATE = 01, current date would be 09/01/88. Another two pieces of information that I needed, was day of the week and hour within the day. Well there are corresponding JCW called HPDAY and HPHOUR. Value of HPDAY will 01 on Sunday, 02 on Monday and so on with Saturday being 07. HPHOUR shows value of the current hour. If it is 17:43 pm value stored in the HPHOUR will be 17.

In order to import values of the above JCWs into the QUIZ routine, I had to use system function called "JCW". That is what QUIZ manual tells us about this particular system function:

> JCW(string)
> Returns an unsigned integer in the range 0 to 63535 inclusive, representing the value of the JCW whose name is specified by string. If the named JCW does not exist, a function result of -3 is returned.

This is how my routine looked like:

```
!
!       Job control statements preceding QUIZ date calculation
!
!QUIZ

SET REPORT LIMIT 1
;
;       SET UP THE CALCULATION VARIABLES AND CALCULATE
;       THE CURRENT DATE
;
DEFINE TD-YEAR   ZONED*2 = JCW("HPYEAR")
DEFINE TD-MONTH ZONED*2 = JCW("HPMONTH")
DEFINE TD-DAY ZONED*2 = JCW("HPDAY")
DEFINE TD-HOUR ZONED*2 = JCW("HPHOUR")
DEFINE TD-DATE ZONED*2 = JCW("HPDATE")

DEFINE TODAY-DATE ZONED*8 = 19000000 + TD-YEAR * 10000 + &
                                       TD-MONTH * 100 + TD-DATE
;
;       NOW LET US FIGURE OUT WHAT DAY OF THE WEEK IT IS
;
DEFINE MONDAY-FLAG  ZONED*1 = 1 IF (TD-DAY EQ 2)              &
                              ELSE      0
DEFINE WEEKDAY-FLAG ZONED*1 = 1 IF (TD-DAY EQ 3)              &
                              ELSE      1 IF (TD-DAY EQ 4)    &
                              ELSE      1 IF (TD-DAY EQ 5)    &
                              ELSE      0
DEFINE FRIDAY-FLAG   ZONED*1 = 1 IF (TD-DAY EQ 6)             &
                              ELSE 0
DEFINE SATURDAY-FLAG ZONED*1 = 1 IF (TD-DAY EQ 7)             &
                               ELSE      0
DEFINE SUNDAY-FLAG ZONED*1 = 1 IF (TD-DAY EQ 1)               &
                             ELSE      0
```

```
;
;       NOW LET US FIGURE OUT DATE OF THE PREVIOUS CALENDAR
;       WORKING DAY. ON TUESDAY, WEDNESDAY, THURSDAY, FRIDAY
;       AND SATURDAY PREVIOUS WORKING DAY WILL BE PREVIOUS
;       CALENDAR DAY.
;       ON MONDAY AND SUNDAY PREVIOUS DAY WILL BE PREVIOUS
;       FRIDAY'S DATE.

DEFINE PREV-CAL-WORK-DAY ZONED*8 =                          &
       DATE(DAYS(TODAY-DATE) - 1) IF WEEKDAY-FLAG = 1       &
ELSE   DATE(DAYS(TODAY-DATE) - 1) IF FRIDAY-FLAG = 1        &
ELSE   DATE(DAYS(TODAY-DATE) - 3) IF MONDAY-FLAG = 1        &
ELSE   DATE(DAYS(TODAY-DATE) - 1) IF SATURDAY-FLAG = 1      &
ELSE   DATE(DAYS(TODAY-DATE) - 2) IF SUNDAY-FLAG = 1

;
;       NOW WE HAVE ALL THE INFORMATION WE NEED TO FIGURE OUT
;       DATE OF THE PREVIOUS WORK DAY
;

DEFINE PREV-WORK-DAY ZONED*8 = TODAY-DATE                   &
             IF (TD-HOUR GT 17 AND WEEKDAY-FLAG = 1)        &
ELSE     TODAY-DATE                                         &
             IF (TD-HOUR GT 17 AND MONDAY-FLAG = 1)         &
ELSE     TODAY-DATE                                         &
             IF (TD-HOUR GT 17 AND FRIDAY-FLAG = 1)         &
ELSE     PREV-CAL-WORK-DAY IF SATURDAY-FLAG = 1             &
ELSE     PREV-CAL-WORK-DAY IF SUNDAY-FLAG = 1               &
ELSE     PREV-CAL-WORK-DAY IF TD-HOUR LE 17

SET SUBFILE NAME DATEPARM

REPORT SUMMARY PREV-WORK-DAY

GO
EXIT
!
!COMMENT       AND SO ON.....
```

In addition to the system function JCW(string) (please note that name of the JCW itself in the "string" parameter must be in quotes), we used two other functions DATE and DAYS. That is what QUIZ manual has to say about these two functions:

DAYS(dexp)

Converts a six-digit YYMMDD date to the number of days since Jan. 1, 1900 inclusive.(Jan.1, 1900 was a Monday). Converts an eight digit YYYYMMDD date similarly if the century prefix is 19 or above. Converts an eight-digit date with a century of less than 19 to a negative number representing days before December 31, 1899.

DATE(nexp)

Converts the number of days since Jan. 1, 1900 to YYMMDD date or a YYYYMMDD date. If CENTURY INCLUDED option is active, converts a negative number to a pre-1900 YYYYMMDD date.

As you can see these functions perform totally opposite calculations. That enables you as a user to calculate any date to number of days from 01/01/1900, add a number to result and convert it back into the date format.

Bibliography

1. QUIZ manual for HP-3000 MPE. published by COGNOS in 1986.

TITLE: Agency Access - From Vision to Reality

AUTHOR: Tom Rayner

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5407

How to use VIRTUOSO to Write an Application

Patrick J. McCombie & Leah Williams
Hercules Incorporated
Post Office Drawer 1517
Brunswick, Georgia 31521
912/265-3550

ABSTRACT

Hewlett Packard has developed a powerful code generator, VIRTUOSO.

A generalized rule-base methodology is developed which allows individuals unfamiliar with VIRTUOSO to create a functioning program. The methodology for using the VIRTUOSO code generator in conjunction with Hewlett Packard's system-wide data dictionary, SYSTEM DICTIONARY is discussed. The presentation provides a functioning example of this approach.

INTRODUCTION

VIRTUOSO is a code generator. At present, the only language supported is COBOL 85, although there is no limitation to the languages which can be supported. The COBOL 85 code produced will compile without error (although it may not perform as intended). The key components used by VIRTUOSO are: a module, the type of element used to define a program in SYSTEM DICTIONARY; a macro, a reusable program function; and a model, a reusable program.

At present, VIRTUOSO has, as part of the product, a sample model and a sample macro library. Models normally perform one activity (such as maintenance to an IMAGE detail data set) and are comprised of references to several macros and some special VIRTUOSO code constructs (used primarily for referencing the SYSTEM DICTIONARY).

Macro libraries are groupings of functions similar in the sub-system accessed. An example is the IMAGE macro library, that contains functions to read a data set record, lock at the specified level, etc. In all cases, the code is generic and specific information is either passed from the calling model or obtained from the SYSTEM DICTIONARY.

One macro library is used to extract information about the program from the SYSTEM DICTIONARY. This allows the program environment (data sets accessed, screens displayed) to be defined in the SYSTEM DICTIONARY. Note that a dictionary relationship is what a CASE, (Computer Aided Software Engineering) analysis and design tool would produce.

VIRTUOSO is a very powerful tool. Provided that a model and the necessary macros exist to perform the program's function, all that should be required is building the definitions and relationships in the SYSTEM DICTIONARY. Include files may also be created that are automatically added to the generated code - entry points are available that allow the code generator to look for an appropriately named file to include. This provides a method of adding specific code, such as a calculation, or an action to perform when a function key is pressed.

In reviewing Figure 1, you will see that there is one item missing from the documentation provided by Hewlett Packard. If VIRTUOSO and SYSTEM DICTIONARY are to become an integral part of a CASE environment, then rules must exist for using the two components together. The VIRTUOSO COBOL Sample Library discusses the general approach, but does not stress the inflexibility.

Hewlett-Packard Manuals to Support Virtuoso	
30189-60001	Virtuoso Programming Environment, Student Kit
30423-60001	Virtuoso Reference Manual
30426-90001	Virtuoso COBOL Sample Library Reference Manual
32245-91001	HP System Dictionary Self-Paced Customer Training
32254-90001	HP System Dictionary/V SDMAIN
32254-90002	HP System Dictionary/V Intrinsics
32254-90003	HP System Dictionary/V Utilities
32254-90004	HP System Dictionary/V General Reference Manual Volume 1
32254-90005	HP System Dictionary/V General Reference Manual Volume 2

Figure 1

There is only **ONE** way to **properly** define an item in SYSTEM DICTIONARY for use by VIRTUOSO - any deviation, even by use of an innocuous alias, is disastrous. The relationship between the two products is rule-based.

What follows is a discussion of the rules for the elements included in version A.00.00 of VIRTUOSO.

USE OF VIRTUOSO WITH SYSTEM DICTIONARY

In general, the VIRTUOSO code generator is executed in the following manner:

```
FILE SYSDIC=SYSDIC.group.account
FILE GENMACRO=macrolib.group.account
FILE GENTEXT=model.VIRTUOSO.SYS
FILE GENOUT=program.group.account
FILE GENLIST;DEV=LP,1
RUN VIRTUOSO.PUB.SYS;PARM=7;INFO="S=DA P=pw D=domain &
    V=TEST MODULE=program"
```

The current version of the VIRTUOSO code generator uses a model to extract information from SYSTEM DICTIONARY. The use of enforceable standards necessitates this use of SYSTEM DICTIONARY.

SYSTEM DICTIONARY contains the relationships between the files (forms, data sets, KSAM and flat files) and the program, called a module. When the code generator is run, it finds the SYSTEM DICTIONARY relationships and generates the necessary code. If a required relationship is missing, the code generator will identify the missing relationship and terminate. Thus, it is important that a model with all necessary relationships exists that will perform the necessary action(s) required by the program being generated.

If the correct model does not exist, it must be created. It will generally require about 50 lines of VIRTUOSO code and/or macro calls to generate a 1000 line COBOL 85 program. Most models will contain 100 to 200 lines when documentation is included. It can take anywhere from an hour to a week to develop a model which is task-specific (such as maintenance to a detail data set) and yet is fully reusable. Single task models, reduced in size and complexity, will more likely be reused.

Once the model to be used has been identified, there is a VIRTUOSO model which can be used to generate the program documentation. This model, MODD1M00.VIRTUOSO.SYS, can be used to generate two important documents. The first is produced by setting the parameter Model-file=NONE. This will create a document identifying all known relationships, and any missing relationships. All possible relationship types are checked and either displayed if found or identified as missing. The model should also be run where MODEL-FILE=model. This will actually produce the program, thereby verifying that all required relationships are present.

The second case will only report on relationships that are missing and necessary. However, it will not identify relationships that exist but are not used by the model. Thus, a comparison of the two reports will determine if any extraneous relationships exist. For example, a comparison indicated that model MOD02M00 does not process flat files.

The output from the generator can be routed to a printer (FILE GENOUT;DEV=LP,1). However, this causes the code generator to fail on the second pass; this is not a 'real' problem.

COBOL SAMPLE LIBRARY

A list of the macro files and model files included with release A.00.00 of the VIRTUOSO code generator is provided. It must be emphasized that, as per Hewlett Packard documentation, these are **sample** models and macros, and may not be functional.

All of the files listed are in group VIRTUOSO in the SYS account.

MACROS

There are currently 7 macro libraries supplied:

. MAC01M00 - the VIRTUOSO macros

. MAC02M00 - the SYSTEM DICTIONARY macros

. MAC03M00 - the IMAGE macros
- DBGetNext (does not work in mode 2 [serial])

. MAC04M00 - the HiLi macros
  - SetKeySet (works, but requires special SYSTEM DICTIONARY setup)
  - HPDSend (not as flexible as it could be)

. MAC05M00 - the Business Report Writer (BRW) macros

. MAC06M00 - the MPE file system macros

. MAC07M00 - the VPLUS macros (which can be used in lieu of HiLi)

MODELS

The following are the models provided. Any known limitations have been identified. All tested models can be used as either main or sub programs.

. MOD01M00 - the outer block driver
  - this model has not been used because it does not provide block mode

. MOD02M00 - the screen processor
  - this model was used as the main menu, but was modified to allow for flat file handling

. MOD03M00 - master data set maintenance
  - this model has been used with no problems encountered

. MOD04M00 - detail data set maintenance
  - this model has been used, but requires special handling due to the use of macro SetKeySet
  - refer to Figure 2 for a list of the commands used to define the relationship

. MOD05M00 - batch updating of a master data set
  - not used

. MOD06M00 - multi-dataset, multi-screen add
  - not used

. MODD1M00 - create module documentation
  - works very well (see preceding section)

. MODR1M00 - create various reports on the entities / relationships in the SYSTEM DICTIONARY

RELATIONSHIP FOR SetKeySet in MOD04M00

```
:RUN SDMAIN.PUB.SYS
>DEF ;S=DA;P=xxxxx;D=MAPPS;V=TEST;OM=EU.
>C E CONFIRMDETLKEYS;ET=HP-SCRIPT;
 ATTRIBUTE-LIST = (DESCRIPTION =
 "FUNCTION KEYS / FORM FOR MOD04M00").
>C R formsfile,CONFIRMDETLKEYS;RT=FORMSFILE,HP-SCRIPT;RC=CONTAINS.
>C E CONFIRMDETLKEYS;ET=FORM.
>C R CONFIRMDETLKEYS,CONFIRMDETLKEYS;RT=HP-SCRIPT,FORM;RC=CONTAINS.
>E.
```

Figure 2

SYSTEM DICTIONARY

The following is a discussion of the various rules required to use the SYSTEM DICTIONARY.

INITIALIZATION

If SYSTEM DICTIONARY has not previously been created using SDINIT, that step must be performed before continuing. One word of warning - if any part of the signon UDC sets job control word, JCW, to a non-zero value the second part of the batch job will have to be performed manually. This is due to the fact that the JCW value will cause the job to fail. An SR has been submitted.

Create the necessary Domains and Versions.

Run SDMAIN using file VCGEXT.VIRTUOSO.SYS as input
(FILE SDIN=VCGEXT.VIRTUOSO.SYS
RUN SDMAIN.PUB.SYS)

This builds the entity types and relationships needed by VIRTUOSO.

LOADING PROGRAM INFORMATION

If the module is using VPLUS forms, two steps need to be performed for the formsfile:

- Load the formsfile into the SYSTEM DICTIONARY running program SDVPD.PUB.SYS

- Create the necessary relationships in SYSTEM DICTIONARY for the formsfile. A two-step procedure can be used. First, use model GENSCRPT.VIRTUOSO.SYS with code generator VIRTUOSO.PUB.SYS to create an input file for SDMAIN. Second, using the output file, run SDMAIN.PUB.SYS to build the relationships in the SYSTEM DICTIONARY.

Optionally, (but recommended), modify the entities created to add the description.

If the program uses a TURBOIMAGE data base, the (existing) data base is loaded into the SYSTEM DICTIONARY by use of program SDDBD.PUB.SYS. This step is repeated for each data base used.

Optionally, (but recommended), SDMAIN may be used to add a description to each of the entities created. Note that if the field names in the form(s) and data base(s) are the same, then this information will only need to be entered once, at the entity level. This is because when documentation is extracted from the SYSTEM DICTIONARY by either of the VIRTUOSO models, the program will look at any lower level for the description (unlike DICTIONARY).

. If any KSAM or 'flat' files are referenced, the file definitions are manually entered into SYSTEM DICTIONARY using SDMAIN.

. Create the 'module' entity type for the program or subprogram.

. Create the relationships between the module and all files which are accessed by the module. The main program should open all files. None of the called subprograms should be related to the formsfile, and should either access or pass-thru the data base(s) and file(s). The object is to put the delay caused by all file opens into the main program (normally just a menu), which will allow the subprograms to be fast in comparison.

. Create the COBOL COPYLIB

- Use model GENCOPYL.VIRTUOSO.SYS with the code generator to generate an input file for SDMAIN. Label the file "A".

- Using an editor, text in "A" and reduce all HP-COPYLIB-MEMBER entity names to eight (8) characters. An example is the use of SET-nnnn for TURBOIMAGE, SCRN-nnn for VPLUS forms, and FILE-nnn for either flat or KSAM files. The name of any flat file RECORD must have '-REC' appended. Label the file "B". NOTE: an SR has been submitted which would allow the use of a counter within VIRTUOSO, eliminating this step.

- Use file "B" as the input file (SDIN) and run SDMAIN

- Use model GENSDCDE.VIRTUOSO.SYS with the code generator to create an input file for use with program SDCDE.PUB.SYS. Label the file "C".

- Use file "C" as input to SDCDE.PUB.SYS to create COBOL COPYLIB members for all VPLUS forms and IMAGE data sets.

- Run program SDCDE.PUB.SYS to manually create new COPYLIB members for files.

This may appear to be a lengthy and time-consuming process, but it creates COPYLIB members for all forms in the formsfile used by the module and all data sets in the data base(s) used by the module. Thus, if everything is opened in the main module the relationships exist for all subprograms.

Use SDMAIN to create entities for any include files used in the module. As a minimum, this would be the indicator of when the module is to end (such as by pressing f8 in block mode).

If the program being defined calls a subprogram, then use SDMAIN to create the hp parameter list entity and associate it to the appropriate entity type RECORDs created in the prior steps. There are 4 additional requirements:

- A record must exist for each database. This is used to pass information about the database (such as the database name).

- If forms are used in the called subprogram, then a record must exist called 'VCG-COMAREA' and a record called 'VCG-FORMS'.

- A relationship must exist between the calling (sub) program, the hp parameter list, and the called subprogram.

- A relationship must exist between the called subprogram and the hp parameter list.

The main program will open all of the appropriate files, and pass the information to the called sub programs. Thus, the hp parameter list for each lower level of subprograms will be a subset of the preceding level; meaning that the initial hp parameter list may take minutes to define, but all subsequent lists will take seconds.

If data is passed between files (such as between a data set and a form), use SDMAIN to define the field movement. If the field names are the same on the "from" and "to" side, the code generator can produce a move corresponding statement. If the names are not the same, the relationship between each field must be defined. This is another example of the VIRTUOSO code generator not forcing standards; the code generator may require a high degree of manual input if the 'suggested method' is not followed.

Figure 3 is an example of the relationship defined in SYSTEM DICTIONARY for a data movement between a data set and a form when the fields have the same name. If the field names are not the same (such as if a 'V' prefix is added to field names for a V+ form), the '/' would be replaced by the actual from and to field names. This data set contains 11 fields, so you would be increasing your work by a factor of eleven.

Data Movement Definition When Same Field Names Used

```
:RUN SDMAIN.PUB.SYS
>DEF ;S=DA;P=xxxxx;D=MAPPS;V=TEST;OM=EU.
>C R CRAFT,/,CRAFTLISTAP,/,,RT=RECORD,ELEMENT,RECORD,ELEMENT;
>RC=HP-MOVES;ATTRIBUTE-LIST = (DESCRIPTION =
>"movement between CRAFT dataset and CRAFTLISTAP form").
>E.
```

Figure 3

The preceding identified the items which typically need to be defined and loaded into SYSTEM DICTIONARY. The first time an account or application is initialized for the VIRTUOSO code generator may require hours of effort. For each (sub) program in the account or application, the task will be significantly faster; ideally, if the model exists, it should require time on the order of minutes to produce a fully functional program from the design!

It is possible to write a program that will prompt the customer for the various types of activities to be performed by the module. The program should then produce an 'SDIN' file for performing the actual loading into SYSTEM DICTIONARY.

RESOURCES

The following is an example of an accounting structure which can be used to implement this approach.

@.VIRTUOSO.SYS - location of HP sample library

@.PUB.VIRTUOSO - location of the system-wide SYSTEM DICTIONARY

@.INCLUDE.VIRTUOSO - location of the system wide include files

@.MACRO.VIRTUOSO - location of the macro files

@.MODEL.VIRTUOSO - location of the model files

MODEL USED

Ideally, the model needed to perform the required activity exists. If this is true, VIRTUOSO would be run using model file MODD1M00.VIRTUOSO.SYS (as previously discussed) to verify that the model references all defined relationships and that all appropriate references are present.

The resulting VIRTUOSO output is used to determine if a match exists between the design (as defined by information entered in SYSTEM DICTIONARY) and the model. If the model has been properly created, it will ignore non-critical relationships not found (such as if there is no forms file to open) and will flag missing critical relationships (such as if the model is for updating a data set and no module-dataset relationship exists).

Figure 3 lists a relationship that must be included to properly use model MOD04M00.VIRTUOSO.SYS. Once these 'rules' are identified, they can be used whenever that model is used. Unfortunately, this is one of the key items missing in existing documentation.

ADMINISTRATION

The macro and model libraries included with release A.00.00 are only a limited part of what is needed. The SYSTEM DICTIONARY, which is an integral part of the VIRTUOSO code generator, needs to be maintained. Both the TEST and PRODUCTION versions of the SYSTEM DICTIONARY are dynamic and need to be controlled. Include files will be created and can be made reusable. Finally, the COPYLIB should be standardized.

These items can best be accomplished if an individual is responsible. This VIRTUOSO Administrator would maintain this information. Hewlett Packard has acknowledged that work is progressing on the development of tools to support this activity. The individual's key responsibility would be to develop, and implement, a site's standards.

MACROs

Refer to pages 6-13 through 6-25 in the Virtuoso Reference Manual for a method of making additions to the macro library. If the code generator is to be generic, then a new macro should be developed for either a new action or for changes to an existing macro. The objective is to allow existing programs generated with the VIRTUOSO code generator to be capable of being regenerated. This would be impossible (or very difficult) if changes to existing macros is allowed. It appears that a more workable solution would be to build a separate structure for maintaining new macros.

It is important that there is a provision for all required standards. If there is a requirement for error handling, then verify that the macro tests for all possible conditions. The 'tombstone' produced by a program generated by VIRTUOSO is very helpful in identifying the problem, but needs may require it to be more specific. A new macro can be developed that will meet specific needs (such as routing it to a specific device class). One item which the supplied macros do not have in all cases is a 'user action' entry point. This allows for the inclusion of code (using an include file) when necessary. HP has an SR to implement user-action points in all macros in the future; and, it is recommended that user actions be included in any developed macros.

MODELs

The supplied macro library may be sufficient, but it is doubtful that the model library will not need additions. Pages 6-5 through 6-12 in the VIRTUOSO Reference Manual are a good starting point.

A new accounting group should be assigned for model files which are produced, but the name must not conflict with a model file received from Hewlett-Packard. The model should be generic, and user-action points should be used freely.

SYSTEM DICTIONARY

Only one SYSTEM DICTIONARY is needed. The domains and versions allow for separation by account and for testing. One individual needs to be responsible for the security and conversion from a test version to production. SDMAIN.PUB.SYS provides the tools to perform these tasks.

INCLUDE FILEs

The include files will probably require a two-tier approach. Some include files will be specific to a particular application, where others will be generic (such as when f8 is pressed to MOVE 1 TO CONTINUE-STATUS).

COPYLIB

The copylib member names need not be known by the individual creating a program using VIRTUOSO. The code generator references the appropriate members. If one copylib file is to be used system-wide (instead of maintaining a separate copylib for each application or account), then a naming convention needs to be in place.

Copylib maintenance does not need to be performed by an individual. The copylib is an integral part of VIRTUOSO code generation. It should be controlled at the system level.

The code produced by VIRTUOSO is disposable. It is not needed once the object has been created. If necessary, it can always be produced again. In contrast, the five items identified must be controlled and maintained in lieu of the source.

SUMMARY

Hewlett Packard should be applauded for the VIRTUOSO code generator. It is a positive step towards eliminating coding and provides a method of maintaining (and enforcing) standards. For anyone who has tried to maintain a program written by someone else (or by themselves years earlier), they know what a challenge, and waste of manpower, the maintenance may be.

The use of VIRTUOSO is in its infancy, and the methods and techniques to implement this product have not previously been documented. The authors have identified a method that has proven successful in producing new applications and at implementing changes to an existing application.

VIRTUOSO can be much more powerful than TRANSACT (and FASTRAN) as a programming language. However, like any new language, there is a steep learning curve that must be overcome before an I S department can take full advantage of this tool. The techniques presented here should minimize the effort.

4GL Mentality

Douglas J. Mecham
AIS Consulting
Palo Alto, CA 94306
(415)856-4039

Abstract: Our computer world has gone crazy with 4th and 5th generation languages. But, have we mentally made the transition from our 3rd generation language position? The answer, of course, is that we are struggling. My thesis is that the thought processes used for 3GL software development are different from 4GL software development. Development of 4GL software with 3GL mentality can be very expensive.

This paper will present a summary walkthru of software design and development characteristics and attributes. Using this summary analysis 3GL and 4GL thought processes will be described. Subsequently, the latter will be expanded into a discussion for 4GL mentality and examples given in applying the thought processes to areas in information systems. Additional discussion is given extending this mentality to testing, conversion, psychology at the terminal, and long term utilization. A conclusion will be presented indicating the greater business leverage using 4GL mentality.

INTRODUCTION:

WHAT IS THE NAME OF YOUR DICTIONARY ???

and

R E M E M B E R

WHERE EVER YOU GO ... THERE YOU ARE!

(N. Mecham, 1966)

We are here in a fast paced time were the bottom line is evaluated daily and long range planning provides direction for next week's work. We must deal with complex situations, in-exhaustable demands, and and political triumvirate of MANAGEMENT - END USER - INFORMATION SYSTEMS. Then on top of all this the flow of INFORMATION and the requirements for its PROCESSING is increasing exponentially. What to do? What to do?

In summary,

A N E X A P L

ANxiety & EXpectations get Ahead of PLanning

We hear talk of artificial intelligence and 5th generation languages but that is a far distance from our 3GL + position at this time. In fact, many information systems still use manual methods and physical file cabinets. The cliches from the IS departments abound - Pay Now Or Pay Later ... There are never enough resources to do the job right in the first place BUT there are always enough resources to do the job over.

MY SUBJECT:

My subject is the human mental thought process associated with fourth generation languages. This is the middle ground between the third and fifth generation languages. While the focus is for programmers and analysts the topic is relevant to management and users alike.

MY OBJECTIVES:

First ..... To contrast thought processes and characteristics for third and fourth generation languages.

Second ..... To develop a basis for 4GL Mentality.

Third ..... To brainstorm some possible exciting & beneficial directions this approach could take.

THE PRESSURE & THE PROBLEM:

To meet the anxieties and expectations the IS departments often bow to the pressure with comments like "Sure - all we have to do is write a few programs ... EASY!" or "We have programmed similar applications and can easily transfer the technology to meet the new requirements." Then when the go-ahead is given the IS team (in some cases a small army) leap into action. The first response is to take the Requirement skip beyond the Systems Analysis and External Design steps to put together a quick sketch of the Internal Specifications so they can begin Development as soon as possible. Following successes in getting developed software to work, the next giant step is to demonstrate in Production Results to the users by sort of sliding thru the steps for Test, Integration, Implementation (conversion, training, operations, and transition), and -naturally- Documentation. Of course, now in Maintenance mode the activity requires heavy effort to iteratively re-design, re-develop, re-implement, and now-document to "fill in the system gaps." The situation dictates "living documents every where with "flexible version freezes".

What is wrong with this approach? Nothing! However, it does pose a PROBLEM. It means we have to change our thought processes to approach the systems problem from another angle and need to be the automated information processing tools available to support the new approaches... the new mentality.

Needless to say the transition is not going well in all sectors, users are frustrated, programmers are frustrated, business managers are upset, the business often falters, and the project managers are mad at everybody. The time and dollar constraints are over-run, the requirements and fixes are changing at a greater rate that that of the development, plus there is a demand for flexibility. Then with great audacity the end users want the "system" to protect them from making mistakes when using the system and insure information integrity at all times.

Is it possible to satisfy these needs and make everyone happy? No! However, some change of mentality will go along way toward greater success.

TRADITIONAL 3GL THOUGHT PROCESSES:

If we were to characterize some of the thought processes associated with third generation languages the following might be a representative sample.

* Information for processing is thought of in terms of the individual data items.

* Process relationships are more important than information relationships.

* The notations made by a 3GL programmer would depict a detailed linear sequence of events necessary to accomplish the processing goal. The notations would be stated by a description of each simple action needed.

* A large amount of energy, 50% to 80%, would be applied to design and development of control, user interface, and data file mechanisms.

* The emphasis would be on the application operational structure versus data groupings because at the 3GL level there is full flexibility to handle any data any where.

* The data structures would be designed at a very detailed level (bits and nibbles) with a focus on accommodating access and control mechanisms.

* The choice of data element names would be cryptic and there would be multiple names used in different processes for the same value.

* When a processing issue arises, the 3GL person's first thoughts are to just add a subroutine to branch out and perform the desired data check, data lookup, or other special process. It is realized that the processing sequence is fully under control of the 3GL software even though the particular process fits with the current information processing at hand. That is, any group of data may be handled in any way and any point in time,

* Logic and decision control of a given automated process is handled at the most detailed level of programming often relying on the most detailed level of data, the bit.

* All data type conversions must be handled at every turn in every detail.

* There is a large difference between the External and Internal specifications; and there is a large difference between the latter specifications and the actual 3GL code generated.

* Many complex operations take place as a direct result of the detailed 3GL coding.

In summary, the third generation language person thinks in terms of the detailed mechanics of the automated information processing.

***** PROCESS DRIVEN *****

CONSIDER PRESENT SITUATION:

Perhaps a review of the spectrum of language generations and how it applies to several relevant categories would help bridge the change in thought process applied to automated information systems.

	Computer System Ages				
	FIRST	SECOND	THIRD	FOURTH	FIFTH
Type	Machine Wiring	Direct Coded	Procedural	Functional	Adaptive
Languages	1GL	2GL	3GL	4GL	5GL
Analysis	Logic Diagram	Descriptive Comment	Flow Diagram	CASE	Intuitive
	By Scientist	By Team	By Analysts	By Staff	By User
File System	SubItem Values	Data Values	Data Groups	Information Network	Conceptual Environment
Processing	Read	Program	Subsystems	Systems	Environment
Support	Plugboard	Assemblers	Compilers	Interpreters	Response
User Interface	Manual	Punches	Edit	Dialog	Image *
	[* Note that a picture = 1000 words ... verbs + nouns]				
Operating System	Manual	Program	Batch	Timeshare	Personal

	Computer System Ages				
	FIRST	SECOND	THIRD	FOURTH	FIFTH
Control	Single Step	Single Task	Multi-TAsking	Multi-Processing	Resource Controller
	Read &	Sequence	JCL	Operations	Heuristics
	Execute	Parameters	Command	Control	Determined
I/O	Switches	Tape	Cards	Terminal	PC
Printing	TTY	Large	Central	Local	Personal
CPU	Single Integrated	Simple Controllers	Single Controllers Special	Multiple Controllers Special	Distributed Systems Dedicated
Data Comm	None	Wire	Modem	Switched	Network
Administration	Logs	Security	+Accounting	+Tracking	?????
Controls	Special	Procedures	Audit	Distributed	Autonomous
Facility	Building	Floor	Room	Office	Desk
Management	Project	Team	Managers	Manager	CIO
Support	Projects	Teams	Departments	Staff	User
Underlying Systems	Project Programming	Team Programming	Department Packages	User & Vendor Tools	Vendor Solutions
Development Maintenance	Project Scientists	Team Programmers	Department Programmers	Staff & Users	Vendor & User
Documentation	Coding Notes	Program Description	System Description	On-Line Help/Doc	Intuitive
Focus	Project	Team	Operations	Service	Enterprise
Strategic Planning	Project	Team	Tactical	Strategic MIS	Strategic Business
Responsibility	Leader	Management	Department	Staff	CIO/User
Integration	None	Teamwork	Interface	Coordinate	Business
Performance Evaluation	Project	Team	MIS Manager	Staff	Enterprise
Purchasing	Project	Team	MIS	Purchasing	User
Vendor Interface	Project	Team	MIS	MIS	User
Information System Leverage	Experiment	Tabulation	Data Processing	Information Processing	Inference Engine
Business	Barter	Mechanize Systematize	Accelerate Performance	Capitalize System	Capitalize Environment
User	n/a	Passive	Remote	Indirectly Active	Directly Active
*** In the Clerical Camp -					
Secretary's Workstation	Typewriter Clerical	Typewriter Coordinate	Typewriter Reports	Terminal Print Control	PC Creative
Support Spectrum	Technical Specialists		Computer Science Professional		End-User Involvement
Excitement	Intensive		Routine		Innovative

CONSIDER 4GL THOUGHT PROCESSES:

If we were to characterize some of the thought processes associated with fourth generation languages the following might be a representative sample.

* Information for processing is thought of in groups and relationship of groups.

* Information relationships are more important that process relationships.

* The notations made by a 4GL analyst would depict displays of information and process functions needed to accomplish the processing goal.

* Only a relatively small amount of energy would be applied to the control, user interface, and data file mechanisms; the large amount of energy would be applied to information description and process functionality. The processing sequence is significantly automatic.

* The emphasis would be on the information structure and application processes since 4GL processing focuses on handling groups of related information.

* The data structures would be designed in terms of completely defined data elements and groups of data elements and their relation to each other and the processes that manage them.

* The choice of data element names would be meaningfully related to the process and use of the information.

* When a processing issues arises, the 4GL person's first thoughts are to evaluate the information groupings and selection criteria. The processing sequence is mainly under the control of the 4GL language processing tool. Only that selected complex of information is available at a particular point during the process; special 4GL characteristics allow independent access or checking of information elements outside the current information complex.

* Logic and decision control is specified with a particular aspect of the fixed 4GL processing logic and utilizes conditional relationships of information at the logical level.

* Often data type conversions are handled automatically within the 4GL processing statement.

* There is a close relationship between the External and Internal Specifications and the actual 4GL statements.

* At each step of a 4GL process multiple complex actions take place on behalf of the programmer.

In summary, the fourth generation language person thinks in terms of the information groups and associated processes.

***** INFORMATION DRIVEN *****

IMPLICATIONS:

There are definitely some implications to this change in thought processes. We have the choice to take advantage of the situation or continue on our own way. To not consider utilizing these new thought processes will put that set of persons very much behind as we enter into the fifth generation languages. Consider some of the following observations,

- A different and broader spectrum of people will be involved directly with the design, development, and implementation of 4GL systems. The users, managers, and IS personnel will have to work as a team.
- Expectations will be even greater with fewer resources.
- Technology underlying the 4GL tool set will change faster than expected.
- With the Information view applications will not be isolated and has the opportunity to take on a new processing dimension for supporting the Business.

These changes (or lack thereof) can cause perceptions to change that in turn magnifies any business up turn ... or down turn. The important flip side of this aspect is the potential of the 4th & 5th generation languages to be powerful business leverage points.

Now the question becomes how to adapt to the everlasting change. Perhaps we need to build Transition Systems with their own methodology, theories, and engineering. In fact, we could look at this situation from another perspective and insist that if the "system" is not changing it should bother us. In this light the fourth generation languages become a significant asset allowing us to adapt application systems to the changing requirements, easily & simply.

If we retain a 3GL mentality as we use 4GL tools energy spent trying to control a process at the lowest level and trying to deal with bits will cause considerable frustration, make 4GL statements and statement sequences un-maintainable, and cause conflict among the project team. The result will be difficulty with development, major design changes, minor design changes, project extension by a factor of 2 or 3 times, and wasted energy.

PROPOSAL:

I propose that the use of a 4GL with proper mentality can produce significant short term and long term benefits.

Among the benefits is a very happy crowd. For sure the new mentality will require new disciplines.

SYSTEM CYCLE:

For a practical use of the 4GL mentality we need to review the system life cycle aspects and how using 3rd or 4th generation languages relate to them.

The first on this list is the VISION/PROBLEM DEFINITION setting forth the basic concepts or issues at hand. The FEASIBILITY provides investigation as to whether the proposed concepts will work at all, ie, can the vision be applied to a practical system? Or, in the case of a problem - what is the real problem and can a solution be found? This is the first point of major decision, to continue or not. The ANALYSIS effort determines all the parts and their relationships in detail. Once disassembled the DESIGN proposes a way to put the parts together, ways to enhance the situation, and/or ways to leap into the yet uncharted paths. The DEVELOPMENT produces a practical result based upon the system plans. After building the system it must undergo TEST to validate correctness. Now the tested result is INTEGRATED into the business areas to insure consistency with current and new operations. Here is the second go - no go decision point. IMPLEMENTATION involves training, data conversion, and operation setup. Once the new system is turned on in a PRODUCTION mode its PERFORMANCE can be measured. Then the system enters the MAINTENANCE mode for ongoing 'repairs' and enhancements. Underlying all of these steps is proper DOCUMENTATION. After significant requirement changes a RE-DEVELOPMENT effort recycles thru these steps.

Below is a simple list of how 3GL and 4GL thought processes and characteristics that might relate to steps in the system cycle.

	3GL	4GL
Vision/Problem	Independent of language except for preconceived notions as to how the vision is formed or problem described.	
Feasibility	Projection only View of existing system	Can be applied to Prototype
Analysis	Focus on current procedures and operational processes	Focus on strategic information and business processes
Design	Detailed/Complicated Not necessarily disciplined	Simplistic Disciplined
Development	Large effort for results Common cod modules re-invented; Need to create control mechanics Emphasis on detail processing	Small effort for results; Standard common code modules Control is given Emphasis on user
Test	Many paths to be tested White box testing Maximum regression	Few paths for test Black box testing Minimal regression
Integration	Difficult at best	Adaptive
Implementation	Transition	Conversion
Production	Fixed operation	Flexible
Performance	Good	Poor
Documentation	Difficult & detailed Never time enough Never completed	Intrinsic; Enhanced with discipline Easy to follow
Maintenance	Long drawn out process Troubleshooting involved Correction not always easy	Rapid turnaround Troubleshooting easy Correction easy
Re-Development	Start over	Adaptive

APPROACH:

The approach for using 4GL tools to support an application system involves developing a prototype from the INITIAL IDEA. From this point the SYSTEM ANALYSIS provides a clear view of the whole system. Then the INFORMATION needed is reviewed for clear definitions and relationships as well as how it is used. The DESIGN then reflects the information and process structures with the RULES used to direct the activity to be applied to the INFORMATION COMPLEX, ie. an information group treated as one entity. The DEVELOPMENT approach is actually an expansion of the prototype to a useful set of operations and a re-development to fit the design specifications. The TEST activity is made simpler through easy data entry to set up test cases and easy reporting to review the results. IMPLEMENTATION is made easier using 4GL tools to convert data; training is made easier by using a common and standard user interface. MAINTENANCE and RE-DEVELOPMENT are simplified by not having to get involved with the detailed code that handles the underlying processing mechanics.

Of all the aspects of a 4GL tool the greatest is the DATA DICTIONARY. This item alone can minimize large amounts of coding and minimize maintenance process.

APPLYING THE 4GL MENTALITY:

TO THE END-USER -

The end-user is very close to the design and thus has a stake in the developed solution. The end-user has direct power over the developed solution, eg. reports. There is greater optimism in meeting end-user requirements and 4GL approach makes it easier adapting to end-user situations. One of the most useful aspects of the 4GL tools is the ease of accommodating clean up of end-user data. The application system design is not limited by previous end-user experience. The end-user can quickly view system results. The end-user is closer to and has some control over the processing logic and can influence the system design with minimal impact.

TO PSYCHOLOGY AT THE TERMINAL -

The 4GL provides a consistent user interface with fixed (although flexible and numerous) common functions that have been tested and are documented. A given application use need only fill in the particular words associated with its information and processes. One of the largest advantages is the established in context HELP functions. This allows great adaptability for a particular application and gives confidence to the user that help is just a key stroke away. The display and formatting functions are among the most effective for an application. The mentality is to create the vision of the user interface then match the 4GL syntax to achieve the desired result.

TO SYSTEM CONVERSION & TRANSITION -

One of the problems with new systems is the entry of "bad" data and the conversion of old data files to the new data files. The 4GL tools allow easy transformation of data items and correction of "bad" data. These processes can easily provide audit trails, data validation, data editing, process validation, methods to check-point a processing sequence, expand generic data, or summarize information. The expense of this effort is minimal concentrates on the application data and not the process to perform the activity.

FOR TESTING -

A desired focus in testing is black box testing of an application's functionality without worry of each detailed processing path tests, white box testing. It is easy to establish test data for test cases and minimize the validation of processing code. The 4GL solutions can be used with automatic testing systems such as AUTOTESTER which focuses on the interface functionality. The bulk data processing functionality of a 4GL can provide an easy way to extract real data for testing. The data resulting from testing is easily analyzed by using proven display & reporting methods. Thus, there is a high degree of confidence for a minimal cost when thinking in terms of 4GL approach.

FOR MANAGEMENT -

The business managers can no longer stay away from automated information systems because of the close relationship of business process and 4GL systems processing. Then need to think in terms of required functionality to enhance the business process AND become involved in the design and development process to a certain extent. Fortunately, the pay-back for the invested time and energy is good. Success can be seen as well as visualized in reasonable time frames. For the enlightened a platform can be created for steps to larger advantages not yet within the scope of vision.

INDUSTRY PROBLEMS:

Not all is goodness and light; there are problems and may turns of the road from 3GL to 4GL and beyond. The 4GL mentality assumes constancy and natural logic with infinite flexibility for such easy and powerful tools. For instance, you may be able to teach a secretary how to create and use a simple application system using POWERHOUSE in about an hour but there are limitations when the secretary tries to create more involved processing. One common problem is the lack of automatic date processing in an expression; another is lack of a TIME data type, and poor text handling. The modules within a particular 4GL tool set are not always the same and can make productivity difficult. Too often the user-programmer needs to discover how the underlying 4GL process is performed through trial and error; there are many implied processes in a 4GL.

Alas, 4GL systems have their problems but they too are in development. They are a great advantage now and will provide even a greater advantage in the near future. The greatest advantage of a 4GL will be when the people involved take on a 4GL mentality.

CONCLUSIONS:

Whether you know it or not you most likely have been developing a 4GL mentality through the use of POWERHOUSE, RAPID, PROTOS, SPEEDWARE, SUPERTOOL, MPEX, MAESTRO, etc. Now the challenge is to fully develop that discipline as an Information Systems Team.

The bottom line is **GREATER BUSINESS LEVERAGE.**

RECOMMENDATIONS:

1. Walk in a user's moccasins
2. Try a 4GL tool system
3. Dream of the possibilities

OPPORTUNITIES:

The opportunities are many: application system standards, end-user flexibility, better information control, business leverage, and stepping stone to 5th generation languages. For 3GL PROGRAMMERS 4GL mentality offers freedom from a large redundant programming effort plus a special advantage in using 4GL tools because of the detailed technical background. For the END-USER and ANALYST 4GL mentality offers a simple & easy way to meet application processing requirements in a changing environment. For the MANAGER 4GL mentality offers a way to achieve a business leverage at a reasonable price within shorter time-frame.

4GL MENTALITY IS A WIN WIN WIN SITUATION

AND A PREREQUISITE FOR 5GL PROCESSING

PRODUCTIVITY IN THE APPLICATIONS DEVELOPMENT ENVIRONMENT

BY BOB HOOVER
INFOCENTRE CORPORATION
5674 STONERIDGE DR., SUITE 218
PLEASANTON, CA 94566
415/460-8220

As a district manager for Infocentre, I have the unique opportunity to visit over 100 HP3000 shops a year. And as I visit these sites I take notes on the similarities of these sites. From my findings over the last year I have come to a conclusion on what is the most productive DP shop. I am not going to discuss any one company today, rather I am going to focus on the DP shop that has the most potential for high productivity. But to have the highest potential I've found that:

- It doesn't matter how big your DP shop is, or even if you have one HP3000 or 100 HP3000's, what it really comes down to is how you as a manager treat your employees.

- It comes down to how you as employees view your job.

- It comes down to how you as a company view the importance of productivity tools.

- It comes down to how your company views the importance of MIS.

My first point is, there is no way to increase productivity in the MIS environment unless you start with good managerial skills so that you treat your programming staff with respect. Whether you are using assembler, cobol, a 4GL or a development environment.

In order talk about productivity we should define it. For this discussion I will define productivity as the ability to increase output while at the same time decreasing expenses, or at least not have them rise as fast as the productivity. This is Utopia for the MIS department. This is how companies become successful in today's competitive environment.

Our goal then, is to build the necessary blocks so that we achieve Utopia in the MIS department.

How do we achieve this first goal? Lets start from the top of the organizational chart and move down to the specifics.

First, if your corporation doesn't view MIS as an integral part of the corporate strategic plan, you are going to have a problem. I am amazed by how many DP shops are located in the basement of buildings. For some reason top management believes we are still coding assembler behind closed doors.

So as a DP manager, I need to install a new outlook on what MIS's importance is to the corporate strategic plan. Once you have commitment from the top, the rest is easy.

Next we focus on a good strategic and tactical plan for MIS. Examples of where we want to be in one to three years and then a good step by step approach on how we are going to get there. This plan will allow us to focus our programming staff so they know exactly where we want them to go and how they'll get there.

This brings us to the programming staff. If you don't start with good people and if you don't treat them with honesty, respect, and award them for their achievements, then you might as well forget about your entire plan. You see, it is the guys in the trenches that are going to make or break the company.

You have got to tell your programmers how it really is so that they know how and when to work productively. Get them involved. Let them move around the department from maintenance, to analysis, to operations. Let them determine the best products to use. A little secret, if you let them choose the products then I can guarantee they'll make them work. They have to because they choose them.

Have enough respect for your staff that you can allow them to go off on their own (once you have given them a good direction of course) and be productive. Remember that we want to hire good people that will work hard at achieving their goals and be creative while doing it.

But productivity needn't stop here. Productivity comes from other sources also. Take 4GL's for instance. 4GL's promise ten times productivity gains compared to cobol. But imagine a 4GL in a shop where no one knows where the MIS department is headed, and no one is treated well.

There are no free lunches. Just because you spend $10 to $50 thousand on productivity tools doesn't mean you'll get productivity. It all goes back to the people using and developing the applications. A productivity tool is a small investment compared to personnel. Employees are most important. Treat them with respect and you will see greater productivity and even greater return on your productivity tool investment.

Lets now assume that we have achieved our first goal and move onto the next step. Lets take a giant step forward and start using something called a 4th generation language. What is a 4GL?

There are typically four categories:

(1) Query and reporting.

(2) Cobol system generators.

(3) Non technical user tools.

(4) Development language.

Query and Reporting tools are typically products that allow programmers to access files and databases for simple lists of information or may get into specific layouts of the data and computations.

Cobol system generators will increase productivity, but you still fall back into a 3GL for compiling, running and maintenance.

Non technical user tools are very helpful to offload backlog from the DP staff but could lead to excessive CPU demand should the tool catch on. Also there may be some limitations as to the functionality of the tool because it is designed to be user friendly.

A development language can be very helpful because it covers all aspects of the programmer's needs including batch, online, transaction processing and reporting.

So why should you use one of these tools? Lets look at the benefits.

You can offload the demand of your programming staff because there are fewer lines of code to write an equivalent application. Because there are fewer lines, the systems become much easier to maintain. Industry estimates are as much as 20 percent of the original cost to develop an application is spent on maintaining that application each year.

Systems will also need less programming documentation because of the high level orientation of the language.

Systems can be fine tuned by the programming staff in much less time. This makes users much happier because they will begin to see programmers putting systems out much faster.

End users can do some of their own programming for reports. This again will offload demand on programmers and allow programmers to work on more sophisticated applications.

So there is no doubt that a 4GL can help to increase productivity. But I know what you are thinking, there is a catch. So lets look at the problems that may occur when using a 4GL.

First of all lets look at the typical or traditional approach we take when developing 3rd or 4th GL applications. I call it the bottom up approach.

(1) Feasibility study.

The benefits of the endeavour are evaluated and a project is given the go ahead.

(2) Systems analysis.

Here the functionality of the system and the data structures are clearly and specifically defined. The users review this specifications document, the user and the DP professionals meet regularly to redefine the specs and finally at long last the specs are signed off so programming can begin.

(3) Database design.

A very sophisticated and carefully structured database is designed that will run most efficiently on the CPU. Typically taking little concern of the logical aspects of the users needs.

(4) Programming and testing.

Here the programmers are off writing and testing programs to meet the specs and testing it from an internal point of view, no involvement with users is taking place.

(5) Implementation phase.

Here the system is sent out with flags waiving and trumpets blowing to a burst of user applause and an office full of excited and appreciative users.

(6) Documentation.

Unheard of and typically ignored until the auditors come around.

(7) Maintenance.

That ugly beast that kills all other plans for development.

This approach assumes that users:

- Know what they want.
- Communicate it effectively.
- Requirements are static.

Which leads to:

- Long development times.

- High costs and time for maintenance.

- The build up of backlog.

- Non functional applications.

This is the typical approach and there is really nothing wrong with it and in fact it can work for you provided that you really concentrate on the specifics and have programmers and users that are willing to communicate on a regular basis. However, at best, you still may see only a 25 percent increase in productivity. Not bad, but remember we want to build the blocks to Utopia.

Lets look at a new approach. This new approach coupled with the inherent productivity of 4GL's will allow us to further increase our productivity.

Why do we need a new approach to development? What's wrong with the traditional way?

Let me answer this question with an example. Lets all try to imagine a picture of a destroyed building. Why is this building being scrapped? Lets make an assumption and say that the architects went out and told the builders that they wanted a 24 story building with windows and 350 rooms made of steel and brick. OK, go build it. Two years later the building was complete and the architect and the owner came to look at the finished building. Remember that the owner and the architect didn't review the building as it was being built.

So they looked at the building and discovered that they forgot to tell the builders to put elevators and stairways in the building. The builders argued that they were not specified in the initial request and therefore could be built but it would cost double the original cost because it was a major design change. Well, you get the picture.

Lets look at the way they really approached things. The architect drew up the plans then built a model showing exactly how the building would look so that everyone involved knew up front what the building would look like and what it had inside. Changes and modifications were made before total completion of the project. No surprises to anyone.

Compare this now to traditional structured approaches to system design. Get specs, go away for 18 months, then come back with a completed application to find out it isn't what anyone wanted.

This is why we will take a new approach.

Lets call this new approach "Prototyping" or "Protocycling" or "Top Down Approach to Development". It really doesn't matter what we call it. What matters is that we work to achieve the most applications for our users in the most efficient way possible.

Here is the scenario:

Users and developers meet to discuss fundamental tasks of the application. They ask questions like:

- What info do we need to collect?

- What info appears on reports?

- What transactions need to be performed?

We start with the very basics.

The developer builds from this information the first prototype. This should exhibit essential features; menus, screens, reports, and a logical database. Developer produces documentation so that user can systematically evaluate prototype.

Developer revises the prototype, adds missing essential features and begins firming up database design and logic.

These last three steps are repeated with the key being timeliners. Each repetition will further solidify the database design and the required logic.

With this methodology we are thinking about an iterative process that starts with a tool to create a visual representation of users requirements, continues to confirm the feasibility of the application and finally evolves to a state where in fact it is a bonafide production system.

So why is this a good approach?

- People are born critics. So why not get them involved early in the design of the applications (after all these are their applications). Let them add, delete, modify and give their input now instead of later.

- Users don't really know what they need or want so by showing them prototypes the entire application begins to come to form directly in front of them.

- Users have an inherent difficulty in communicating with DP people. So this means that both DP and users must commit to working together towards one goal.

- Reduces risk management. User's needs are constantly changing so rather than presenting them with the wrong finished product, we now have the opportunity to present applications that can be changed or even scrapped before the big expensive coding begins.

Most importantly:

- Reduce the amount of training because at least one key user will be familiar with the applications.

- System is presold. Users get excited early on in the development cycle. No surprises and no misconceptions. There is immediate acceptance of all applications sent out by data processing. The application is authored by the users.

- Programming staff are heroes. DP is putting applications out ahead of schedule and they are exactly what the users need and want. It really reduces the pressure on programmers because everyone knows what to expect.

It now becomes a win-win situation!!

So now we have implemented a new approach to development. Getting involved with the users to define and modify prototypes until they eventually become true production systems. This new approach will help to increase the productivity in using a 4GL. But there are some inherent problems with 4GL's.

Most 4GL's will allow programmers to generate simple applications much quicker which is why we invest in them. But quite often I hear programmers say that they use their 4GL for about 80 percent of their applications and then fall back to 3rd GL's for the rest.

Falling back to 3rd GL's is something the 4GL's must overcome if we are to reach Utopia.

This moves into another dilemma called the performance V's ease of use. As programs get to be more complex, the 4GL's become much more difficult to use. 4GL's must provide tools that allow for ease of use but must also be able to get complex in their computations.

One of the reasons they become more complex is that typically you have to have a number of different modules or products in order to do different things. For example a report writer that generates cobol instead of a 4GL or a different module to do reporting and a different module to do screens. This is the perfect example of products getting more confusing as the applications become more complex.

This leads to another problem of keeping consistency of application designs. Using different products or modules for your applications will lead to many different approaches to the actual coding. This is a major problem in controlling maintenance costs. When programmers leave they typically take with them their own special approaches to program writing and we all know how well we all love to document our programs. Having structured methods of approaching screen code, and report code will help cut maintenance costs.

Another problem occurs when the report writer, query language, or development language is dependent on a dictionary. In many cases this can be to the advantage of the DP shop provided it is used the way it was meant to be used. Many times dictionary driven tools require a full time person to maintain the dictionary. If not, you loose all the benefits. A dictionary can cause many tools to run much slower than they should because of the constant accessing of dictionary information.

The last problem and probably the most important is the fact that most development languages are simply that, a programming language. They allow you to write the program at least 10 times faster than compared to a 3rd GL. There are hundreds of products and dozens of approaches to augmenting application development currently on the market including all the types I mentioned above. Each product attracts the designer's attention to those aspects of the development process it addresses. But to see the true productivity that we are striving for we must deal with all aspects of the applications life cycle. From the creation of the database, generation of code, prototypes for user involvement, generation of documentation, tutorial training facilities, simple "user friendly" screens, and simple maintenance of the system through the entire life cycle of the application.

I stated before that we were going to take a giant step forward when we discussed 4GL's and we did. However, we also came up with some serious limitations with these 4GL's.

So lets take another giant step forward to the next step in programmer productivity and talk about a complete environment that will address these inherent defects of 4GL's.

- 80 percent of applications written by 4GL. Lets make a commitment and say that we have created a 4GL that is so powerful that it can do 95 to 100 percent of all your programming needs. Lets achieve this by keeping the code a little more procedural than the typical 4GL. This will allow us to get very complex in our computations without the code getting exponentially more confusing to work with.

- Lack of programming standards. Lets create a product that creates or generates most of our code for us so that from the user point of view all applications will work alike, and from the programmers view much of the code will be very similar for all applications. The systems will function in the same manner (i.e., menus, screens, prompting, message display). The time needed for the user to become productive on a new system is minimized. The DP professional can then be spending the time previously spent on training the user on new development. These standards manifest themselves also in the code itself. Not only do the applications function the same, but the code will be very similar. Readability and standardization of code account for enormous savings in development and maintenance time.

- Dictionary dependent. Lets take the best of both worlds and have a system wide dictionary (HP system dictionary for instance) for use if your specific site needs it for development; but lets use it in a passive fashion meaning that your program accesses it during the development phase but not during production. Active dictionaries are a major reason for the slowness of 4GL's. Or lets also leave you the ability to run the applications without a dictionary should you not require one. This saves time by not forcing initial dictionary set up.

- The last, and most important as far as true productivity goes, lets create a tool that gives us full life cycle support of the application. A tool that would take care of most of the mundane and repetitive aspects of developing production application systems. Tools that understand what is involved in the fundamental functional areas of an application, such as data storage structures, menu driven end user interface, online data collection and retrieval screens, online and batch reporting, transaction processing, application security and is able to assist the DP professional in pulling it all together. And, most importantly, be able to achieve this without significant performance degradation on the CPU. To add a few extra flowers with the icing on this cake, lets allow these applications to run on either the HP3000 or PC's and network the data between the two. Lets call this a full 4th generation environment.

Well known database industry export Dr. George Schussel states that a typical database application is made up of:

- 60 percent screen code.

- 25 percent database access code.

- 15 percent computational and control code.

That means if we can use a Fourth Generation Environment to design screens without writing a single line of code we have just eliminated close to 60 percent of the redundant coding. Now lets eliminate the database access code using our environment and that will leave us with 15 to 30 percent of the application that actually needs to be hand coded. This allows our programmers to concentrate on code that is meaningful and takes some serious thought. The only code to write is the challenging code which will again increase the productivity of the programming staff; and don't forget that this remaining code is still written in a 4GL because we can write close to 100 percent of the entire application in our 4GL.

The last aspect is ongoing maintenance which over time is the biggest cause of expense and backlog. Lets allow our environment to maintain our program application even after it has been turned over to the users. Even years later by keeping all the screens, menus, databases, user documentation, help screens and code within this environment. This allows the programmers to quickly locate areas of the source code so that they can be changed or modified and then tested without ever having to exit the environment.

There you have it. A complete 4th Generation Environment that supports the programmer/analyst through the entire life cycle of the application, from the initial prototype to the old and patched ten year old production application.

It is my belief that Utopia does exist here and now as I speak or I wouldn't be here to talk about it. This is a new and exciting challenge in today's backlogged-laden DP shops and I urge you to seek out these new and exciting ways to boost your staff's productivity.

This is the solution to the perfect DP shop that will experience 10 to 30 times the increases in productivity by making a corporate commitment to invest in the people, tools and techniques necessary to make the application development and maintenance process the most productive and cost effective possible.

TITLE: The Production Report Generator Re-examined

AUTHOR: Patrick Fioravanti

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5432

Migrating to Native Mode With Cognos...

Marilyn Petty Verill
Talley Defense Systems
3500 N. Greenfield Rd.
Mesa, Arizona 85205
(602) 898-2434

Introduction.

Cognos version 5.03D runs in native mode on the HP3000 Spectrum series computers. It takes advantage of the Hewlett-Packard Precision Architecture allowing it to be much less computer resource intensive than previous versions. This is how we at Epson America did our migration and what we learned in the process.

Our major Cognos application is the Epson Reporting System (ERS) running on four series 70 machines using version 5.01F. This system is made up of approximately 219 QUICK screens, 179 QUIZ reports and 10 QTP routines. There are also a few routines used in other applications that would migrate at the same time. We run all of our screens in block mode. Our goal was to migrate to two series 950 machines and run in native mode. We expected our applications to run faster and thereby cause performance to be at least not worse than it had been with four series 70's.

Planning the Upgrade

Our System Engineer from Hewlett-Packard was very helpful in setting up plans for the migration. We documented what we intended to do and how each step would be accomplished. This planning was invaluable as the migration progressed.

In planning the migration to native mode Cognos, we first gathered documentation. We got Cognos release notes for version 5.01F and the manual updates for version 5.03D. When 5.01F was released, syntax that had become obsolete was noted. With the release of 5.03D, this syntax was no longer supported. Since we had not updated obsolete syntax in our current code, it had to be done with the migration.

We generated a list of all Cognos routines used in the accounts that were to be migrated. We used the SCAN function of SPEEDEDIT (Bradford) to locate them and to find any obsolete syntax they might contain. We also checked for any code that might work differently on the new release.

Updating Program Code

The next step was the removal of obsolete syntax. We wrote an EDITOR batch routine to do most of the obvious changes, such as string to character. Then we looked through each program to see if there was anything else that might need modification. Running the Cognos products with INFO = "OBSOLETE=ERROR" was helpful in finding anything that was missed. The changes we made were minor and if you had only recently begun using Cognos, you might not need to do this step at all.

Notice that your QKGO files must be upgraded to the new version. The QSCHEMAC should not be re-compiled using 5.03D QDD until you have re-compiled all other routines because 5.03D routines can use a 5.01F schema but 5.01F routines can NOT use a 5.03D schema.

Testing

We tested every screen, report and QTP routine. First, we needed to make sure that they would run without errors. Then we ran representative reports of each type using the same options and databases with version 5.01F and again with version 5.03D. These reports were then compared to be sure that format and contents matched. Then we ran all routines that needed more than just obsolete syntax changes on the Series 70 using the old version 5.01F to be sure results would still be the same when they were used in that environment.

Problems and Solutions.

Dates.

During testing it was noticed that the default dates that should have been today were yesterday. This was a problem with the synchronization of the hardware and software clocks on the machine. Our Hewlett-Packard System Engineer fixed this by bringing the system completely down and setting both clocks.

"End of Program" overwrote fields on screen.

One of our screens runs a program while data is displayed. When that program finished, the end of program message overwrote some of the data; it didn't change results, but would have been confusing to our users. We fixed this by immediately doing a refresh upon completion of that program.

QUIZP and QTPP.

QUIZP and QTPP are part of previous versions of Powerhouse. They use PM and run faster. They are not part of version 5.03D. Cognos customer support says they are no longer needed because QUIZ and QTP run so fast in the new version. We didn't see that mentioned in their documentation. We changed UDC's for QUIZP and QTPP to run QUIZ and QTP respectively in order to minimize modifications. This approach also allowed us to continue to use these programs on the series 70 machines with the same code. We again used the SCAN function of SPEEDEDIT (Bradford) to find all explicit runs of QUIZP and QTPP and then changed them to run QUIZ or QTP.

QTP Aborts on intrinsic GETDSEG.

EAI uses OMNIDEX (DISC) for accessing IMAGE datasets through non-key fields. This product is not yet available in native mode for the Spectrum series machines. Their SL routines use the intrinsic GETDSEG. This caused QTP in particular to abort. We couldn't give DS capability to QTP as none of our tools will work on native mode programs. We were stuck. To work around this problem, we decided to keep a schema compiled with 5.01F on the machine and run that version of QTP whenever an OMNIDEX key must be updated. For all other QTP runs the native mode QTP is used.

QTP KSAM Access.

We found that a QTP routine that accesses a KSAM file but does not update it gets the error, "operation inconsistent with access type" on the KSAM file and terminates. We fixed this by changing the routine statement, SET LOCK FILE RUN, to SET LOCK FILE REQUEST. Cognos has generated a patch tape (D7) for version 5.03D that is supposed to fix this problem.

QKGO.

Our QUICK routines would abort at initialization. The error message told us the common area size was too small. The common area size should be set to more than 0, (Zero is the default.) even if you do not use DO EXTERNAL.

QUICK Fatal Error 251.

Epson uses QUICK to build a STREAMX (VESOFT) job which when executed will use that user's log-on in the job card. When trying to run that job, it should request character input; STREAMX didn't request the input, but ran the job anyway. QUICK aborted with fatal error 251. This occurred when the response was input at the information response line of the routine. Cognos says error 251 has to do with the roll back buffer. We moved the STREAMX run into a menu screen and it worked fine.

Subscreen error.

A subscreen that used a file opened by an upper level screen generated the error, "illegal operation due to multiple file access" when STREAMX tried to run the job that the screen created. That job file was accessed and closed in the immediately upper level screen. We changed the subscreen to open that file separately to fix this.

Installation.

We installed 5.03D in one evening on the first machine. We had some problems with some screens being left out of the compile job and one screen that was not compiled because the source was missing. The next week-end we installed on the second machine. Naturally, the second install was much easier than the first. The extensive planning and testing paid off.

Post-Installation.

Our users had been informed that their application was being converted to native mode. The problems they found were minor. A couple of screens had not been re-compiled. We did the required compiles to take care of that. Our users set off jobs to run some reports themselves. They can look at SHOWJOB to see that their jobs are running. We had users asking what happened to their jobs - and finding that they were finished and were already printing! The speed really surprised them.

During our analysis, we read about the real number changes for native mode. Since we are primarily a COBOL shop, we use packed numbers and integers. For that reason, we thought we would not have any problems in this area. WRONG! When a number is defined in Cognos, the default is a float size 8. Floats are real numbers. Extract files created on the Spectrum machines, that contained this type of fields could not be read by the 70's. They caused Cognos to abort with a fatal error. When we called Cognos customer support, they didn't know what it meant. Once we realized what the default is, and we knew what to do, we changed our extract programs to define numbers as NONIEEE FLOAT SIZE 8. The solution was found. You must remember this if you intend to move files between Spectrum and classic HP3000's. The error message just won't tell you what the problem is.

Benefits.

We were very pleased and surprised at the speed of native mode Cognos. Run times are considerably shorter. Some reports that previously ran in about 1 1/2 hours now run in 5 minutes or less. Some of the nightly processing now takes longer to check than to run.

A prior problem we had been working around when using 5.01F was resolved. In this application, records in extract files were on byte boundaries rather than word boundaries. This caused QUIZ to abort because it was unable to read the subfiles it had made. With 5.03D this problem just disappeared because of the way the operating system (MPE/XL) lays out files.

Observations on Vender Support.

In general we found Cognos customer support to be sporadic in their ability to help us with this migration.

We found Hewlett-Packard very responsive and helpful.

Conclusion.

We are very happy with the 5.03D release of Cognos. It exceeded our expectations.

Performance: How do I get more in both MPE/V and MPE/XL?

Francisco Alfredo Rego

Adager

Sun Valley, Idaho
83353-0030
U.S.A.

In December 1986 Dick Breon, noted HP3000 consultant, invited me to participate in the BARUG database benchmark. This was a welcome opportunity, since I have always dedicated myself (in a private kind of Technological Olympics) to the constant quest for more performance. In the process, I have faced many challenging technical issues which are relevant to *any* application that runs on the HP3000 computer (whether Classic under MPE or Spectrum under MPE/XL).

My theories are nicely backed up by numbers: What took over 4 hours on a Series 42 in 1987 took only 15 minutes on a Series 950 in 1988. My goal is to increase our performance dramatically, to bring our time down to 9 minutes on a Series 955 in 1989.

Naturally, theories are nice *but* we need to implement them to make them useful. I have been fortunate. Since I have enjoyed many different types of Hewlett-Packard computers as platforms for the various implementations of my theories, a bit of nostalgia is appropriate.

I learned my first lessons on Hewlett-Packard computer performance in 1974, when I worked on an HP2100 with RTE II at the Telephone Company in Guatemala. The challenge was to create an automatic billing system for international direct-dial calls. The resources were extremely limited by today's standards: 64k bytes of magnetic-core memory, 5 megabytes of disc (2.5 fixed and 2.5 removable), one paper-tape reader, one teletype with paper-tape punch, two 800 bpi tape drives (no read-after-write), 1 optical card reader, 1 CDC 600 lpm printer and 5 Datapoint 2600 terminals (with upper case only). HP did not make *everything* in those days!

From 1975 to 1986, I spanned the 70's and the 80's working under MPE on pre-Classic and Classic HP3000 computers. During 1987, 1988 and 1989 I have enjoyed the dramatic evolution of Hewlett-Packard's MPE/XL operating systems and HP Precision Architecture (Spectrum) hardware.

What have I learned in the last two decades? How does knowledge evolve? How can I apply this understanding in the decades ahead? In this essay, let's explore (in a nicely relaxed manner) *what* I have done, *why* and *how*.

Warm up

"Performance" is certainly a heavy subject. Let's just get the juices flowing and limber up stiff intellectual joints, as we would do before embarking on any strenuous physical activity. Let's also stretch our mental muscles by mentioning other important concepts which we cannot afford to ignore while we strive for performance, such as: clarity, relevance, ease of use, reliability, functionality, robustness, accuracy, maintainability, precision, privacy, security, and so on.

Let's take reliability and robustness as examples. A reliable and robust system requires a significant amount of software effort to check for (and hopefully correct) all kinds of unacceptable conditions. Some people might think that such effort detracts from the system's performance. I beg to differ: Designed-in reliability improves overall performance, because the system does not have to pick itself up all the time as it "walks" (or, rather, "stumbles") towards its objectives. Lots of *redundancy* is the price we pay for reliability.

Let's consider the computer's relationship with the human user as another illustration. "Performance" has but one final motivation: to improve the productivity of the end user. A machine that performs miracles but requires gurus does not perform as well as a less lofty machine that allocates a lot of its power to communicate nicely with the end user.

A powerful interface allows the user to specify, precisely, what the computer is required to do. Ideally, the system should immediately catch any ambiguous or potentially damaging requests from the user *and* enter into a friendly discussion before a serious misunderstanding develops. This way, the machine is less prone to go off in the wrong direction to do something inappropriate blazingly fast.

An intelligent user interface requires a lot of software and computer juice. It takes a lot of MIPS for the computer to reach the conclusion that a message like "Hey, you!" is in order. Particularly if the user gets the message on the spot, in time for corrective action.

On the other side of the interface coin, the computer should communicate to the user, in clear terms, the results of its tasks. Instead of a 6-inch thick report of *all* widgets, the user will be better off with a 1-page report of those widgets that, given some quick action by management, will increase profits by 63%. This way, the user is able to make sense of the whole thing in less time, with less frustration and with better understanding (so the user does not go off in the wrong direction to do something inappropriate).

Perhaps we must accept different standards of "computer performance" when we are dealing with a human and when we are dealing with a machine.

I believe that raw performance is a worthy goal when dealing with raw bits and bytes, with operating-system intrinsics, and with any machine-oriented aspect of computing. But let's keep our perspective and let's not get lost in the game of "performance per se". We can spend our life saying "let's see how many times we can run this loop in a second" instead of thinking "let's see if we can eliminate this loop while still performing the same function".

Running

Let's remember a strange computing paradox. Running *any* program on *any* computer (even the world's most efficient program on the world's fastest computer) has a measurable negative effect: the net result is a degradation of the computer's performance! At first, we don't notice any difference. But as we slowly add more and more running programs, the machine

eventually and suddenly comes to a dramatic and humiliating standstill. This is the infamous "knee in the curve" that we see in slides during HP performance presentations.

I became painfully aware of this back in 1975, when running on a venerable Pre-Classic HP3000 CX computer (which we brought from Guatemala to Sun Valley and scheduled for an appearance during the INTEREX 1989 San Francisco Conference). Since then, I established a policy of having at most one programmer per computer (in my personal case, I feel quite satisfied about the fact that I have three computers at home, two Classics and one Spectrum).

The same holds true for you: get as many computers as you can afford, to spread the load!

Inertia

Newton said, "*inertia* is a property of matter by which it remains at rest or in uniform motion ... unless acted upon by some external force". Now, after having warmed up, we can appreciate the significance of Newton's insight regarding our own bodies, celestial or not: it takes a while and some effort to get going.

The same happens to computer entities. Every time you use ":run", or "dbOpen", or "fOpen", or "hpfOpen", or ":prep", you create a minor crisis in your computer system, as it goes through the anti-inertial motions of *causing* something to happen. By the same token, you cause great grief whenever you use ":abort", or "deAllocate", or "dbClose", or "fClose", as the computer wraps up a few things on your behalf.

The moral? Don't spend energy working against inertia. Don't call on any resources until you absolutely need them. And once you have the resources, don't allow them to remain idle: exploit them at the maximum and then let them free so somebody else may enjoy them.

I have a personal example to illustrate the power of inertia. In 1985 and 1986, I met with some top-level managers at Hewlett-Packard regarding an enhancement request for IMAGE that would cure a chronic case of *inertiae databasis*. I also expressed the request at Doug Spreng's session during the Washington 1985 INTEREX Conference. In those days, I was told "not to worry", since that enhancement was part of the new HPIMAGE. Now that TurboIMAGE/XL is "in" and HPIMAGE (for the HP3000) is "out", we seem to be back at square one.

The inertial problem: Currently, if you want to change the value of a critical (search or sort) field in a detail entry, you have to do two high-overhead operations: (1) DBDELETE the entry and (2) DBPUT the entry again after having changed the values in your program buffers.

This approach ("the way we have always done it") causes some horrible happenings.

> (1) As you delete the entry with DBDELETE: The de-linking of all the chains where the entry is a link, the updating of the FreeEntry count (which is decremented by 1) and the addition of a new link to the FreeEntry list.
>
> (2) As you invoke DBPUT to re-add the recently-deleted entry: The re-linking of all the entry's chains, the updating of the FreeEntry count (which is now incremented by 1 to go back to what it was in the beginning anyway) and the deletion of the newly-added link to the FreeEntry list!

This is not very elegant. Not to mention its negative impact on performance. A solution: Enhance DBUPDATE (with a new mode, perhaps "2", to preserve the current behavior of those millions of current callers of DBUPDATE with mode 1). DBUPDATE mode 2 would allow

changes to critical detail fields. At worst, if you changed all the critical fields for all the paths, DBUPDATE would have to de-link and re-link all the entry's chains (but it would not have to update and re-update the FreeEntry count and it would not have to link and de-link the FreeEntry list). At best, DBUPDATE would have to de-link and re-link only one chain out of 16, leaving the other 15 untouched, for a well-connected dataset.

Several people have asked me, since 1985, why I bother banging my head against HP's inertial wall instead of writing my own replacement for DBUPDATE, which I could then ship to my customers. The reason is simple: This is an enhancement to IMAGE that *only* Hewlett-Packard can do. Anybody can modify the *static* database structures (such as root files and datasets) but only Hewlett-Packard can modify the *dynamic run-time* database procedures (such as DBUPDATE) and structures (such as DBCB's, DBG's, DBUX's, or whatever). So, once again, I bring this topic up, hoping that some perceptive high-level Hewlett-Packard manager will pick it up!

Workout

Now that we have warmed up with a few casual topics, we are going to increase our heart rate by touching on a few controversial subjects. Let's begin with trying to define "performance".

What do we mean by "performance"?

There are, at least, two basic issues to consider: Throughput and response time. We, obviously, want to maximize throughput and minimize response time. Unfortunately, when we have *many* concurrent users, we usually end up optimizing a few at the expense of the others. If you ask "the few", they will tell you that the computer's performance is terrific. If you ask "the others", they will tell you unprintable things.

There are so many ways to define and measure performance! In your particular case, performance should be measured according to *your* definition, which ultimately depends on the productivity of *your* end users, following priorities specified by your management. One person's performance junk is another person's performance treasure. That's why this is such a touchy subject!

Consider the following: (1) a race car, with lots of horse power and raw speed which needs a smooth road surface; (2) a small 4-wheel-drive vehicle that can climb volcanos; (3) a mountain bike with just human power and very low gears and knobby tires; (4) a small, fragile helicopter that can land on delicate cornices at the top of snowy peaks. Each of these vehicles is a specialist, infinitely better than the others at a given task. Which "performs" better? All of them do, and none of them do! It all depends. The same applies to computers!

Performance and good citizenship

We have all been passed on the highway by a performance-oriented driver who almost blew us off the road. I don't believe that performance is incompatible with good manners toward fellow users of the same resources (be they roads, discs, memory, or whatever). The fact that

Spectrum HP3000 machines have much more memory (and neat facilities such as mapped files) does not mean that we should simply grab all the resources. We should still be considerate and use only the resources that are necessary (and sufficient) to achieve decent performance for ourselves and for our fellow users. From this perspective, we should try, even within the riches of an MPE/XL system, to follow the discipline that MPE V *imposed* on us.

Conceptual models: Bit maps and stacks

A good conceptual model is a map that will guide you through the performance wilderness. A lousy conceptual model will get you hopelessly lost in time-dated performance trivia.

For instance, instead of moving massive amounts of data around, you may choose to appoint *ambassadors* or *class representatives*. The ultimate example is a bit map. On a Classic (MPE) HP3000, you can place more than half a million bits in a single extra data segment. With a smart model, you can use the bit map to find out what you need about an entity in, at most, one disc access (to virtual memory, if your extra data segment was swapped out). Even better, since you will probably want to know a lot about lots of entities, the memory manager will automatically keep your extra data segment resident in memory and you will be able to get answers relating to hundreds of thousands of entities without a single disc access! Compare this with a model that would require you to wade through 2000-byte entries in a file instead of breezing through tiny bits in a bit map.

The idea is that it's a lot easier to shuffle symbols than to shuffle the entities represented by the symbols. The trick is to come up with smart symbols and good paradigms. Fortunately, Hewlett-Packard has championed some excellent ones, such as stacks and RPN.

Stack architecture and Reverse Polish Notation (so beautifully implemented in HP hand-held calculators and in the Classic HP3000 under SPL) are very powerful tools that you should consider keeping in your bag of tricks. The fact that the Spectrum HP3000 computers are register-oriented does not mean that you cannot implement your *own* stacks. As a matter of fact, I have been successfully using, since 1987, software-implemented stacks that are amazingly effective on *both* Classic and Spectrum HP3000 computers. I would go insane if I even tried to model the complex things required when transforming databases with recursive functions (such as adding a path to a non-existing dataset that will require fields whose items are also non-existing). I wrote the stack-management modules once and then forgot about them. Now I can spend my time and energy thinking about more useful things, such as mapped files.

Mapped files and extra data segments

If your application runs exclusively on the Classic HP3000, by all means use extra data segments for buffers. If your application *also* runs on the Spectrum HP3000, you may choose to take advantage of mapped files for some cases and you still may decide to continue using extra data segments for other cases. As a matter of fact, MPE/XL simply offers you more choices in buffering technologies. Depending on *your* quickness of mind, you can make one non-obvious technology (for instance extra data segments) perform much faster than some other obvious technology (for instance mapped files). Don't be swayed by buzzwords: after all, you measure performance with numbers, not with words!

Parallel processing

Serial processing is the simplest kind of scheduling, but it may also produce the slowest results from a performance viewpoint. Parallel processing may become a scheduling nightmare, but it may be the only way to gain significant performance improvements.

Scheduling involves the allocation of resources such as time, memory space, disc space, central processors, peripheral processors, etc. Parallel processing involves the *concurrent* allocation of resources. Not all parallel processing is necessarily good. Witness a street intersection: if we "allocate" all vehicles from all directions into the intersection without any rules at all, we will certainly get a messy collision. This analogy has been present in the minds of most implementors of parallel technologies, and the common vocabulary reflects it (for instance, the term *semaphore* means "traffic light"). As if scheduling vehicles through a busy intersection were not challenging enough, we must still allow for *priority* vehicles (such as ambulances or police cars) that, theoretically at least, should breeze through intersections, even against the flow controlled by automatic traffic lights.

Think of an orchestra conductor who has quite a few instruments as *resources* (assuming that they are all played by competent musicians, which is another story in itself). Most of the time, the composer calls into *active duty* only a small subset of all possible instruments. And, most of the time, not all instruments in this small subset are playing at the same volume with the same intensity. You might say that instruments have various priorities, which range from *nothing at all* (even less than *pianissimo*) to *all out* (*fortissimo*). The object of the game is to obtain the best performance possible. The person in charge of ensuring the necessary *dynamic* allocation of resources is the conductor, who should certainly follow the original specifications of the composer.

Absolutely the same holds true for computers! You can design systems that take advantage of *many* concurrent processors focused on various aspects of the same complex task, or you can design systems that present a single processor with many tasks to be performed on a given MultiBuffer that you have brought from disc to memory. In any case, you want to avoid extra "notes" and extra "musicians". If you can accomplish a task with one disc access, please do not use 20. Even better: If you can accomplish a task without going to disc at all, please don't go to disc at all!

Parallel processing implies organization of tasks before executing them. This brings us to the topic of normalization, which I have covered in previous papers. Let's review.

Normalizing

I have established this simple definition of *normalizing:* "Put together those things that belong together and separate those things that don't belong together".

The methods for achieving the various normal forms in database theory apply this simple definition in a step-by-step fashion.

Normalizing applies to time as well as to space. Note that *separating* can sometimes be easier than *consolidating* (two incompatible things can be in the same place at different times or at the same time in different places).

"Sorting things out" usually refers to the process of normalizing. Getting organized before embarking on a task usually minimizes wasteful operations. I always try to consolidate as many operations as possible while scanning a disc file. (See "Parallel Processing" above).

The normalization of time involves deciding the sequence of operations that has the highest likelihood of minimizing the total elapsed time. I think it's impossible (and udesirable) to eliminate all backtracking, but constant backtracking is a sign of poor time normalization.

I became fully aware of time normalization after I read an issue of INTERACT in early 1983. Before then, my software was fairly primitive in this regard. Thereafter, I spent a lot of effort to improve its interface (by allowing automatic recursive operations when a user request finds a missing prerequisite entity) and its efficiency (by designing a dynamic scheduler that minimizes backtracking during the actual mapping of database files). Implementing this all-important aspect of computer software is no easy task. It took me five years to design and construct my version!

Caching, monitoring and self-tuning of intensive I/O operations

By definition, an intensive I/O (input/output) operation spends a lot of time thrashing around between main memory and disc (not to mention other, even slower, peripherals). Instead of covering significant ground, the application just seems to run on a treadmill.

One obvious solution for this performance bottleneck is to eliminate (or at least to minimize) the unnecessary transfer of data, back and forth, between memory and disc. This is the idea behind the concept of caching.

There are all kinds of caching schemes: central processor caches, MPE caching, disc-controller caching, and so on. I certainly like all these caching technologies and I use them whenever it is appropriate. But there is one kind of caching that has turned out to be the most effective, according to our measurements in the Adager Labs: Application-defined disc caching. Central processors, operating systems and disc drives do not know the application; therefore, they can at best guess what is going to happen next. The application itself, on the contrary, knows *exactly* what is going to happen next and can prepare for it by flushing cache buffers or by increasing/decreasing cache buffers or by performing any of the various caching operations.

An often-overlooked class of intensive I/O operation is *remote* file or database access. In this case, the thrashing happens between (or among) processors that are connected through some sort of communications link. You can bring two connected Series 955's to their knees (and you can allow two Series 37's to fly with the *same* task) if you do things stupidly on the 955's (and smartly on the 37's). It's as simple as that!

As an example of bringing two machines of *any* kind to their knees, take this task: "Access a remote database and report those customers who have not bought anything in the last 90 days". A not-so-smart way to perform this task is to run a local program that will examine *all* the records in the remote database (while passing them to your local computer through the communications line). If the entries of interest to you are relatively few, you will have wasted a lot of your preciously-narrow bandwidth transporting junk. A smarter way would be to run a *remote* program that will still have to access all the remote records (remotely for you but locally for the remote program). This is actually not too bad, since the communications line will not be touched at all while the remote program goes through entries that do not interest you. If and when it finds qualified entries, the remote program will send them to your local computer.

In this example, the performance figures are vastly different, depending on your choices of what to do locally versus remotely. It is amazing that you have basically the *same resources*. You are just orchestrating them differently!

Batch processing and sorting

Accessing a file *randomly* is an incredibly expensive operation, if we have to access *most* of its entries to accomplish a task (which is typical of *batch* processing).

Sorting is a way to reduce a file's randomness according to *one* dimension (a primary sort key and optional secondary keys). As a side effect, we usually increase the file's randomness according to its other dimensions, if any (the other non-key fields).

The HP3000 computer has been blessed with a good sort subsystem. Unfortunately, in the Classic (MPE) environment, programs with large data stacks did not have enough space to take advantage of direct calls to the sort intrinsics. Fortunately, several people developed ingenious ways to work around these limitations. In the Adager Labs, we developed an internal subsystem called "fwSort" (in Fred White's honor, since he worked out all the algorithms). Other companies have developed commercial sort subsystems that they market to the general public. And, naturally, Hewlett-Packard is constantly working to improve the performance of the HP3000 System Sort. So, there is no excuse for not sorting in batch mode!

A small ancient computer (such as a Series 44) may outperform a big modern computer (such as a Series 925) in batch mode *if* a big file is nicely sorted on the Series 44 and hopelessly randomized on the Series 925. You would be using the 44's miserable MIPS in an effective way and you would be wasting the 925's awesome MIPS thrashing all over the place. Remember the tortoise and the hare? Complacency is a fatal flaw!

OnLine Applications and their data structures

Accessing *all* of the entries of a file sequentially is crazy, if we only have to access *a few* entries (which is typical of *OnLine* processing).

My definition of *OnLine* is: "If you can answer a question over the counter or over the telephone without first having to say *please have a seat* or *I'll call you back*, then you have a true OnLine application".

For OnLine applications, sorting large files on the fly is out of the question, at least with the current sorting technologies, even though they can sort millions of entries in a few minutes. You would still have to say "please have a seat" or "I'll call you back", and you would not have an OnLine application any more. For OnLine applications, we must use techniques based on hashing, trees, lists, stacks, and mixtures of these and other data structures.

A humble computer (such as a Series III) may outperform a fancy computer (such as a Series 955) in OnLine applications *if* a big file is nicely structured on the Series III and carelessly disorganized on the Series 955. You would be using the III's lowly MIPS in an effective way and you would be squandering the 955's magnificent MIPS by sequentially scanning the whole file whenever you need to access a single entry. Remember David and Goliath? Self-satisfaction accompanied by unawareness of actual dangers or deficiencies is catastrophic!

Performing a square dance on a round table?

We can now see that "performance" is a relative thing. Using sophisticated data structures intended for OnLine use (such as IMAGE databases) in an exclusively batch environment will degrade performance and will waste valuable disc space. By the same token, using sophisticated sorting techniques intended for batch use in an exclusively OnLine environment will drive your waiting customers towards your competition.

Remember the distinction between "efficiency" and "effectiveness". You should be effective (i.e., you should do the right thing) and you should be efficient (i.e., you should do it quickly). You must not fall into the trap of being efficient while doing the wrong thing (or being inefficient while doing the right thing).

Cool down

It is now time to decrease our heart rate by mentioning a few calming items. After all, we could go on forever and we have to stop at some point, before we collapse from exhaustion.

A cool issue nowadays is migration from Classic to Spectrum machines. My experiences of 1987, 1988 and 1989 can be summed up by a few questions and a few answers to these questions.

Where do we stop? How much is enough? Compatibility mode? Object code translation? Native mode? Optimizing compilers? Before you worry sick about getting into these kinds of things, I would recommend that you check for the less esoteric kinds of things that we mentioned in this essay. Do a lot of macro optimization of performance *before* you fiddle with micro optimization. Take an infinite loop as an example. You can certainly execute an infinite loop most efficiently in native mode, but doing so would be even more stupid than executing it (less efficiently) in compatibility mode, because of the higher expectations.

An amazingly simple progression that has worked wonders for me is:

(1) Move your software from the Classic HP3000 to the Spectrum HP3000 in compatibility mode. Hewlett-Packard did a *fantastic* job with compatibility mode.

(2) Use the Object Code Translator to increase the performance (and the size!) of your programs. Hewlett-Packard did an unbelievable job with OCT.

(3) Incorporate specific procedures in native mode to be invoked *automatically* by your software when it detects that it is running on Spectrum hardware. Don't bother to do this for rarely-used stuff: Do your native-mode song and dance for those operations that you will perform a million times on raw bits and bytes.

(4) Go fully native with your current programs if you have the spare programming resources!

(5) Write your new applications in native mode, but don't forget to use sound management practices. The fact that you are taking advantage of Hewlett-Packard's new Precision Architecture hardware, new Operating System software and new optimizing compilers is not a license to kill. **Don't let your performance guard down, even when you have MegaMIPS at your disposal!**

Understanding Migration

Victoria Shoemaker
Taurus Software, Inc.
770 Welch Road Suite 3A
Palo Alto, CA 94304
(415) 853-6893

Introduction

If you're considering purchasing an HP3000 Series 900 machine, then there's a very good chance you will be "migrating" once it arrives. By migrating, I mean moving your existing programs and data files from your "classic HP3000" to your new "Spectrum Machine".

Migration has never been much of an issue within the Hewlett-Packard world, because up until now, all HP3000's have had the same hardware instruction set. This means that compiled programs will run on any HP3000, with the only difference being performance.

As you probably know by now, the Series 900 machines have a completely different instruction set than classic HP3000's. This instruction set has fewer instructions than traditional computer instruction sets and is based up RISC (Reduced Instruction Set Computer) technology. Hewlett-Packard calls this technology Hewlett-Packard Precision Architecture (HPPA). Within this paper, the terms HPPA machine, Spectrum machine, Series 900 machine, and MPE XL machine will all be used interchangeably to indicate HP's new computer. The term "classic HP3000" will be used to indicate older machines that run MPE/V.

When you begin learning about migrating and the new HPPA machines, you soon realize that there is an abundance of terms you've never heard before. Terms like "native mode", "mapped files", "Sysgen", and "MPE XL" are some of them. This presentation will focus on what these terms mean to you and your migration. Specific topics that will be discussed are as follows:

1. What is migration?
2. What is the difference between Compatibility Mode and Native Mode?
3. How do I prepare for migration?
4. What are some of the new MPE XL features and commands?
5. What are some of the new MPE XL utility functions?

What is Migration?

Migration is simply moving programs and files which run on MPE/V based machines to HPPA based machines. The question then is: "Why is any migration effort needed at all?"

When the HP3000 was first introduced, it had an operating system called MPE, meaning Multi Programming Executive. Subsequent revisions of MPE were released, and most of you are probably now running the MPE V/E operating system. On the HP3000 Series 900 machines, the operating system is called MPE XL. The operating system on all classic HP3000's is written in a language specifically designed for the HP3000 hardware called SPL, Systems Programming Language. For the most part, MPE XL is a complete rewrite of MPE/V in a language called Pascal/XL. Pascal/XL is standard Pascal plus a variety of programming extensions added by Hewlett-Packard.

When MPE XL was designed, paramount importance was given to the issue of compatibility. With few exceptions, MPE XL is, and was designed to be, completely compatible with all MPE/V software. To make all MPE/V software run on MPE XL, Hewlett-Packard had a big problem because all MPE/V programs contained instructions for the old hardware which won't run on the new machines. The solution HP chose, was to write software that would emulate the classic HP3000 hardware and to integrate this emulator deep within MPE XL. By doing this, Hewlett-Packard has made the differences between MPE/V and MPE XL almost completely invisible.

Compatibility Mode versus Native Mode

"Compatibility Mode" (CM) is the term HP has chosen to mean running a program that contains instructions for the classic HP3000. When a compatibility mode program runs on MPE XL, the instructions are actually being emulated by the emulation software.

Almost all, if not all, of your programs will execute in compatibility mode with little or no effort on your part. Compatibility mode programs are typically created on an MPE/V machine, and then moved with `:STORE/:RESTORE`, or `:DSCOPY` to an MPE XL machine. You can create a compatibility mode program by running an MPE/V compiler in compatibility mode on a MPE XL machine. Running an MPE/V compiler in compatibility mode generates object code with classic HP3000 instructions. This means they can run on the classic HP3000 using its instruction set or in compatibility mode on the HPPA machine emulating the classic HP3000 instruction set.

"Native Mode" (NM) is the term HP has chosen to mean the running of a program that executes HPPA instructions. A native mode program must be created by a native mode compiler on MPE XL, and will not run on MPE/V.

Native mode programs execute much faster than compatibility mode programs because the process of emulation takes time. On the average, a native mode program will execute about 12 times faster than its compatibility mode counterpart. Currently, there are four native mode compilers available on MPE XL. They are: Pascal/XL, FORTRAN 77/XL, COBOL/II/XL, and C/XL.

OCT

You are probably not surprized that HP has created a middle ground. The OCT (Object Code Translator) produces programs that can be executed in native mode and on the classic HP3000 with out having to recompile.

What the object code translator does is translate the classic HP3000 instructions contained in executable code (PROG or SL files) to native mode instructions. The native mode instructions are then added to the program or SL file, leaving the classic HP3000 instructions in place.

There are two reasons for leaving the old instructions in place. One reason is because the old code is used by the translated instructions, and the other is so that the translated file may be moved back to MPE/V and still work. It will still work because MPE/V ignores the HPPA instructions within the file. Translated programs tend to run about 4 times faster than untranslated programs.

Switching Modes

Sometimes it is necessary to have a program run in both native mode and compatibility mode. For example, a native mode COBOL program needs to call an SPL procedure. Since there is no native mode SPL compiler, it must remain in compatibility mode. Hewlett-Packard has provided a mechanism for doing this called the "Switch Subsystem".

The Switch Subsystem is a set of procedures that enable a program to call a procedure that runs in the opposite mode than the program is currently running in. In order to understand how the switch stub works, we need to understand the internal data structures.

On MPE/V, the only data structure that is created when the program is executed is the program's stack. (See Figure 1) However on MPE XL three different data structures are created: the Compatibility Mode Stack (Same format as MPE/V), the Native Mode Stack, and the Native Mode Heap. (See Figure 2)

Environment for MPE/V Program

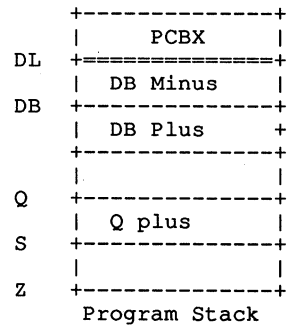


Figure 1.

Environment for MPE XL Program

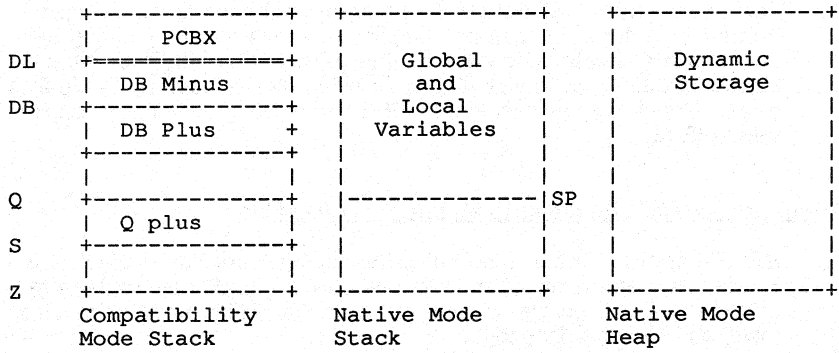


Figure 2.

When a program is executing in CM, it has access only to the Compatibility Mode stack. When a program is executing in NM, it has access only to the Native Mode Stack and Heap. Because of this, a "switch stub" is required to change modes. The switch stub is responsible for calling a switch procedure with the correct parameters to direct it to copy the procedure parameters from the current stack to the stack for the opposite mode. An easy fill-in-the-blank utility program is supplied with MPE XL to help the programmer write switch stubs.

Which is better for my program - CM or NM?

When it comes time for you to migrate, you'll be faced with many decisions, not the least of which is whether your programs will run in native mode or compatibility mode. The advantage of native mode is that your programs will run much faster. The advantage of compatibility mode is that you can move your programs from MPE/V to MPE XL and run them all within a matter of minutes.

In the long run, an attempt should be made to get as many programs as possible to run in native mode. Pascal and COBOL/II programs should not present much of a problem to migrate to a native mode compiler, and FORTRAN 77 programs should migrate to native mode easily but watch out for the IEEE floating point.

Relatively Easy Programs to Migrate to Native Mode:

1. Pascal programs. Pascal/XL has many new features over Pascal/V, however you should be careful when using them because they will not work on MPE/V.

2. Cobol/II programs. Cobol 66 programs will need to be changed to Cobol/II, before they can be compiled in native mode.

Potentially Difficult Programs to Migrate to Native Mode:

1. Fortran Programs. All Fortran 66 programs will have to be changed to Fortran 77 before they can be compiled in native mode. Floating point numbers are stored differently internally on HPPA machines than on classic HP3000's, so any floating point data that has been written to files or data bases must be changed to IEEE format using an intrinsic supplied with MPE XL.

Difficult or Impossible Programs to Migrate to Native Mode:

1. SPL Programs. There is no native mode SPL compiler available from Hewlett-Packard, although there is one called SPLASH available from a third party. SPL programs must run in compatibility mode or must be rewritten in another language.

2. Basic Programs. Basic/3000 will never be available in native mode on MPE XL, however HP Business Basic will be.

3. Programs that use privileged mode (PM). Many privileged mode programs will run correctly in CM, and many will not. Because the MPE XL operating system is completely different internally than MPE/V, there is very little chance that privileged mode programs can be run in native mode without major changes.

Native Mode Considerations

In general, there are two big differences between the behavior of compatibility mode programs and native mode programs. One is data alignment, and the other is floating point. Compilers like to align data on word boundaries; the trouble is that classic HP3000's have 16 bit words, and Series 900 machines have 32 bit words. This will cause the data to be stored differently in compatibility mode programs than in native mode programs. This could cause problems for programs that use existing data files, or programs that call external procedures expecting the data in a different format. Hewlett-Packard has supplied a compiler option for the native mode compilers to direct the data to be aligned the exact same way as it would be using a compatibility mode compiler.

Classic HP3000's use their own format for storing and manipulating floating point (Real) numbers, and HPPA machines use the IEEE Standard for storing and manipulating floating point numbers.

Single precision HP3000 floating point numbers have a precision of 6.9 digits and a range of ± 1.2E77 to ± 8.6E-78. Single precision IEEE floating point numbers have a precision of 7.2 digits and a range of ± 3.4E38 to ± 1.4E-45.

Double precision HP3000 floating point numbers have a precision 16.5 digits, and the same range as single precision numbers. Double precision IEEE floating point numbers have a precision of 15.9 digits and a range of ± 1.8E308 to ± 4.9E-324.

This format difference will probably have a negligible affect on mathematic results, however it is a problem for floating point data stored within files. Classic HP3000 floating point numbers stored within files, will have to be converted to the IEEE format using the HPFPCONVERT intrinsic if you wish to access them in native mode.

Preparing for your Migration

HP has provided a number of facilities to allow you to prepare for you migration prior to your HPPA machine arriving. One of the utility programs which runs on the classic HP3000 is called RTM (Run Time Monitor).

The RTM is a utility intended to help you identify areas within your MPE/V programs that could cause a problem when ported to MPE XL. Because of fundamental differences between MPE/V and MPE XL, there are some things that may work on MPE/V that will fail on MPE XL. The RTM is intended to help you detect these things by logging calls to MPE/V intrinsics that could be a potential problem on MPE XL.

Logging is controlled using a program called `RTMSYS.PUB.SYS` and the logging results may be printed using a program called `RTMREP.PUB.SYS`. Typically, the user will monitor an application being considered for migration for a number of days. Then the RTM reports will printed and any potential problems will be investigated. The advantage of the RTM is that it runs on MPE/V, and problems may corrected long before migration actually begins.

OCA

Another utility which that can be run on the classic HP3000 to help your prepare for migration is the OCA (Object Code Analyzer). The OCA is a utility program that scans program or SL files for potential MPE XL problems.

It is intended to perform the same function as the RTM, however its method of operation is different. The RTM logs intrinsic calls when they actually happen; the OCA scans a program or SL file looking for intrinsic calls. The OCA has the advantage of being able to obtain the results immediately without waiting for days of logging. Its disadvantage is that it is may not be as accurate because it cannot always tell what parameters an intrinsic is being called with. Like the RTM, the OCA runs on MPE/V, and may be used before migration actually begins.

MPE XL New Features

A number of new features have been added to MPE XL machines. The features fall into three areas: intrinsics, mapped files, and command interpreter changes.

Intrinsics

As you may have guessed, new intrinsics have been added to MPE XL. Many of the new intrinsics have to do with switching between NM and CM. Others are provided to access features of the new command interpreter, including a new HPCICOMMAND intrinsic that can execute any MPE XL command including User Defined Commands (UDCs). Another new intrinsic is HPFOPEN that opens a file just as the FOPEN intrinsic does, but HPFOPEN is implemented with a couple of new features and is designed to be easily expanded.

Mapped Files

Using the HPFOPEN intrinsic, a program may now open a file "mapped". When a file is opened as a mapped file, a pointer is returned to the calling program. This pointer may be used to access the file directly without going through the file system. The mapped file may be treated exactly as if it was a data array within the program's stack. The advantage of mapped files is a tremendous performance improvement because the file system overhead is virtually eliminated.

MPE XL Command Interpreter

The MPE XL Command Interpreter has almost all of the commands that the MPE/V Command Interpreter has. The commands that have been deleted were deleted because they have no place within MPE XL. The deleted commands include CACHECONTROL, DATA, FULLBACKUP, GIVE, LISTVS, PARTBACKUP, PTAPE, and VINIT.

Some of the existing MPE/V commands have been modified or enhanced on MPE XL. Some of these commands are: IF has been greatly enhanced, LISTF has had some new options added, LISTACCT, LISTGROUP, LISTUSER have been changed to give output similar to LISTDIR5, REDO has been enhanced, RUN has had some options added.

Several new commands have been added to MPE XL that give dramatic improvement over MPE/V. The SETVAR, SHOWVAR, DELETEVAR, and INPUT commands have been added to manipulate variables. Variables are similar to JCWs except that they may contain string and boolean values in addition to numeric values. About 60 variables are predefined within MPE XL and they contain a variety of information about the users environment, such as the CI prompt, user's jobname, user, group and account among others.

A redo stack has been added that saves that last 20 or more commands. A DO command that does any command within the redo stack has been added, and a LISTREDO command that displays the redo stack has also been added.

A fast COPY command has been added that copies files 10-20 times faster than FCOPY. A PRINT command has been added to display the contents of a file, or prints the file upon a line printer without having to use an editor.

Perhaps the greatest improvement is the addition of command files. Command files are similar to UDCs except that they may only contain one group of commands, and they do not need to cataloged. Whenever a command is entered that MPE XL does not recognize, it is assumed to be a command file or program name, and MPE XL searches the user's group, PUB group, and PUB.SYS group file a program file or command file with the same name.

SYSGEN

There is no SYSDUMP program on MPE XL to configure your system. SYSDUMP has been replaced by a program called SYSGEN to configure your system and make cold load tapes. Backup is performed using STORE. SYSGEN has a much different approach than SYSDUMP; it is command oriented rather than question oriented. This approach is much more direct than SYSDUMP's.

In addition to SYSGEN, there is a program NMCONFIG that is used to configure your LAN (Local Area Network), DTC (Distributed Terminal Controllers), and terminals connected to DTC's. Since all terminals on MPE XL except the console must go through a DTC, the NMCONFIG program must be used to configure all of your terminals.

A program called VOLUTIL replaces VINIT and is used to configure your disc drives.

DIRMIG

If you wish to replace one of your classic HP3000's with a Series 900 machine, you will want to move your entire accounting structure along with all of your files to the new machine. Hewlett-Packard has supplied a program to assist you in doing this called DIRMIG. DIRMIG runs on your MPE/V system and creates a tape containing accounting and configuration information to be moved to MPE XL.

Utilities

Many of the system utility programs have been either removed or replaced on MPE XL. The following list summarizes the major changes:

1. LISTDIR5 has been removed. Its functionality has been moved to standard MPE XL commands. LISTF options 3, and is similar to the LISTDIR5 LISTF command. LISTF option 4 is similar to the LISTDIR5 LISTSEC command. The LISTUSER, LISTGROUP, and LISTACCT commands now have output similar to LISTDIR5, and the octal dump mode has been eliminated.

2. The FREE5 program has been replaced with the DISCFREE program. DISCFREE performs the same function as FREE5, but the output format is different.

3. SPOOK has been modified internally, but no differences are visible to the user.

4. The LISTEQ5 program has been eliminated. It no longer has a use in MPE/V or MPE XL because of the LISTEQ and LISTFTEMP commands.

5. DEBUG has been completely rewritten and does not bear any any resemblance to the MPE/V debugger. The MPE XL debugger is many times more powerful and makes extensive use of windows to display information.

Summary

The first step towards a successful migration is education. MPE XL contains many new things that at first can be overwhelming. What is comforting is that when you begin to use MPE XL, you don't even need to know you're using it. All of the commands you are likely to use perform just the same, and programs moved to MPE XL in compatibility mode just run. Only when you are ready to maximize the benefits of your new machine do you need to have a good understanding of the migration process.

A General Guide: Exploiting HP's Powerful New Precision Architecture

Jason M. Goertz
Mattedor Computer Services
5105 Highland Drive
Bellevue, WA USA 98006
206-746-0212

Introduction

The development of Hewlett-Packard Precision Architecture (HPPA) is arguably the most significant undertaking attempted by HP in their 20-odd year history in the computer industry. The development of HPPA served several purposes and accomplished several milestones, both for HP as a company and for its customer base. It is important for us as HP 3000 users to understand the power available to us, and how to exploit these features for the good of our applications and companies.

This paper will attempt to answer the question: "Why should I care about HPPA?" It will illustrate some of the main features afforded to us by HPPA. First we will examine how HP can exploit this new architecture for their purposes (and thus for their user base), and how these features can be incorporated into our applications as well. Just as important, if not more so, is to understand how the new features of MPE-XL can be used. We will examine not only the hardware architecture of this new breed of HP computers, but will examine many of the new features and will give examples of how these can be used to our advantage.

The technical level of this paper is intended to satisfy both management and technical readers. Managers will find some technical detail which they can gloss over, while technicians will see enough detail to know the techniques and ideas presented are possible. Further detail can be obtained from HP manuals and other sources.

Fundamentals of RISC and HPPA

Before discussing any of these topics, lets first see a brief description of HPPA and RISC.

The fundamental concepts which shaped HPPA were those proposed by the paradigm of Reduced Instruction Set Computing (RISC). RISC was first proposed as a theoretical concept several years ago. The precepts of RISC are: only hardware-based instructions, no microcode, and an instruction set that is very small and compact. The original RISC documents espoused an instruction set of around 30 instructions. The MPE V-based HP 3000 has an instruction set many times that size, counting the instructions specifically designed to assist COBOL II. This does not include the various addressing modes possible with these instructions, which could easily count as additional instructions. All that is really needed, so say the advocates of RISC, are simple arithmetic, control, branch, shift, load and store instructions. This simple set of fundamental commands will allow any computer to execute any type of program.

HPPA can indeed be called a RISC implementation. However, it is unfair to describe it in these terms only. HP Precision Architecture has the following attributes:

o An instruction set that is very simple. While HPPA has over 30 instructions, they are limited

to the general classification presented above. They do, however, execute in one clock cycle (8 Mhz for the HP 3000 Series 930), except for some load, store and branch instructions, which execute in two cycles. The size of the instruction set is large compared to a "pure" RISC machine at 140 instructions.

- A large number of general-purpose registers, all of which are visible to the software.
- A scheme which allows programs to address extremely large areas of memory.
- A very flexible and simple I/O architecture, which allows drivers to be easily written.
- Complex operations (instructions) implemented in the compilers, allowing compile time optimizations not possible with microcode. In addition, the compilers are equipped with an optimizer that greatly reduces code sizes and modifies the instruction stream to maximize the instruction pipeline.
- Virtual memory address translation performed by very fast hardware. Infrequent exceptions are handled with software (kernel) assist.
- Four-level ring protection, allowing for multiple definitions of "privileged mode."
- Separate instruction and data cache memory areas, which can be manipulated explicitly by software.

The specific features of this architecture were determined by very careful measurement and studies performed by HP Laboratories. Billions of instruction sequences, taken from actual customer's machines, were analyzed to determine the exact instructions that should be implemented. Because of the precision of these measurements, HP chose to call the architecture HP Precision Architecture.

Why Did HP Choose This Path?

HPPA followed the precepts of Reduced Instruction Set Computing (RISC) when designing the fundamental architecture which makes up HPPA. HP made history with its introduction of HP 3000 900 Series and HP 9000 800 Series. The delivery of these machines marked the first large scale products delivered to market by a major computer vendor that incorporated the RISC paradigm.

Because of the simplification afforded by RISC concepts, HP is afforded numerous advantages in the manufacturing process. Using modern hardware design methodologies, primarily CAD software, HPPA processors can be designed in relatively short periods of time. Because of the hardwired nature of RISC and thus HPPA, the processors are very fast, opting for hardware execution of instructions as opposed to the more common use of microcode. Finally, HPPA designs can be produced very cheaply in virtually any type of semi-conductor technology available today. Currently, HPPA machines exist using off-the-shelf Schottky-TTL logic circuits (HP3000/930 and HP9000/840), HP's NMOS-III VLSI chipset (HP3000/950 and 955). Other technologies can certainly be employed, such as ECL, CMOS, and perhaps GaS (Gallium Arsenide) when this becomes more common and cost effective. Single HPPA processors could be built with MIP ratings of 40 or 50 or more, and when used in multi-processor systems, MIP ratings in the hundreds would be possible. These ideas could come to market as soon as the mid 1990's, allowing for incredibly powerful machines at a relatively low cost.

Not only does all of this allow HP to be more profitable, and thus be around in the future, it lends itself to providing a steady stream of newer, more powerful products to meet the continued growth needs of HP's customer base in the future.

Now that we have discussed why HP would choose this path of development/manufacture, let us look at the specific features of HPPA and their value to us.

Since a full description of the HPPA architecture has been given several other places[1], we will discuss here only those facets of the architecture of which we can directly apply.

Larger Memory Addressing Capability

Probably the most exciting feature now made available to users of HP 3000 computer systems with these new machines is the large addressing capabilities provided to the programmer. To those of us who have labored for so many years under the 32K stack limitation of the Classic 3000, this feature is indeed manna from heaven. In a nutshell, this addressing capability means that the programmer can now declare very large arrays within programs for easy manipulation. HPPA extends this concept even more by allowing the space to be used to access data on disc, which MPE-XL calls mapped files. We will look at how addressing is done, then look at its applications in both areas.

Theory of HPPA Addressing

HPPA defines an entity called an *address space*. Technically, an address space is a range of contiguous byte addresses, accessible by user code. The addresses can range from 0 to 2^{32}-1. These byte addresses are mapped to physical memory, disc addresses, or both by hardware and software. It is important to understand that an address space in and of itself really has **no physical reality at all**. It is only the mapping to memory or disc which gives it reality. An address space in its purest form only represents **potential** address that can be accessed by software.

The concept of a space is often hard for people indoctrinated in the Classic HP 3000 to understand. The closest thing conceptually to an address space is the data or code segment. It too is a contiguous range of addresses which can be accessed, but the analogy breaks down quickly after this point. While segments are sharply differentiated by the hardware by what they contain (code or data), spaces are not. Data segments are modifiable, code segments are not, properties which are enforced by the instruction set (microcode). Spaces, on the other hand, can contain code, data, or a combination thereof, and can be created with read, write, read/write or execute access. Instead of the contents of the space defining its character and access rights, as is the case with segments, the contents and access rights of a space are just another characteristic of that space.

Another difficulty is in how the space is allocated. Segments are given physical manifestation upon creation. When a data segment is created, an area of virtual memory is allocated, and that data segment "lives" at that disc address at all times for its lifecycle. Code segments come from either a program file or SL file, and the disc address of that file becomes the swap area where a fresh copy is brought into memory when needed. Additionally, a segment is an atomic unit. It cannot be separated into components. When a stack (data segment) is swapped out, the IO that is accomplish writes the entire structure out to virtual memory. Likewise when a data segment is brought in. Address Spaces do not share these properties at all. Allocating even one entire space on today's machines would be very difficult if the entire thing had to be present at once. An entire address space represents about 1 1/2 Eagle disc drives, a large investment for any company. Allocating the entire space in memory is actually impossible. HPPA defines 1 **real** address space (0-4 Gbytes) for the semiconductor memory and the IO hardware. Since the IO hardware takes up about 1/4 megabyte by itself, and MPE-XL has several megabytes of tables and code, there is not enough room to allocate an entire space in real memory. Therefore, HPPA defines fixed length **pages** of 2K bytes

---

[1]. *Beyond Risc!*, chapter 5. Wayne Holt, et al. Software Research Northwest, Inc., publisher. "Beyond RISC - A Technical Evaluation of HP Precision Architecture", *SuperGroup Magazine*, Vol 6, No. 12 & Vol 7, No. 1, SuperGroup Association, publisher.

each[2], which are the units actually manipulated by the machine. Unlike the Classic 3000, disc space for these spaces is not in a fixed location on disc (like virtual memory) but is spread out over all discs and is called **transient space**. (Permanent files are kept in **permanent space**). An attempt is made to keep the transient space together for contiguous pages, and the memory manager optimizes the IO's to and from these areas of disc.

In summary, parts of a space can be allocated and manipulated randomly in fixed length pages. Segments are allocated and manipulated *in toto*.

HPPA further defines that many spaces can be created and active at once. The architecture defines different **addressing levels**, which really define the maximum number of spaces possible. Level 1 systems define a space number as being 0-64K. The HP 3000 Series 930, 950 and 955 are all level 1 machines. Each space, when created, is assigned a number by the OS, and this number is then used to access that space. The accessing is done automatically by the memory load and store instructions when the D and C bits are set in the Processor Status Word[3].

Figure 1 shows the use of the registers to perform this addressing. HPPA defines 8 **space registers**. Before a memory access is performed, the proper space register is loaded with the space number (commonly called space id, or SID). One of the 32 general purpose registers is loaded with the desired byte address. The load and store instructions specify which space register to use. The net effect is that a 64 bit address is created, which the hardware uses to load or store a byte, halfword or word (32 bits).

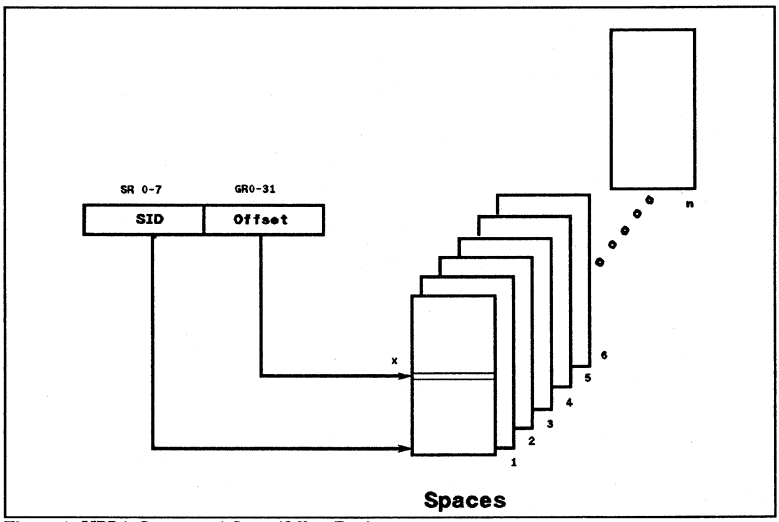


Figure 1. HPPA Spaces and Space/Offset Registers.

---

[2]. This figure is defined by the hardware architecture. MPE-XL actually manipulates pages in 4K increments, thus reducing the number of IO's performed by the memory manager. For our purposes, the difference is not important.

[3]. These bits instruct the hardware to perform virtual address translation. When off, then the 32 bit real address space is accessed. This mode of access is used exclusively by the very lowest levels of the operating system.

Two types of virtual addressing is possible. If the space register specified is 1,2, or 3, then **long pointer** addressing is being employed. In this method, the software must load the proper SID into one of these three registers each time it is used. If space register 0 is specified in the load or store instruction, the hardware executes **short pointer** addressing. In this case, the upper two bits of the byte address (stored in the general register) is added to 4, and this space register is used. Thus, for a byte address whose upper two bits is zero, space register 4 is used. This continues through space register seven[4].

Since long pointer addressing requires SR1,2 or 3 to be loaded, more code is executed for each memory access. It is desireable to optimize the most common access, and this is indeed what short pointer access is for. The space registers used by short addressing (SR4,5,6 and 7) are pre-loaded by MPE-XL with the SID's of the most commonly used spaces for a given process. These are the SID's of the currently executing code and the process's private data area. In addition, since a great deal of a process's execution is done within the operating system, MPE-XL loads the SID's of it's code and data as well.

Along with this optimized approach to data access comes a penalty. In order to make the hardware easier to build, the upper two bits (used to determine the desired SID) is considered as part of the byte address to access. The net effect is that only 1 quarter of a 4 gigabyte space can be accessed via a given space register using short pointer addressing. Each of these quarter spaces is called a **quad**, and is illustrated in Figure 2. While at first this sounds like a waste, on further examination it is not. First, 1 gigabyte is still a large amount of address space. Further, since only those parts of a space that are actually used are allocated, the other 3 quads are not really wasted space. These quads represent virtual address ranges that are merely not used.

Note that in either method, a 64 bit address is ultimately used to access the virtual space. The only difference between short and long pointer access is how the space register (the upper 32 bits of the 64 bit address) is specified in the load and store instructions.

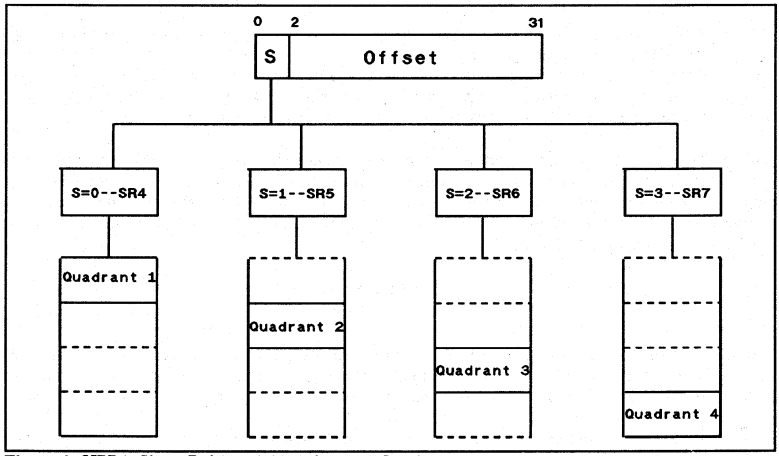


Figure 2. HPPA Short Pointer Addressing and Quadrants.

---

[4]. Since a zero in the load or store instruction implies short pointer addressing, space register 0 is not actually accessible. This register is used only as the object for the Branch and Link External (BLE) Instruction.

Application of Virtual Addressing

Now that we have seen how the machine performs the accesses to the virtual address space, let's see how we can apply this technology.

The most obvious use of up to 4 gigabytes of space is to allocate large arrays and buffers in a program's data area. Any application needing large data areas can benefit from this capability. Statistical analysis comes to mind offhand. Thus, a COBOL data definition could be:

```
01    STAT-SUMMARY-TABLE.
      03    STAT-SUMMARY-ENTRY  OCCURS 10000000 TIMES,
            INDEXED BY STAT-SUM-NDX.
            05    STAT-TOTAL          PIC  S9(09) COMP.
            05    STAT-NUMBER         PIC  S9(09) COMP.
```

On the Classic 3000, large data areas like this have to be implemented with multiple extra data segments. In this case, relatively complex software would be written to figure out which extra data segment actually contained the entry desired, then either retrieve or update it from WORKING STORAGE. Using the large address space provided by HPPA, this extra work is not necessary. In addition, since the data is within the data area of the code, it will reside within the private data space of the program, and thus be accessed via short pointers, which is more efficient than long pointer access, and **much** less overhead than using extra data segments.

The disadvantage of this method is that the compiler and linker create an image of the global data area which is placed in the program file. This is used at program load to time initialize the globals area in the memory stack. In the example above, this one array is 80 megabytes, and would force the program file to be at least this large on disc. This could be avoided by creating a non-dynamic subprogram. This would force the data area to be built the first time the procedure was called, and thus save the disc space of the program file.

For users of Pascal, another area of the private data area is available for use, called the **heap**. This area is expandable, and entries can be dynamically allocated and deallocated with special function calls built into the HP Pascal language. These routines return pointers which point to areas of the heap, and can be manipulated by the program. Unfortunately, the heap is **only** for use by Pascal. It does provide these users with a very nice and simple to use dynamic area.

The advantage of these capabilities is obvious to anyone who has ever written a program to manipulate large amounts of data. Very little code has to be written to manipulate tables like this, and very complex processing can be done very quickly. The bottom line is that certain very complex applications can be written very quickly, improving those applications and potentially giving the application's users an additional edge.

Mapped Files

An extension of the idea of large arrays of data within the programs data space is to have large data areas external to the program's space. Remember that the program's data area resides within its data quad. If more area is necessary, then mapped files can be employed.

As mentioned before, the virtual array is mapped (via MPE-XL tables) to specific disc locations, called extents. As the space is allocated, these areas are created and the links established. In a normal data space, when the space is deallocated (such as when a program terminates) the transient space's extents allocated to that space is released for re-use. Thus, the mapping of the space is outward, from virtual memory to disc.

A mapped file reverses this method, mapping existing file extents in permanent space to their corresponding addresses in a virtual space. Figure 3 shows this relationship. In essence, the file

is viewed not as logical records, physical records and blocks, as with the Classic HP 3000. Instead, the file is viewed as just a byte stream.[5] Once the file is mapped, long (or short) addressing can be performed to modify or add to the file.

The real advantage this gives is the speed with which the file can be accessed. In a normal file system, layers of software must be called to access the file. Once the file is mapped under MPE-XL, the data in the file is accessed at hardware speeds. Large volumes of data can (theoretically) be moved around at CPU speeds, with absolutely no intervening software.

The really exciting thing about mapped files is that data on disc is now accessible as an array, or table. In essence, all disc data is just table entries of data, stored in a permanent fashion, as opposed to the transient nature of data stored in a stack. Now such things as sort algorithms, which are always described in texts in terms of arrays, can be written staying true to the paradigms presented in these texts. B-trees and other types of data structures can be treated similarly.

As with all things, the TANSTAFL[6] principle applies. Because the file system software is totally bypassed when mapped access is used, it is up to the programmer to make sure the file system is informed whenever some property of the file has changed, such as the current end of file marker. This, however, is a small price to pay for the speed and ease of data access provided by mapped files.

Due to the way memory management is handled under MPE-XL, however, certain applications are better performed by traditional file system access. Primary among these is large sequential file accesses, such as those provided by SUPRTOOL from Robelle. Again, this is due to the fact that true mapped access is done with no knowledge by the software. Whenever a new page is hit by the sequential access, a trap occurs which causes the Memory Manager to do a disc IO to bring in the page. This incurs a lot of overhead. The file system, since it knows that sequential access is being done, can ask the Memory Manager to **prefetch** a large number of pages, reducing the total

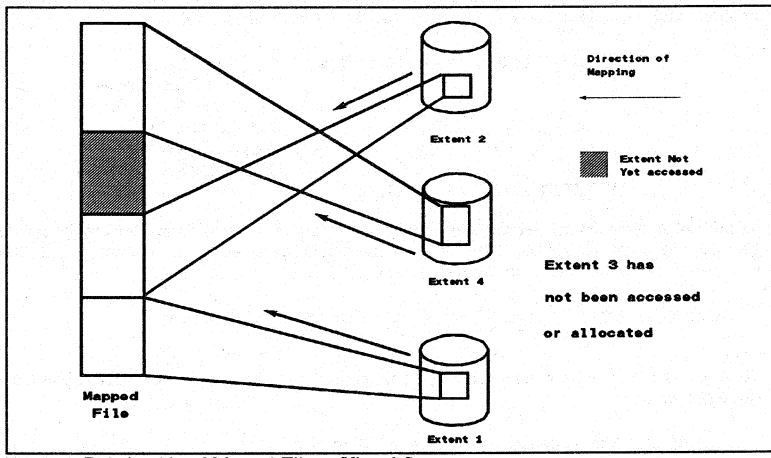


Figure 3. Relationship of Mapped File to Virtual Space

[5]. The fact that MPE-XL and MPE-V structure files is easily the subject of another whole paper.

[6]. There Ain't No Such Thing As a Free Lunch.

overhead imposed to bring the data into memory.

For this reason, sequential file access is not recommended using mapped files. However, random access to relatively small files can result in incredible performance benefits for the application.

Mapped File Example

Access to mapped files is only currently possible with three languages: Pascal/XL, C/XL and SPLash!. These languages can define and manipulate both long and short pointers. In Pascal, the definition would look something like:

```
TYPE
   ExampleFileType      = RECORD
                            Name,
                            Street,
                            City       : PACKED ARRAY [1..30]
                                         OF char;
                            Zip        : PACKED ARRAY [1..10]
                                         OF char;
                            State      : PACKED ARRAY [1..2]
                                         OF char;
                          END;

VAR
   LongFilePtr          = ^$EXTNADDR$ExampleFileType;
   ShortFilePtr         = ^ExampleFileType;
```

In the code, a call is made to HPFOPEN. This is a new extensible intrinsic provided under MPE-XL. HPFOPEN takes paired parameters, the first specifying a code describing the pointer, the second the parameter itself. To open the file with mapped access, you pass the pointer (either long or short) and the code telling HPFOPEN to pass back the pointer:

```
HPFOPEN(FileNum,MPEXLStatus,  2,FileName,
                              3,PermDomain,
                              6,FixedFormat,
                              10,AsciiType,
                              11,AccessType,
                              21,LongFilePtr);
if MPEXLStatus <> 0 then .....
```

At this point, the pointer is set to the beginning of the data in the file. To manipulate the pointer, the reserved construct ADDTOPOINTER is called to change where the pointer points. For example, to move the pointer to the next record:

```
ADDTOPOINTER(LongFilePtr,sizeof(^LongFilePtr));
```

To access the data, merely dereference the pointer just like you would when accessing any data through a pointer:

```
LongFilePtr^.Name := 'Jason Goertz';
```

At this point, the **data is changed in the file.** It's that simple. No calls to file system intrinsics at all. Of course, it is up to the memory manager to eventually post that page in memory to disc. If it is desired to make the data durable on disc immediately, then FCONTROL should be called by the programmer to post the data. (This is the same as on MPE-V).

While it may seem like the file system has to jump through some hoops to give you this pointer,

this is actually not true. The file system **always** deals with the files using mapped access at the lowest levels. This is the main reason that the file access on MPE-XL is very fast and efficient. The HPFOPEN call merely instructs the file system to return the pointer that it already has allocated.

Before leaving mapped files, a word on the difference between long and short mapped files is in order. Long mapped files use a 64 bit long pointer, and thus requires more code to access. Short mapped files return a 32 bit short pointer, but must be mapped into a commonly accessible space in order for this type of access to work. In the case of MPE-XL, the files are mapped into the system space, and are accessed via SR6 and SR7. For this reason, however, the size of short mapped file space is limited to 6 megabytes for a process. If more data is needed, long mapped files must be used. However, if 6 megabytes or less of data is required for the application, short mapping the file will result in significant performance gains for the application.

Again, the value of this feature is that applications and system tools can now be made to accomplish more with less work and fuss. Many applications manipulating large quantities of data on MPE-V systems must jump through many hoops to access that data. With mapped files, the data is accessible with a minimum of code.

A great deal of benefit can be derived by the user from this feature indirectly. That is, the benefits of others' use of mapped files will be seen. Probably the widest-spread application using mapped files is TurboImage/XL. Rather than using an Extra Data Segment for a file buffer as with TurboImage/V, TurboImage/XL performs all reads with mapped access. Writes, however, still are performed through normal file system intrinsics. This is done because Transaction Management (which is responsible for maintaining physical integrity of the files) must know the before and after images of the data records. Mapped file access bypasses software, so updating the data with this method is not possible. However, most IO's to data base files are reads, so the performance benefits of this HPPA feature are seen by every TurboImage/XL user.

MPE-XL Features

The other major change in the new Series 900 HP 3000's is the MPE-XL operating system. While billed by HP as being just an enhanced version of MPE, it is by no stretch of the imagination the same code with enhancements. MPE-XL is a brand new piece of software, written in a different language than MPE-V, and written with a different architecture in mind. It is, however, a superset of MPE-V. Virtually every feature of MPE-V has been incorporated, with additional capabilities. In the next section, we will examine some of these additional capabilities and how they can be used to make the 3000 a more usable and flexible computer.

Command Interpreter

While many things have changed in MPE-XL, the area with the most useful changes to applications is the Command Interpreter, commonly called the CI. The CI is the portion of MPE that prompts the user for commands, accepts them, then calls the appropriate piece of code, called a **command executor** to actually perform the command. In MPE-V, the CI is just a procedure which is created at logon time by MPE. In MPE-XL, the CI is a program in PUB.SYS, and can actually be run like any other program.

Compared to other machines that came before the 3000, the CI has always been relatively user-friendly and easy to use. With the enhancements in MPE-XL, this ease-of-use is extended even further, while still maintaining compatibility and the flavor of MPE-V's CI.

The enhancements are centered in four areas: Command Files, Variables, user IO, and Flow of Control Constructs. We will examine each of these individually.

Command Files

When UDC's were introduced in MPE-III in the late 1970's, MPE took a great step forward in allowing the user to configure the commands possible from his system. This ease-of-use is enhanced even further with command files.

A command file can be thought of as a UDC in a file by itself instead of being in a file with many others. While at first this may sound like a marginal advantage over UDC's, in actuality this gives the programmer/user a great deal of flexibility to manage these commands. An example will help explain the differences.

Lets take a simple case, where the main program of an application is to be run. Before this, a file equation is laid in with an optional file passed in the command line:

UDC	Command File
`File name = ORDUDCS`	`File Name = ORDENTRY`
`ORDENTRY FILE=ORDFILE`	`PARMS FILE=ORDFILE`
`FILE INFILE=!FILE`	`FILE INFILE=!FILE`
`RUN ORD001P.PROG;LIB=G`	`RUN ORD001P.PROG;LIB=G`
`**`	`**`

At first glance, the two appear to be identical. The difference is really in the very first line. In the UDC, the first line defines the name of the UDC command, and the parameters specified by the user. In the case of the command file, the first line just specifies the parms, with the command name being dictated by the MPE file name. Herein lies the main advantage of command files. While UDC's must be "set" by executing a :SETCATALOG command, the command file is active as soon as the file is created. The MPE-XL CI now has a hierarchy it searches when a command is entered. It first searches for UDC's, then command files, then for a program file with the name of the command entered[7]. Since command files do not have to be set, they can be changed with much greater ease than UDC's. Users do not have to re-logon to activate command files as with UDC's, and thus the system manager can activate new commands or change existing ones while users are logged on and running.

Variables

Probably the most exciting addition to the CI are command variables. This really is an extension of the idea of JCW's that MPE-V has had for many years. Philosophically, a JCW is just a 16 bit integer variable that can be set, read, and used in MPE :IF statements. MPE-XL has expanded this idea by creating variables of many different types, including strings and 32 bit integers.

The variables are set with the :SETVAR command. To use the variable, an exclamation mark ("bang" in Unix parlance) preceding the variable name instructs the CI to substitute its current value before executing the command. Some examples are:

```
:SETVAR RUNGROUP 'OBJECT'
:RUN ORDENTRY.!RUNGROUP;LIB=G

:SETVAR RUNPARM 123
:RUN ORDENTRY.!RUNGROUP;LIB=G;PARM=!RUNPARM
:COMMENT *** ORDENTRY SETS TRANTYPE VAR TO LAST
:COMMENT *** TRANSACTION TYPE
```

---

[7]. This actually is another nicety that the new CI has incorporated, the implied :RUN command. While a major enhancement to the ease-of-use of the CI, it is not important enough in and of itself to warrant more than a footnote.

```
:IF TRANTYPE = 'ENTRY' THEN
:   RUN FINENTRY.!RUNGROUP;LIB=G
:ELSE
:IF TRANTYPE = 'UPDATE' THEN
:   RUN FINUPD.!RUNGROUP;LIB=G
:ENDIF
```

Using these variables, it is possible to turn command files, UDC's or jobstreams into virtual programs. When coupled with the CIPUTVAR and CIGETVAR intrinsics, an incredibly powerful application environment can be created. The examples above barely scratch the surface.

Several built-in functions exist which allow for string manipulation. The plus ("+") operator allows strings to be concatenated. STR extracts substrings, POS locates the position of one string within another, and UPS upshifts a string.

An example of concatenation could be:

```
:SETVAR PROGNAME "S" + !SUBSYS + ".PUB.APPL"
:RUN !PROGNAME
```

Another predefined function is FINFO. This is essentially an interface to the FFILEINFO intrinsic, accessible from the CI. Codes are passed into the function (matching the codes passed to FFILEINFO), and a result is returned which varies with the type of information requested.

User IO

MPE-XL now has two new commands, :INPUT and :ECHO. These allow UDC's and command files to prompt the user for input, and display results. Here is a somewhat simple example that will verify that a file exists:

```
:INPUT PROMPT="Input File to be checked->";NAME=FNAME
:COMMENT FINFO(0) RETURNS TRUE IF FILE EXISTS
:COMMENT FINFO(1) RETURNS THE NAME
:IF FINFO(!FNAME,0) = TRUE THEN
:  ECHO "File !FINFO(!FNAME,1) exists"
:ELSE
:  ECHO "File !FINFO(!FNAME,1) does NOT exist"
:ENDIF
```

This presents an incredible labor saving device. In MPE-V, if it is desirable to have the user redirect the flow of control, a program has to be written, and several hoops jumped through. With these simple commands, the necessity to do this is null and void.

Flow of Control

All major programming languages have flow of control statements. MPE has now entered to this world with the addition of the :WHILE and :ENDWHILE commands. MPE has had the :IF, :ELSE and :ENDIF statements for many years. With the addition of the :WHILE statement, it is now possible to do a great number of things that had to be done by programs before. Coupled with variables, this can be a very powerful application tool.

For example, a simple UDC could be created to prompt for and purge files:

```
:SETVAR FNAME 'XXX'
:WHILE FNAME <> ""
:  SETVAR FNAME ''
:  INPUT PROMPT="Enter File Name->";NAME=FNAME
:  IF FNAME <> "" AND FINFO(!FNAME,0) = TRUE THEN
:     PURGE !FNAME
:     IF CIERROR = 0 THEN
:        ECHO "File !FINFO(!FNAME,1) Purged Successfully"
:     ENDIF
:  ENDIF
:ENDWHILE
```

The real weakness of this command is the conditional statements limited access to information. A tremendous addition to the CI would be the ability to define user-written procedures that could be called by the CI, that would return strings or whatever was appropriate. This would effectively give MPE something that other operating systems have had for years, user-intercepts for various functions. While this example is relatively primitive, it would be a major step forward in making MPE a truly usable tool.

Miscellany

There are several other features that have been added to the CI in MPE-XL that are nice or enhance the functionality described above, but do not warrant discussion in and of themselves.

Intrinsics

Several intrinsic additions and changes have taken place in the new CI. First, the COMMAND intrinsic can now execute the :RUN command, a deficiency that has been the bane of menu writers and the programmers of sophisticated application drivers for years. Virtually all other limitations of the COMMAND intrinsic have been maintained, foremost among them the inability to execute UDC's. This limitation was intentionally left in to maintain compatibility with MPE-V, the theory being that someone, somewhere, depends upon the COMMAND intrinsic to operate the way it does for security reasons.

To alleviate these glaring problems, a new intrinsic was created. Now, the HPCICOMMAND intrinsic performs all the functions one would expect an intrinsic of this nature to do. In essence, the HPCICOMMAND intrinsic duplicates the CI's execution path, allowing :RUN, compiler commands, command files, the implied run command, etc. The only commands no longer programmatically executable are ones that make no sense programmatically, such as the :IF statement.

As mentioned before, the CIPUTVAR and CIGETVAR intrinsics allow programmatic access to the variable interface. Again, this extends the JCW capabilities of MPE-V.

The CI now has a redo stack, which buffers up the last n commands, where n is configurable by the user. A new command, :LISTREDO, is included to list out the redo stack. To redo a specific command, the user can enter :REDO number, where number is the command number in the buffer. Or, more conveniently, the user can enter :REDO string, where string is the first few letters of the desired command. The CI will then search backwards for a command whose first characters match the string entered. In either case, the command is presented for modification to the user. If the command does not need to be modified, and the user just wants to re-execute the command, a new command :DO is provided. The use and syntax is exactly the same as :REDO, but no correction is allowed. As an example, if the user wants to redo the last :RUN command with no modification, he or she only has to enter :DO R. The CI will find that last :RUN command, display it on the screen, and execute it. This command is very handy in a program development environment where

a compile, a link, and a run command are entered often. As long as the first character of these commands are different, very short commands can be entered by the user to repeatedly execute these commands.

Conclusion

As stated before, MPE-XL and Hewlett-Packard Precision Architecture are by far the most significant advancements for HP. The use of the features provided can give a shop the most flexibility yet for transaction oriented applications. Using these features, information becomes easier to access, and businesses can obtain competitive advantages much more difficult to obtain with previous incarnations of the HP 3000 computer line.

"Classic HP3000" Emulation in Native Mode

Carl Burch

Data and Languages Division
Hewlett-Packard Company
19447 Pruneridge Avenue
Cupertino, CA 95014

As part of the overall software migration plan, the native mode compilers on MPE XL offer directives to compensate for the various architectural differences between the 16-bit ("Classic) HP3000s and the 32-bit Series 900 machines of the Hewlett-Packard Precision Architecture. A knowledge of these options and their implications can greatly aid the programmer migrating software to the Series 900s, both in terms of ease of migration and application performance delivered.

Degrees of Migration

There are four general levels of migration for MPE V programs to MPE XL. They are Compatibility Mode, Translated Object Code, Emulated Native Mode, and Full Native Mode. Each respectively gives more performance for more migration effort.

Compatibility Mode (CM) is the easiest and the slowest. Simply transfer the program and data files, then type :RUN - and it works. There are a number of cases that aren't supported, but I've never seen one. The performance can be expected to be roughly that of a Series 68 or 70, depending on your program and the model of Series 900 you're running on. As indicated in Figure 1, CM performance is significantly better than the Series 42 I use for my MPE V work.

Translated Object Code is produced by the Object Code Translator (OCTCOMP.PUB.SYS). It "compiles" the Classic 3000 instructions into HP-PA instructions and appends them to the CM program file. The MPE XL loader then causes the Native Mode (HP-PA) instructions to be executed instead of the CM program, unless a code sequence is encountered that must be emulated at runtime (e.g., XEQ TOS). For the time it takes to translate the CM program file, a very substantial increase in performance can be obtained (see Figure 1). For programs that use many CM services (i.e., calls to CM SLs like SL.PUB.SYS), translated code can beat Emulated Native Mode code (see the Whetstone and Distance Points benchmarks in Figure 1). Since translated code runs in compatibility mode (i.e., uses only data on the CM stack and routines in SL.PUB.SYS, not XL.PUB.SYS) but with native mode instructions, the cost of switching between CM and Native Mode in Emulated Native Mode can overwhelm the advantages of running the main program in Native Mode. The Whetstone benchmark displays this behavior because (in full Native Mode) it is about half math library calls, which require cross-mode switch calls to emulate in Native mode - but not for CM programs (even with translated instructions).

Emulated Native Mode programs have been compiled with the HP-PA compilers on a Series 900, but with one or more compiler directives in effect to emulate some feature(s) of the Classic 3000 architecture. Emulated Native Mode requires that source code be available. By inserting appropriate compiler directives and recompiling, a full Native Mode program can be produced with the compiler generating code to compensate for the architectural differences between the Classic 3000 and HP-PA. In particular, compiler directives are available that control alignment, floating-point format, and (in FORTRAN 77/XL) string-move semantics. Some or all of these factors may be needed to successfully migrate an MPE V program. This paper attempts to present the use of these options and their effects.

Full Native Mode (NM) programs use HP-PA alignment/packing rules, IEEE floating point, and do not depend on the "ripple-move" order of the Classic 3000 string move instruction (MVB). This can be a significant revision to some programs, or a complete NM migration may not be reasonable at all - as in the case of programs using IMAGE data bases that depend on IMAGE's 16-bit alignment

rules or containing Classic 3000 format reals. While it is possible to fully migrate any program regardless of the features used, in many cases all that would be accomplished would be for the applications programmer to do the emulation chores instead of the compiler. For this reason some programs will stop short of full NM on the migration trail.

Performance of the four Migration Levels (CPU Seconds) *					
	S/42	CM	OCT	Emulated NM	Full NM
D.P. Whetstone	100.781	82.832	58.213	81.354	5.031
BnchIO	239.871	90.894	88.904	18.189	17.469
Sieve	6.624	13.633	2.890	0.347	0.347
Distance Pts.	809.000	510.000	355.000	760.000	96.000

Figure 1.

In Figure 1 (as well as the remainder of the paper), S/42 represents the time to execute the program on an HP3000 Model 42 running MPE V version G.03.05; CM represents execution of the program file from MPE V in Compatibility Mode on the 930; OCT represents execution of the Compatibility Mode program after translation by the Object Code Translator; Emulated NM is the same source program compiled by HP FORTRAN 77/XL with $HP3000_16 and level two optimization on; Full NM is the same source program compiled with just level two optimization.

Language Factors in NM Emulation

The three issues mentioned above as being part of Emulated Native Mode migration (alignment/packing, floating-point format and string move algorithms), all must be handled either by the compiler or the programmer to ensure a successful migration. Alignment refers to the rules used by the compiler to assign variables to memory, relative to the surrounding word boundaries. The fact that two different floating-point formats are available on MPE XL implies that floating-point data must be processed only with the correct operators for its format. String moves that overlap from source to target also can change results if not considered in the migration process.

Alignment Emulation

Loading one word (32 bits) into a register on HP-PA takes one machine cycle. Loading half a word (16 bits) also takes one cycle, as does loading one byte (8 bits). Therefore, if you want to load a 32-bit integer to do some arithmetic on it, there are several ways of doing so. They range in cost from the obvious choice, one cycle for a Load Word (LDW) instruction, to a seven- instruction sequence of Load Byte/Deposit instructions that assemble the value one byte at a time. Why would anyone ever use anything other than the LDW? Well, here is the result of executing an LDW on an address that wasn't a multiple of four bytes (i.e., wasn't the beginning address of a full word in memory)!

```
**** Data memory protection trap (TRAPS 68).

ABORT: $OLDPASS.CARL.XL13

DEBUG/XL A.01.16
**** PROCESS ABORT TRACE ****

NM PROG  1cc.0000406c sub+$8

run $oldpass;xl="xl.pub nl.pub.sys"
PROGRAM TERMINATED IN AN ERROR STATE.  (CIERR 976)
```

* All benchmark results in this paper (less the Series 42 times) are from an HP3000 Model 930 running MPE XL version A.01.20. Classic 3000 programs are compiled with the A.02.00 release of HP FORTRAN 77/V; HP-PA programs are compiled with HP FORTRAN 77/XL release A.03.03.

Actually, the LinkEditor protects you against this sort of problem by default - the above results are after turning off parameter checking. With parameter checking on :

```
HP Link Editor/XL (HP30315A.02.15) Copyright Hewlett-Packard Co 1986

LinkEd> link $oldpass;xl=xl.pub nl.pub.sys
INCOMPATIBLE ALIGNMENT: sub ($OLDPASS, $OLDPASS)  (LINKERR 1044)
(PARAMETER #1)
FOUND 1 TYPE CHECKING ERROR(S)  (LINKERR 1010)
LINK FAILED  (LINKERR 1116)

link $oldpass;xl=xl.pub nl.pub.sys
PROGRAM TERMINATED IN AN ERROR STATE.  (CIERR 976)
```

Like most things in life, computer architecture obeys the TANSTAAFL law: There Ain't No Such Thing As A Free Lunch. In this case, you don't get the added speed and simplicity of the LDW instruction unless you'll guarantee that it will never be fed an address that isn't **aligned** on a word boundary.

This restriction existed but wasn't very tough on the Classic 3000- words were only 16 bits long, so only character items ever had any reason to be allocated off a word boundary. Since the 16-bit address space also required the use of byte addresses as distinct from word addresses, we all got used to keeping the two separate and otherwise not worrying about it.

On HP-PA all addresses are byte addresses, but not all addresses are created equal. Three out of four addresses will get you the disappointing results shown above if you try to load or store a word there. Only addresses pointing at the beginning of a machine word can be the subject of an LDW or STW. Similarly, only even byte addresses can be used to store or load a halfword (16 bits). A Load Byte or Store Byte instruction can reference any address with no alignment problems, but who cares? Nobody wants to pay seven instructions to load something they can get in one.

Obviously, the compiler-writers are going to be smart enough to always align things during memory allocation according to their size so they can always be loaded with one instruction. Character items aren't restricted at all; 16-bit items are allocated on half-word boundaries, 32-bit items on word boundaries, and 64-bit items (Fortran DOUBLE PRECISION or Pascal longreal) on double-word boundaries. The latter restriction is for use by the Floating Load/Store Double instructions - again, the idea is to load and store each primitive data item in one instruction. If that implies (and it does) leaving holes in data structures so that things will be aligned correctly, we'll do it. What's a few bytes in a multi-megabyte data space?

Other than proving HP's compiler-writers can manage to grasp the obvious, why do we care? Well, the problem comes in when we deal with structures that require data to be laid out in memory a certain way - most importantly, when calling IMAGE to store and retrieve data. IMAGE (and some other MPE utilities) expects data being passed to it to be laid out the Classic 3000 way - the only alignment holes occur after an odd-length set of character items and before a noncharacter item, and are never more than one byte. The same situation on HP-PA will result in either a one- or three-byte hole if followed by a 32-bit item and a one-, three-, five-, or seven-byte hole if followed by a 64-bit item. If any answer except one comes up, you get out of sync with the data base record and trash your data.

What do your friendly compiler-writers do about this? It depends on the language. FORTRAN programmers have to build up records "by hand", usually using the true data items EQUIVALENCED to an artificial array that holds the data items together in the format the data-base assumes. In this case, FORTRAN 77/XL will give you a fatal 'ILLEGAL OR INCONSISTENT EQUIVALENCE STATEMENT' error - the compiler-writers' idea of a subtle hint. Pascal RECORDs used the same way are even more dangerous - they are "silent but deadly". Since the compiler is laying out the data in a RECORD, it thinks it knows how to do so - unless you tell it otherwise. Alignment of fields within a RECORD is called packing, hence the references to this problem as "alignment/packing".

Whether or not the syntax of the language allows the compiler to flag this problem as an error, the general solution is to allow the programmer to tell the compiler that nondefault alignment is required. Yep, we're referring to the last refuge of the compiler-writer, the compiler directive. Compiler directives are what compiler-writers invent when they can't figure out how to solve a problem solely from the information in the source code, or if the performance hit for doing something is large enough that they want to be able to say "But you told us to!"

In this case, the various directives effecting alignment tell the compiler not only to allow nonaligned EQUIVALENCES and RECORD packing, but also to generate load/store sequences that are safe for the degree of nonalignment requested. If the compiler can see that the data being accessed actually was allocated correctly aligned, the default instructions will be used instead of the nonaligned sequence. This is important whenever a global directive is being used, as in FORTRAN 77 or Pascal's $HP3000_16 directive. Pascal's $ALIGNMENT **n** $ directive directly controls alignment and code generation for a particular data structure - all others are aligned and packed according to the global packing algorithm in effect.

Reference parameters are the major performance problem with $HP3000_16 alignment. Since (in the general case) the compiler cannot judge if all calls to a procedure pass a particular reference parameter only aligned arguments, we must assume that all reference parameters may be nonaligned and generate nonaligned code sequences to load or store them. This policy is safe, if somewhat slower. It explains why programs with no actual requirement for any data to be nonaligned run somewhat slower when compiled with $HP3000_16 alignment on. It also explains why compiling some procedures with HP3000_16 alignment and not others is a good way to abort abnormally - it is **much** safer to use these global directives truly globally.

Performance with Different Alignments (CPU Seconds)				
	16-Bit	32-Bit	HP3000_16	Full NM
D.P. Whetstone	8.414	7.329	5.685	5.031

Figure 2.*

Figure 2 illustrates how bad the performance hit can be if you force **everything** to be allocated nonaligned. The first two columns are for programs that have been edited to ensure that **every** data item is nonaligned as labelled. The "HP3000_16" column is more likely - if you take the same exact stock source that yields the "Full NM" results and compile it with $HP3000_16 ALIGNMENT, you get the 5.685 seconds results. Other than the influence of the longreal operands being hit more than shorter items by being nonaligned, the Whetstone can be considered typical of most programs.

Fortunately, getting everything nonaligned only occurs if it is explicitly required or you are very unlucky. "Very unlucky" because both FORTRAN 77 and Pascal's $HP3000_16 alignment directives work by simply exchanging the Classic 3000's alignment rules for HP-PA's - be that good, bad, or indifferent. This comprehensive approach implies that while all EQUIVALENCES and data base calls will be accepted and work correctly, all variables (not just those that need it) will be aligned according to the Classic 3000 rules - even if that means making something nonaligned that need not be, with whatever performance impact that entails. If you're lucky, the performance hit will be limited to reference parameters; if not, you'd better check out the $TABLES listing and figure out which of your data structures are nonaligned.

Understanding how the compilers allocate memory can help a lot to know what to do about a heavily-used data item (variable, array, record, etc.)that is nonaligned. A few minutes looking at the $TABLES listing reveals that FORTRAN 77 currently allocates non-EQUIVALENCED local variables in alphabetical order. Therefore, your problems may be as simple as having a CHARACTER or Integer *2 variable named "AAA_TEMP", which throws off the alignment (from HP-PA's point of view) of all later variables and arrays. Items in COMMON, of course, are allocated in the order of their appearance in the COMMON statement(s) - $TABLES will show you the items that are nonaligned, but you will have to introduce explicit dummy names into the COMMON list (of the

* Due to the labor involved in rewriting the benchmark and ensuring all data items in it are nonaligned, the Whetstone was the only benchmark rewritten for this test.

appropriate types/sizes) in place of the holes the compiler would leave when observing the HP-PA alignment rules.

Unfortunately, such tricks are rather harder in Pascal - variables are allocated in hash table order, which might as well be random from the user's point of view. A better strategy in Pascal might be to use $ALIGNMENT 2 on those records and fields that need it and otherwise leave $HP3000_16 off. If you need floating point support as well, you're in trouble - Pascal has no suboptions like FORTRAN 77 to allow separate global control of the reals format.

COBOL has two alignment options for SYNCHRONIZED data items, $CONTROL SYNC16 and $CONTROL SYNC32. These are primarily advisory performance directives and do not have the reference parameter implications that FORTRAN and Pascal experience. While programs that match the SYNC directive to the machine will run faster, it would be a rare COBOL program that found the difference to be significant.

Floating-Point Emulation

The Classic 3000 uses an HP proprietary format for floating-point data; HP-PA machines use the IEEE 754 standard format. Compared to the Classic 3000 formats, the IEEE 32-bit format has one less exponent bit and one more mantissa bit. The IEEE 64-bit format has two more exponent bits and two less mantissa bits than the Classic 3000. In decimal terms, the 32-bit IEEE is about a quarter of a decimal digit more accurate, but can only represent numbers in the range of 10 to the powers from -43 to +38. The Classic 3000 formats (both 32- and 64-bit) can represent numbers ranging from 10 to the powers -77 to +77. The 64-bit IEEE format is about half a decimal digit less accurate than the Classic 3000, but can represent four times as wide an exponent range (from 10 to the -324 to +308). IEEE numbers range farther into negative exponents than positive due to the use of a denormalized format that gradually trades off mantissa bits for exponent bits as underflow is approached.

One specific implication of the change in formats is that the trick of extending a 32-bit Classic 3000 real to 64 bits by merely appending a second longword of zero is gone. Since IEEE formats have differing numbers of exponent bits, this trick will go seriously wrong on an IEEE machine.

While there is an MPE XL intrinsic function (HPFPCONVERT) that translates between floating-point formats, requiring all programs that need to access CM databases on MPE V binary files to be rewritten seemed unsatisfactory as a migration solution. The program model we had in mind was an application that had a little (or no) floating-point data and performed only a few operations on it. Floating-point-bound applications are rare on MPE to begin with, and we assumed they would choose to migrate directly to IEEE to enjoy the much faster floating-point hardware on the HP-PA line. Hence, for Emulated Native Mode we chose an emulation approach designed to give bitwise identical results (in most cases) to those on the Classic 3000. A translated approach that promised higher performance but less similar results (including some emulation failures due to the smaller 32-bit IEEE exponent range) was rejected. There were four areas we found it necessary to emulate for the HP3000_16 reals option: Constants in the source code, operators, math library calls, and I/O. Constants have to be put in the object file in their Classic 3000 bit patterns. This should be remembered when using the System Debugger. Arithmetic operators generate code for millicode calls instead of IEEE hardware instructions.

Math library calls are prefixed with 'EM_' before the external name of the library routine to retarget it to an analogous math library that computes the same function in 3000 instead of IEEE format. For instance, a call to the FORTRAN DSQRT() routine is compiled into a call to FTN_DSQRT (to avoid colliding with the Compiler Library routine DSQRT); if $HP3000_16 REALS is on, the call is to EM_FTN_DSQRT. The 'EM_' versions of the math library routines either do a cross-mode call to the CM routines in SL.PUB.SYS, or use emulated arithmetic to mimic the algorithm in the IEEE version. Some, in fact, do both : emulated arithmetic operating on the results of calls to lower-level routines that eventually do cross-mode calls (in a dizzying bootstrap arrangement that **still** confuses even some of those that helped design it).

I/O is a little simpler (for once). Binary I/O still just moves the bits between the file and memory - no problem. Conversions to and from ASCII were a problem, though. Though we could assume math library performance to be secondary, I/O is done a lot and needed high performance. Instead of emulating it by making a slow cross-mode call to SL.PUB.SYS, we took advantage of the inherent

sloppyness of the decimal conversion process, noticing that no conversion routine can be as accurate as the difference in the number of mantissa bits in the two 64-bit formats. In the 32-bit case we could translate to IEEE and do the conversion to ASCII with impunity, as the IEEE format is more precise than the Classic 3000. If the Classic 3000 value is out of range for 32-bit IEEE, we promote it to 64-bits and then convert to IEEE and ASCII. The 64-bit translation to IEEE always works, but you may be somewhat concerned to know that the low order two bits of these conversions can be assumed to be so nearly random that a translation tromping on them can be accepted as doing no more damage than the double-to-ASCII algorithm already does. The reverse case (on input) is analogous.

Floating-Point Performance by Migration Level					
	S/42	CM	OCT	Emulated NM	NM (IEEE)
D.P. Whetstone	100.781	82.832	58.213	81.354	5.031
BnchIO	239.871	90.894	88.904	18.189	17.469
Distance Pts.	809.000	510.000	355.000	741.000	96.000
D.P. Linpack	*	*	*	157.565	45.532

Figure 3.

In Figure 3, the Emulated NM column is for compilation with $HP3000_16 REALS - no nonaligned data influences. The Linpack benchmark is pure floating-point arithmetic, no library calls. The BnchIO benchmark is an HP Fortran I/O benchmark that does many types of I/O, but does include a significant amount of conversions from 3000-format reals to ASCII and vice versa. The Whetstone, of course, is about half math library operations and half arithmetic. So is the Distance Points program, a production program which computes the points among several thousand that are within 500 miles of each other and prints their distances.

So, what does all this mean to the programmer? Mostly, that you can use Classic 3000-format reals with a minimum of fuss and bother - but with a large performance penalty if you do it a lot. While emulated floating-point arithmetic is about three times slower than the IEEE hardware instruction on a 930, the cross-mode calls used to emulate math routines run about thirty times slower than the corresponding IEEE routines.

The style of the emulation support varies considerably between FORTRAN 77 and Pascal. FORTRAN's approach is all or nothing. Either every real variable is assumed in Classic 3000 format or everything is in IEEE - mix the two at your own risk, the practice is unsupported. In Pascal, the floating point format is an attribute of each individual variable - $HP3000_16 only sets the default, which may be overridden by the $HP3000_32 directive. It is debatable how much more control $HP3000_32 gives you, since about all the compiler allows you to do with such items is pass them to HPFPCONVERT. That's what the Pascal manual means by "String, set, and real operations are illegal on HP3000_32 strings, sets, and real numbers." - you can't even add A + B if A and B are HP3000_32 reals.

String Move Emulation

FORTRAN 77/XL has one suboption that Pascal does not have, $HP3000_16 STRING_MOVE. It causes the compiler to generate calls to a millicode routine that does a ripple move like the Classic 3000's microcode, instead of the HP-PA default block move. The block move is much faster, and only overlapping substring moves like the following get different results (from the HP FORTRAN 77/SL Programmer's Guide):

```
character *10 ch

ch(1:1)= '*'
ch(2:10)= ch(1:9)
```

* Too much data space to run on MPE V.

This was a popular trick in SPL that probably got carried over into the rest of the 3000 languages by linguistic osmosis. If you know you have this kind of code, you probably don't have much of it - it might be easier to just replace it with the corresponding ANSI FORTRAN 77 loop:

```
      DO 10, I = 1,10
         ch (I:I)= '*'
10    CONTINUE
```

You might ask - where is the corresponding option for Pascal? There isn't one. Pascal strmove()'s apparently always generate the block move millicode call. There is a new Pascal/XL predefined procedure "move_l_to_r" which seems to do the ripple move corresponding to the Pascal/V semantics.

Language Emulation Features

Fortran has the most extensive collection of directives of the languages. In each case, the option is global in nature and likely to cause problems if not used truly globally. It's control over alignment isn't as flexible as Pascal's $ALIGNMENT directive, but seems sufficient as a migration option for maximum ease of source code migration and maintenance. It is the only language offering control over the string-move semantics to be used.

Pascal provides very good control over alignment, to two levels of control, through the $HP3000_16 and $ALIGNMENT directives. The floating-point support is also invoked by the $HP3000_16 directive, making these functions difficult to separate. Pascal code should be inspected for overlapping strmove() calls that expect a ripple move; they should be converted to use "move_l_to_r". Also, watch those -32768 .. 32767 subranges - they're 32 bits on HP-PA, not 16 as on MPE V! See below for an idea on how to shift all these declarations into one conditionally compiled set of types/constants.

COBOL has very few migration problems from MPE V to MPE XL. Not having floating-point and using eight-bit alignment for almost every data type leaves very little to emulate. The $CONTROL SYNC16/32 directives allow a minor tweak to performance for those programs making heavy use of SYNCHRONIZED items. There have been requests for ripple moves in non-string cases, but these were not accepted as having been documented as both unsupported and in violation of the ANSI standard.

C has very little installed base on MPE V, but is regarded as a good choice for Native Mode migration of SPL code. As such, support for 16-bit alignment would be a reasonable extension for CCXL, though none has been planned at this writing. Support for Classic-3000 reals and ripple moves both seem unlikely.

All these languages offer some form of conditional compilation that can be used to adapt the source code to the architecture. Conditionally compiling declarations and symbolic constants for the architectural differences can largely limit the necessary changes to the declarations, letting the compiler make the semantic changes necessary.

Conclusions

Many programs can migrate to HP-PA by a straight native mode compilation on the first try. For those with Classic 3000 dependencies, however, the compiler options offered by the native mode compilers provide an intermediate migration point on the way to full native mode. If 3000-format reals must be processed or IMAGE calls made to databases assuming 16-bit alignment, the emulation options allow an immediate and easy port of any software with available source code. On the other hand, such quick migrations have the possibility of disappointing performance, depending on the facilities used - the prudent programmer will do well to understand what the compilers are doing and why.

MPE XL Switch Subsystem

by

R. Gregory Stephens
Hewlett-Packard Company

This paper addresses the subject of the MPE XL Switch Subsystem. The Switch Subsystem is that portion of the MPE XL operating system which allows a Native Mode program to call a procedure that resides in Compatibility Mode. It also allows a Compatibility Mode program to call a procedure that resides in Native Mode. At first this may sound like a rather esoteric feature. However, in situations where migration of an application to Native Mode is not a simple recompilation, the Switch Subsystem can provide a means of getting increased performance for incremental migration.

This paper will address several important aspects of using the Switch Subsystem, but it is by no means a substitute for the MPE XL Switch Programming Guide (p/n 32650-60030). In addition, it will not cover the subject in the depth that the accompanying presentation will. To be more specific, the paper will discuss a strategy for using the switch subsystem and writing switch stubs as well as addressing some of the potential pitfalls to watch for in writing switch stub routines. It does not provide a complete tutorial on how to write switch stubs nor how to use each of the switch intrinsics and their parameters (although the presentation will cover these subjects in more detail).

Compatibility Mode and Native Mode Addressing

There is a fundamental technical reason that we need the Switch Subsystem and that is the fact that addressing of data structures is completely different in Native Mode versus Compatibility Mode.

To provide object code compatibility between the Stack Architecture based MPE V HP 3000's and the new HP Precision Architecture MPE XL HP 3000's, the 'stack-3000' addressing used by the instructions in MPE V programs had to be supported on HPPA machines. This means that Compatibility Mode addressing has to be the same as the addressing on MPE V systems. So, in Compatibility Mode, the stack is still a 16 bit data structure that is addressed via registers like DL, DB, Q, S, and Z. Compatibility Mode programs are still segmented into 16k-word segments that are addressed via the P, PB,and PL registers.

In Native Mode the full features of the HP Precision Architecture are exposed. 32 and 64 bit addressing can be performed and the Native Mode stack and heap together, can theoretically be up to 1Gb. Code and data addressing conventions are also completely different from Compatibility Mode.

If the goal is to allow a Native Mode program to call a Compatibility Mode procedure, we have, by definition, a challenging problem of differing code and data addressing conventions to overcome. In Compatibility Mode, instructions still reference a 16 bit stack which is an entirely different data structure from the 32 bit Native Mode stack that is accessed by Native Mode code.

Phased Migration

Migration from MPE V to MPE XL offers a number of choices to the application developer. The first step in migrating to the 900 Series HP 3000 systems in most every situation is to restore an

application and run it in Compatibility Mode with little or no changes to non-privileged mode programs. Even in situations where long-term execution of an application in Compatibility Mode may not be desired, simply running an application in Compatibility Mode requires no recompilation and makes this a natural first step to establish that the software and hardware configuration required to execute the application is in place before proceeding.

Having tested an application in Compatibility Mode, the next step is to determine which portions of the application should be recompiled into Native Mode. This may include all pieces of an application, none of the application, or any combination in between. (For those portions remaining in Compatibility Mode (CM), the Object Code Translator (OCT) provides a simple means of improving CPU performance of programs and SL's that will remain in CM.)

For programs that we want to be able to take advantage of NM performance or features (such as mapped files) we have to look closely at the language that they are written in and any dependencies that these programs may have on other code. The first question is whether or not the program itself was written in a language that is supported in Native Mode (as of this writing, Feb 1989, COBOL II, Pascal, Fortran 77, RPG, and HP C Native Mode compilers are all available from Hewlett-Packard, with HP Business Basic planned for the future release. A Native Mode SPL compiler is available from Software Research Northwest).

If the program is written in a language that is supported in Native Mode, the next question is concerning dependencies that the program may have on other code. This may be procedure libraries that reside in RL's or SL's which the program is dependent upon. If so, we then have to look at the language this code is written in and the availability of source code for the procedures. We must also determine which other programs are dependent upon this procedure library or SL.

In many situations, the library routines have been written in SPL or they were supplied by another party and source code is not available. In these situations, the Switch Subsystem can act as the mediator between the code that you migrate to Native Mode and the code that remains in Compatibility Mode. Switch can also provide flexibility when migrating code from CM to NM by allowing the code to be migrated to NM in stages. When a number of programs are dependent upon the same library routines, a set of switch stubs allows these programs to continue to call the library routines from either mode as programs are migrated over time to NM.

Switch Intrinsics

The Switch Subsystem is called via 3 intrinsics. The 'HPSWITCHTOCM' intrinsic is used for Native Mode to Compatibility Mode switch calls. The 'HPSWTONMNAME' and 'HPSWTONMPLABEL' intrinsics make the Compatibility Mode to Native Mode procedure calls.

At this point it is worth emphasizing that switch calls are made on a procedure basis. If you are using the process handling intrinsics (CREATEPROCESS, CREATE, ACTIVATE, etc.) to spawn a son process and the process to be spawned is a program in a mode other than the currently executing mode, the switch subsystem is not needed since the creation of a process is essentially the same whether you type the :RUN command or use the process handling intrinsics. In both cases the MPE XL loader is invoked and it will determine which environment (CM or NM) the program needs to be loaded.

Call by Name or PLABEL

There are two ways to invoke a procedure on MPE V systems. The first and most obvious way is to call the procedure by its name. For instance, if we have a procedure called "JULIAN'TO'GREGORIAN'DATE" and we want to call it via our COBOL program we would write

a statement such as:

```
CALL "JULIAN'TO'GREGORIAN'DATE" USING JULIAN-DATE-FIELD,
                                      GIVING GREGORIAN-DATE-FIELD,
                                      STATUS.
```

There is another, less known way to invoke a procedure and that is to dynamically load and invoke the procedure by its PLABEL (or Procedure Address). To dynamically load and invoke this procedure on MPE V in SPL first requires a call to the LOADPROC intrinsic. This intrinsic loads the procedure into MPE's system tables and returns the PLABEL for that procedure.

The next step is to push all of the parameters onto the stack, push the PLABEL onto the stack and perform a PCAL 0 instruction which will take the PLABEL off the top of the stack and call that procedure.

This dynamic procedure calling is fine, but who cares? On MPE V the answer to that question was 'Not many people'. Its primary use was to allow you to determine, at run-time instead of compile-time, which procedure you want to call.

The reason it is relevant to a discussion of Switch is that we can now use this same capability to increase the performance of Switch calls in both directions. When a call to Switch is made using the procedure name, Switch, by default, dynamically loads the procedure, hashes the name to create an entry in a hash table for that procedure, and then invokes it. This means that the initial call to the procedure requires additional overhead, to search one or more SL's for the procedure to be called, while repeated calls only require the name to be hashed to arrive at the callee's PLABEL.

If we are going to invoke a procedure hundreds or thousands of times within a program, we can get even better performance by loading the procedure ourselves and saving the PLABEL. On subsequent calls to switch we can use the PLABEL and no loading or hashing takes place.

Switching to Compatibility Mode

There is one intrinsic provided that makes the Native Mode to Compatibility Mode switch whether we want to make the switch using the procedure name or PLABEL. This intrinsic is HPSWITCHTOCM. Because it requires us to pass explicit pointers (as opposed to parameters that are passed by reference, in which case the compiler implicitly generates and uses pointers on your behalf) it can only be called from HP Pascal/XL or HP C/XL. These are the only HP supported Native Mode languages that allow the programmer to generate and manipulate pointers.

To successfully use HPSWITCHTOCM requires a complete understanding of the procedure to be called. Some of the more obvious information that we have to supply to the switch intrinsic is the name of the procedure, the SL search path (equivalent to the ;LIB=G/P/S parameter on the :RUN command), the number of parameters to be passed, and whether we are calling a procedure or function.

For each of the parameters, we must provide the length of the parameter, whether the parameter is passed by value or reference. For parameters passed by reference we must also specify whether a byte or word address is expected by the callee. For performance reasons, which will be discussed in more detail during the presentation, we can specify that a reference parameter is used as input and or output to the callee.

Switch Stubs

If we look at the scenario mentioned earlier with a Native Mode COBOL program calling a Compatibility Mode SPL procedure, a good approach to structuring our calls to switch is to create a set of routines, called switch stubs. By creating our own library of Native Mode switch stub routines that mimic the actual Compatibility Mode library routines we want to call, we maintain a familiar structure and we isolate the switch stubs from the caller and callee code.

By taking this structured approach, we maintain flexibility in migrating the CM library in phases. This allows us to port the CM library to NM one routine at a time. As each routine is recompiled in Native Mode, its corresponding switch stub is deleted from the switch stub library. The new NM version of the routine is put in its place, while doing this we never had to modify or even recompile the calling COBOL program.

Switching to Native Mode

The concepts involved in switching from Compatibility Mode to Native Mode are similar to its NM to CM counterpart when it comes to the parameters that switch requires. The primary differences are a result of the fact that the Native Mode callee will have no problems addressing reference parameters that are on the Compatibility Mode stack or heap since NM code has the ability to use the full 32 bit addressing of the HPPA. As a result, CM to NM switching does not care about byte or word addresses or whether a reference parameter is input and/or output to the callee.

The other obvious difference is that there are two intrinsics supplied for CM to NM switches. Switches by name are made via 'HPSWTONMNAME' and switches by PLABEL are made via the intrinsic 'HPSWTONMPLABEL'. To initially load a Native Mode procedure and retrieve its PLABEL we use 'HPLOADNMPROC' before we call 'HPSWTONMPLABEL'.

Advanced Switch To NM Features

One of the first performance features that may be worth taking advantage of is the switch by PLABEL capability. To take advantage of this capability requires a call to 'HPLOADCMPROCEDURE' when making a NM-to-CM switch call. This intrinsic loads the procedure, like the original 'LOADPROC' intrinsic does. It also establishes an entry in switch's hash table as was discussed earlier. From that point on the 'HPSWITCHTOCM' intrinsic can be called using the PLABEL option to get better CPU performance. Because we called the 'HPLOADCMPROCEDURE', instead of 'LOADPROC', it also means that switch's hash table is loaded with an entry for this procedures PLABEL.

Because of the fundamental addressing restrictions that Compatibility Mode programs have, reference parameters that are passed from a Native Mode program to a Compatibility Mode procedure must be copied onto the CM Stack before the CM procedure can be invoked. On return from the CM procedure switch will copy the reference parameter back to the Native Mode data structure (normally the NM stack or heap) from which it came. Obviously, this could amount to a significant amount of overhead, imaging copying at 32k byte buffer 2 times for each call to a procedure that will be called thousands of times in a program.

To provide better performance in these situations, the HPSWITCHTOCM intrinsic has a couple of options. The first is the input/output copying option that was discussed earlier. For example, if a routine you are calling with a reference parameter does not require that reference parameter as input, you can specify it is 'output only' from the callee procedure. This tells switch not to copy the actual reference parameter in Native Mode over to the Compatibility Mode stack, saving us CPU time.

Another parameter which can provide increased performance for NM to CM switch calls is the 'method' parameter of HPSWITCHTOCM. This parameter allows you to specify that the callee routine is 'split stack callable'. The split stack feature on MPE V based HP 3000's allows a privileged program to point the DB register away from the stack to an extra data segment as a means of addressing a structure without having to copy it to the stack. When the split-stack option is used in the HPSWITCHTOCM call, it tells the Switch Subsystem that it doesn't have to copy a reference parameters from the Native Mode data structure to the CM stack. Instead switch creates a CM extra data segment that is 'wrapped-around' any reference parameters, switches DB to the newly created extra data segment and then invokes the callee.

It is extremely important to note that this capability REQUIRES PRIVILEGED MODE AND A THOROUGH UNDERSTANDING OF ADDRESSING AND THE 'STACK-3000' INSTRUCTION SET AND ADDRESSING. It also requires that the callee routine must have been written with the specific ability of being able to be called in split-stack mode. If you call a non-split-stack callable routine with DB split away from the stack, the results will at the least be unpredictable and cause data loss, program abort, or even system abort.

Switch Stub Pitfalls

One of the easiest ones to make and most difficult to duplicate is related to NM to CM switches with the input/output option specified incorrectly. On the surface, we might be writing a switch stub to call our FREAD-type procedure that resides in a Compatibility Mode SL. Recognizing that we could be reading data structures of up to 32 or 64k bytes and that our FREAD intrinsic does not need this potentially large buffer copied as input, we might specify that the 'buffer' reference parameter is 'output-only'. By doing this when we call our FREAD procedure, we have told switch that it doesn't have to copy the contents of our 32k byte buffer that is sitting in NM over to the CM stack. It only has to make room for a 32k byte buffer on the CM stack and copy that buffer from the CM stack to the NM data structure when our CM FREAD procedure has completed.

But what happens if we call our FREAD procedure and the procedure encounters some kind of normal (or abnormal) failure like EOF? In this case switch would have allocated space for this 32k buffer on the CM stack, called the FREAD procedure, and after completing (with errors), copied the CM buffer back to NM. Assuming that no data was successfully read by the procedure, switch will be copying whatever garbage was on the CM stack at the location of the 32k buffer it had allocated to the NM stack. This is different than how our FREAD worked on MPE V since our FREAD would probably have left the buffer untouched on a failure.

This may or may not be a problem. It is a problem if the calling program assumes that the buffer is unchanged on a failure (say EOF) condition and it accesses the buffer, which now contains garbage, based upon an assumption which is no longer valid!

Summary

The MPE XL Switch Subsystem can be an extremely valuable piece of a long term migration plan. It also requires a thorough understanding of the application program and libraries.

(apr '89)

DEBUGGING APPLICATIONS ON MPE XL

The Best Tool for the Job

Presented by:

TOM JACK

Hewlett-Packard Company
Commercial Systems Division
10670 N. Tantau Avenue - 46S
Cupertino, CA 95014-4143

INTRODUCTION

SECTION 1

1.1 Statement of Purpose

Debugging Applications on MPE XL - The Best Tool For The Job

MPE XL provides a new application development environment. This new environment is abundantly rich with tools that can be used for debugging applications.

Among these tools are a new system debugger (**System Debug/XL**), a new source level symbolic debugger (**Symbolic Debugger/XL**), as well as a MPE XL version of the MPE V source level debugger (**Toolset/XL**).

The variety of new tools makes a difficult choice, of which tool to use, for even experienced MPE programmers.

This paper shows for each of the debugging tools available - supported languages, main features, limitations, commands for debugging, and product documentation.

My paper and presentation will compare and contrast the various tools that are available for debugging applications on MPE XL.

TOOLSET XL

SECTION 2

Toolset/XL is the one debugging tool on MPE XL that MPE V programmers may already be familiar with. Toolset/XL is largely identical to Toolset/V.

2.1 Toolset XL - Languages Supported

o Languages Supported:
COBOL
PASCAL
FORTRAN (for Symbolic Debugging only)

o To prepare program for Symbolic Debugging
COBOL - compile program with $CONTROL SYMDEBUG
PASCAL - compile program with $SYMDEBUG 'TOOLSET'$
FORTRAN - compile program with $SYMDEBUG TOOLSET
or $SYMDEBUG ON

2.2 Toolset XL - Features

o Designed to facilitate three program development activities:
1) Coding - contains editor
2) Compiling - contains program translator (to handle compile and link)
3) Debugging - contains symbolic debugger

o User Interface.
Access and communicate with TOOLSET XL through the use of commands and function keys. Commands may be strung together. Separate commands with a semicolon. Continue lines with an ampersand.

o Background compiling.
Other TOOLSET XL operations can be performed while your source file is being compiled. These operations must not access the object or list file.

o Versioning facility.
Allows you to maintain multiple versions of source files.

o Editing facility.
Toolset XL includes a basic file editor.

o Can log terminal output to a file.

o On-line Help facility
Three basic levels of explanation within the Help Facility: Overview, Features, and Command Descriptions.

o Workspace
A working environment in which you develop programs. Contains a file directory which controls all of the files that are used in constructing a program (source, listing, object, and program files).

o Symbolic Debugger
Allows you to interactively debug your program without knowing memory locations or code addresses.

2.3 Toolset XL - Limitations

o Toolset XL is unable to symbolically debug Compatibility Mode code.
A program migrated from MPE V to MPE XL must first be recompiled using a MPE XL Native Mode compiler before TOOLSET XL may be used to symbolically debug it.

2.4 Toolset XL - Command Set

o Symbolic Debug Commands

AT	- Set a breakpoint in a user program.
BREAK	- Return to Symbolic Debug from a command list or XEQ file.
CALLS	- Display the names of the currently executing main and subprograms.
CLEAR	- Remove a breakpoint in the user program.
DATATRACE	- Monitor the value of a data item.
DISPLAY	- Display the current contents of data-items.
END RUN	- Terminate the running of the current program.
MOVE	- Transfer the value of a literal, figurative constant, o data item to another data item.
RESUME	- Start or restart program execution.
RETRACE	- List the most currently executed paragraphs, procedures, subroutines, or functions.
SHOW DEBUG	- Display the names of data items being traced.
SYSDEBUG	- Access the MPE XL System Debugger.
TRACE	- Identify each subprogram, section, paragraph, or procedure before it executes.

2.5 Toolset XL - Product Documentation

HP Toolset/XL

HP Toolset/XL Reference Manual (PN 36044-90001)

The *Toolset/XL Reference Manual* contains chapters on:

- Introduction
- Function Key Sets
- Summary of Commands and Function Key Sets
- Toolset/XL Commands and Function Keys
- Workspace and File Management
- Editing
- Program Translation
- Symbolic Debug

The *Toolset/XL Reference Manual* is largely identical in format to the *Toolset/V Reference Manual*. The XL manual has been updated in areas where the nature of the product has changed - most changes reflect changes in the operating environment itself (i.e. substitute "RL" for "USL", substitute "LINK" for "PREP", etc.).

HP Toolset/V

HP Toolset/V Reference Manual (PN 32350-90001)
HP Toolset Customer Training Kit (PN 32350TA)

SYMBOLIC DEBUGGER XL

SECTION 3

Symbolic Debugger XL is a new source level symbolic debugger. Symbolic Debugger was developed to run on both HPPA platforms; as "Symbolic Debugger XL" on the MPE XL HP 3000 System 900 Series, and as "Symbolic Debugger" on the HP-UX HP 9000 800 Series.

On both platforms, the Symbolic Debugger is commonly referred to as "xdb".

3.1 Symbolic Debugger XL - Languages Supported

o Languages supported:
  HP C
  HP Pascal
  HP FORTRAN 77

o To prepare program for Symbolic Debugging:
  HP C – compile with -g
  HP Pascal – compile with $SYMDEBUG 'XDB'$
  HP FORTRAN 77 – compile with $SYMDEBUG XDB

3.2 Symbolic Debugger XL - Features

o Interactive debugger - interact with your program as it runs.

o Symbolic debugger - execute and detect errors in compiled programs using the same high level constructs that are used to write the program.

o Display and modify variables.

o Trace program flow control.

o Single step through a program.

o Set, Delete, Suspend, and Activate breakpoints.

o User defined macros.

o Windows.
  View the internals of your program (in source, disassembly, or split screen mode).

o Assertions
  Execute a list of commands before each source level statement is executed.

o Record and playback features.
Reproduce the state of an HP Symbolic Debugger session by saving debug commands in a file, and then play back the commands.

o Help facility
Print a command summary.

3.3 Symbolic Debugger XL - Limitations

Symbolic Debugger XL is unable to symbolically debug Compatibility Mode code.
A program migrated from MPE V to MPE XL must first be recompiled using a MPE XL Native Mode compiler before Symbolic Debugger XL may be used to symbolically debug it.

3.4 Symbolic Debugger XL - Command Set

Symbolic Debugger XL has over 50 commands. Following are the Symbolic Debug XL commands which are most useful for debugging application code.

o Display and Modify Variables.
p - Display and Modify variables.

o Trace program flow control.
t - stack trace
T - stack trace with variables

o Step through a program.
s - single step
S - single step over procedure calls

o Breakpoints.
b, ba, bb, bc, bp, bt, bu, bx - Set Breakpoints
db, dp - Delete breakpoints.
sb - Suspend breakpoints.
ab - Activate breakpoints.

o User defined macros.
def - Define a macro.
lm - List macros.
undef - Remove macro definitions.

o Windows.
w - Change the size of source window.
td - Toggle the source window between disassembly mode and source m
ts - Toggle split screen mode.

o Assertions
  a - Create assertion.
  aa - Activate assertion.
  la - List assertions.
  da - Delete assertion.
  sa - Suspend assertion.
  ta - Toggle assertions.
  x - Exit assertion.

o Record and playback features.
  >file - Turn recording on.
  > - Display the recording state.

o Help facility.
  h - Access help facility.

3.5 Symbolic Debugger XL - Product Documentation

Symbolic Debugger XL

HP Symbolic Debugger/XL User's Guide Kit (PN 31508-60001)
HP Symbolic Debugger/XL Quick Reference Guide Kit (PN 31508-60002)

The *HP Symbolic Debugger/XL User's Guide* contains chapters on:

Program Overview
Using the HP Symbolic Debugger
Language Conventions
Commands
Sample Sessions
Error Messages
Debug Differences
Language Operators
HP-UX Dependencies
MPE XL Dependencies

Symbolic Debugger HP-UX (S/800)

HP Symbolic Debugger User's Guide (PN 92435-90001)
HP Symbolic Debugger Quick Ref Guide (PN 92435-90002)

The User's Guide and the Quick Reference for HP-UX are largely identical to the User's Guide and Quick Reference for MPE XL.

SYSTEM DEBUG XL

SECTION 4

System Debug XL is a new system level debugger that was designed for the MPE XL operating system. The new system level debugger is a vastly improved superset of the system level debugger "MPE Debug" that exists on MPE V. A number of features have been added to make the new system debugger much more complete and easy to use.

4.1 System Debug XL - Languages Supported

o System Debug XL can be used to debug applications at the assembly level. The System Debugger can be used to debug applications written in any of the Native Mode or Compatibility Mode languages.

o To prepare to debug Native Mode programs:

Compile programs to get listings of code and symbol offsets -

PASCAL	: $TABLES ON$	to get identifier map
	$CODE_OFFSETS ON$	to get statement number/offset for executable statement
COBOL	: $CONTROL MAP, VERBS	to get symbol table map and listing of statements used in Procedure Division with code offsets
FORTRAN	: $CODE_OFFSETS ON	to generate code offsets for FORTRAN statements
	$TABLES ON	to generate symbol table information
C	: -Wc,-m,-o	to generate identifier map (+m), and code offsets (+o)

o To prepare to debug Compatibility Mode programs:

1) Recompile programs and libraries to get listings of code and symbol offsets (if possible).

2) Ensure that the program and libraries contain FPMAP symbolic records.

For program files, use the FPMAP option when you prep your program:

PREP uslfile,progfile:FPMAP

For libraries, use the FPMAP option each time you add a segment to the library:

ADDSL seg;FPMAP

4.2 System Debug XL - Features

o Set, delete, and list breakpoints in a program.

o Set, delete, and list data breakpoints.
Data Breakpoints cause a break when a specified address is written to.

o Arm trace traps within application code.
Trace traps can be armed to cause the debugger to stop at all procedures, labels, or statements.

o Single step (multi-step) through a program.

o Display and/or Modify the contents of memory locations.

o Display a symbolic procedure stack trace.

o Calculate the value of expressions.

o Use full screen displays (windows).

o On-line help for all commands.

o Create and reference user-defined variables.

o Define macros.

o Define aliases for commands and macro names.

o Execute commands from a file, record all user input to a logfile and record all debug output to a listfile.

o Useful for debugging unoptimized code at the assembly level.

4.3 System Debug XL - Limitations

System Debug XL is a assembly level debugger. Because the System Debugger works at the assembly level, it requires that its users have a lower level knowledge of the MPE XL operating system environment. Familiarity with assembly code, procedure calling conventions, parameter passing conventions, and HP Precision Architecture is helpful for using the System Debugger.

With some practice, most application problems can be found with the System Debugger. The system debugger may require somewhat more practice to be proficient than the symbolic debug tools.

4.4 System Debug XL - Command Set

System Debug XL has over 300 commands. Following are the System Debug XL commands which are most useful for debugging application code.

o Breakpoints.
- **B** – set breakpoints.
- **DB** – delete breakpoints.
- **BL** – list breakpoints.

o Data Breakpoints.
- **DATAB** – set data breakpoint.
- **DATABD** – delete data breakpoint.
- **DATABL** – list data breakpoints.

o Trace traps.
- **TRAP** – arm/disarm/list traps.

o Step through a program.
- **S** – step.

o Display/modify memory.
- **D** – display contents of memory locations.
- **M** – modify contents of memory locations.

o Display stack trace.
- **TR** – display a symbolic procedure stack trace.

o Calculate value of expressions.
- **=** – calculate the value of expressions.

o Use full screen displays (windows).
- **WON** – turn windows on.
- **WOFF** – turn windows off.

o On line Help.
- **HELP** – on line Help for all commands.
- **WHELP** – overview of window commands.

o User-defined variables.
- **VAR** – create user-defined variables.

o User macros.
- **MACRO** – define user macro.
- **MACDEL** – delete macro.
- **MACECHO** – echo macro.
- **MACLIST** – list macro.
- **MACREF** – reset reference count.
- **MACTRACE** – trace macros.

o Alias commands.
  ALIAS – define alias for command.
  ALIASDEL – delete alias.
  ALIASINIT – restore alias.
  ALIASLIST – list aliases.

o Execute commands from a file.
  LOG – execute debug commands from a file.

4.5 System Debug XL - Product Documentation

System Debug XL

System Debug Reference Manual (with Binder) (PN 32650-60017)

The *System Debug Reference Manual* contains chapters on:

DEBUG Introduction
User Interface
MPE XL Debug Interface - Commands & Intrinsics
DAT/DEBUG Command Specifications
Symbolic Formatting Symbolic Access
DAT/DEBUG Windows
DAT/DEBUG Window Commands
Standard Functions
Dump Analysis Tool (DAT)
Standalone Analysis Tool (SAT)
Patterns and Regular Expressions
Expression Diagrams
Emulated/Translated CM Code
Reserved Variables/Functions
DAT/DEBUG Command Summary

SUMMARY

5.1 Conclusions

Each debugging tool has strengths and weaknesses. My paper has outlined the features and limitations of each tool.

When you select from among these tools, you should balance each tool's features and limitations against what you need the tool to do.

Some of your selection criteria might be:

Languages you will be programming in.

If you program in COBOL than you should choose a tool that supports that language - Toolset XL or System Debug XL.

Systems you will be using.

If you work on both the MPE XL and HP-UX platforms, then you might want to select a tool that you can use on both platforms - Symbolic Debugger.

If you work on both MPE/V and MPE XL, then you might want to consider that Toolset XL is the only tool that has a largely identical product on MPE/V.

Whether you prefer a symbolic debug tool or a assembly level debug tool.

If you prefer a Symbolic Debugger then your choice is between Symbolic Debugger XL and Toolset XL. Symbolic Debugger has more complete functionality as a debugger. Toolset offers a built in editor and versioning facility.

If you prefer a Assembly Level debug tool then System Debug XL is the tool for you.

In closing "The Best Tool for the Job" is based on many criteria that will vary from shop to shop.
MPE XL has a wide range of debugging tools available that can help us all to become better software developers.

Table of Contents

TITLE: The Growing Importance of a 4GL's Data Dictionary

AUTHOR: Roy Sylvain

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5630

5660 Getting HP3000 Data to Users in Real Time

Author: Robert Ross
Hewlett-Packard
Office Systems Division
Roseville, California
National INTEREX Conference; September, 1989

The Problem: Who Makes Decisions?

At every level in any organization, people make decisions. An inspector decides whether a part is defective, an assistant may have to decide which airline to book a flight on for the boss, an engineer makes design decisions that may affect a product's eventual success in the marketplace, and so on. But the people who usually make the company's most critical strategic and tactical decisions are the managers, who end up prospering from the success or languishing in the failure of the results of those decisions.

How do They Make Them?

Many factors are critical for the managers making those decisions - past experience, projections for the future, and maybe most important of all, *CURRENT* data. I emphasize the word "current", because more and more opportunities are won or lost today because someone else knows more than you do, or knows it before you do. Who has not experienced the agonizing disappointment of investing many months and millions of dollars in a new product, only to see a competitor release the same thing before you even get a chance to announce your product? During the drought of last summer, investors made or lost hundreds of thousands of dollars based solely on who was first to get their hands on an accurate (?) weather forecast. Timely data in an easily understood format is imperative for the decision making capability of the managers running today's businesses.

Where do They Get Their Data?

Where does this data come from and who controls it? The raw data may be generated from any number local or remote sources, but generally the people responsible for entering, collecting, manipulating, and reporting on the data are NOT the same people who need that data in an easily understood format for making crucial business decisions. The data processing personnel may do an expert job of storing the data, generating the typical reports, archiving "old" data, and accessing "current" data, but ordinarily their "lives do not depend" on getting quick and easy access to the same data that they are trained to take care of. And the manager whose "life may depend" on it does not know how to extract what he or she needs.

So What's the Problem?

We do have a problem!

- Managers make decisions
- Managers need current, easily understood data to make those decisions
- Non-managers generally manipulate and control access to the data
- Non-managers do not make the decisions

How do we get the right data, in the right format, to the right people, in a timely manner?

The Environment

Much of the decision making done in the past was based on experience with the product, discussions with current customers, surveys, study of competitors products, etc. However, none of these methods will tell you why a customer cancelled your largest order for the month. In fact, you may not even be aware of the cancellation until the monthend reports are run at the beginning of the next month.

What would happen if you were notified of the reversal the day after it occurred, giving you an opportunity to call the customer and ask what happened while all of the circumstances are still fresh in everyone's mind? You might still be able to work out any problems and recoup an order that would have been permanently lost if you had been forced to wait until the next month when the daily orders report arrived. To accomplish this, you need fast access to data, and the only reasonable method for handling the glut of data produced by our accounting systems is some sort of computer based system.

Let's Try Large Computers

We store all sorts of data on computers, and in the case of accounting data, it is usually stored in some sort of database system. There are many advantages to using these database systems on large computers:

- They have massive storage capacities
- They can support hundreds, and possibly thousands of users
- They can handle a large number of simultaneous transactions

There are also many disadvantages to using these large mainframe or mini computer systems:

- They are frequently "opaque" to the very people who need the decision-making information that they can provide - these people may know what they want, but they do not know how to get it

- This information, when made available, is invariably too late to help in making critical daily decisions

Let's Try Small Computers

For more than a decade now we have had small "personal" computers that are well-suited for the quick, "ad-hoc" inquiries that daily, or even hourly, provide details that help us make useful choices in our business planning. Their advantages almost parallel the disadvantages of the larger systems:

- Data is more accessible, and usually retrieved more rapidly
- Personal computer software continually becomes more powerful and less intimidating to the infrequent user

But personal computers have their own set of disadvantages, some of which make it virtually impossible to use them effectively in a large, varied environment:

- Storage capacity is severely limited when compared to large systems, and PC's simply cannot store and control enough data to give the overall picture for any large company
- Multi-user PC systems generally mean 3 or 4 users, not hundreds
- PC database software becomes intolerably slow when more than a few users are actively pounding away at it

The Solution: Let's Try Both Together

Connecting PCs to large computers is nothing new - we have been doing it for years. Using PCs to download small subsets of data from large databases is also nothing new - we have been using various data communications packages and some sort of spreadsheet to do this, also for years. However, the process frequently goes something like this:

Step 1:	The large system collects and stores data input from various sources throughout the day
Step 2:	A nightly batch job strips off the required data, saving it in a file on the large computer
Step 3:	The file is then converted and downloaded to the PC in the proper format
Step 4:	Finally, the next morning, the data is available for the manager to analyze and use

So what happens if the manager decides at 11:45am that he or she wants the data after lunch? Most likely, the data programming personnel will be sending out for pizza so they can work on producing the required report!

We Need to Use New Products

What can we do today that might help alleviate this problem? What is going to get data from a large computer database into the hands (or computer) of the decision-maker in real time, rather than the next morning or next month?

Fortunately, more and more products are coming to market that fulfill these demands for getting easily understood information from the central repository on the main computer into the proper format for end users in "almost" real time. Instead of running a nightly batch job to strip off information and make it available the next morning, the program can be coupled to requests from the user's PC so that the details can be had at any time. The two programs, one on the PC and one on the main computer, can even be more tightly coordinated so that the large system can "notify" the PC user of an unexpected or outlandish transaction or event that just happened, allowing the manager to respond as quickly as possible. In other words, the next time your payroll manager decides to write him or herself a $10,000 bonus check, you can find out as soon as the data is entered into the computer, before the check even has a chance to be printed. Assuming, of course, that this is really what you want your large computer to spend it's time and resources looking for.

How are New Products Going to Help?

Why is this solution any different from simply asking the programmers or system administrators to run the nightly batch job during the day? What has been added to this new software that make things simpler, more convenient, and more useful? Several things:

- Even though all data on the large computer is available, only the data necessary for a particular request is extracted
- This small subset of data usually allows the process to be done swiftly, before opportunities may be lost
- Instead of being tied to standard reports, which may produce much more data than is needed, or may not even contain the item you were looking for, the end user can modify his or her requests to allow for a changing environment
- **Mostly, managers can get the data they want, looking the way they want it, whenever they want it, and often without further help or intervention!**

Do They Really Work?

Now, all is not heaven in this world of software solutions that tightly couple personal computers to large computers. It may take quite a bit of work to get the raw data to look and feel like what the manager wants. But once the initial time investment has been made, the results are extremely useful and can often be easily leveraged to work on other needs in the company.

The rest of this paper is geared towards a specific solution using the same method described above as requested by a particular company, using a given computer system and software.

The Solution Process

Rarely does a "spark of insight" lead to an immediate result - the spark usually acts as a catalyst that gets ideas flowing and eventually produces a solution that satisfies everyone. In this case, if the manager who needs the information were able to generate the files and code necessary to yield the proper answers, the rest of us might be twiddling our thumbs instead of working overtime. In the real world however, getting a good result requires a good deal of give and take between the two main groups involved - the decision experts (manager) and the computer experts (programmers). The programmers need to fully understand what it is the manager wants - what data is needed, what form it should take when the manager sees it, and when the data is absolutely required to be available.

A Description of the Problem

Our case study deals with a favorite human pastime that should be familiar to many of you - gambling. A large casino with the usual assortment of gaming tables and "electronic gambling machines" wanted to keep better track of their unattended gambling devices, specifically their slot, poker, and blackjack machines.

They had the following requirements:

- At any time, day or night, they wanted to be able to find out the dollar balance of any machine on the floor of any of their gaming rooms without stopping play on the machine
- They also wanted to know the total net balance of any group or all of the active slot, poker, and blackjack machines
- When a customer hit a "jackpot", as may be defined internally, they wanted to know about it immediately
- On a less frequent basis, they wanted a statistics report generated that would pinpoint high and low traffic machines, and also show the times during the day when the highest density traffic and largest dollar volume traffic would frequent the machines

What Kind of Machines Do We Have?

There are three types of machines to keep track of - slot machines, poker machines, and blackjack machines. All types of machines have some minimum betting amount ranging from $0.05 to $20.00, and they also allow betting in multiples of this amount, with a maximum of 5 times the minimum bet. The following table describes the different possibilities:

Machine Betting Structure

Machine Type	Minimum Bet	Maximum Bet
Slot	$0.05, $0.25, $1.00, $5.00, $20.00	$0.25, $1.25, $5.00, $25.00, $100.00
Poker	$0.05, $0.25, $1.00, $5.00, $20.00	$0.25, $1.25, $5.00, $25.00, $100.00
Blackjack	$0.05, $0.25, $1.00, $5.00, $20.00	$0.25, $1.25, $5.00, $25.00, $100.00

How Do We Collect The Data?

The electronic game machines produce the original data to be stored on the minicomputer. Unfortunately, the sheer number of machines make it impossible for the system to service all of the interrupts that would be generated during peak business periods. Because of this we decided to make it a 2-step process with personal computers handling the initial data from the game machines and passing it to the minicomputer at specified intervals.

The Gaming Machines

Two types of transactions were defined for each machine. A "normal" transaction was specified as the complete act of a customer inserting 1 to 5 coins or tokens of the required value and playing the "game" to completion, whereby the machine would either keep all of the bet or return a certain amount to the better based on the outcome of the game. The computerized record-keeping for each "normal" transaction was simplified to 2 items - a time stamp and an amount. Thus each time a better plays a single game, either 1 or 2 records would be created and stored - an input time and amount, and possibly an output time and amount. This is the only information actually generated by each gambling machine to be stored by the computer.

The PCs

PC's are used in the data collection process to log all transactions for each machine, along with the machine's unique ID code. Part of the PC's job is to use the machine's ID code, which at this point is simply the serial number, and append a machine "type" prefix and manufacturer code to yield the actual unique ID as stored in the HP3000 TurboIMAGE database. At the end of each completed transaction, the game machine sends an interrupt to the PC requesting a read on its serial line, 4 of which are connected to 4 gambling machines. The PC will then store 3 data items - the machine ID (prefix, manufacturer, serial number), the transaction time stamp, and the amount.

The PC software will then interrupt the minicomputer on either of 2 conditions - the transaction file now has a specified number of transactions waiting in it (programmable, with a default of 16), or a transaction file has records that have been waiting long enough for a timer to expire (again programmable, but the default is 1 minute).

The Minicomputer

An application running on the HP3000 services these interrupts by uploading the transactions to a TurboIMAGE database, then deleting the PC file. As soon as the HP3000 application indicates via a flag to the PC program that a transaction file read is about to take place, the PC will open a new file and write all subsequent transactions to it. This prevents the PC from being

unable to accept an interrupt from a machine because the transaction file is busy being read by the HP3000. Also, if no one is using any of the machines hooked up to a PC, the minicomputer will not receive any interrupts from that PC, improving response time for other interrupts.

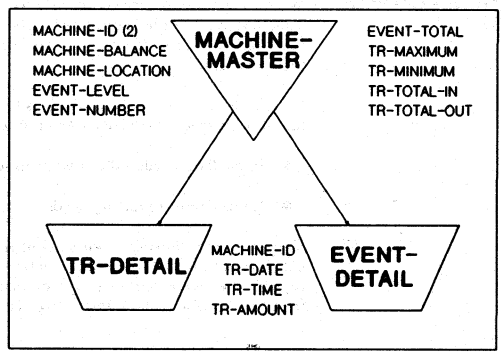
What are 'Events'?

The second type of transaction is called an "event". You might know it better by another term - "JACKPOT!" It is also an exception to the 2 conditions listed above for the PC interrupting the HP3000. Each game machine has an "output level" programmed into it, and any output transactions at or above that level will trigger an "event", which signals a jackpot. Although the records generated by this "event" are identical to normal transactions, an extra step is introduced to allow quick recognition in the control room of what has happened and give management a chance to respond, if necessary. (You know - get your social security number for tax reporting purposes, take your picture, give you lots of free drinks so you will spend the rest of the night losing the jackpot back to the casino, etc.)

The extra step, added to allow recognition of "events", is an immediate interrupt of the HP3000 by the PC requesting a read of the existing transaction file. Part of the process of loading these records into the TurboIMAGE database is to check the master dataset record for that machine, which contains the "event level" amount as programmed into the machine itself. If the incoming transaction amount is at or above this level, the data record is stored in two different places - the normal transaction detail dataset, and a special "Event- Detail" dataset so as to be easily distinguished from "regular" transactions.

How Do We Store the Stuff?

The overriding concern in developing this application was access speed, first for knowing about "events" or jackpots happening as soon as possible, but also for being able to know the precise status of a machine or group of machines without opening the back or shutting them down. These needs led to a simple TurboIMAGE database design, with only 3 datasets, as the following picture illustrates:



All pertinent information about each gambling machine is stored in the MACHINE-MASTER manual master dataset, and every transaction is stored in the detail dataset called TR-DETAIL. In addition, a copy of the transaction is stored in the EVENT-DETAIL if it falls in the "jackpot" category as defined for each machine by the item "EVENT-LEVEL" in the master record.

As each transaction is written to the TR-DETAIL dataset, a "summary" of it is added to the master record. Depending on how much money is involved, the master record items MACHINE-BALANCE, EVENT-NUMBER, EVENT-TOTAL, TR-TOTAL-IN, and TR-TOTAL-OUT may be updated with the transaction amount, or in the case of EVENT-NUMBER, merely incremented. This design is critical for generating an almost instant response when looking up the machine's current balance.

How Does Information Access Get Involved?

What we have so far is a method for taking transactions generated by customers using electronic gambling machines and getting them summarized and stored in a TurboIMAGE database on an HP3000 minicomputer. This is information that floor managers can use effectively, but it is most certainly NOT in a format they can use easily, nor is it available in anything approaching "real time". To get this information properly presented to the right people without delays that would make it obsolete, we used a product offered by Hewlett-Packard called Information Access.

What is Information Access?

As mentioned before, there are many products on the market today that, in one way or another, extract data from large computer systems and present it to end users, either via terminals or PCs, in a more readily understood and usable format. This solution uses Information Access, which is a product that combines HP3000 and PC code to extract data from a large number of formats on both the HP3000 and PC, and either translates and/or transfers that data into a large number of other formats, also on the HP3000 or PC.

How are We Going to Use It?

We have already described how the raw data gets placed into the TurboIMAGE database on the HP3000. Now we need to know how Information Access can help us retrieve, summarize, and display the data for the managers.

First, we should remember what it is that the managers want to see. Their initial requests can be broken down into 2 types:

- Immediate needs - the "I want to see it RIGHT NOW!" data
- Longer term reporting needs

Since only the statistics reports about traffic density fall into the second category, and even though Information Access was used to develop those reports, they were fairly straightforward and not "real time", so I will not take time to discuss how they were generated.

What Do We Really Want to See?

The "immediate needs" data can also be broken into 2 types:

- Jackpots - we want to know in 15 seconds or less, and the display must be triggered by the computer, not the user
- Machine balances - displayed at any time within 15 seconds of user request for individual machines, in up to 5 minutes for groups of machines

The database design revolved around the second group of data, while some interesting features of Information Access tables and batch files produced the first data output.

Who Hit That Jackpot?

The original idea as proposed by the casino's management was to have a PC dedicated to each gaming room floor, and have the floor displayed on the screen with a map showing each machine as a small icon situated on the map in the same place it was located on the actual game floor. Whenever a player hit a jackpot on that machine, the icon would turn a different color, blink, and have a dollar amount be written on the screen near the icon. This effect would last for an agreed upon amount of time, and then the screen would return to its normal display. Although possible, this approach would have required a very committed programming staff writing some special custom software, an idea which management was somewhat leery of.

When asked what was the basic information that they really needed, management responded that they wanted to know the following whenever an "event" occurred:

- How much the jackpot had paid out
- Which machine had 'done the job'
- Where in the room that machine was located
- They also wanted a permanent record of the "events"

Finally, they needed to know all of this immediately, or at least within a few seconds (no more than 15) after it took place.

The Data is There - How Do We Get It?

Since the data flow for jackpot records is already set up to get the information into the HP3000 database within a few seconds, all we needed to do was continually check the TurboIMAGE database for "new events" in the EVENT-DETAIL dataset. This was done using 2 features of the latest version of Information Access.

Batch Files

Information Access can be set up to work in batch mode, both on the PC and the HP3000. Although most often aimed at unattended operation at night or while the user is busy with other tasks, in our case we used the batch

facility to continuously read through the EVENT-DETAIL dataset, each time looking for any "event" records that were not there during last search.

MPE Binary Files

After cycling through the EVENT-DETAIL dataset, the batch file also stores the date and time stamp of the last record in the dataset into a separate MPE binary file. The next time through, the lookup table will only look for any records with a date/time stamp greater than what was stored in the MPE binary file. This is very helpful in making sure we do not scare the managers into thinking the same machine has been hit with a jackpot twice in the last few seconds.

Continuously Logging All Jackpots

Once the batch file creates a table containing only the new "event records", the information is output to a printer and the batch file starts over again. As you can see, they get a continuous log of all jackpots throughout the day (and night) for all game rooms.

DATE	TIME	MACHINE	LOCATION	AMOUNT
89/12/31	14:27:05	SL1400A06	BL-A6-12	$ 50.00
89/12/31	14:28:22	P02250C15	BL-D1-04	$ 50.00
89/12/31	14:30:49	P02250C67	BL-D1-19	$ 200.00
89/12/31	14:35:02	BJ9000X24	BL-M8-53	$ 100.00
89/12/31	14:41:32	SL1400A12	BL-A6-92	$ 150.00
. .	. .	.	.	.
. .	. .	.	.	.
. .	. .	.	.	.

Thus we created an ongoing written record that will give a manager watching the system enough information in a readable format to be able to know what, where, and when an "event" occurred. They get this within a reasonable enough time (average = 7 to 12 seconds) to take any action deemed appropriate. This also has the added advantage of producing a document (that can be reviewed at any time) that shows all transactions whose magnitude is great enough to fall within a manager's "level of interest".

Guess How Much Money is in that Machine?

The second type of "immediate data" requested by management was the existing balance of any machine or group of machines without having to physically go to the machine, open the back, dump the money out, and take it to the coin counter.

Because of the database design on the HP3000, this data was very simple to generate. To get the balance of an individual machine, the manager selects a menu option, enters a machine ID, and within a couple of seconds the most recent balance is displayed. The balance can be off by, at most, 1 minute's worth of transactions. However, we discovered that aside from a few "high-visibility" (meaning their jackpots could add up to a lot of money) machines, managers rarely wanted to know the balance of a single machine. They were far more interested in specific groups of machines.

Where do Customer's Spend the Most Money?

The groups of machines that were of most interest to the managers were broken down into the 3 major categories (Slot, Poker, Blackjack) and also subdivided by the 6 rooms in which they were situated. The result of this interest was an additional 27 menu choices (= 6 rooms + 3 machine types + 6 rooms * 3 types) which corresponded to 27 preset Information Access tables. Each table, when selected by the manager, would proceed to extract from the MACHINE-DETAIL dataset only the machines that met the criteria for that group, then summarize the balances and display the net balance for the whole group.

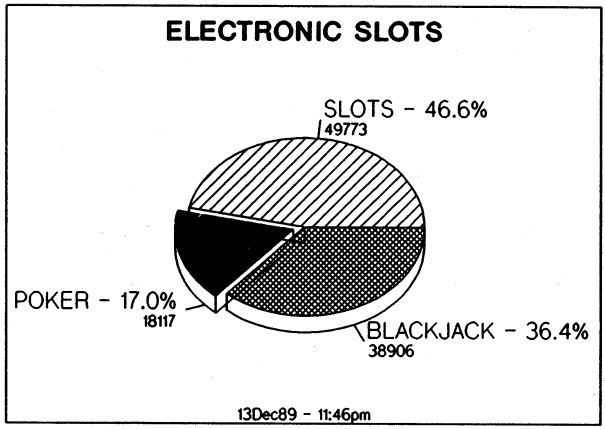
These net balance numbers seemed to be exactly what management was looking for, but once we offered them a further enhancement of the visual display of the information, they liked it so much that they refused to have it any other way.

Getting a Piece of the Pie

It was a fairly simple matter to include as some additional menu choices a graphical display of the business given to each type of machine. Hewlett-Packard's graphical display package for PCs, called Graphics Gallery, can readily use the output generated by Information Access as input for line, pie, and bar charts.

We modified the automated batch files executed by the menu program to use Information Access to extract and summarize the balance data, write it out to a file on the PC, then run Charting Gallery and pass in the file as input. This would generate a sequence of bar and pie graphs displaying (in color) the dollar percentages and comparisons of any combination of machines and rooms. The following pie chart shows the simplest one - a comparison of the total dollar input values for each type of machine.

Sample Pie Chart



Time Sequencing the Data

An additional enhancement was added later to the bar and pie charts. This change was implemented using a graph that displays what we call "time sequenced data". Every night Information Access is used to create a file that summarizes the transactions during the last 24-hour period. Each of these files contains not only a summation of the day's activity, but also a date stamp. These files are then appended into a large file that can be used to generate a line chart that shows revenue growth day by day, and could even be modified to display the growth hourly if a time stamp were added to the summarization record.

Did We Solve the Problem?

As the charts graphically demonstrate, we were able to provide the right information, in a readable format, to the right people, in the right timeframe. Although we did not give the managers what they were dreaming of - a nice map of each floor with game machines represented by icons that would change and draw attention to themselves to alert someone watching of unusual activity - we did provide them with the data they wanted within the amount of time specified. Aside from the subjective comfort of being able to know what is happening monetarily to all machines in all of their rooms from a single control center, the system did provide the real benefit of allowing the casino to reduce their floor management staffing levels by almost 30%. They were also able to respond to situations with the proper level of management much more quickly than before.

Tightly Coupled PC Connections Really Work!

The solution also demonstrates that close software connections between PCs and larger computers can be very effective in solving problems that do not seem to have any obvious answer. However, as networked systems continue to grow in power and ease of use, the differences between data on the PC, minicomputer, and mainframe should almost disappear from the user's perspective, giving him or her quick access to any information needed in today's world of rapid decision-making.

Table of Contents

Chapter 1: Getting HP3000 Data to Users in Real Time

TITLE: Preparation For The Migration From Turbo Image to HP SQL

AUTHOR: Richard Irwin

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5705

Trends in Data Management

Sean Bandarkar
Oracle Corporation
1 Davis Drive
Belmont, CA 94002
(415) 598-4234

Introduction

Businesses and government agencies around the world recognize that their data is a key strategic resource. In order to collect and use data appropriately, computer professionals must deal with a brave new world of database and information management technology. Powerful low-cost personal computers and super-minis, coupled with high-speed networks, have revolutionized database computing. In addition, massive industry-wide investments in software research and development have produced a commercial data management landscape that was considered impossible just a few years ago. Assumptions that have gone unchallenged for many years no longer apply.

This paper summarizes the most significant 'points of interest' in modern information management. It focusses primarily on major developments in 'back-end' or database management system (DBMS) technology. This paper will not emphasize 'front-end' disciplines, like Computer-Aided Systems Engineering (CASE) or application-development tools.

To help you understand some of the underlying influences on the technology, I will include a brief historical overview. I will also attempt to predict broad data management trends in the 1990s.

Yesterday: Alternate Database Models

The first DBMSs received widespread commercial acceptance in the late 1960s. These Database Management Systems provided an order of magnitude improvement in productivity over the File Management Systems that they replaced. File systems, of course, have a sequential data structure based completely on the way data is laid out on disks.

DBMSs provided a way to deal with multiple related files, control redundancy, and improve data security and consistency. DBMSs consist of data structures and associated operators (namely a Data Definition Language and a Data Manipulation Language).

The 1960s, from today's perspective, can be characterized as the era of 'hardware crises'. Computers were big, expensive, and unreliable. Storage, both primary and secondary, was very limited. Only the most basic applications in the biggest organizations could be economically computerized. It's no surprise, then, that the earliest DBMSs were oriented towards optimizing operations in this environment. Early DBMSs were essentially file systems with pointers (links).

Hierarchical Model

Hierarchical systems like IMS and System 2000 were the first to appear. In these models, data organization was restricted to 'one-parent-to-many-child' relations, in which each parent contained a pointer with the address of its children. This design had the singular advantage of mapping data almost identically to the layout of data on a physical storage device.

For example, in a two-level hierarchy—parts and suppliers—all suppliers that have supplied a particular part could be a single file. The parts could be included as the first record in these files or as a separate file with pointers to the suppliers.

After several years, the limitations inherent in this method of data organization became apparent. Many real-life situations cannot be translated into a one-to-many relationship without introducing further complexity and the need for time-consuming workarounds. For example, problems with redundancy and integrity can arise when suppliers provide multiple parts.

Network Model

A variation on this approach, network systems—like IDMS, ADABAS, and IMAGE—also permitted 'one-child-to-many-parents' relations. This provided additional flexibility and allowed more real-world situations to be mapped efficiently into a database. Network models were also close to physical storage layout, as the conceptual model consisted of files with embedded pointers. In addition, the multi-directional pointers often took on a "rat's nest" appearance that produced a high degree of complexity. Complex operators were required to manipulate network data structures. This overall complexity made design, modification, and learning difficult.

Relational Model

In the theoretical world of the laboratory, researchers focussed on developing the ultimate data model and set of operations. From extensive studies of set theory and predicate logic they designed a simple data model—a relation or table—that could express all entities and their relationships. The concept of embedded pointers was removed. Any common field could be used as a connector between tables.

The table is independent of the underlying implementation. For the first time, a clear separation was made between logical and physical models. The first paper to discuss this so-called "relational" approach was published in 1969 by E. F. Codd, then at the IBM San Jose Research Laboratory. In Codd's words,

> "[traditional DBMSs]...burdened application programmers with numerous concepts that were irrelevant to their data retrieval and manipulation tasks, forcing them to think and code at a needlessly low level of structural detail...."

The simplicity of the data structure meant that only a simple set of operators was required. A set of English-like operands was developed that allowed for virtually complete data manipulation.

While the relational model represented improvements in ease-of-use, power, and flexibility, it ignored the hardware constraints of the early 1970s. To run efficiently, databases using the relational model required computers with multi-MIPS CPUs, large address spaces, and multiple megabytes of main memory. Since even the largest computers of the time could barely provide this capability, relational DBMSs (RDBMS) were not commercially viable in the 1970s and remained in the research labs.

Hardware Advances

The hardware explosion in the early 1980s ended the crisis of the underpowered CPU forever. Within a few years, advances in semiconductor technology made possible an entirely new generation of machines. Soon even the smallest machine, the desktop computer, had hardware resources (processing power and main memory) comparable to those of high-end minicomputers a few years previously.

A new era had dawned, this time characterized by 'software and people crises'. The expertise and software required to harness this new hardware power became the critical resources.

In this environment, Relational Database Management Systems (RDBMS) became commercially viable and a new generation of products like ORACLE and DB2 became popular. While their productivity benefits were unquestioned, early RDBMSs did not make efficient use of the hardware resources available to them. As a result, RDBMSs performed poorly, especially in applications originally designed for hierarchical models. RDBMSs thus gained a reputation for unsatisfactory performance in runtime and Online Transaction Processing (OLTP) environments, and were relegated to less transaction-intensive applications.

Today: The Rise of Relational

"It is safe to say that poor performance is, in fact, inherent to the relational environment...."

William H. Inmon, *Computerworld*
November 28, 1983

"In the past few years, relational database management systems have delivered on their promise of productivity and flexibility, putting to rest forever the idea that relational databases would never be fast enough."

Richard Finkelstein, Database Consultant, 1988

SQL-based relational databases are the model of choice for the vast majority of new applications. IBM has declared that their relational product, DB2, and not the hierarchical IMS, will be their strategic database of the future. In the world of the early 1980s, relational OLTP was an oxymoron. However, several RDBMSs currently deliver 'industrial-strength' OLTP capability. On VAX minicomputers, UNIX multiprocessors, and IBM mainframes, for example, relational databases have executed over 50, 120, and 250 Transactions Per Second (TPS) respectively. In many environments, well-designed RDBMSs are outperforming their hierarchical and network counterparts. These results are the product of massive software R&D investments finally yielding a technology that uses the potential of the new generation of hardware.

Hardware Imbalance

One of the less well-known side-effects of the hardware revolution is that CPU speeds are increasing at a higher rate than I/O speeds. The result is that CPU-intensive software environments tend to benefit disproportionately.

The HP experience is an excellent example of this phenomenon. When customers upgrade from the Series 70 to a Series 950, I/O and CPU-intensive applications which use a 4GL or RDBMS see a higher performance boost than applications that are I/O-intensive and CPU-light (such as COBOL and IMAGE). Similarly, the huge increase in addressing capacity on the HP 3000 tends primarily to benefit address-intensive software, like relational DBMSs.

Just as the small storage, weak CPU machines of the 1970s required hierarchical/network systems, today's large storage, powerful CPU machines favor relational systems.

Software Advances

Research and development investments in RDBMSs have far exceeded the level of investment in pre-relational models. As much as 95% of all DBMS R&D is in relational systems. For RDBMSs, therefore, the state of the art is increasing at a more rapid rate.

Locking serves as an example. In relational systems, there is no need to explicitly define the locking strategy. The system handles locking for you. Until recently, RDBMSs locked entire tables for reads and writes. Thus any access to a table was serialized. In contrast, many of today's RDBMSs don't lock at all for reads. Also, writing requires locks only at the row level.

Another example is 'array interfaces' between applications and tables, ensuring that multiple rows of data are transferred at a time. 'Clustering' provides the capability of physically locating data from multiple tables close together to speed access.

Other advances include better optimization techniques, index creation, and multi-server architectures. In the true relational spirit, these features continue to be transparent to end-users and applications.

Standards

The proliferation of computers with proprietary environments created a veritable Tower of Babel. This in turn led to an extreme fragmentation of expertise, as computer professionals' skills became less transferable across environments. Also, computer users began to realize that enterprise networking is more than a mere communications problem: the issue of dissimilar data managers is more significant.

Partially in response to this situation, the American National Standards Institute (ANSI) and the International Standards Organization (ISO) declared SQL to be the standard language for data management, giving relational systems further impetus.

Tomorrow: Client-Server Computing and Distributed Databases

In the late 1980s a new dilemma has appeared—one that might be termed 'the communications crisis'. The central focus of the 1990s will be integrating personal computers and other heterogeneous computers with high-speed networks. In the 1990s, entire networks of computers need to act like a single computer. Client-server computing (also known as distributed or cooperative computing) and distributed databases will make these goals possible.

Client-Server Computing

Until very recently, all parts of an application program executed in the same minicomputer or mainframe computer. Applications requiring continuous interaction with a person eventually bog down. As more and more users are added to the system, each additional user experiences poorer response time. Soon the computer has to be replaced with a larger, more expensive model to accommodate the increased computing load. Moreover, in a non-client-server architecture, personal computers can participate only as 'dumb' terminals or as limited development workstations.

A client-server architecture makes it possible to split an application into two parts. One part, the server or back-end, resides and executes on a large-capacity, shared computer. The server software controls a shared database on behalf of its clients. The second part, the client or front-end, runs on other computers, frequently a PC or workstation. The client software interfaces with the user's keyboard and screen, requesting data from and sending data to the database server, and performing application-specific data processing.

In a client-server architecture, multiple client computers can concurrently access the same server. As more clients are added to the configuration, the server has to perform some additional work on behalf of each, but the effect on overall system performance is minimal. Thus, the performance of the other client applications, and the response time seen by their users, are not adversely affected.

When server capacity has been exceeded, additional resources may be added in two ways. A server may be scaled vertically, meaning that it is replaced with a computer that runs faster and has larger disk capacity. Or it may be scaled horizontally, which means that it is augmented with an additional computer.

In the latter case, the data is partitioned or distributed between the multiple servers. The database servers must cooperate to ensure that their client applications continue to perceive and access the database as a single logical entity.

Distributed database management systems (DDBMS) allow data to be stored where it is most frequently used. For example, sales data can be stored in multiple field offices and customer data can exist at headquarters. This distribution strategy greatly reduces network traffic.

Application functions such as database queries and relational joins must continue to work identically regardless of whether the database tables being queried reside on separate servers or on the same server. This concept is termed 'location transparency' and is a fundamental requirement of a distributed database management system. With location transparency the user (or program) can reference data on multiple nodes in a single statement. With the above example set-up, a single query can be executed from headquarters to determine all the orders placed by all operations of a multi-national customer in a particular timeframe. The system automatically determines the data locations based on table name and transparently routes join criteria and other SQL statements to the multiple remote nodes for execution.

'Site autonomy' is another key DDBMS architectural element in network-based computing. Site autonomy means that each server in the network—whether mainframe, minicomputer, or PC—is capable of operating independently of the others. Features such as a central catalog (containing network-wide data definitions), central scheduler, or central deadlock detector are precluded. A centralized resource, like a global dictionary, would be a single point of failure for the entire network.

With site autonomy, each server has an owner who can set up their own system administration procedures, within network-wide guidelines. In this way, each server can maintain sovereignty even while it participates in the network, and each node can feel protected from the actions of the others. Site autonomy also allows for easier system growth.

Distributed databases have always been relational. The independence of logical and physical models allows the same logical concept (set of tables, relational operands) to be implemented on a network rather than a stand-alone computer. Managing embedded pointers across a network is generally considered to be too complex a task. Thus, pre-relational systems are not distributable and offer limited networking capabilities.

Networks

Clients and servers will be connected using a variety of networks simultaneously, based on business needs. The most popular connection will probably be the Local Area Network (LAN) using network protocols such as TCP/IP (e.g., HP's LANs) or SPX/IPX (Novell).

Other protocols may be in concurrent use for connection over long distances, for connectivity to IBM systems, connectivity over phone lines, etc.

Preparing for the Future

Looking ahead, it is safe to say that hardware technology will continue to surge ahead. New hardware architectures, featuring multiple processors, will offer ever better price/performance. New communications alternatives, featuring fiber-optics for example, will emerge. Network protocols will change and evolve as standards are agreed upon.

Portability for applications and databases—across hardware, operating systems, and network protocols—is important to allow maximum flexibility. The network will need to adapt to changing business or technological considerations. Thus the choice of clients, servers, and networks will not be cast in stone but can be re-assessed as the business need or technology evolves.

Databases, in turn, will become more powerful and start storing more types of information, like text, and rules associated with the manipulation of this information (object-orientation). Close links with Application Generation and CASE products will permit an increasingly productive analysis, design, and construction environment.

Conclusion

Information management technology has been driven by changes in computer and communications technology. From the 'hardware crisis' to today's 'software, people, and communications crisis', data managers have undergone major changes to respond to customers' most pressing needs.

Standards-based relational databases, client-server computing, and distributed databases permit a powerful, productive foundation that takes full advantage of the key technologies of the 1990s. This will allow businesses to tap the full potential of their data, and thus improve the competitive posture of their enterprises.

INFORMATION MANAGEMENT TECHNOLOGIES INTO THE 1990'S

Orland J. Larson
Hewlett-Packard Company
19091 Pruneridge Avenue
Cupertino, California 95014

ABSTRACT

Enterprises have come to depend more and more on the accessibility, accuracy and timliness of information. During the next few years, new database technologies will provide solutions to many of the problems and difficulties facing today's MIS and user departments.

This paper begins by reviewing the changing world of information management and the challenges facing MIS. This will be followed by the major trends associated with information management over the next several years. These include the continued acceptance and importance of relational database technology, the increased interest in distributed database applications and the emergence of the new object-oriented database management capabilities. Finally, this paper will address the significance of these technologies in the cooperative computing environment.

INTRODUCTION

THE CHANGING ROLE OF DATA PROCESSING

Before looking forward into the 1990's, we should first look back at how data processing has evolved over the past twenty years.

The general trend of the 1970's was the use of centralized computers and resulted in systems that were often difficult to use, inflexible and usually did not meet the end user's needs. Database management systems (mostly hierarchical and network) became widely used and provided the basis for on-line, interactive applications. In addition, the computers and operating systems provided programmers the capability of developing applications on-line, while sitting at a terminal and interactively developing, compiling and testing these applications. The end users were also provided easy-to-use, on-line inquiry facilities to allow them to access and report on data residing in their databases.

During the 1980's, the emphasis has been on the decentralization of data processing. This includes the proliferation of personal computers which has resulted in both the "islands of automation" and the corresponding "islands of information" problems. This in turn resulted in reduced control of corporate data for the MIS department. In addition, relational databases became commercially viable and experienced wide acceptance even though performance was often an issue. Relational database performance has now improved significantly; and they are currently proving effective in on-line transaction processing (OLTP) environments. Software tools such as 4th generation languages (4GL's) continue to be used successfully as an effective way of developing applications through the concept of information systems prototyping. This required that the end user be more involved in the development of systems and has resulted in more effective systems that

that meet the users' needs. This has helped to reduced the backlog of applications but usually also has contributed to the "islands of automation" problem.

As we move into the 1990's, relational database will continue to gain wider acceptance. It is the enabling technology and the basis for distributed database management, which provides transparent access to data which is distributed over several sites.

There are also new technologies called object oriented database (OODB) and object oriented programming systems (OOPS), which will manage more complex data structures and will result in improved programmer productivity and more flexible systems.

An additional technology, cooperative processing, is evolving which will help integrate those "islands of automation" back together and allow for the data and programs to be accessed and shared in a cooperative computing environment.

THE IMPORTANCE OF SQL

According to a recent Gartner Group Report, in 1988 only about 7% of the applications developed used relational database or Structured Query Language (SQL). However by 1992 their prediction is that 65% of applications developed will use SQL.

There is no doubt that SQL will be the basis for applications developed in the 1990's. One of the main advantages of SQL is data independence or the immunity of applications to changes in storage structure and access strategy. Another main advantage of SQL is the simplicity of the underlying relational model which is the easiest to understand - at least at the most basic level. In this model, data are represented as tables, with each horizontal row representing a logical record and each vertical column representing one of the attributes, or fields, of the record.

The following are the key points associated with relational technology:

* Relational concepts are easy to understand and use.
* SQL is a multifunctional language
  - Database definition and creation
  - Data retrieval and manipulation
  - Authorization and security
  - Transaction management and recovery
  - Database environment management and restructuring
  - Interactive and programmatic use
* SQL allows you to specify which information you want - not how to retrieve it.
* SQL increases programmer productivity and raises programming close to the level of problem solving.
* Data independence is ensured and minimizes maintenance of programs
* Data access can be automatically optimized as the DB structure changes.
* The DBA has power and control over the database.
* New systems can be implemented much faster.
* SQL assists in cross-system connectivity.
* Relational databases can provide a cost effective, powerful solution.

There are, however, some areas of SQL that need improvement. The current SQL standard is weak and missing many important features and many of the standard features are implementor defined. In other words, no vendor fully supports the complete "standard" and no two SQL implementations are exactly alike. The SQL language has many inconsistancies and the same SELECT command may yield different results on systems from two different vendors.

As we enter the 1990's, relational database will become a dominant technology in today's information management marketplace. There are several enhancements planned to improve functionality and performance. It eventually will be appropriate for most applications and gain wide acceptance by all users.

Relational databases can improve the quality, control and accessibility to your organization's extremely important and valuable information resources. It can result in an improved competitive position by aiding business analysis that can help to determine ways to improve products and services.

Unlike non-relational database environments, relational databases adapt easily to dynamic business requirements. In addition, unrestricted access to important data means better information for more effective decision making.

Relational database can also have a positive effect on many MIS development environments by reducing the application backlog and reducing the time and cost required to develop applications. The improved database flexibility and ease of change can also result in a significant reduction in the maintenance of applications.

Overall, the use of relational technology can increase the MIS professional's effectiveness and productivity, which results in improved user satisfaction and confidence. Choosing relational now will position your organization to take full advantage of the technological advances of the 1990'S.

DISTRIBUTED DATABASE

One of the hottest topics in the commercial database world is the growing trend towards the use of distributed database management systems. After many years of research, distributed databases are becoming more viable. However, there is still much to be done to provide more than just read access to distributed data. Chris Date, one of the world's leading experts on relational database, recently presented a paper (see reference 3) in which he provided a working definition of distributed database. "A distributed database system is a system involving multiple sites connected together in a communication network, in which each site is a database system in its own right, plus a user at any site can access any data in the network exactly as if the data were all stored at the user's own site. Thus, a DDB is a virtual DB whose components are physically stored in a number of distinct real databases at a number of distinct sites."

Chris Date follows this working definition with an "alternate" or more elaborate definition. "A distributed database system is a system that allows an arbitrary collection of relations, from an arbitrary collection of databases, on a variety of different machines, running a

variety of different operating systems, connected by a variety of different communication networks to function as if they were all stored in a single database on a single machine. The user is completely insulated from all details of distribution."

Distributed databases can allow the structure of the database to mirror the structure of the company, while simultaneously solving the "islands of information" problem. Some additional advantages include local control of local data, accessibility to remote data, increased capacity, incremental growth, data availability, efficiency of storage, flexibility and cost effectiveness.

There are also some potential problems or disadvantages associated with distributed database such as the complexity of implementation - but this is the vendor's problem. Some additional potential problems include the problem of how to design systems for distributed environments, the complexity of administration and control, the impact on local operations, the political problems dealing with the ownership and protection of the data and the possibility of a node or line "crash". In addition, solutions that are appropriate in a centralized environment may frequently not be appropriate with distributed systems.

Chris Date is the author of 12 rules of a distributed database system. He begins with the fundamental principle or "rule zero" that states, "To the user, a distributed database system should look exactly like a nondistributed system". The subsidiary rules follow:

1. Local autonomy
2. No reliance on a central site
3. Continuous operation
4. Location independence
5. Fragmentation independence
6. Replication independence
7. Distributed query processing
8. Distributed transaction management
9. Hardware independence
10. Operating system independence
11. Network independence
12. DBMS independence

These rules are fairly self explanatory and will not be expanded in this paper. No current DBMS vendor adheres to all of these rules. A few vendors claim adherence to rules 1-8 and almost none to rules 9-12.

AJ Brown, a Project Manager at Hewlett-Packard's Database Lab, recently spoke at the SPARC (South Pacific Area Regional Conference) and presented HP's seven phased approach for developing a distributed database management system. The following is a summary of that approach:

Phase 1 - REMOTE DATABASE ACCESS - Program can read/update a remote DB without coding for communication and remote processes.

Phase 2 - MULTI-REMOTE DATABASE ACCESS - Program can read and update more than one remote database at a time.

Phase 3 - FOREIGN DATABASE ACCESS - Program can access multiple vendors databases without coding for DBMS differences.

Phase 4 - DISTRIBUTED READ ACCESS - Retrieval-only transaction can span databases without coding for coordination of results.

Phase 5 - DISTRIBUTED READ/WRITE ACCESS - Distributed DB appears completely local to the program for both retrieval and updates

Phase 6 - REPLICATED DATA - Enhanced availability and performance through multiple copies of data, with automatic synchronization of copies when updates occur.

Phase 7 - PARTITIONING - Enhanced performance and availability through partitioned tables.

Some of the issues regarding distributed database that will have to be addressed in the 1990's include:

* Distributed query optimization and decomposition
* Fragmentation, recombination and optimizability of data
* 2-phase commit and recovery
* Referential integrity across sites
* Management of replicated/partitioned data
* Controlling authorized user access
* Update synchronization
* Degree of Transparency
* Flexibility to move data around the network
* Cost of mainframe vs. mini vs. micro MIPS
* Provision of foreign (non-HP) DBMS gateways (gateways are a way of processing data in a foreign DBMS or file system)

There are some factors that will help propel the distribution of data. These include company mergers, the downsizing of computers, the increased database needs and the general industry push toward distributed and cooperative processing. A company with distributed operations will gain competitive advantage through the support of distributed databases.

OBJECT-ORIENTED SYSTEMS

Each decade, one or two key advances emerge to change the practice of software development. Object-oriented systems and methods are rapidly entering the mainstream of software engineering and systems development. Leading consultants are heralding object-oriented approaches as one of the most important trends to affect businesses in the 1990's. But even among its strongest advocates, disputes abound over key issues, content, and definitions of the object-oriented approach. Object-oriented technologies are moving out of the academic world and into the business world.

With the object-oriented approach, processes revolve around the data, not the other way around. Using the traditional approach, programs are structured around data rather than procedures. For example, when using a traditional programming language, parameters are used to pass data structures and values between routines. The object-oriented approach attaches routines to data structures. In other words, the behavior of the data is kept with the data. This is called encapsulation.

An object-oriented programming language allows the programmer to define and manipulate objects. Some object-oriented programming languages are

extensions of classical languages - C++ and ObjectiveC are in this category. Others are brand new languages, eg. Smalltalk and Eiffel.

An object-oriented DBMS also supports the definition and manipulation of objects, plus providing the classic DBMS functions of persistent storage, transaction management, concurrency control, security, backup and recovery.

A "message" is an important concept used with the object-oriented approach. It is defined as a request sent to an object to change its state, or to return a result to the sender. Objects respond only to well-defined messages. The only information needed to use an object is knowledge of the messages it can receive. An object-oriented program is a flow of messages among cooperating objects.

Messages ensure the modularity of a system. To interact with any object, you only need to know what messages to respond to, not how the object is represented. Knowledge of how an operation is accomplished is of interest only to the programmer responsible for the definition of the object itself. Messages make an object's functionality available to other objects, while hiding the implementation details.

Maintaining and modifying software has been a real drain on programming resources. Maintenance programmers must understand a complex system well enough to fix its problems or enhance it. However, these programmers often did not develop the code, and often operate without adequate documentation or guidance. Changes often introduce new, unanticipated problems to the system. A programmer working with existing code must read and understand it; this may require a mental translation back into the original design specifications, which is extremely difficult for complex or poorly-coded systems. An additional concern is that programming languages allow unchecked access to data structure internals.

Objects can dramatically improve the problem of maintenance. Modularity and encapsulation limit interdependence, allowing changes that do not disrupt the rest of the system. Objects' natural organization make it easier to learn and understand relationships between parts of a system. The original programs are easier to write and debug and fewer errors occur. Programs read like designs, making changes clear and easy to make. Reuseability is an important advantage of using objects and libraries of these software components provide leverage. What has been written once need not be written again. Model features like inheritance allow existing components to be incrementally modified to suit changing needs. Together the representational advantage of encapsulation and the features of inheritance dramatically improve software development and can greatly improve programmer productivity.

Libraries of high-quality, tested software components will radically alter the way software is written. Software will routinely consist of a series of software components glued together. Application programming will no longer mean rushing to a text editor to begin coding. It will require understanding the capabilities and restrictions of available components, plus knowing how to combine them.

Some of the challenges facing object systems include the time it takes to learn about existing libraries of software components. Programmers also may resist accepting this new approach. Objects also consume more resources, however emerging 80386 PC's and RISC workstations will help

to alleviate this problem. Applications with promising potential include: prototyping, user interfaces, graphics, telecommunications, computer aided design (CAD) and computer aided manufacturing (CAM).

Object oriented products are still in their infancy and the commercialization of object-oriented technology has barely begun. Now more suitable for advanced technology projects, object database systems should become viable for commercial projects over the next few years and widespread adoption in the 1990's.

Hewlett-Packard has developed a prototype of an object-oriented database management system (OODBMS) called Iris. Development started in 1985 and Iris has been presented and demonstrated at several major conferences in the past few years including: SIGMOD (Special Interest Group on the Management of Data) in June 1988 in Chicago, OOPSLA (Object-oriented Programming, Systems, Languages and Applications) in September 1988 in San Diego, DB/EXPO in February 1989 in San Francisco and at the Patricia Seybold Forum on Object-oriented Technology in April 1989 in Boston.

Object-oriented environments herald the dawn of new programming paradigms. Business people will be empowered to perform tasks that, in the past, required professional programmers. Programmers will be empowered to design complex applications in smaller, modular, more fool-proof pieces.

Neither end users nor application programmers will need to concern themselves with the mechanics of networking, peripheral support, or file handling. Object based architectures lend themselves to the creation of a much richer information environment. Digitized voice, music, images, video clips, and animation will begin to populate our information systems.

COOPERATIVE COMPUTING ENVIRONMENT

The environment in which today's business must operate is changing quickly and becoming more complex. To meet the challenges produced by this changing environment, organizations need greater amounts of information to make the key decisions required for success. Keeping pace with the rapid change that is occurring means gathering information and making decisions faster than ever before.

As we look at the computer industry, we are about to embark upon the next revolution in computing. This is not a unique case. There have been multiple revolutions in the computing industry and computation in general, going back to the early days of mechanization, tabulation and so on, through the first computer mainframes. The mainframe was a very centralized processor, still oriented toward batch and was really a carry-over from punchcard tabulation systems. The next move was into the distributed processing and personal computer revolution, causing an explosion of workstations and personal computers on people's desks, which fueled distributed processing and distributed computing.

Then came the communications revolution, where the objective was to integrate all this computing power in the corporation in a way that moves information around and enables different types of devices to

participate in the solution of business problems. This has become the basis or launching point for the next revolution of cooperative computing.

Cooperative computing is the notion of tying together all the information computation resources in a corporation into a single entity, and making all those things interact in some efficient manner, transparent to the end-user in such a way that each user has access to all the information computation resources in the network as though they were local to the user's workstation.

The cooperative computing environment is Hewlett-Packard's vision of the future of computing: a network of heterogeneous computers that can work together to solve a single problem and are extremely easy to use.

To support the cooperative computing environment requires a cooperative computing strategy. HP has focused its product development efforts in three key areas - systems, end-user computing and networking. And perhaps most importantly, the integration of the three.

The corresponding three technological pillars of HP's strategy are:

* HP Precision Architecture - HP's powerful, flexible architecture that serves as a common foundation for HP computers. It includes an advanced, optimized and integrated application development environment that allows programmers to be more productive.

* New Wave Environment - Simple, natural consistent user interface, which can provide one "window" into the entire network of computing resources.

* HP AdvanceNet - HP's networking strategy based on standards, both industry and defacto.

This paper has addressed many of the important technologies that are essential to the cooperative computing environment. ALLBASE/SQL is the strategic DBMS of the future for cooperative computing. ALLBASE/SQL runs under both MPE and HP-UX. The application development environment for both of these platforms is an excellent 4th generation language called ALLBASE/4GL and an easy to use end-user oriented inquiry and reporting facility called ALLBASE/QUERY. ALLBASE/SQL will also be the basis for distributed database technology in the future. ALLBASE/NET is the first phase of HP's distributed database technology and will also use HP's networking capabilities.

The concept of "objects" in the New Wave environment is similiar to the object-oriented capabilities described in this paper. The object management facility (OMF) is the main component of the New Wave Environment. The OMF tracks all data in the PC, whether it be text, graphics, spreadsheets, scanned images, even voice. These objects are represented as icons which can be combined into compound documents containing different types of data.

The OMF allows users to create "hot-links". Hot-links allow users to share data between different reports, memos, even file folders. When data is changed in one place it is automatically changed every other place it has been shared.

The OMF knows what application to use to work with a particular object. When an object is selected, the New Wave environment will automatically load the proper application.

SUMMARY

The important technologies briefly addressed in this paper: SQL, distributed database management systems and object-oriented systems are extremely important to the future of the cooperative computing environment. HP's Cooperative Computing Strategy is focused upon helping our customers meet the challenges of today's changing business environment. We believe that these technologies and strategy, coupled with HP's commitment to industry standards and reputation for high quality, reliable systems, can provide our customers with the solutions they will need into the next generations.

REFERENCES

Brown,A.J.,"Distributed Data Management From HP",Proceedings SPARC'89, Sydney, Australia,April 11-13, 1989.

Connors, T. and Lyngbaek, P.,"Providing Uniform Access to Heterogeneous Information Bases",Software Technology Laboratory,Report STL-87-16, Hewlett-Packard Laboratories,December 4, 1987.

Date, C.J.,"Distributed Database Technology", Proceedings DB/EXPO'89, San Francisco,CA,January 31-February 2,1989.

Eunice, J.,"Leaders in Object Databases",D.H. Brown Associates, Inc., December 1988,400 Benedict Avenue,Tarrytown,NY,10591,(914)631-7859.

Fishman,D.H.,Beech,D.,Cate,H.P.,Chow,E.C.,Connors,T.,Davis,J.W., Derrett,N.,Hoch,C.J.,Kent,W.,Lyngbaek,P.,Mahbod,B.,Neimat,M.A., Ryan,T.A.,Shan,M.C.,"Iris: An Object-Oriented Database Management System",ACM Transactions on Office Information Systems,5(1),Jan 87.

Larson, O.J.,"Strategic Importance of Relational Database Technology", Proceedings INTEREX Computing Management Symposium'89,Nashville,TN, March 8-10, 1989.

Tash, J.,"Application Architecture and the Strategic Role of Relational DBMS",Database Decisions, Inc.,41 Marcellus Drive, Newton,MA,02159, (617)332-3101.

Schussel,G,"Database Developments in the 5th Generation",Proceedings 1989 Database and Cooperative Processing Symposium,Washington,DC, March 20-22, 1989.

Stein,J., and Maier,D.,"Concepts in Object-oriented Data Management", Database Programming and Design,1(4), April 1988, pp.58-67.

White, C.,"The Importance of SQL",Proceedings DB/EXPO'89,San Francisco CA,January 31-February 2, 1989.

Design for Performance with HP SQL

Edward C. Cheng
May Kovalick
Distributed SQL Laboratory

Hewlett-Packard
19447 Pruneridge Avenue
Cupertino, CA 95014

ABSTRACT

In reality, multiple users will be retrieving and updating information stored in a database. It is important for a user to accomplish a task with the database management system (DBMS) in the quickest possible time without degrading the performance of other users' jobs running on the same system. It is for this reason that transaction design and query design are important in writing database applications.

HP SQL allows a wide range of consistency level to be declared for a transaction to provide maximum concurrency and still maintain certain level of data consistency. It also has the ability to optimize a query to find the fastest access method. Writing applications with transaction and query design in mind would help users to achieve high performance with HP SQL.

This paper discusses the features provided by HP SQL in improving performance in both single user and multi-user environment and how users can design their applications on the transaction and query levels to take advantage of these features.

Key Words and Phrases: Consistency and isolation levels, concurrency, query optimization, selectivity factor, dynamic query, relational databases, SQL

1. Introduction

Databases are critical resources to today's enterprises. Information is stored to and retrieved from databases. Users require a fast response time from the DBMS while data integrity must also be ensured. Depending on the kind of applications running, sometimes a transaction may be operated under a lower consistency state in order to achieve a higher performance.

Performance of a DBMS can be measured in terms of elapsed time it takes to execute a query. The elapsed time depends on the efficiency of the DBMS in executing the query

as well as the optimizer decision in picking the most optimal access plan [1]. In a multi-user environment, performance can also be measured in terms of number of transactions executed per a unit of time. This is refered to as throughput performance. A system which can execute a single user transaction within a short elapsed time may not necessarily give a high performance throughput when multiple users are running on the system. The latter really has to do with how much concurrency is allowed on the system. In view of this, HP SQL provides various features to allow a transaction to be executed under a desired consistency level dictated by the user. It also provides a number of optimization features so that an issued query can be executed efficiently with the best possible access plan.

In this paper, we will first discuss the concurrency control features that HP SQL provides and how users should go about designing their applications on a transaction level for maximum performance. We then describe the optimizer features of HP SQL and how users can design their queries to take full advantage of them.

2. Concurrency and Transaction Design

Transaction design involves the design of the environment under which a transaction is going to operate. By changing some of the parameters manipulated by the transaction management component of the DBMS, one can control the deadlock handling and the level of concurrency on the system.

HP SQL permits users to set the priority and isolation level for a transaction. The *priority* is an integer ranged from 0 to 255. This number is used when two transactions are entangled in a deadlock situation; the transaction with the larger priority number will be aborted by the system. If the two transactions have the same priority number, then the one which has started later will be aborted. The default priority is 127. Users can decide to use a lower number if the application in question is more important than others, or use a higher number if it is of less significance.

Besides priority, the following isolation levels can also be specified on a transaction basis: Repeatable Read (RR), Cursor Stability (CS), Read Committed (RC), and Read Uncommitted (RU) [2]. The list is given in descending order of isolation severity. A lower degree of isolation implies more concurrency in the database environment, but data touched by the user are also maintained in a lower level of consistency state throughout the transaction. By default, all trasactions are run on a high level of isolation to ensure repeatable read.

The syntax of assigning the priority and declaring an isolation level for a transaction in HP SQL is shown below:

```
BEGIN WORK [Priority] [RR]
                      [CS]
                      [RC]
                      [RU]
```

It must be noted that with any of the consistency levels we provided, *atomicity* of a transaction is always guaranteed to ensure data integrity of the entire database. By atomicity we mean that either all actions in a transaction are done or none of them is done [3].

Besides the various isolation levels, a feature called Keep-Cursor (KC) is also available to be used in certain application environment. A cursor opened with the KEEP CURSOR option will retain its cursor position even after the transaction is committed. With this feature, you can improve concurrency over the entire system without giving up the efficiency of the application coded with KC. All the isolation levels and the use of KC shall be discussed in the following sections.

2.1 Repeatable Read

Repeatable Read or RR fullfilled the most stringent consistency requirement in the database world. It assures the users that all data which has been accessed, either read or write, are protected till the end of the transaction. A transaction running in RR is guaranteed that it will not see any uncommitted data, nor will it lost its update if the transaction is committed, and finally, it can repeatedly read the same block of consistent data from the beginning to the end of the transaction. Table 1 shows the locking strategy with RR:

			Select	Update/Intent Update
Access Methods	Relation Scan	Table	S	SIX
		Page	-	SIX
	Indexed Scan	Table	IS	IX
		Page	S	S: Non-leaf; SIX: Leaf & Data

Table 1 Lock Modes with Repeatable Read

Applications should be written with RR when they required the above restrictions to ensure integrity. RR should also be used when performance of a particular application is more important than others. For instant, a debit/credit application for a bank will be a typical example of this kind. In this type of applications, it is essential to have the highest data consistency as well as a fast response time.

2.2 Cursor Staility

As its name has implied, cursor stability or CS maintains data consistency on the block of data that the user is currently scanning. To allow this level of consistency, appropriate locks are put on table and pages while data are read and shipped to the user. When the scanning goes on and the next block of data is locked, the system will release the locks from the last fetch. This allows other users to access the segment of data that this application has read without being blocked till the end of transaction as RR would have done. It is important to know that even with CS, locks on updated data will not be release until the end of transaction to ensure no uncommitted or dirty data be read by any other people on the system. Table 2 shows the locking strategy with CS:

			Select	Update/Intent Update
Access Methods	Relation Scan	Table	IS	IX
		Page	S*	SIX*
	Indexed Scan	Table	IS	IX
		Page	S*	S: Non-leaf; SIX*: Leaf & Data

*: Release lock on the next fetch

Table 2 Lock Modes with Cursor Stability

Applications written in CS are protected from dirty reads and lost update, but unlike RR, it does not guarantee repeatable read property. In other words, if a user scanned a piece of data twice within the same transaction in CS, he might see different values since the data can be changed by another user on the system. In theory, it is always possible to design applications in such a way that when they run concurrently inconsistent results will be produced. It is therefore very important for the programmer to decide if CS should be used or not.

Since CS requires frequent acquiring and releasing of locks in a small granule, it usually consumes more CPU time than RR. The advantage is that it allows more parellelism or concurrency on the system. The user is advised to code an application with CS when repeatable read property is not required and performance of this application is not on the first of the priority list. Application which scans a large block of data but just updates a few tuples would be an ideal candidate of CS; report writer application is another.

2.3 Read Committed

Read Committed or RC is another isolation level allowed in HP SQL. RC does not guarantee repeatable read nor cursor stability. Nevertheless, only committed data or undirty data will be read. With RC, by the time information is returned to the user, the data might have been deleted or modified by other transactions since locks are released right after the fetch. It is obvious that RC can provide even more concurrency to the system than CS, but it also holds a lower consistency level than CS. Also, as in CS, acquiring and releasing locks in RC adds more overhead to the lock manager of the system and therefore transaction running in RC may not get the peak performance when compared to an RR application. The lock modes used for RC is the same as in CS except that locks are released before the data is shipped back to the user.

The user is recommanded to write read-only application which scans a major portion of a table in RC. Once again, if performance of this application is more important than anything else, then RR should be used instead. For write also application, users should either use CS or use RC along with the REFETCH command. The REFETCH command is provided to allow the user to refetch a particular tuple and verify the data before the update. The lock on that tuple will be held until the end of the transaction. The following example demonstrates the use of RC and REFETCH:

```
DECLARE CSStudent CURSOR FOR
  SELECT NAME, ADDRESS, GRADUATED
  FROM STUDENT
  WHERE MAJOR = 'CS'
  FOR UPDATE OF GRADUATED;

BEGIN WORK RC;

FETCH CSStudent INTO :host;
      .
      .
      .
if (condition) then
  begin
  REFETCH CSStudent INTO :host;

  { verify the data }

  UPDATE STUDENT SET GRADUATED = this_year
    WHERE CURRENT OF CSStudent;
  end;
      .
      .
COMMIT WORK;
```

2.4 Read Uncommitted

Read Uncommitted or RU allows the maximum concurrency. It is sometimes refered to as dirty read capability. RU does not ensure repeatable read, nor does it protect users

from reading uncommitted data. It does, however, guarantee no updates will be lost; updated data will be locked exclusively until the end of transaction. Unlike CS or RC, RU does not have the overhead of the complicated locking scheme. It can scan the data even if they are locked by some other users. Similar to RC, users are advised to do update or delete in RU only along with the REFETCH command. Table 3 shows the locking strategy with RU:

			Select	Update
Access Methods	Relation Scan	Table	-	IX
		Page	-	X
	Indexed Scan	Table	-	IX
		Page	-	S: Non-leaf; X: Leaf & Data

Table 3 Lock Modes with Read Uncommitted

Applications which do not require reading very accurate data, such as from statistical databases, can use RU to improve performance for themselves as well as for the system.

2.5 Keep-Cursor

Up to this point all the tools we have in transaction design are limited to a single transaction only. With Keep-Cursor (KC) feature, cursor positions are retained across transactions. This allows the user to do cursor operations and others within a transaction, commit the work and release all or some of the locks already held. A new transaction can then be started to continue the work with the cursors at the positions where they were left.

The KEEP CURSOR option is associated with each individual cursor and is declared when the cursor is opened at execution time. KEEP CURSOR enhancement is designed to work under all isolation levels as described in the former sections. The locking schemes described in the corresponding lock tables are still preserved with the use of KC.

Along with the CKC declaration, user can also specify if he wanted the locks associated with the cursor to be held across transactions or not. The default is with lock. The advantage of holding locks is that repeatable read property can be guaranteed over multiple transactions. The advantage of releasing all locks is that it allows more concurrency on the system. The syntax for KC is described below:

```
OPEN CursorName [KEEP CURSOR [WITH LOCKS  ]]
                             [WITH NOLOCKS]
```

whenever the KEEP CURSOR option is specified, the default is WITH LOCK.

KEEP CURSOR feature should be used when an application has to do a lot of updates on a single but large table, or when updates on multiple small tables are done based on the scan of a large table. In the case of updating lots of records in a large table, the operation can be cut down into multiple update blocks. After updating each block of data, a commit work should be issued to free all the resources and then move on to the next block. Similarly, cursor which scan a large table can be opened with the KEEP CURSOR option; based on the result of this scan, the application may decide to update some other tables in the database. After a set of updates, it may be desirable to issue a commit work to release the locks on different tables before moving on to the next scan.

3. Optimizer Decision and Query Design

We discuss query design here in the scope of writing queries to achieve best possible performance from a DBMS. Ideally, once a query is issued, the optimizer in the DBMS should be intelligent enough to reconstruct the query internally so that the best access method can be evaluated inspite of how the question is asked. But before the optimizer becomes such a perfect AI engine, it is important for the users to know the capability of the optimizer so that its strengths can be fully utilized.

As we have pointed out in [1], optimization in a relational model can be classified into two categories; namely optimization of single-relation queries and multiple-relation queries. The former involves choosing the cheapest access path for that query on the relation. The later also adds the decisions of picking the best join order over the tables and choosing the best join method for each join in the query. Currently HP SQL provides index and sequential scans as the two basic methods for accessing data. Hashing will be provided as another access method in an upcoming release. For joins, HP SQL employs both nested loop join and sort merge join. The choice between the two depends on the size of the tables and whether indices are built for the tables or not.

Users can help the system to use the best access method by designing their queries in the right way. In the following sections, we will highlight some of the optimizer characteristics that are available in HP SQL and describe how users can use them.

3.1 Propagate Filter in Join

Consider the following query:

```
Q1: SELECT *
    FROM STUDENT, RECORD
```

```
WHERE STUDENT.NAME = RECORD.STUDENT_NAME
AND NAME = 'JOIN TOM';
```

Since there is an equal-join between relations STUDENT and RECORD, the equal predicate on STUDENT NAME = 'JOIN TOM' can be distributed to the RECORD table as well. Thus, logically, Q2 below is equivalent to Q1:

```
Q2: SELECT *
    FROM STUDENT, RECORD
    WHERE STUDENT.NAME = RECORD.STUDENT_NAME
    AND NAME = 'JOIN TOM'
    AND STUDENT_NAME = 'JOIN TOM';
```

Note that Q2 is more efficient than Q1 since the amount of tuples returned to be joined from the RECORD table in Q2 will be less due to the filter on the STUDENT_NAME column. The ability of being able to propagate equal and range expressions from one table to multiple join tables is known as propagate filter capability. HP SQL has this functionality and therefore users do not have to worry about propagating filters themselves in designing their queries.

3.2 OR Predicate Optimization

In HP SQL, OR predicates on the same table are optimizable in that the system will consider using index scan to satisfy the query. Consider the example below, assuming indices exist for both GRADUATED and MAJOR columns:

```
Q3: SELECT *
    FROM STUDENT
    WHERE GRADUATED = :this_year
    AND (MAJOR = 'CS' OR MAJOR = 'EE');
```

The system will consider using table scan, or index scan on GRADUATED, or index scan on MAJOR. Since students satisfying GRADUATED this_year would in general be a much large domain than people in the CS or EE department, the index on MAJOR is most likely to be picked.

Another thing that is noteworthy is that the above query is written in conjunctive form. Conjunctive form queries have their predicates arranged so that the OR factors are lying on the two sides of the AND factor. e.g. (C1=5 OR C2=2) AND C3=1. Contrary to conjunctive form, disjunctive form predicates have the OR factors linking up the AND's: e.g. (C1=3 AND C1=7) OR C2=5. Even though the optimizer has the ability to normalize a disjunctive form query into conjunctive form before the optimization process on OR predicate can take place, it is always advisable to write the queries in conjunctive form if possible.

3.3 LIKE Predicate Optimization

LIKE predicates are being used in pattern matching queries. A system which does not have the ability to optimize on the LIKE predicate would have to force the users to write queries with extra range predicates. Without the addition of range predicates, Q4 will always be using table scan:

```
Q4:  SELECT *
     FROM STUDENT
     WHERE  NAME LIKE 'RO%';
```

In this case, users will usually rewrite their queries as:

```
Q5:  SELECT *
     FROM STUDENT
     WHERE  NAME LIKE 'RO%'
     AND NAME >= 'ROA'
     AND NAME <= 'ROZZZZZZZZZZZZZZZZZZ';
```

With the ability of optimizing on LIKE predicate, users can forget about adding filters to the query to allow the system to consider index sacn on the LIKE column.

HP SQL can optimize on the LIKE predicates including LIKE with host variable. Consider the following query:

```
Q6:  SELECT *
     FROM STUDENT
     WHERE  NAME LIKE :host;
```

The value of host is not given until execution time. If the system blindly picked index scan and the user entered a pattern such as '%CY', the response time of this query could be fairly poor. In HP SQL, a plan is first built at compiled time based on the default selectivity factor of the LIKE predicate. At execution time, the host variable is read and analyzed; if the host variable begins with a wildcard character, HP SQL will be able to switch to use table scan even the default plan is index scan. However, if users do not want to rely on the default selectivity factor, they should use dynamic queries which will be optimized at run time when the host variables are known.

4. Conclusion

This paper has reviewed and updated some of the issues and designs in HP SQL in the aspect of improving performance. We believe that among other components such as availability, usability, standard conformance, connectibility, and a set of complete tools, performance is also a significant parameter in a successful DBMS. Continuous effort will be spent in this area to further improve the response time in the OLTP environment as well as for ad-hoc queries.

Reference

1. E. C. Cheng, HP SQL Performance, Distributed Data Management Laboratory, Cupertino, CA. 1988

2. A. Lutgardo, The HP SQL Advance - Towards the OLTP Market, Distributed Data Management Laboratory, Cupertino, CA. 1988

3. P. Griffiths Selinger, Access Path Selection in a Relational Database Management System, IBM Research Laboratory, San Jose, CA. 1979

Distributed Data Management From HP

AJ Brown, Project Manager
Hewlett-Packard Company
19447 Pruneridge Avenue, MS44UW
Cupertino, CA 95014

WHAT IS A "DISTRIBUTED" DATABASE?

The term "distributed database" has been kicked around for some time now, and everyone seems to have developed their own interpretation. The concept can really be defined in two ways: purely academically, and realistically. This is not to suggest that the academic world breeds theories not reasonable to real-world situations, but there does seem to be a difference between what scientists solve and what customers need.

In the academic sense, data within a distributed database is spread over multiple nodes in a networked environment. The logical collection of this data is treated as a single database by the distributed database management system and typical operations such as select, project, and join may be performed on database objects without regard for the actual location of the data. Even though this definition is quite simple, it's difficult to visualize the benefits of using a distributed database for any given application. In fact, the intentions are not to use a distributed database at all, but to use distributed data.

Realistically, data in any large operation today is already distributed. One does not have the luxury of deciding whether or not to disperse data throughout a network. Instead, as multiple hardware boxes are connected to common local and wide area networks, one is faced with the reality that data is already distributed. Managing and integrating this data is the real goal at hand, and the distributed database management system is a tool to this end.

Consider the following example which will be used throughout this paper. A hypothetical manufacturing organization has the following equipment:

> An HP3000/950 is used by the MIS department to store customer data and sales history data as well as personnel data

An HP9000/855 is used by the manufacturing department to maintain parts and finished goods inventory

Three workstations are used in the marketing department to track sales, orders, and forecast data

The data stored at each location is useful on its own, but there are many applications where data from multiple sites would need to be pulled together. Consider the following requests:

Refine a product forecast based on current sales trends, sales history, and the current product forecast.

Create a new software system which coordinates product shipments. The system will check new orders to see if inventory exists; if it does, inventory will be reduced and a shipment record generated along with a packing list and mailing address. Customer records will be updated to reflect cumulative amounts received.

In the previous examples, data at more than one location is required to complete the request, and the integrity of the data at each site must be retained. For example, a single transaction must reduce inventory, generate a new shipment record, and update customer receipts. In order to preserve data integrity, the transaction cannot be partially completed; all parts or no parts must be performed. For these reasons, the power of a distributed database management system is needed to manipulate this data, and a logical distributed database can be created.

The selection of which data to include in the production of our distributed database is flexible and would include the order history data, the finished goods inventory data, and the data stored on the marketing department's workstations for sales, orders, and forecasts. The distributed database does not need to include the personnel data or parts inventory since this information is unrelated to our requests.

BUILDING A DISTRIBUTED DATABASE MANAGEMENT SYSTEM

At Hewlett-Packard, we believe the best technology on which to build a distributed database management system is the relational model. Relational systems, such as ALLBASE/SQL, offer great amounts of flexibility and power in retrieving data and are excellent candidates for transparent distribution. However, given the size and complexity of the problems to be tackled in creating a distributed database management system, it is obvious that complete functionality can not be introduced overnight. Distributed database management functionality can be broken into seven layers or concentrations. Each of these layers incorporate

more and more distributed features offering greater benefits to the application programmer and system administrator. The layers are discussed in detail below.

LEVEL 1 - REMOTE DATABASE ACCESS

The first level addresses the need to have data and application independence. Remote database access allows an application to execute on one machine and access data which resides on a different machine. The machines may be connected using whatever communication services are provided, for example a local or wide area network. In this first level, the application is only allowed to access a single remote database at any one time. Connections to multiple databases are not permitted.

In this example, the first level would support an application which executes on one of the workstations and accesses data stored in the MIS department's customer database.

LEVEL 2 - MULTI-REMOTE DATABASE ACCESS

The second level enhances the capabilities provided by the first level to include simultaneous connections to multiple databases. A single application will be able to maintain more than one open database and can communicate with any chosen database at any point in time. Transactions are not allowed to span databases.

In the example, an application which executes on the HP3000 and accesses the manufacturing inventory tables and the marketing sales data would need level 2 capabilities to be most efficient.

LEVEL 3 - FOREIGN DATABASE ACCESS

Also referred to as heterogeneous or multi-vendor access, level 3 allows an application to be written which manipulates databases managed by other vendors' relational database management systems. For example, an ALLBASE/SQL application could be written and executed against an IBM DB2 database in place of an ALLBASE/SQL database. Three modes of operation would be permissible:

1) a system-common mode where only SQL statements which are common to all vendor implementations are allowed within the application

2) a system-exclusive mode where vendor-specific SQL may be used and is passed through to the server machine without regard to what is sent

3) a translation mode where one vendor's SQL is translated to another's

Independent of the mode chosen above, the ability to download data from a foreign vendor database to the local database would be provided through a SNAPSHOT capability. A SNAPSHOT can be thought of as a physical view; the definition of which is specified using a SELECT command referencing the foreign data, and whose result table physically exists at the local site and may be updated on a periodic basis.

Again using the previous example, consider a corporate IBM mainframe enters into the picture with vendor information for parts availability. An application written on the HP3000 using ALLBASE/SQL would be able to access the mainframe data.

LEVEL 4 - DISTRIBUTED READ ACCESS

This level introduces the ability to perform standard relational operations such as SELECT, PROJECT, and JOIN over tables which reside at multiple nodes in a network. The tables must all belong to the same logical distributed database and are accessed transparently by the distributed database management system. That is, the application need not specify the physical location of the data.

For our example, a distributed database might be created containing order history, finished goods inventory, sales, orders, and forecast data. An application might be written running on any of the nodes which reads and joins information from any combination of these tables.

LEVEL 5 - DISTRIBUTED READ AND WRITE ACCESS

Level 5 enhances level 4 to provide write and update capabilities within a defined distributed database. Operations such as INSERT, UPDATE, DELETE, and DROP would be allowed on objects located at multiple sites within a single transaction. Data integrity is guaranteed through the use of a two-phase commit protocol.

In this example, when inventory is moved out of the inventory database, a shipment record can be generated and customer receipts increased all within one logical transaction. The transaction is guaranteed to either finish completely or not at all.

LEVEL 6 - REPLICATED DATA

This level begins to offer advanced mechanisms for tuning application performance by allowing specified data to be duplicated throughout the network at sites where it is most

often used. This provides higher availability of the data in the event a source node is down and also can help achieve higher read performance since data is held locally in addition to being held at a remote site. An overhead is incurred when updates are made to the base data.

For our example, the order history data may be duplicated at one of the workstation sites so that marketing has faster access to this information and so that the MIS department's HP3000 is not overloaded with time-consuming queries.

LEVEL 7 - PARTITIONING

Level 7 allows data in a table to be split over multiple nodes. A division made at the row level, typically based on a given key range, is said to be a horizontal partitioning. A division made placing certain columns at one site and remaining columns at another site is called vertical partitioning. In either case, access to any portion of the table is transparent to the application.

For example, the sales table could be split into regional categories onto separate machines. One machine would maintain sales data for the United States, another for Europe. Logically, there is only one table containing data stored at both sites.

Each of the above levels offers specific features and benefits to the application programmer for enhanced data distribution. Not all the levels will be pertinent to all applications, and the systems analyst will have great flexibility in choosing what is right for any given system.

ISSUES IN IMPLEMENTATION

Several issues face designers of distributed database management systems. Some of the more obvious include incorporating two-phase commit protocols to ensure data integrity, and managing replicated data. The research sector has done an excellent job at devising algorithms to handle such obstacles and there are many pre-definedmechanisms at our disposal. More intricate design hurdles exist, however, and these are such topics as how to optimize queries which span multiple nodes for the best performance; how to ensure transparent access to data even when it is moved from one node to another; and how to coordinate transactions and recovery.

Take, for example, what appears to be a simple task of retrieving data from three tables located at three different sites:

```
SELECT C1, C2, C3
  FROM T1, T2, T3
 WHERE T1.J1 = T2.J2
   AND T2.J3 = T3.J4
   AND T1.C4 = 'EASTERN'
   AND T3.C6 = 'SALES';
```

The distributed database management system must go through a process that goes something like this:

1. Break down the query to pull out the names of tables being accessed. In this example we are using T1, T2, and T3.

2. Find out where on the network the data for these tables physically resides. A data dictionary would be an appropriate depository for this kind of information. We can assume for this example that SITE1, SITE2, and SITE3 contain data for T1, T2, and T3 respectively.

3. Formulate sub-queries to be performed at individual sites and send each sub-query to the appropriate site for execution. In this example, we have:

   [SITE1] SELECT C1, J1 FROM T1 WHERE C4 = 'EASTERN'

   [SITE2] SELECT C2, J2, J3 FROM T2

   [SITE3] SELECT C3, J4 FROM T3 WHERE C6 = 'SALES'

4. Decide at which site(s) to perform the JOIN operation(s). Move data from the other sites to the JOIN site. In our example, we will assume data will be joined at SITE1.

5. Join the data at the JOIN site and retrieve the results to the application site.

In the end, the original query turned out to be quite involved, and there are many decision points which could affect the final performance of the data retrieval operation. A distributed database management system must understand the trade-offs involved and must be tuned for optimal performance. Another aspect of this query was that data had to be located on the network based on the table names provided in the select list. Typically a data dictionary will be used for this purpose and must be able to translate logical table names to physical names and locations. The translation must be performed at execution time dynamically in the event tables are moved from time to time. Finally, the entire

query must be performed within a single transaction which the application has started. Since the query was broken into three sub-queries, distributed transaction management becomes the responsibility of the distributed database management system.

SUMMARY

It is safe to say that distributed data is here today and is here to stay. Integrating this data through the use of a distributed database management system that provides complete data integrity and ease of use to the application programmer and end user is the key to success in a distributed environment. The technology will not arrive overnight, however, and the design entails many issues which are of academic study and current practice. Hewlett-Packard is committed to offering distributed capabilities based on the relational model and has identified the steps necessary to reach an exceptional offering.

ALLBASE/TURBO CONNECT - SQL GATEWAY TO TURBOIMAGE/XL

Tu-Ting Cheng, Jim Stratton, Dolly Hu, Manoj Mittal
Hewlett-Packard
Data and Languages Division
Cupertino, California 95014

INTEREX Conference; September 11-15, 1989; San Francisco, CA

ABSTRACT

Allbase/Turbo CONNECT provides transparent read access to TurboIMAGE/XL data in a standardized relational format. Applications may issue queries on TurboIMAGE/XL data via ISQL or preprocessed code using standard HP SQL syntax.

The benefits of this product to the user are numerous: TurboIMAGE/XL data can be accessed using Allbase/SQL's rich and flexible query language; applications that currently access TurboIMAGE/XL directly may continue to do so concurrently with Allbase/Turbo CONNECT without change; access to TurboIMAGE/XL data via Allbase/4GL and Allbase/Query will be possible.

This paper describes how to set up Allbase/Turbo CONNECT, some runtime design considerations and various issues which arose when we tried to bridge the gap between these two dissimilar database management systems.

1. INTRODUCTION

Often, users wonder why they need to know the underlying physical implementation of the database. Wouldn't it be nice if they could address a standard query against any kind of a database? HP's answer is yes, and we have taken a step forward by providing access to a non-relational database via a standardized query language in the form of Allbase/Turbo CONNECT.

Allbase/Turbo CONNECT makes HP's concept of "Allbase" a reality in providing access to HP SQL (relational database) and TurboIMAGE/XL (network database) via a common query tool, SQL. Access to TurboIMAGE/XL data using tools like ISQL, preprocessed 3GL code, Allbase/Query and Allbase/4GL is possible in the same way it is done for a relational access - the user need not be concerned with the underlying organization of the data being accessed.

The second section gives a brief overview of HP SQL and TurboIMAGE/XL, the differences which exist between the two and our approach in overcoming these differences with Allbase/Turbo CONNECT. The third section describes how to set up Allbase/Turbo CONNECT via a utility as well as some aspects of the utility. The fourth section gives an overview of the global and process local data structures used at run time and an example SQL query against a TurboIMAGE/XL database.

2. TURBOIMAGE/XL AND HP SQL - TWO DIFFERENT WORLDS

To start with, there is a difference in the terminology of the two database management systems. Table 1 shows how specific database concepts are described in each system.

HP SQL	TurboIMAGE/XL
System Catalog	Root File
DBEnvironment/DBE	Collection of Databases
DBCON	Root File
Table/Relation	Data set
View	--
Row/Tuple	Entry
Column	Field
DBEFile	--
DBEFileSet	--

Table 1. Conceptual Mapping Between HP SQL and TurboIMAGE/XL

Note that the above pairs of terms represent similar but not necessarily identical concepts and that in some cases a concept exists in HP SQL but not in TurboIMAGE/XL. Figure 1 shows a simple block diagram of the HP SQL Architecture. There are three main components in this structure: the user interface, *SQLCORE* and *DBCORE*. The user interface consists of administrative tools like ISQL, end-user tools (e.g. Allbase/4GL, Allbase/Query), preprocessed applications, etc. by which the user has the ability to query the database. The query is passed down to SQLCORE for processing and optimizing, which in turn, calls DBCORE for the appropriate data retrieval. In other words, SQLCORE in a very generic sense is the query processor and DBCORE is the data manager. SQLCORE parses and linearizes the input query as part of its processing. As we will see later, the tasks SQLCORE performs are fully exploited by Allbase/Turbo CONNECT. DBCORE is responsible for access methods (Input and Output), concurrency control and logging/recovery. HP SQL files consist of two kinds of MPE/XL privileged files - log files and DBEFiles plus another MPE/XL privileged file: the DBCON file. These files collectively make up the DBEnvironment. A DBEFileSet is a logical grouping of DBEFiles. The data is stored in DBEFiles. A logical view of this data is the table, which consists of rows and columns.

Figure 2 shows a simplified block diagram of the TurboIMAGE/XL architecture. There are two components in this architecture: the user interface and the TurboIMAGE/XL database manager (analog of DBCORE). The user interface consists of user programs with embedded "DB" intrinsic calls or interfaces such as BRW and Query/3000. The file architecture consists of two types of MPE/XL privileged files - a root file and one or more data sets. Note that there is no query processor in this architecture and it is the responsibility of the interface tools (BRW, Query/3000) to provide this functionality. This has lead to the development of non-standard query languages.

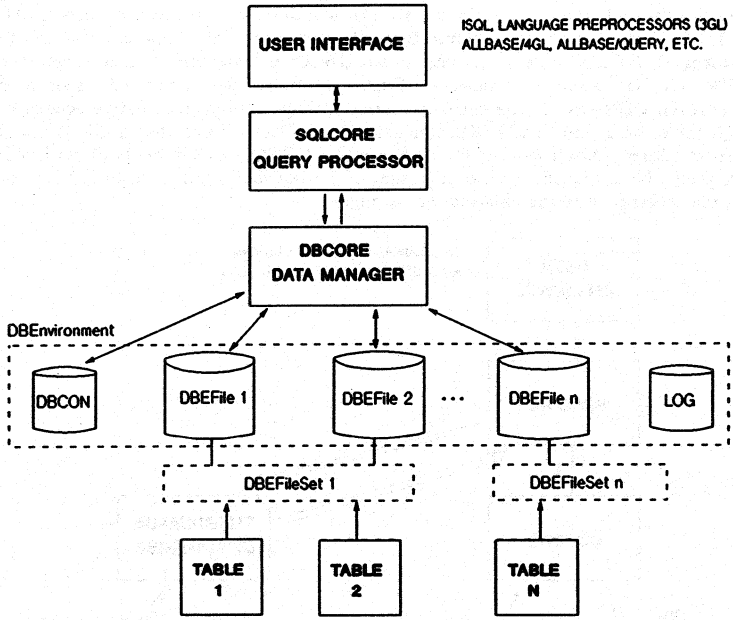


Figure 1. HPSQL Architecture

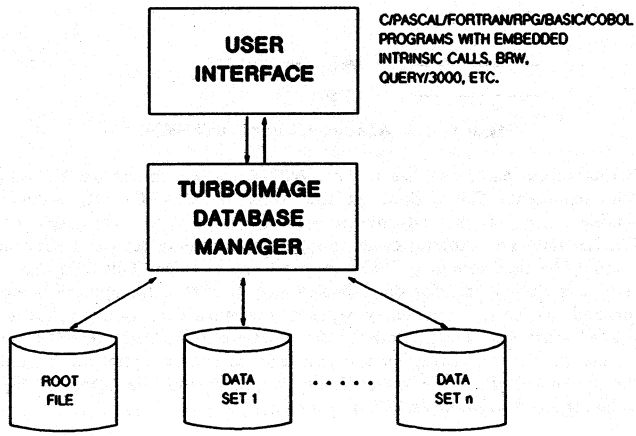


Figure 2. TurboIMAGE/XL Architecture

How does Allbase/Turbo CONNECT use HP SQL's architecture to access a TurboIMAGE/XL database? If we look at the two architectures, the answer becomes apparent: instead of letting the processed HP SQL query be passed to DBCORE, intercept it and pass it to the TurboIMAGE/XL database manager. Figure 3 shows the integrated environment of Allbase/Turbo CONNECT. The benefits of this architecture are that native access to either HP SQL tables or TurboIMAGE/XL data sets is not disturbed and that a query can retrieve data from either TurboIMAGE/XL data sets or HP SQL tables or both with complete transparency! Note that there is no migration of TurboIMAGE/XL data to HP SQL data - it is only the access path to the data that has changed.

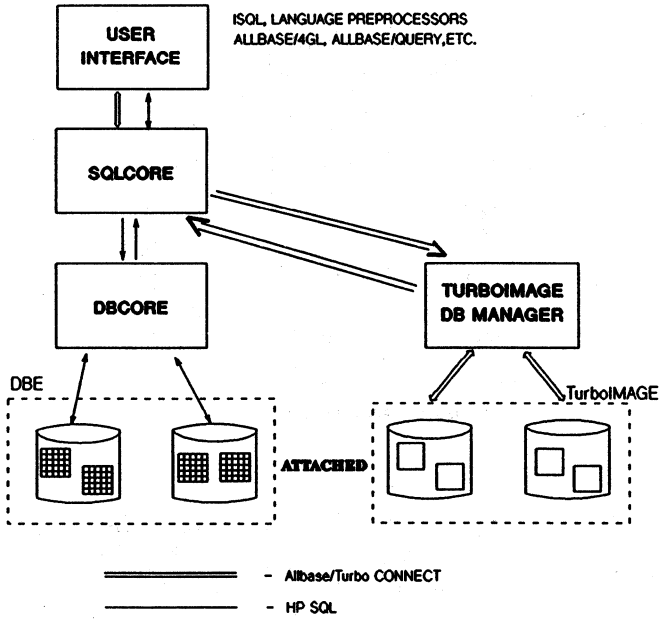


Figure 3. Allbase/Turbo CONNECT

To establish this access path, one has to first **ATTACH** a TurboIMAGE/XL database to an HP SQL DBEnvironment. This is done via the utility called ATCUTIL. Attaching involves defining a table entry in the DBEnvironment system catalog for each corresponding TurboIMAGE/XL data set. Among other things, for each data set the definition comprises the data set name, the corresponding field names, and their types. This information is used by SQLCORE for data formatting, for data display and in arithmetic operations performed on the data if needed. When the table entry is made, it is marked to indicate that it represents a TurboIMAGE/XL data set. This is done to differentiate a TurboIMAGE/XL data set from an HP SQL table so that the correct access path can be chosen. Multiple TurboIMAGE/XL databases can be attached to the same DBEnvironment and the same TurboIMAGE/XL database can be attached to multiple DBEnvironments.

Given that the structural information of the TurboIMAGE/XL data sets is recorded in the system catalog, it becomes imperative to translate the TurboIMAGE/XL information into an equivalent HP SQL representation. This is where the differences in the two systems become

apparent. This section describes how the main differences between the two systems affect the design of Allbase/Turbo CONNECT.

2.1 The Access Mechanism

In a relational database, the semantics of the data are not reflected in the physical organization of the data. The user queries the data with explicit semantics in the query.

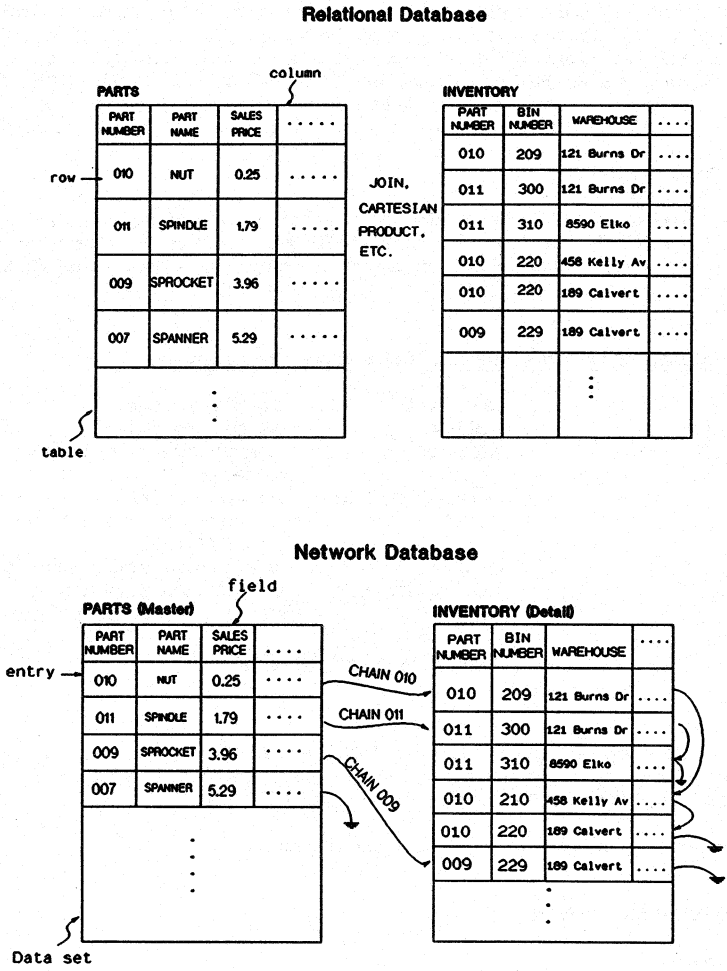


Figure 4 - The Access Mechanism

In a network database, the semantics are represented in the physical organization of the data itself. This is illustrated with an example from the schema described in Figure 4. In the network database, all the entries in the data set PARTS that have the same PARTNUMBER are linked together and hence, for the retrieval of this information, the database manager has only to follow this chained link. In a relational database the entries exist as separate tables and this information will have to be obtained by joining the two tables over the key word, PARTNUMBER. For faster access, the data in a relational database can be indexed.

Since the access mechanism is different for each database, we had to translate the relational access calls to DB intrinsic calls that operate on the network database. Most of the relational access is either in the form of indexed scans or serial scans on the data. The information on data access that is intercepted by Allbase/Turbo CONNECT is translated into DBFINDs and DBGETs at run time. Once intercepted, Allbase/Turbo CONNECT does its own optimization in terms of DBFINDS and DBGETs. Access to HP SQL tables is still handled by DBCORE.

2.2 Data Types

Allbase/Turbo CONNECT takes a close look at data types because in describing the TurboIMAGE/XL data set structure, TurboIMAGE/XL data types must be mapped into HP SQL data types. Table 2 shows the default mappings that Allbase/Turbo CONNECT applies:

TurboIMAGE data type	HP SQL data type	Description
I,J	smallint	16 bit integer
I2, J2	integer	32 bit integer
K1, K2	integer	conversion from binary to integer
Pn	decimal(n-1,0)	packed decimal
R4	float	conversion from HP3000 real to IEEE real
U(n), X(n)	char(n)	n byte character string
Zn	decimal(n,0)	conversion from zoned decimal to packed decimal

Table 2. Data Type Mappings

There are data types in both HP SQL and TurboIMAGE/XL which do not have an exact mapping to one another. In such cases, a "best fit" default mapping is chosen, and the DBA has the option of changing this mapping via the utility, ATCUTIL.

For the purpose of illustration, consider the following schema:

```
BEGIN DATABASE TIPART;

PASSWORDS:
   14 CLERK;
   12 BUYER;
   18 DO-ALL;
```

```
ITEMS:
   BINNUMBER        , K     (12,14/18);
   COUNTCYCLE       , 3I    (12/18);
   ITEMNUMBER       , I2    (/14,18);
   LASTCOUNTDATE    , X8    (12/18);
   PARTNAME         , X32   (14/12,18);
   PARTNUMBER       , X16   (14/12,18);
   SALESPRICE       , P12   (/12,18);
   LARGE-ITEM       , I158  (/12,18);
   WAREHOUSE        , X32   (/12,18);

SETS:
   NAME: PARTS, MANUAL;
   ENTRY:
      PARTNUMBER(1),
      PARTNAME,
      SALESPRICE,
      LARGE-ITEM;
   CAPACITY: 301;

   NAME: INVENTORY, DETAIL;
   ENTRY:
      PARTNUMBER(PARTS),
      BINNUMBER,
      WAREHOUSE,
      LASTCOUNTDATE,
      COUNTCYCLE;
   CAPACITY: 200;
END.
```

When compound items are mapped to separate items, the item name is appended with an underscore and a sequence number. For example, in the above schema, item COUNTCYCLE maps to columns COUNTCYCLE_1, COUNTCYCLE_2, COUNTCYCLE_3. Large TurboIMAGE/XL data types, such as I158 (item LARGE-ITEM in the above example), which are greater than 64 bits, are mapped to char and the user is given the option of splitting this into smaller individual data types (the option of splitting an item is available only if it is greater than 64 bits in size).

All of the TurboIMAGE/XL data types have at least one mapping which requires no conversion and is a direct move of bytes (we have chosen HP SQL's char type for this in most cases). This is necessary so that the data can be retrieved exactly as stored in TurboIMAGE/XL. Mapping exists for almost all TurboIMAGE/XL data types. All things considered, the various options in mapping give the DBA a lot of flexibility in deciding what to choose.

2.3 Security

The structuring of security in TurboIMAGE/XL and HP SQL is quite different. The fundamental difference in the two systems is that the HP SQL security is based on the user's logon name, while TurboIMAGE/XL security is based on a password defined in the schema. To account for this, the TurboIMAGE/XL class number of each user is stored in a separate privileged file. The class number is used at run time to do a DBOPEN with the password

corresponding to the user logon. The other difference in security is that in TurboIMAGE/XL there are read/write restrictions at the item and data set level, whereas in HP SQL there is a notion of Groups, and group access authorities. These similar concepts are mapped one-to-one to insure that no level of TurboIMAGE/XL security is bypassed.

To do this security mapping, an HP SQL Authorization Group is created corresponding to each TurboIMAGE/XL user class. These groups define the scope of HP SQL access rights for the end users. The authorization group is named GROUP*n* where *n* corresponds to the class number. Groups are not created for class 0 or classes with no users assigned to them.

For our sample schema, the group for class 14 (password CLERK) is created by ATCUTIL as follows:

```
CREATE GROUP TIPART.GROUP14;

GRANT CONNECT TO GROUP TIPART.GROUP14;
```

Similarly, groups for the other classes are created. Next, a series of views are created for each TurboIMAGE/XL data set. A view, in TurboIMAGE/XL terminology, can be regarded as a subset of a data set. It defines an access specification and also a selection criteria. Views are created for a data set only when there are read access restrictions associated with the user class at the data set and/or field level. The name of each view is based on the data set name, with the string "_V*n*" appended to it. Each *n* matches the corresponding HP SQL group number.

Each group is assigned read-only authority on its corresponding views since this group level security corresponds to the security level of each user class in TurboIMAGE/XL.

In our example, the following view and grant authority is created by ATCUTIL for class 14, data set PARTS:

```
CREATE VIEW TIPART.PARTS_V14 AS SELECT
PARTNUMBER, PARTNAME FROM TIPART.PARTS;

GRANT SELECT ON TIPART.PARTS_V14 TO TIPART.GROUP14;
```

The name of the view that an end user accesses varies, depending on the user class in use.

The DBA must add individual DBEnvironment users to the appropriate group(s) using ISQL. The DBA may also change the group names, or even assign views to individual users, as long as the changes are consistent with TurboIMAGE/XL security. For example, to add user JACK@PARTACCT (HP SQL understands the user logon in the format username@acctname), the DBA issues the following:

```
isql> ADD JACK@PARTACCT TO GROUP TIPART.GROUP14;
```

Now, the user JACK@PARTACCT can access view TIPART.PARTS_V14, with the same access restrictions as defined in the TurboIMAGE/XL schema.

2.4 Concurrency and Updates

HP SQL allows explicit locking at the table level only, whereas in TurboIMAGE/XL the granularity of locking goes down to the item level. HP SQL does implicit page level locking whenever a data page is accessed. The potential of a deadlock is increased when doing a mixed retrieval. That is, a retrieval involving both TurboIMAGE/XL and HP SQL data with locks on the data. A simple example is one user waiting on data set A and holding a lock on table B, while a second user is waiting on table B and holding a lock on data set A. Since TurboIMAGE/XL's deadlock detection mechanism is not compatible with DBCORE's, locks cannot be held in both database managers simultaneously. Thus:

1) Allbase/Turbo CONNECT does not allow update access. Hence, no locking is required.

2) Because no locking is done in TurboIMAGE/XL, all reads are "dirty".

Since there is no update access, logging and recovery procedures are not a concern. Any native update of an HP SQL table or a TurboIMAGE/XL data set is logged as usual by the respective database manager.

3. HOW TO SET UP ALLBASE/TURBO CONNECT

The Allbase/Turbo CONNECT utility, ATCUTIL, is used to establish the access path from HP SQL to TurboIMAGE/XL. ATCUTIL is strictly an administrative tool and is used by the DBA only. ATCUTIL, like ISQL and DBUTIL, is a command driven interface. Wherever possible, the commands are designed so as to assume practical default values.

3.1 Attaching

Establishing the access path minimally takes only 3 commands - (i) declare the TurboIMAGE/XL database (ii) declare the HP SQL DBEnvironment and (iii) attach the former to the latter. ATCUTIL enters the structural information of the TurboIMAGE/XL database in the system catalog of the DBEnvironment. This process is known as **ATTACHING** in Allbase/Turbo CONNECT terminology. This allows SQLCORE to recognize access to a TurboIMAGE/XL database as opposed to HP SQL tables. For example:

```
:ATCUTIL

>>SET TURBODB TIPART
>>SET SQLDBE PARTSDBE,MAINT=SQLMAINT
>>ATTACH
```

ATTACH couples the sample TurboIMAGE/XL database TIPART to PARTSDBE (a hypothetical DBEnvironment which already exists) with the default data type mapping. This mapping is stored in a separate privileged file, called ATCINFO. Information about other TurboIMAGE/XL databases, attached to this DBEnvironment, is also stored here. There is one ATCINFO file per DBEnvironment. In this example, assume that the DBA is also the

creator of the TurboIMAGE/XL database. Hence the maintenance word is not required. At this point, only the creator is ready to use Allbase/Turbo CONNECT.

3.2 Adding Users

The **ADD USER** command is used to add more users to the Allbase/Turbo CONNECT environment. For example:

```
>>ADD USER JACK@PARTACCT WITH CLASS=14,MODE=6
```

This user information is also stored in the ATCINFO and is used at run time when a DBOPEN is done. At this time all the security for JACK@PARTACCT gets defined as explained earlier (see 2.3). This command can be only issued by the TurboIMAGE/XL database creator who has DBA authority over the DBE.

3.3 Changing Default Mapping Information

The **UPDATE TYPE** command is used to update the mapping information in ATCINFO. For example:

```
>>UPDATE TYPE IN TABLE TIPART.INVENTORY TO SMALLINT WHERE &
  SQLITEM = BINNUMBER
```

This command updates the data type of item BINNUMBER in data set INVENTORY from K to smallint. The update is restricted to those data type mappings presented earlier. One can see that the table INVENTORY has been qualified with the TurboIMAGE/XL database name. This is because HP SQL table names have an 'owner'. The default used by **ATTACH** is the TurboIMAGE database name. **ATTACH** has an extended syntax which allows you to specify the owner name of your choice.

Item LARGE-ITEM of table PARTS is of type I158 and has been mapped by the **ATTACH** command to char(316) (see the data type mapping section above.) There is a **SPLIT** command used to divide these large items into a union of small items which reflects the true structure of the data. For example:

```
>>SPLIT TIPART.PARTS.LARGE_ITEM INTO PART_ID_CODE:I2:INTEGER,&
                                    SHORT_PART_DESC:X56:CHAR(56),&
                                    PART_VERSION_NO:I1:SMALLINT,&
                                    LONG_PART_DESC:X254:CHAR(254)
```

Note that item LARGE-ITEM in data set PARTS has been stored in the system catalog as LARGE_ITEM. This conversion is made because SQLCORE does not understand the "-" in data names.

In addition to the above commands, ATCUTIL has a host of other commands to do specific tasks e.g. **DISPLAY** the mapping and security information, **DELETE** users from the

Allbase/Turbo CONNECT environment, **LOG** all the user specified commands, **DETACH** a TurboIMAGE/XL database from a DBEnvironment, **UPDATE** user security, execute an ATCUTIL command file, **HELP** facility, etc.

4. RUN TIME CONSIDERATIONS

Some of the runtime aspects of Allbase/Turbo CONNECT are the conversion between data types, handling compound and split items, and security enforcement.

4.1 Data Types and Security

For data types that do not have a compatible mapping, we convert the data values that DBGET retrieves into the desired HP SQL type. For example, "classic" HP3000 floating point must be converted to the IEEE floating point format that is native to HP SQL and MPE/XL. Compound items and split items are treated specially, as a single TurboIMAGE/XL item must be returned as several HP SQL column values. Each of these may in turn require type conversion. When one or more items appear in an HP SQL predicate (described below), the predicate value(s) must be converted from the HP SQL format into the TurboIMAGE/XL format so that retrieved data entries may be qualified by the predicate and discarded or returned, as appropriate.

As outlined earlier, security is enforced by SQLCORE through the use of views. Allbase/Turbo CONNECT takes steps to insure that only the data item values that appear in a view are returned to the user making the query.

Each process has runtime access to the global information in the ATCINFO file as well as a private set of process local information. The process information encompasses the state of the current transaction, the optimization technique in use for the current SELECT request, etc.

4.1 Examples

Now that we have shown the mechanics of Allbase/Turbo CONNECT, let us illustrate some examples of an SQL query applied to TurboIMAGE/XL data. In general the syntax of a **SELECT** statement is:

SELECT *item names* **FROM** *table names* **WHERE** *predicate*

The example of Figure 5(a) shows a simple example of a range retrieval. Here, only those entries are retrieved from data set INVENTORY whose field BINNUMBER is less than equal to 300 and greater than 220. The predicate could be a complex boolean function of the boolean operators **and**, **or**, **not**, to choose just the entries one wants. As stated earlier, all the semantics of the data have to be expressed in the query.

The example of Figure 5(b) shows a pattern matching example. The symbol "%" specifies a wild card and "_" (not illustrated) specifies any single character. "_" and "%" can appear multiple times and in any combination in a pattern. The use of the ORDER BY clause displays the entries in a sorted order on the field specified after the clause. In the example, the entries from data set PARTS, whose field PARTNAME matches "SP%", are retrieved, sorted on the field PARTNUMBER

QUERY: SELECT * FROM tipart.inventory
WHERE binnumber <=300 AND
binnumber > 220

011	300	121 Burns Dr	....
012	229	189 Calvert	....

(a)

QUERY: SELECT * FROM tipart. parts
WHERE partname
LIKE 'SP%' ORDER
BY partnumber

007	SPANNER	5.29	....
009	SPROCKET	3.96	....
011	SPINDLE	1.79	....

(b)

QUERY: SELECT SUM (salesprice) FROM tipart. parts

11.29

(c)

Figure 5 - Examples

QUERY: SELECT partname, warehouse FROM tipart. parts, tipart. inventory
WHERE tipart.parts.partnumber = tipart.inventory.partnumber

NUT	121 Burns Dr
NUT	458 Kelly Av
NUT	189 Calvert
SPINDLE	121 Burns Dr
SPINDLE	8590 Elko
SPROCKET	189 Calvert

010	NUT	0.25	
011	SPINDLE	1.79	
009	SPROCKET	3.96	
007	SPANNER	5.29	

010	209	121 Burns Dr	
011	300	121 Burns Dr	
011	310	8590 Elko	
010	210	458 Kelly Av	
010	220	189 Calvert	
009	229	189 Calvert	

(d)

Figure 5 (contd) Examples

The example of Figure 5(c) illustrates the use of aggregate functions. Aggregate functions specify a value computed using the data in the argument. In this example, the SUM of the field SALESPRICE is computed over the data set PARTS. There are other aggregate functions like AVG (average), MAX (maximum), MIN (minimum), and COUNT (counts the number of rows that satisfy a given argument), that can be used.

The example of Figure 5(d) shows a join operation. Combining the data from two or more tables is referred to as a *join*. In the example, PARTNAME from data set PARTS and WAREHOUSE from data set INVENTORY are retrieved where the PARTNUMBER of the two data sets match. The graphical illustration shows which entries are joined with which. The predicate field gives the flexibility to achieve various kinds of joins. It is this feature of HP SQL which permits data retrieval from multiple TurboIMAGE/XL data sets. It is worthwhile to mention here that a join of a HP SQL table and a TurboIMAGE/XL data set is possible too.

The above examples are by no means exhaustive, but give an idea of the power of a SQL query.

5. SUMMARY

The paper has presented an overall picture of Allbase/Turbo. Hopefully, this has given the readers an insight into how Allbase/Turbo CONNECT operates, and how the differences in the two database environments have been accounted for.

The product is quite easy to set up via the utility, and though we did not spend much time in describing the power of SQL as a query language, readers with knowledge of SQL know that this opens up an entirely new world of powerful data reporting tools to the TurboIMAGE/XL users. Relational tools like Allbase/Query, Allbase/4GL, preprocessed SQL code can now be used to access TurboIMAGE/XL data.

Though the concept of independence of physical structure of the data is still new, HP has made a step forward by providing an integrated Allbase environment for its relational and network database management systems.

TITLE: Objected-Oriented Databases

AUTHOR: Mary Loomis

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5771

ALLBASE/NET: Remote Database Access

Leigh Anne R. Glasson
Hewlett-Packard
Distributed SQL Laboratory
19447 Pruneridge Avenue Mailstop 44UW
Cupertino, CA 95014 USA

Abstract

Distributed database management systems are fast becoming a focal point for many companies in the computer industry. ALLBASE/NET is Hewlett-Packard's first product in its evolutionary path toward a true distributed database management system. ALLBASE/NET allows users to access Allbase/SQL databases which are physically located on a different machine. Access to the remote database is completely transparent to the user.

This paper discusses the benefits provided by ALLBASE/NET. It will describe the internal workings of ALLBASE/NET, including what information is needed to access remote ALLBASE/SQL databases and how that information is provided for ALLBASE/NET to use.

The target audience of this paper is Application Developers who would be using ALLBASE/SQL remotely. Prior knowledge of ALLBASE/SQL is not required, however a basic understanding of networking would be helpful.

1 Introduction

As users move away from single CPU processing and dumb terminals toward intelligent workstations, the need for remote database access becomes apparent. Data may be stored on different machines for many reasons such as ownership, size, performance, or security, and ALLBASE/NET provides a way to access

this dispersed data as though it were stored locally. ALLBASE/NET provides flexibility. It will allow databases to be moved from one machine to another as needed without requiring change to existing application programs.

ALLBASE/NET allows an application on one Hewlett-Packard machine to access an ALLBASE/SQL DBEnvironment on another HP machine, provided the two machines are connected via a local area network (LAN) or a wide area network (WAN). The local machine is known as the client node, and the remote machine is known as the server node. An application runs on the client node, and accesses an ALLBASE/SQL DBEnvironment residing on the server node.

The ALLBASE/NET product is composed of several parts (Figure 1). On the client node there is an application program and the AliasDB file. On the server node is the ALLBASE/SQL backend process, the NETUsers file, and the listener daemon. The following applications can use ALLBASE/NET to access a remote ALLBASE/SQL DBEnvironment from the client node:

- ISQL
- ALLBASE/SQL Preprocessors (all languages)
- ALLBASE/QUERY
- ALLBASE/4GL
- a user-written application which has been preprocessed with an ALLBASE/SQL preprocessor.

2 The Listener Daemon

The listener daemon provides two important functions: (1) it "listens" for connection requests and (2) it sets up a direct communication line between the user and the database process. The listener monitors a specified datacommunications port for any incoming connection requests. When it receives one, it determines the identity of the client user and spawns a database process for that user. After spawning the database process, the listener redirects communications so that the user and database process communicate directly with

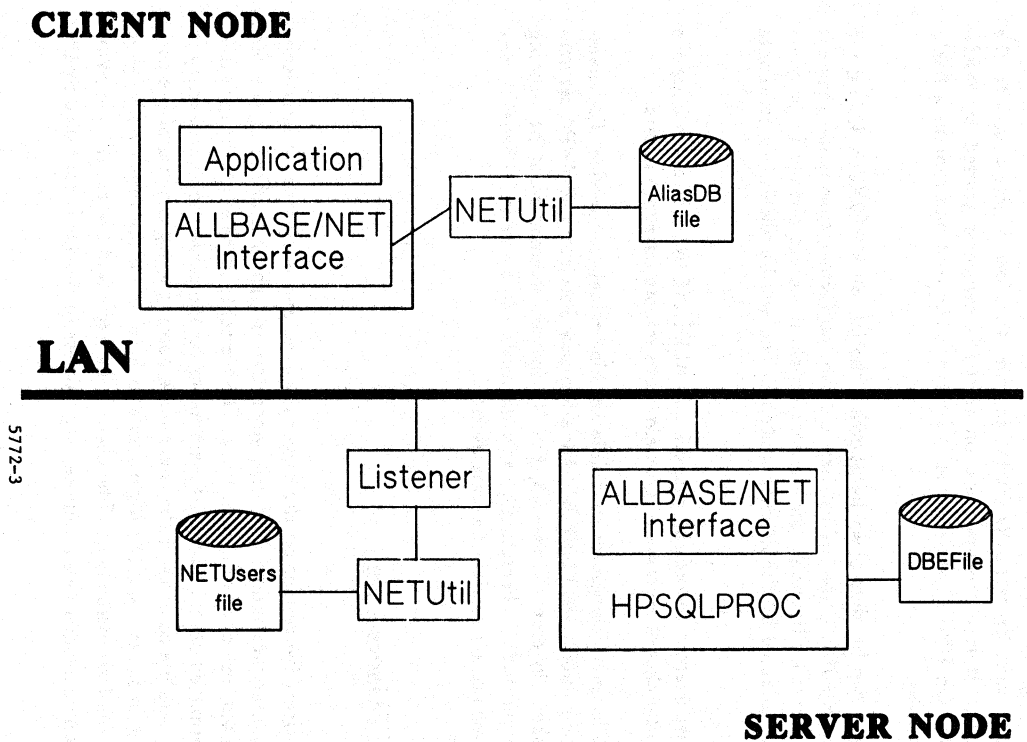


Figure 1 : ALLBASE/NET Architecture

one another over the network, thus leaving the listener daemon free to resume listening for more connection requests.

The listener design is consistent across all supported HP platforms. Therefore, the use and installation is the same regardeless of the machine being used. This eliminates any relearning for the different platforms.

The listener must be manually started in order to allow remote access to the server node. It is a streamed job on MPE/XL which is started by MANAGER.SYS. On HP-UX, the listener is started up by root and runs as a background process.

The listener can be shut down by aborting the job on MPE/XL or killing the process on HP-UX. Once the listener is down, no further connection requests will be accepted. However, any remote database access already in progress will not be affected by the listener shutdown and will be allowed to terminate normally.

The listener daemon records all network and ALLBASE/NET errors in a log file which can be checked in case of problems with remote database access.

There is a listener daemon for each different kind of network service. For example, on the HP9000 S800 both NS, HP's Network Services, and ARPA services are available. In order to use NS, the NS listener daemon must be running on the server node. Theoretically, new daemons can be added whenever new datacommunication services become available.

3 NETUsers File

The NETUsers file resides on the server machine and is created automatically by the maintenance utility, NETUtil, when the first entry is added to the file. The main purpose of this file is security. The file contains an entry for each user that is permitted access to a DBEnvironment on the server node. An entry in the NETUsers file consists of the client node name, the login name of the user trying to gain access, and the login name to be used on the server node when accessing the DBEnvironment. The client node name and the client login name together uniquely identifies a remote user. Access to this file is controled by the

NETUtil utility and is covered below.

The first message that the listener daemon receives will contain the client node name and the user's login name on the client node. The listener will use this information to locate the corresponding entry in the NETUsers file and map these to a login name to be used on the server node when spawning the database process. If a matching entry is not found in the NETUsers file, an error message is returned.

4 Transparency

Complete location transparency is one of the main goals of the ALLBASE/NET product. Location transparency means that an application doesn't have to know which machine the database is on. In fact, location transparency means that a database can be moved from machine to machine without causing any changes to the applications that access that database. With ALLBASE/NET, the only modification necessary to move a database to a different machine is a change to the AliasDB file entry for that database. The AliasDB file is discussed below.

In ALLBASE/NET, location transparency is achieved through an aliasing mechanism. An alias is an alternate name used in place of the actual DBEnvironment name in a START DBE or CONNECT statement. Therefore, the information associated with an alias name can be changed (i.e. the actual location of the DBEnvironment) without requiring any changes to existing code.

ALLBASE/NET handles connections in the following way:

- First, a connection to a LOCAL database with the given name is attempted
- If the local attempt fails, a REMOTE connection is attempted :
  - the alias name is looked-up in the AliasDB file
  - if a matching entry is found, the entry is used to determine which server node to communicate with and which datacomm protocol to use

- if a matching entry is not found, an appropriate error message is returned

Note that even if ALLBASE/NET has not been purchased, two error messages will appear: the first to indicate the DBEnvironment was not found locally, the second to indicate it was not found in the AliasDB file either.

5 AliasDB File

The AliasDB file resides on the client node. It contains an entry for each aliased DBEnvironment that can be accessed remotely from that node. An entry consists of the alias name, the nodename of the server, the actual location of the DBEnvironment on the server node, the type of machine the server is, and the type of data communication protocol being used. This information is needed to set up the communication line between the client and server machines and to make the connection to the remote database.

It is easy to change the location of a database or the type of datacomm protocol being used simply by altering the contents of the the AliasDB file. In fact, by changing the server node name to that of the client node, a loopback situation can be achieved.

Another flexibility that the AliasDB file provides is the ability to have multiple alias names for the same DBEnvironment. This would be helpful for developing a new application. While developing, a smaller database could be used to test out the application. When the application was completely tested, a new alias for the real database could be added to the AliasDB file, thus allowing access to the real database rather than the test database.

6 Internal Algorithm

A typical run-through of a remote connection using ISQL and ALLBASE/NET would look something like this:

On the client:

```
ISQL > CONNECT TO 'ColorDBE';
```

STEP 1: send the connect statement to the parser and linearizer and get back a linear tree.

STEP 2: check for a local file named 'ColorDBE'. If one exists, is it a DBEnvironment? If it is a DBE, make a local connection. If not, continue.

STEP 3: Search in the AliasDB file for an alias name of 'ColorDBE', if not found then return error.

STEP 4: If the datacommunications protocol being being used is NS, initiate a connection with the server node using the ipcconnect call, else if ARPA is being used, create a stream socket with the socket call and establish a connection with the connect call.

STEP 5: Send a message containing the client node name and client login name.

On the server:

STEP 6: The listener on the server node waits for a connection request. When received, the listener reads the NETUsers file to map the client node name/client login name pair to a server login name.

STEP 7: The listener starts up an ALLBASE/SQL database process.

On the client:

STEP 8: Send the linearized connect statement using the ipcsend command if using NS, or for ARPA the send command.

A connection is now established between the client application and the server database. Any valid SQL command will be passed through to the server for processing.

When a database is released, a final message is sent back to the client signaling the end of communication. When this LAST message is received, the virtual circuit is released with the shutdown call for ARPA and the ipcshutdown call for NS.

7 Maintaining the AliasDB and NETUsers Files

NETUtil is a command driven utility that resides on both the client and server nodes. It is used to create and maintain the AliasDB and NETUsers files. Entries can be added, deleted, modified, and displayed with NETUtil.

NETUtil is intended for use by the system administrator. The system administrator (MANAGER.SYS on MPE/XL and root on HP-UX) is the only user with write privileges on either file and the only user with read access to the NETUsers file. However, anyone is allowed to read the AliasDB file. Both the AliasDB and NETUsers files are PRIV files on MPE/XL.

Access to these files was restricted for security reasons. The NETUsers file tells ALLBASE/NET which users are allowed access to the server node. If this file were modifiable by anyone, then users could access any DBEnvironment they wanted to by creating an entry for themselves and mapping their login name to another login name that they know has certain privileges.

Update access to the AliasDB file was restricted to system administrators simply to ensure that the contents of the file are complete and correct. If write access to the file were given to everyone, the file would be difficult to maintain and contents could grow out of control.

8 Software Updates

ALLBASE/NET maintains a version stamp in both the AliasDB file and the NETUsers file. This version is checked to ensure compatibility. It is our goal to always allow a newer server to talk to an older client and when possible, allow older servers to talk to newer clients. This will allow gradual updates of software. First, the server software will be updated, then the clients can be

gradually updated as time permits since all clients are not required to be running the same version of ALLBASE/NET.

9 Platforms

The following Client/Server configurations are currently supported:

Client	Server
HP3000 S900	HP3000 S900
HP9000 S800	HP9000 S800
HP9000 S300	HP9000 S300

The following Client/Server configurations will be supported in the future:

Client	Server
HP9000 S300	HP9000 S800
HP9000 S300	HP3000 S900
HP3000 S900	HP9000 S800
HP9000 S800	HP3000 S900
HP9000 S800	HP9000 S300
Discless S300	HP9000 S800
Discless S300	HP9000 S300
Discless S800	HP9000 S800

10 Summary

The benefits of ALLBASE/NET include:

- transparent access to remote data
- flexibility
- security

ALLBASE/NET provides the foundation for Hewlett-Packard's distributed database strategy which is discussed further in Distributed Data Management From HP, a paper being presented at this conference by AJ Brown.

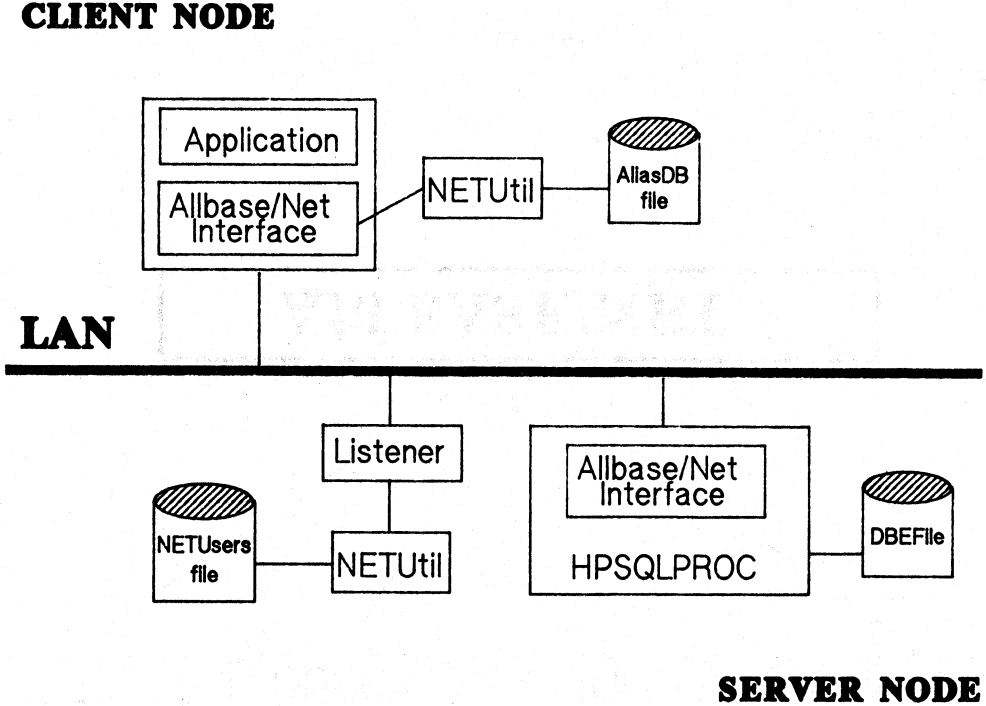
ALLBASE/NET:
Remote Database Access

Leigh Anne R. Glasson
Software Design Engineer

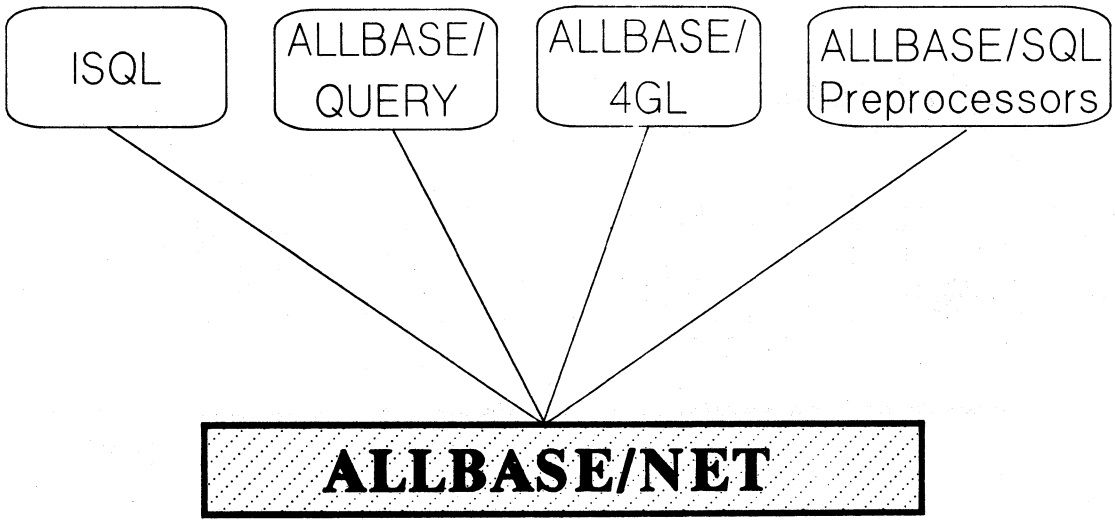
Outline of Presentation

- What ALLBASE/NET does
- How it does it
- How ALLBASE/NET fits into HP's Distributed Database strategy

HEWLETT PACKARD



ALLBASE/NET Architecture


ISQL
ALLBASE/ QUERY
ALLBASE/ 4GL
ALLBASE/SQL Preprocessors
ALLBASE/NET

Benefits of Remote Database Access

1. Increased quality and timeliness of information
2. Reduced data duplication
3. Increased application integration
4. Increased flexibility

ALLBASE/NET Listener

1. Listens for connection requests
2. Spawns a database process
3. Sets up a direct communication line between the user and the database process
4. Returns to listening for further connection requests

ALLBASE/NET Listener

- design is consistent across all HP platforms
- manually started and stopped
- Records all failures (ALLBASE/NET and network) in a log file
- Different listener for each network service

NETUsers File

- resides on server machine
- contains an entry for each remote user
- two main purposes
    1. security
    2. name conflict resolution

NETUsers File

Contains:

- Client node name
- Client login name
- Server login name

Contents of first message

. . .	client node name	client login name	. . .

- the listener uses this information to locate the corresponding entry in the NETUsers file
- If an entry is found, remote user is mapped to a local user
- If no entry is found, an error is returned
- NOTE: all authority must be granted to the LOCAL user

AliasDB File

- resides on client machine
- contains an entry for each local DBE that can be accessed via ALLBASE/NET
- two main purposes
    1. transparency
    2. communication line set-up

AliasDB File

Contains:

- Alias Name
- Database server type
- DBEnvironment name on Server
- Server node name
- Machine type
- Datacomm type

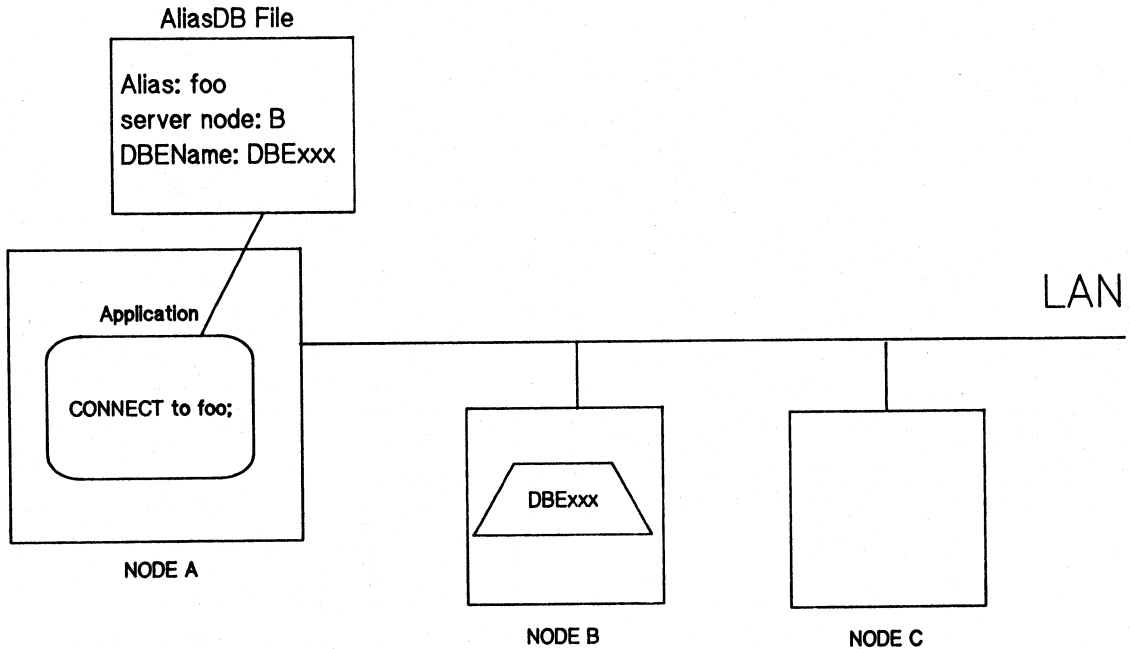
Connections using Alias names

- First, a connection to a LOCAL database with the given name is attempted
- If the local attempt fails, a REMOTE connection is attempted:
  - ◇ the alias name is looked-up in the AliasDB file
  - ◇ if a matching entry is found, the entry is used to determine which server node to communicate with and which datacomm protocol to use
  - ◇ if a matching entry is not found, an error message is returned

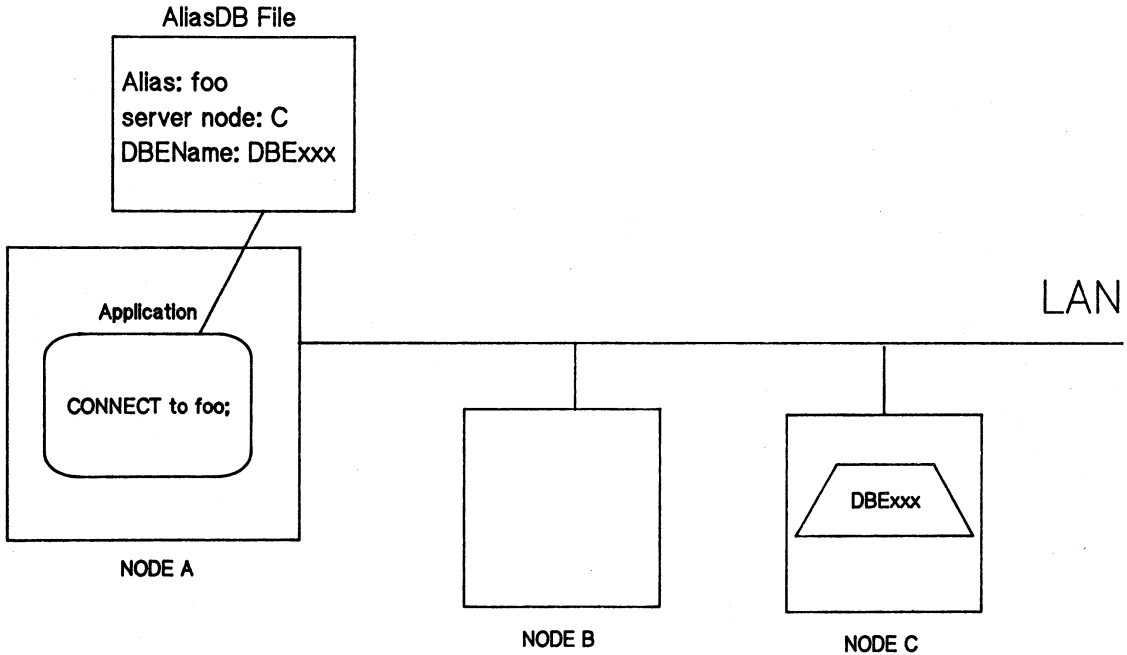
NOTE: even if ALLBASE/NET has not been purchased, two error messages will be returned for an unknown DBE name.

Transparency

- Location Transparency: applications don't have to know where a database physically resides
- Alias: an alternate name used in place of actual DBE name in a 'START DBE' or 'CONNECT' statement


AliasDB File
Alias: foo
server node: B
DBEName: DBExxx
Application
CONNECT to foo;
LAN
DBExxx
NODE A
NODE B
NODE C

5772-25


AliasDB File
Alias: foo
server node: C
DBEName: DBExxx
Application
CONNECT to foo;
LAN
NODE A
NODE B
DBExxx
NODE C

5772-26

INTERNAL ALGORITHM

On the client:

ISQL > CONNECT TO "ColorDBE";

STEP 1: send the connect statement to the parser and linearizer and get back a linear tree.

STEP 2: check for a local file named 'ColorDBE', If one exists, is it a DBEnvironment? If it is a DBE make a local connection. If not, continue.

STEP 3: Search in the AliasDB file for an alias name of 'ColorDBE', if not found then return an error.

STEP 4: If the datacommunications protocol being used is NS, initiate a connection with the server node using the ipcconnect call, else if ARPA is being used, create a stream socket with the socket call and establish a connection with the connect call.

STEP 5: Send a message containing the client node name and client login name.

On the server:

STEP 6: The listener on the server node waits for a connection request. When received, the listener reads the NETUsers file to map the client node name/client login name pair to a server login name.

STEP 7: The listener starts up an ALLBASE/SQL database process.

On the client:

STEP 8: Send the linearized connect statement using the ipcsend command if using NS, or for ARPS the send command.

NETUtil

- utility residing on both the client and server nodes
- used to create and maintain the AliasDB and NETUsers files
- used by the system administrator

Software Updates

- Version stamp maintained to ensure client/server compatibility

- Allows gradual software updates
  - ◇ server software is updated first
  - ◇ client software can then be updated gradually since multiple clients can be running different versions of ALLBASE/NET

PLATFORMS

The following configurations are currently supported:

Client	Server
HP3000 S900	HP3000 S900
HP9000 S800	HP9000 S800
HP9000 S300	HP9000 S300
*HP9000 S300	HP9000 S800
*HP9000 S800	HP9000 S300
*Diskless S300	HP9000 S800
*Diskless S300	HP9000 S300

* Available in 1989

The following configurations are planned for future releases:

Client	Server
HP3000 S900	HP9000 S800
HP9000 S800	HP3000 S900
Diskless S800	HP9000 S800

ALLBASE/NET and HP's future database strategy

ALLBASE/NET could be used as a foundation for future products such as

- Distributed Database Access
- Foreign Database Access

Summary

- allows transparent access to remote data
- provides flexibility
- promotes data sharing
- is Hewlett-Packard's first step into the Distributed Database arena

MAKING THE MOST OF MESSAGE CATALOGS

Glenn Cole, Consultant

10306 Hickory Ridge Rd., #332
Columbia, MD 21044-4613
telephone (301) 993-5840

Purpose

The purpose of this paper is to illustrate how message catalogs can improve system security and system identification, increase application flexibility, and save critical data stack space within an application. A technique is presented for modifying an existing application in order to make use of a message catalog. Also included is a brief discussion of the Application Message Facility, the new and improved version of message catalogs for Native Language Support. This material is directed toward both systems managers and applications programmers.

Introduction

The MPE message catalog facility provides a means of programmatically accessing messages that are contained within a specially-formatted file. These messages can be accessed by using standard MPE file system intrinsics. Parameter substitution is supported, and messages can be routed directly to the list device, retrieved for use by the application program, or both. This facility is provided by Hewlett-Packard (HP) as part of the Fundamental Operating System (FOS), and is available under all versions of MPE on all HP3000 computer systems at no extra cost.

Concepts

A message catalog (Figure 1) is created initially as a standard MPE flat file. It consists of from 1 to 62 "sets," each containing up to 32767 messages. Sets are indicated by "$SET *n*" beginning in column 1, where *n* is the set number (1 - 62). The sets must be in ascending sequence by set number, but the numbers need not be consecutive. For example, a message catalog with only 3 sets — numbered 2, 7, and 15 — is valid. The remainder of the line following the space after the set number can be used as a comment (e.g., "$SET 1 *** System Messages ***"). Other comments can be included by entering a "dollar-sign space" ($) in columns 1 and 2, with the remainder of the line (columns 3 to 72) used for the comment.

Each message in the file is uniquely identified by its set number and message number (1 - 32767). Message number 100 in set 1 can be completely unrelated to message number 100 in set 2.

Messages are entered under the appropriate $SET heading with the message number beginning in column 1, followed by a single space, followed by the text of the message. The message numbers within a set must be in ascending sequence, though not be consecutive. (Review Figure 1.)

There are three special-purpose characters that can be included in the message text. These include the continuation characters ampersand (&) and percent (%), and the parameter substitution character exclamation point (!). When the last non-blank character on a line is a continuation character, it indicates that the next line is to be included as part of the current message. All blanks immediately preceding the continuation character are ignored (Figure 1).

```
$SET 1  ***  SYSTEM MESSAGES  ***
$       ^The above comment begins HERE (2nd space)
$This comment line is INVALID (no space after "$")
1 LDEV #! IN USE BY FILE SYSTEM
05 IS "!" ON LDEV #! (Y/N)?
$  Note: (1) Numbers need not be CONSECUTIVE
$        (2) Leading zeros are allowed
$        (3) Comments may be inserted between msgs
9 ANOTHER MESSAGE (NOCIERROR)
$SET 3  ***  Non-Consecutive Set Number
10 This message is           &
continued.
$ The above prints "This message iscontinued." (Why?)
```

Figure 1 - Sample Message Catalog

The application program can elect to have the message routed directly to the user's terminal. This is where the two continuation characters differ. If the percent sign (%) is used, then a carriage return and line feed are printed before continuing with the message. The ampersand (&) indicates that the continuation line should be printed without first printing a carriage return and line feed. In this case, the continuation line will be printed adjacent to the current line. These symbols correspond roughly to the SPACE (%40) and %320 carriage control characters.

Each message can contain up to five exclamation points (!) for parameter substitution. Each passed parameter is inserted in the message where the corresponding exclamation point occurs, with the first parameter replacing the first exclamation point, the second parameter replacing the second exclamation point, until all parameters are included.

All CIERR and FSERR messages — and many others — are located in the system message catalog, CATALOG.PUB.SYS. When it is understood how a message catalog works, it is a small matter to customize this file as desired. The procedure for doing this is documented and does not require privileged mode (PM) capability.

Improving System Security

Some of the error messages that occur when signing on are downright "user-affectionate." These particular messages help the would-be intruder determine which part of the :HELLO command needs to be changed. Instead of telling the user

```
NON-EXISTENT ACCOUNT.  (CIERR 1437)
```

or

```
ACCT EXISTS, USER NAME DOESN'T.  (CIERR 1438)
```

respond to all invalid logon attempts with

```
INVALID LOGON ATTEMPT.
```

(or some other similar response). The specific messages related to system security are listed in Appendix A.

Improving System Identification

The system message catalog contains more than CIERR and FSERR messages. Many MPE commands get information from there. For example, when the operator (using the logical system console) sends a message via the :TELL command, the users receive data such as:

```
FROM OPERATOR:  System going down in 5 minutes.
```

The "FROM OPERATOR:" portion is taken directly from the system message catalog, where the message appears as:

```
8 FROM OPERATOR: !
```

In a DS network, most systems are assigned "names," like GUMBY or POKEY. Modify the system message catalog to read:

```
8 FROM GUMBY OPERATOR: !
```

then the same message appears as:

```
FROM GUMBY OPERATOR:  System going down in 5 minutes.
```

The message number is critical, and the exclamation point (!) is used for parameter substitution; the other characters are used "as is." The message for "normal" :TELL messages can be modified in a similar manner.

As another example, the :SHOWME command gets some information from the system message catalog. Following is an example of how this can be improved. The "SYSTEM:" line was added to the system message catalog.

```
:SHOWME
USER: *S149,MANAGER.SYS,PUB       (IN BREAK)
MPE VERSION: HP32033G.02.04.  (BASE G.02.04)
CURRENT: SUN, DEC 18, 1988,  1:42 PM
LOGON:   SUN, DEC 18, 1988,  1:23 PM
CPU SECONDS: 1          CONNECT MINUTES: 20
$STDIN LDEV: 39          $STDLIST LDEV: 39
SYSTEM:      GUMBY
:
```

The system welcome message, which normally appears after the "$STDIN LDEV:" line, would appear after the new "SYSTEM:" line. The identification line is at the end because the "$STDIN LDEV:" entry occurs in only one place in the system message catalog (versus three places for "USER:"). The line that occurs in the system message catalog as

```
65 $STDIN LDEV: !          $STDLIST LDEV: !
```

is modified to read

```
65 $STDIN LDEV: !          $STDLIST LDEV: ! %
SYSTEM:      GUMBY
```

Use a percent sign (%) to display a carriage return and line feed before continuing with the message. The remainder of this particular message is copied "as is." The "SYSTEM:" line is left justified to omit leading spaces when the message is retrieved.

The specific locations in the system message catalog for these messages are given in Appendix A.

Mechanics — Modifying the System Message Catalog

The mechanics of modifying the system message catalog are shown in Figure 2. The program, MAKECAT.PUB.SYS, performs the actual installation.

MAKECAT uses INPUT as the formal file designator for the input file. When using the BUILD entry point, it renames the existing file CATALOG.PUB.SYS as CAT*n*, where *n* is the first available number, zero suppressed. (The *Systems Operation and Resource Management* Manual, Part Number 32033-90005, is in error on this point.) If the file is not valid, then a short error message is displayed for each line believed to be in error. If this occurs, then the new catalog is not installed. If errors did not occur, then the new catalog is installed as CATALOG.PUB.SYS.

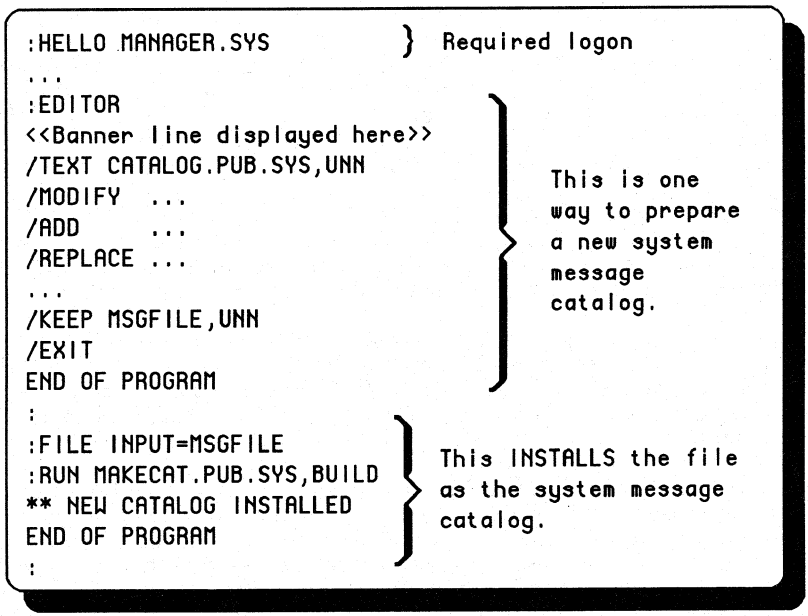


Figure 2 - Modifying the System Message Catalog

When a COLDSTART is performed, the system message catalog is brought in from tape. Therefore, the catalog should be customized prior to creating a new cold load tape. The *Systems Operation and Resource Management* Manual states that it is possible to install a new system message catalog as part of the ":SYSDUMP" dialogue.

When installed, as shown in Figure 2, if the input file is valid as a message catalog, it *will* be installed as the system message catalog. If an application message catalog was accidentally named, then it becomes the new system message catalog. To recover, install the backup copy CAT*n* just created; see the section "Mechanics - Modifying the System Message Catalog." If this fails, COLDSTART the system. Among other things, this installs the system message catalog that is present on the tape.

When a new version of MPE is installed, it includes a new system message catalog. Customizing the system message catalog must be repeated for each new version of MPE. This is accomplished by performing the modifications in a batch job. A liberal method is to place both the editing and the installation in a single batch job to be streamed as needed.

A more conservative approach is to perform each task separately, perhaps as two batch jobs. The first job could edit the system catalog, save the new version (as *myfile*, for example), and then run a comparison program (e.g., DIF.PUB.ROBELLE) against *myfile* and CATALOG.PUB.SYS. The output from this comparison should be examined manually. If all changes proceed as expected, then the actual installation of the new system catalog can be performed either manually or in a second batch job.

As an alternative, the author has submitted an application ("CATFIX") to the swap tape (originally on the VEGAS account) that simplifies the customization process. Documentation is present on the swap tape. The remainder of this section presents the ideas behind the application.

The system message catalog is treated as an *unnumbered* text file (Figure 2). Only the second record in the file is unnumbered; all other records are numbered. One feature of CATFIX is that it attempts to put line numbers where there are none, but it will not change existing line numbers.

The primary function of CATFIX is to merge two numbered text files. To use this application, first determine the line numbers (not just the *message* numbers) of the entries that need to be changed. Next, add the new version of that record to our merge file using both the same line number and the same message number, if any. Some lines will not have a message number. Examples of this are $SET lines, comment lines, and continuation lines).

There are two "$SET 2" entries in virtually every system message catalog. There are no real messages in between these entries (only comment lines), and the catalog seems to work fine, but clearly this is a mistake. To rectify this situation, the second (or first) "$SET 2" needs to be deleted.

In the following example, a comment line is added as well for illustration. The area of interest in the system message catalog looks like this:

```
2080   $
2085   $SET 2  CIERR Messages
2090   $
2095   $ ****************************************************
2100   $ *****       Set 2  ----  The CIERR messages       *****
2105   $ ****************************************************
2110   $
2115   $SET 2  CIERR messages
2117   10 EXECUTION OF THIS COMMAND HAS BEEN DISABLED (CIERR 10)
```

If the following data is entered in a file...

```
:EDITOR
<<Banner line displayed here>>
/ADD 2114
    2114    $ <<Line 2115 used to say "$SET 2" redundantly>>
    2115    $
    2116    //
...
/
```

...and then merged with the system message catalog, the following is obtained (changes indicated by ">>>"):

```
      2080  $
      2085  $SET 2  CIERR Messages
      2090  $
      2095  $ ******************************************************
      2100  $ *****       Set 2  ----  The CIERR messages       *****
      2105  $ ******************************************************
      2110  $
>>>   2114  $ <<Line 2115 used to say "$SET 2" redundantly>>
>>>   2115  $
      2117  10 EXECUTION OF THIS COMMAND HAS BEEN DISABLED (CIERR 10)
```

Line 2114 is inserted between existing lines 2110 and 2115, and the new line 2115 replaces the old line 2115. The second "$SET 2" was not actually deleted. Instead, it was replaced with a comment line. Appendix A lists many entries in the system message catalog that can be customized. **The line numbers in the system message catalog are not guaranteed to remain constant.**

Miscellaneous System Catalog Changes

Systems that still use multi-point (MTS) software have a problem where responses to prompts for account/user/group passwords are displayed on terminals. This can be alleviated by inserting a control-N (change to alternate character set) after the password prompts in the system message catalog. These messages are listed in Appendix A.

Some systems have a full-time operator. For these systems, it would be helpful for certain messages (e.g., those that require a =REPLY) to be displayed in inverse video with a bell (control-G) at the end. This can be accomplished with the system message catalog. These messages are NOT listed in Appendix A.

Any message displayed by MPE is probably contained in the system message catalog. If changing or adding to these messages would be helpful, the tools are available.

The remainder of this paper centers on using a message catalog with a user-written application program.

Preliminaries — Application Message Catalog

Two decisions must be made before using a message catalog with an application.

First, decide how the messages are going to be retrieved — in line or in a separate subroutine. Several factors can influence this decision. Is the application large or small? For small applications, in line calls offer both optimum flexibility and optimum convenience. In what language is the application written — FORTRAN, COBOL, etc.? With the global variables available in FORTRAN, in line calls are reasonable. Large COBOL applications almost require using a separate subroutine.

Secondly, decide on the numbering scheme for the messages. This includes deciding on whether to use a single set or multiple sets. If the messages are grouped strictly by program (or by subprogram), then any "common" messages will appear in several places throughout the catalog. If the messages are grouped differently, then it can be difficult to determine which program uses a given message. A combination of these ideas — grouping messages by program, with all "common" messages together — can be useful.

Mechanics — Preparing an Application Message Catalog

There are two primary differences between preparing a new system message catalog and preparing an application message catalog.

First, a new system message catalog can be installed while the system is up and running (therefore, the "old" catalog is still in use). An application message catalog, however, must not be in use when a new version is installed.

Secondly, MPE "knows" the final destination of an intended system message catalog (because of the BUILD entry point in MAKECAT). A new application message catalog must go through an intermediate step before being ready for use.

The mechanics of preparing an application message catalog are shown in Figure 3. The messages are entered using a favourite text editor, then saved as one would save FORTRAN or SPL source code, that is, as a numbered file with 80-byte, fixed-length records. As with the system message catalog, the key is the program MAKECAT.PUB.SYS. It still reads the data from formal file designator INPUT, but here it builds a *temporary* file called CATALOG. A short error message is displayed for each line believed to be in error.

When the input file is validated, the message "** VALID MESSAGE CATALOG" is displayed and the CATALOG file is built as a temporary file. Any existing temporary file named CATALOG is renamed to CAT*n*, where *n* is the first available number (i.e., CAT1, CAT2, etc.). The new catalog file includes a directory as a single user label. This directory includes the largest set number present, how many records are in the file, where each set begins, and the first message number in each set. By inference, performance can be improved by splitting a single large set into two (or more) smaller sets.

At this point, :SAVE CATALOG and then :RENAME CATALOG,*myfile*. (Of course, if a file named CATALOG exists, use the :RENAME CATALOG,*myfile*,TEMP command and then :SAVE *myfile*.) When this has been accomplished, the message catalog is ready for use. (Please review Figure 3.) The message catalog can still be modified by returning to the EDITOR, /TEXTing the file, making the changes, keeping the file either under the same name or under some other name, and rerunning MAKECAT.

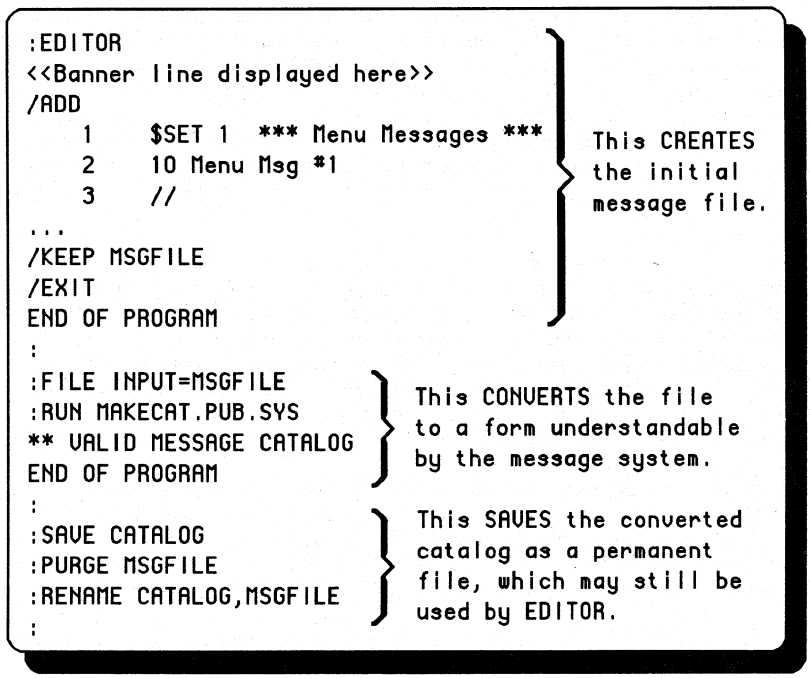


Figure 3 - Preparing the Message Catalog for Use

Mechanics — Using an Application Message Catalog

In order for the application program to access the message catalog, the file must first be opened with the standard FOPEN intrinsic. The AOPTIONS (access options) parameter must include Multi-Record and Nobuf (%420).

The intrinsic GENMESSAGE is the key. A brief description of this intrinsic is given in Figure 4. Performing parameter substitution with character strings is discussed in detail later in this paper.

The balance of this section assumes that the messages are to be accessed through a subprogram (for purposes of this discussion is called GETMSG). This is done without loss of generality as all other techniques are involved in the creation of this subprogram.

If GETMSG is written in COBOL, then it must be declared either as DYNAMIC or as SUBPROGRAM. When a DYNAMIC subprogram is called, it gets an "original" copy of its WORKING-STORAGE section. The variables in WORKING-STORAGE are re-initialized to the values defined when the subprogram was compiled. However, for a subprogram that is *not* declared as DYNAMIC, the variables in WORKING-STORAGE are *not* re-initialized. For example, imagine a non-DYNAMIC subprogram with a variable named COUNT described initially with a value of zero. Imagine also that during the course of this subprogram, COUNT is incremented by one. For the first execution of this subprogram, COUNT will be zero upon entry, and one upon exit. The second time through, since COUNT is not re-initialized to zero, COUNT will be one (the previous value) upon entry and two upon exit.

A single subprogram can be designed that performs both the FOPEN call and the GENMESSAGE call. The subprogram defines the file number initially with a value of zero. This value is checked and the FOPEN executed only when the file number is zero, i.e., on the very first call. A successful FOPEN changes the file number to a non-zero value, so subsequent calls will not execute the FOPEN. This approach means the message catalog will be closed implicitly by the file system when the application terminates, instead of being closed explicitly by the application.

The primary concern in the subprogram is retaining the file number for the message catalog from one call to the next. If this routine is coded in FORTRAN, the concept of a non-DYNAMIC subroutine can be implemented by placing in labelled COMMON the variable for the file number, then initializing it to zero in a BLOCK DATA subroutine.

The balance of this section assumes that the messages are to be accessed through a subprogram (for purposes of this discussion is called GETMSG). This is done without loss of generality as all other techniques are involved in the creation of this subprogram.

```
   I                          IV       IV       IV
msglen := GENMESSAGE (filenum, setnum, msgnum,

                          BA       IV
                       buff,    buffsize,

                         LV       LV       LV
                       parmask, parm1,...,parm5,

                         IV       I        O-V
                       msgdest, errnum);


buff     : A byte array to which the message is returned
buffsize: Length, in bytes, of "buff"; passed TO intrinsic

parmask:  16-bit Logical mask describing parms 1 thru 5
     Bit  (0:1) = 1.  Ignore rest of word & parms 1 thru 5
                = 0.  Rest of word, in 3-bit groups, defines
                      parm types for parms 1 thru 5
     Bits (1:3) = (Parm 1 type)
                  0 - byte address of string
                      (string terminated by ASCII "null")
                  1 - integer
                  2 - word address of "double" identifier
                  3 - ignore this parm
     Bits (4:3) = (Parm 2 type; same values as Parm 1)
     Bits (7:3) = (Parm 3 type; same values as Parm 1)
     (etc.)

msgdest :  destination of assembled message;
           0=$STDLIST;  >2=File number of destination file
```

Figure 4 - The GENMESSAGE Intrinsic

If GETMSG is written in COBOL, then it must be declared either as DYNAMIC or as SUBPROGRAM. Each time a DYNAMIC subprogram is called, it gets an "original" copy of its WORKING-STORAGE section. The variables in WORKING-STORAGE are re-initialized to the values defined when the subprogram was compiled. When a subprogram that is *not* declared as DYNAMIC is called, the variables in WORKING-STORAGE retain their previous values — they are *not* re-initialized.

For example, imagine a non-DYNAMIC subprogram with a variable named COUNT described as PIC S9(4) COMP VALUE ZERO. Imagine also that during the course of this subprogram, COUNT is incremented by one. For the first execution of this subprogram, COUNT will be zero upon entry, and one upon exit. The second time through, since COUNT is not re-initialized to zero, COUNT will be one (the previous value) upon entry and two upon exit.

A single subprogram can be designed that performs both the FOPEN call and the GENMESSAGE call. The subprogram defines the file number initially with a value of zero. This value is checked and the FOPEN executed only when the file number is zero, i.e., on the very first call. A successful FOPEN changes the file number to a non-zero value, so subsequent calls will not execute the FOPEN. This approach means the message catalog will be closed implicitly by the file system when the application terminates, instead of being closed explicitly by the application.

The primary concern in the subprogram is retaining the file number for the message catalog from one call to the next. If this routine is coded in FORTRAN, the concept of a non-DYNAMIC subroutine can be implemented by placing in labelled COMMON the variable for the file number, then initializing it to zero in a BLOCK DATA subroutine.

In reviewing the GENMESSAGE intrinsic (Figure 4), each parameter to be substituted in the message must be specified by data type (character string, integer, or double word). If the main application is coded in COBOL, then everything can be passed easily as a character string. Working with only one data type greatly simplifies the task of parameter substitution.

The parameter **parmask** (parm mask) specifies that character strings are passed as a *byte address*, and parameters **parm1** thru **parm5** are passed each as *logical* by value. In FORTRAN/3000, this can be implemented as

```
      CHARACTER*10  CPARM1
      ...
      IMSGLEN = GENMESSAGE (IFILE, ISET, IMSG, CMSGBUFF,
     +                      IMAXLEN, %0, CPARM1,,,,,, IERROR)
```

The compiler gives a warning for CPARM1, saying that the argument type is inconsistent. This is okay — the generated code performs as intended. If this variable is EQUIVALENCEd to one that is LOGICAL, then the compiler does not complain, but the code that is generated is not what is intended.

When FORTRAN '77 was implemented on the HP 3000, the developers saw this problem. Instead of specifying the lone parameter CPARM1 in the call to GENMESSAGE (which is now flagged as a fatal error), a special function is called explicitly to make the conversion. The code in the preceding example, when implemented in FORTRAN '77, becomes

```
      CHARACTER*10  CPARM1
      ...
      IMSGLEN = GENMESSAGE (IFILE, ISET, IMSG, CMSGBUFF,
     +           IMAXLEN, %0, BADDRESS(CPARM1),,,,,, IERROR)
```

The function BADDRESS is what makes this work, and no warnings are generated.

COBOL II is a little more bizarre. An example of this follows:

```
05  MSGFILE-PARM-1          PIC X(10).
05  MSGFILE-PARM-1-ADDR     PIC S9(4)  COMP  SYNC.
...
CALL INTRINSIC ".LOC."  USING @MSGFILE-PARM-1
                        GIVING  MSGFILE-PARM-1-ADDR.
...
CALL INTRINSIC "GENMESSAGE"  USING MSGFILE-FILENUM
                                   MSGFILE-SETNUM
                                   MSGFILE-MSGNUM
                                   MSGFILE-BUFF
                                   MSGFILE-BUFFLEN
                                   MSGFILE-PARM-MASK
                                   MSGFILE-PARM-1-ADDR
                                   \\ \\ \\ \\
                                   \\
                                   MSGFILE-ERROR
                             GIVING MSGFILE-REALLEN.
```

The pseudo-intrinsic ".LOC." is what enables parameter substitution with character strings in COBOL. This pseudo-intrinsic is described more fully in the *COBOL II/3000 Reference Manual*, Part Number 32233-90001. Since GETMSG is a non-DYNAMIC subprogram, the variable addresses are static once they are established. The procedure ".LOC." can be called just after opening the message catalog, once for each character string to be substituted. These byte addresses are retained from one call of the subprogram to the next.

When using parameter substitution with character strings, the end of the character string must be denoted explicitly. As stated in Figure 4, this is accomplished by terminating the string with an ASCII "null" character. As an example, if the tenth byte of a character string is ASCII "null," then the first nine bytes will be substituted into the message. This is true even if the last several bytes of the string to be substituted are blank (spaces). The responsibility of "deleting" these trailing blanks lies with the user. It is convenient for the calling routines to have this done in the subprogram GETMSG prior to calling the intrinsic GENMESSAGE.

Sample listings of the subprogram GETMSG are given in Appendices B through D. These routines are coded in COBOL II, FORTRAN/3000, and FORTRAN '77, respectively.

Modifying Existing Applications

This section describes the steps that were necessary to modify an existing COBOL application to take advantage of a message catalog. In this case, there were several subroutines that contained about 40 messages of 80 characters each in WORKING-STORAGE. The subroutine calls were nested to a maximum of four (4) levels deep. The application also made extensive use of IMAGE and VIEW. When the application called the subroutine that was four levels deep, it resulted in a STACK OVERFLOW and the program aborted.

First, the basic structure for GETMSG was laid out using the guidelines set forth previously in this paper. Next, all the messages in all the subroutines were checked to see how many parameters were needed for substitution. Since each subprogram had all of its messages grouped together in WORKING- STORAGE, this did not consume much time. As an alternative, one can assume five parameters are present — the maximum number supported by GENMESSAGE. Then, if fewer parameters are found, GETMSG can be modified to strip the trailing blanks off of the appropriate number of character strings and to ignore the rest.)

Each subroutine needed to know which set number and message numbers to use since these were the primary input parameters for GETMSG. These numbers were assigned, arbitrarily, based on the number of messages used by a given subprogram. All subroutines were assigned set number 1, and the message numbers were assigned as 1-99, 100-199, 200-299, etc.

In each subprogram, the messages were grouped together in WORKING-STORAGE. When a subprogram was /TEXT into the editor, this range of lines was kept in a separate file. This auxiliary file was modified later so that it could be included as part of the message catalog file. This meant that the actual messages did not have to be re-entered.

In the subprogram, the variable names assigned to the messages remained the same. Each description was changed from two FILLERs, each PIC X(40), to no FILLERs and PIC S9(4) COMP, and a VALUE equal to the message number was assigned. The message numbers were created in sequence, with the first message getting the lowest message number assigned to the subprogram. (See Figure 5 for a "before" and "after" look at this area.) Fields were added also to hold the set number, the generic message number, the retrieved message, and three 8-byte parameters for substitution.

```
01  PROGRAM-MESSAGE-AREAS.
    05  PM-NO-SUCH-ENTRY          PIC X(40)  VALUE
        "** No such entry; please try again.     ".
    05  PM-DUPLICATE-TRAILER      PIC X(40)  VALUE
        "** That Trailer already exists!         ".
     ⋮
        Working-Storage before GETMSG

01  PROGRAM-MESSAGE-AREAS.
    05  PM-NO-SUCH-ENTRY      PIC S9(4) COMP VALUE 101.
    05  PM-DUPLICATE-TRAILER  PIC S9(4) COMP VALUE 102.
     ⋮
01  GETMSG-PARMS.
    05  GP-SET-NUMBER         PIC S9(4) COMP.
    05  GP-MSG-NUMBER         PIC S9(4) COMP.
    05  GP-PARM-1             PIC X(08).
    05  GP-PARM-2             PIC X(08).
    05  GP-PARM-3             PIC X(08).
    05  GP-MESSAGE            PIC X(80).
⋮
        Working-Storage after GETMSG
```

Figure 5 - Working-Storage before and after GETMSG.

The PROCEDURE DIVISION was searched for all moves to the VIEW window area. In general, each of these MOVE statements was replaced by two other statements — a MOVE of the desired Message Number to a holding area, and a PERFORM of a utility routine to retrieve the message. The set number did not have to be moved, since it remained constant throughout the subprogram. If parameter substitution was required for a given message, then the necessary value(s) also were moved prior to the PERFORM.

The utility routine consisted only of a CALL to GETMSG, followed by a MOVE of the returned message to the VIEW window area. It was placed in a separate paragraph both for possible future optimization, and in case the basic technique for message retrieval changed. If an error occurred in GETMSG (e.g., a missing set number/message number combination), this was indicated by the returned message.

The data area saved by these modifications was tremendous, and the "longest path" worked, so this was indeed a solution. Also significant is that it took two days to research and implement. There was little new coding involved, and the technique was not involved. Regarding performance, the delay between this and the previous hard-coded method was barely perceptible.

This message catalog appeared high on the I/O usage report produced by the contributed program FILERPT. This program can be found in the contributed library. If Robelle's products are used, FILERPT is also in QLIB.ROBELLE. All messages were grouped in one set.

In retrospect, one optimization would be to keep the same message numbers, but to split them out over multiple sets. For example, set 1 could contain message numbers 1 - 500, set 2 messages 501 - 1000, etc. Next, GETMSG could be modified to select the set number based on the requested message number. Other subprograms need not be modified. This technique does not require knowledge of how frequently any particular subprogram is being accessed.

Another Alternative: Native Language Support

In the time since GETMSG first was implemented, HP introduced Native Language Support (NLS) in late 1984. As with the standard MPE message catalog facility, this is included as part of the Fundamental Operating System. Native Language Support is an excellent idea, and it embodies far more than an improved version of message catalogs (known as the "application message facility" under NLS). Only the application message facility is discussed below; most of NLS goes beyond the scope of this paper. Users are encouraged to read this manual.

A chart comparing MPE message catalogs with NLS is shown in Figure 6. The two techniques are similar. The GETMSG routine described earlier can be re-written to use NLS without modifying other subprograms.

With MPE message catalogs, all spaces immediately preceding a continuation character are ignored. This is not true with the application message facility. In NLS, all characters preceding a continuation character are significant. The worst place for the continuation character, in terms of data compression, is in the last position of the record.

The amount of disc space consumed by a message catalog under NLS is directly proportional to the average message length. An NLS message catalog whose average entry takes up one-half of the record consumes about one-half of the disc space of the equivalent MPE message catalog. As an example, the system message catalog takes up about 40% less space when formatted for NLS.

Regarding physical disc I/O's, a brief study of the File Close records in the system log file shows that while the number of *records* processed remains the same whether or not NLS is used, the number of *blocks* processed is reduced anywhere from three (3) to over ten (10) times. That is, NLS can require less than one-tenth of the disc I/O's of an equivalent MPE message catalog. Note also that the blocks are smaller under NLS — 128 x 1 = 128 words for NLS versus 40 x 16 = 640 words for the MPE message catalog.

	MPE	NLS
Limits:		
Set #'s:	1 - 62	1 - 255
Message #'s:	1 - 32767	1 - 32766
Directory:	By Set #; single user label	By Message #; first few records
Data Compression?	NO	YES
Physical Characteristics:	REC=40,16,F,BINARY; CODE=0	REC=128,1,F,BINARY; CODE=MGCAT
Formatting Program:	MAKECAT.PUB.SYS	GENCAT.PUB.SYS
Intrinsics Used:	FOPEN, FCLOSE, GENMESSAGE	CATOPEN, CATCLOSE, CATREAD
Order of Parameter Substitution:	Determined by calling sequence.	May be forced by message. If not, then determined by calling sequence.

Figure 6 - MPE Message Catalog vs. NLS

Conclusion

All users are encouraged to examine the possible uses of message catalogs. A message catalog is relatively easy to use and customize. As with the case presented in this paper, it can prove critical to the success of an application. NLS also should be examined closely. HP has put considerable time, thought, and effort into this, and has made these tools available to all users at no extra cost.

Appendix A.
Some system messages in CATALOG.PUB.SYS.

EDITOR line #	Message (as of MPE V/E V-Delta-3)
50 (op :TELL)	8 FROM OPERATOR: !
55 (:WARN)	9 OPERATOR WARNING: !
270	53 ENTER GROUP (!) PASSWORD:
275	54 ENTER ACCOUNT (!) PASSWORD:
295	55 ENTER USER (!) PASSWORD:
385	80 CAN'T INITIATE NEW SESSIONS NOW
491	125 ENTER OLD USER PASSWORD:
491.5	126 ENTER NEW USER PASSWORD:
492	127 ENTER NEW USER PASSWORD AGAIN:
860	251 ! ABORTED BY SYSTEM MANAGEMENT
2085 (comment	$SET 2 CIERR Messages
2115 out one)	$SET 2 CIERR messages
6180	1402 EXPECTED HELLO, :JOB, :DATA, OR (CMD) AS LOGON...
6225	1421 (COMMAND) LOGON IS MISSING RIGHT PARENTHESIS ...
6230	1422 (COMMAND) LOGON IS MISSING LEFT PARENTHESIS ...
6235	1423 EXPECTED JOB NAME. (CIERR 1423)
6240	1424 EXPECTED [SESSION NAME,] USER.ACCT [,GROUP] ...
6245	1425 EXPECTED USER PASSWORD. (CIERR 1425)
6250	1426 EXPECTED ACCOUNT NAME. (CIERR 1426)
6255	1427 EXPECTED ACCOUNT PASSWORD. (CIERR 1427)
6265	1429 EXPECTED GROUP NAME. (CIERR 1429)
6270	1430 EXPECTED GROUP PASSWORD. (CIERR 1430)
6290	1434 FIRST CHARACTER IN NAME NOT ALPHABETIC. ...
6295	1435 NAME GREATER THAN 8 CHARACTERS LONG. ...
6300	1436 ACCT/USER EXIST, GROUP NAME DOESN'T. ...
6305	1437 NON-EXISTENT ACCOUNT. (CIERR 1437)
6310	1438 ACCT EXISTS, USER NAME DOESN'T. (CIERR 1438)
6315	1439 ACCT/USER EXIST, NO HOME GROUP FOR USER. ...
6320	1441 INCORRECT PASSWORD. (CIERR 1441)
6335	1444 MISSING PASSWORD. (CIERR 1444)
13335 (:TELL)	3 FROM/!/!
13510	50 END OF PROGRAM
13530	54 END OF REMOTE PROGRAM
13585(:SHOWME)	65 $STDIN LDEV: ! $STDLIST LDEV: !
13590	66 END OF REMOTE PROGRAM

Appendix B.
COBOL II /3000 source for GETMSG.

```
$CONTROL SUBPROGRAM
 IDENTIFICATION DIVISION.
 PROGRAM-ID.  GETMSG.
*
*  This subroutine is responsible for retrieving the named message.
*  It opens the message catalog (if needed), delimits the character
*  strings used in parameter substitution (max. 3), and places the
*  returned message in a buffer.  If the desired message is not
*  found, then the message returned so indicates.
*
*  Note:  COBOL '85 features are used, but these are not critical.
*
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.  CONDITION-CODE IS C-C.

 DATA DIVISION.
 WORKING-STORAGE SECTION.
 01  PROGRAM-INDICATORS.
     05  PC-YES                      PIC X      VALUE "Y".
     05  PC-NO                       PIC X      VALUE "N".
*
     05  IS-MSGFILE-OPEN             PIC X      VALUE "N".
         88  MSGFILE-IS-OPEN                    VALUE "Y".

 01  PROGRAM-CONSTANTS.
     05  PC-NULL                     PIC X      VALUE %0.
     05  PC-MAX-STRING-LEN           PIC S9(4)  COMP  VALUE 08.
     05  PC-MASK-EXAMINE-PARMS       PIC S9(9)  COMP  VALUE %0.

 01  PROGRAM-INTRINSIC-AREAS.
     05  PI-ERROR                    PIC S9(4)  COMP.
         88  ERROR-SAYS-BAD-SET-OR-MSG      VALUE 3, 4.
         88  ERROR-SAYS-OVERFLOW            VALUE 6.
*
     05  PI-REALLEN                  PIC S9(4)  COMP.
*
     05  PI-MSGFILE-NAME             PIC X(28)  VALUE "CATFILE.PUB.PROD ".
     05  PI-MSGFILE-NUM              PIC S9(4)  COMP.
     05  PI-MSGFILE-BUFF             PIC X(80).
     05  FILLER                      PIC X(80).
     05  PI-MSGFILE-BUFFLEN          PIC S9(4)  COMP  VALUE 80.
     05  PI-MSGFILE-PARM-1           PIC X(10).
     05  PI-MSGFILE-PARM-2           PIC X(10).
     05  PI-MSGFILE-PARM-3           PIC X(10).
     05  PI-MSGFILE-PARM-1-ADDR      PIC S9(4)  COMP.
     05  PI-MSGFILE-PARM-2-ADDR      PIC S9(4)  COMP.
     05  PI-MSGFILE-PARM-3-ADDR      PIC S9(4)  COMP.

 01  PROGRAM-WORK-AREAS.
     05  SUB1                        PIC S9(4)  COMP.
*
     05  PW-STRING                   PIC X(10).

 01  PROGRAM-MESSAGE-AREAS.
     05  PM-OPEN-FAILED.
         10  FILLER                  PIC X(44)  VALUE
```

```
            "* * * OPEN failed on message catalog; error ".
        10  PM-OPEN-ERROR          PIC -ZZZ9.
        10  FILLER                 PIC X(31)  VALUE
            ".  Please tell someone. * * *  ".
    05  PM-GENMESSAGE-CCL.
        10  FILLER                 PIC X(40)  VALUE
            "* * * Message could not be retrieved; er".
        10  FILLER                 PIC X(40)  VALUE
            "ror CCL.  Please tell someone. * * *    ".
    05  PM-GENMESSAGE-ERROR.
        10  FILLER                 PIC X(44)  VALUE
            "* * * Message could not be retrieved; error ".
        10  PM-GENMSG-ERROR        PIC -ZZZ9.
        10  FILLER                 PIC X(31)  VALUE
            ".  Please tell someone. * * *  ".
    05  PM-MISSING-SET-MSG.
        10  FILLER                 PIC X(27)  VALUE
            "* * * Message missing; Set ".
        10  PM-SETNUM              PIC -ZZZ9.
        10  FILLER                 PIC X(06)  VALUE ", Msg ".
        10  PM-MSGNUM              PIC -ZZZ9.
        10  FILLER                 PIC X(37)  VALUE
            ".  Please tell someone. * * *       ".

LINKAGE SECTION.
01  LS-SETNUM                      PIC S9(4)  COMP.
01  LS-MSGNUM                      PIC S9(4)  COMP.
01  LS-PARMS.
    05  LS-PARM-1                  PIC X(08).
    05  LS-PARM-2                  PIC X(08).
    05  LS-PARM-3                  PIC X(08).
01  LS-BUFFER.                     PIC X(80).

PROCEDURE DIVISION  USING  LS-SETNUM, LS-MSGNUM, LS-PARMS, LS-BUFFER.

0000-DRIVER.
    PERFORM 1000-INITIALIZATION  THRU  1000-EXIT.
    PERFORM 2000-MAIN-ROUTINE    THRU  2000-EXIT.

*   File closes will be done by MPE when main program ends.

    GOBACK.

1000-INITIALIZATION.
    IF MSGFILE-IS-OPEN
        GO TO 1000-EXIT.
*
*   %5=FOptions old, perm, ASCII; %420=AOptions MR, NoBuf
*
    CALL INTRINSIC "FOPEN"  USING PI-MSGFILE-NAME \%5\ \%420\
                            GIVING PI-MSGFILE-NUM.

    IF PI-MSGFILE-NUM  NOT =  ZERO
        MOVE PC-YES  TO  IS-MSGFILE-OPEN
    ELSE
        CALL INTRINSIC "FCHECK"  USING PI-MSGFILE-NUM, PI-ERROR
        MOVE PC-NO                TO  IS-MSGFILE-OPEN
        MOVE PI-ERROR             TO  PM-OPEN-ERROR
        MOVE PM-OPEN-FAILED       TO  LS-BUFFER.
*
    CALL INTRINSIC ".LOC."  USING @PI-MSGFILE-PARM-1
```

```
                          GIVING  PI-MSGFILE-PARM-1-ADDR.

    CALL INTRINSIC ".LOC."  USING @PI-MSGFILE-PARM-2
                          GIVING  PI-MSGFILE-PARM-2-ADDR.

    CALL INTRINSIC ".LOC."  USING @PI-MSGFILE-PARM-3
                          GIVING  PI-MSGFILE-PARM-3-ADDR.
1000-EXIT.

2000-MAIN-ROUTINE.
    IF NOT MSGFILE-IS-OPEN
        GO TO 2000-EXIT.
*
    PERFORM 2100-SETUP-PARMS.
    MOVE SPACES  TO  PI-MSGFILE-BUFF.
    MOVE ZERO    TO  PI-ERROR.
    PERFORM 2001-GET-MSG.
2000-EXIT.

2001-GET-MSG.
    CALL INTRINSIC "GENMESSAGE"       USING    PI-MSGFILE-NUM
                                               LS-SETNUM
                                               LS-MSGNUM
                                               PI-MSGFILE-BUFF
                                               PI-MSGFILE-BUFFLEN
                                               PC-MASK-EXAMINE-PARMS
                                               PI-MSGFILE-PARM-1-ADDR
                                               PI-MSGFILE-PARM-2-ADDR
                                               PI-MSGFILE-PARM-3-ADDR
                                               \\ \\ \\
                                               PI-ERROR
                                     GIVING    PI-REALLEN.
    IF C-C = ZERO
        MOVE PI-MSGFILE-BUFF             TO  LS-BUFFER
    ELSE
    IF C-C < ZERO
        MOVE PM-GENMESSAGE-CCL           TO  LS-BUFFER
    ELSE
        IF ERROR-SAYS-BAD-SET-OR-MSG
            MOVE LS-SETNUM               TO  PM-SETNUM
            MOVE LS-MSGNUM               TO  PM-MSGNUM
            MOVE PM-MISSING-SET-MSG      TO  LS-BUFFER
        ELSE
        IF ERROR-SAYS-OVERFLOW
            MOVE PI-MSGFILE-BUFF         TO  LS-BUFFER
        ELSE
            MOVE PI-ERROR                TO  PM-GENMSG-ERROR
            MOVE PM-GENMESSAGE-ERROR     TO  LS-BUFFER.
2001-EXIT.

2100-SETUP-PARMS.
*
*  Note that PW-STRING is longer than any of the LS-PARM-... fields
*  to guarantee at least one blank at the end of string.  This is
*  important for the next routine.
*
    MOVE LS-PARM-1                       TO  PW-STRING.
    PERFORM 2110-SETUP-A-PARM.
    MOVE PW-STRING                       TO  PI-MSGFILE-PARM-1.

    MOVE LS-PARM-2                       TO  PW-STRING.
```

```
     PERFORM 2110-SETUP-A-PARM.
     MOVE PW-STRING                      TO  PI-MSGFILE-PARM-2.

     MOVE LS-PARM-3                      TO  PW-STRING.
     PERFORM 2110-SETUP-A-PARM.
     MOVE PW-STRING                      TO  PI-MSGFILE-PARM-3.
 2100-EXIT.

 2110-SETUP-A-PARM.
*
*  This is the only routine that uses COBOL '85 features.
*
*  To eliminate use of these features, set up PW-STRING as a group with
*  PW-STRING-CHAR under it as PIC X(01)  OCCURS 10 TIMES.  Search this
*  table backwards, character by character, until a non-blank character
*  is found.  Insert the string delimiter just after this position.
*
*  That is why the substring feature of COBOL '85 is used here.
*
     IF PW-STRING  =  SPACES
         MOVE PC-NULL  TO  PW-STRING (1:1)
     ELSE
*
*  Unlike the FORTRAN examples of GETMSG, this routine places the
*  end-of-string delimiter after the last non-blank character.
*
         MOVE 1  TO  SUB1
         PERFORM  UNTIL SUB1 > PC-MAX-STRING-LEN
                     OR PW-STRING (SUB1: ) = SPACES
             ADD 1  TO  SUB1
             END-PERFORM
         MOVE PC-NULL  TO  PW-STRING (SUB1:1).
 2110-EXIT.


*END OF SUBPROGRAM*
```

Appendix C.
FORTRAN/3000 source for GETMSG.

```
      BLOCK DATA GETMSGI
C
C  This BLOCK DATA area initializes the COMMON fields for GETMSG,
C  thus simulating a (COBOL) DYNAMIC subprogram.  This way, the
C  subroutine "knows" whether or not it needs to open the message
C  catalog file.  As an aside, the name of the message catalog is
C  given here, as is the maximum length of the retrieved message
C  that GETMSG can handle.
C
      COMMON /GMCAT/  IFILE, IMAXLEN, CCATNAME
      CHARACTER*28  CCATNAME
      INTEGER       IFILE, IMAXLEN
      DATA IFILE /0/, IMAXLEN /80/, CCATNAME /"CATFILE.PUB.PROD "/
      END
C
C
      SUBROUTINE GETMSG (ISET, IMSG, CPARM1,CPARM2,CPARM3, CMSG)
C
      CHARACTER*08  CPARM1, CPARM2, CPARM3
      CHARACTER*80  CMSG
      INTEGER       ISET, IMSG
C
C  This subroutine is responsible for retrieving the desired message.
C  It opens the message catalog (if needed), delimits the character
C  strings used in parameter substitution (max. 3), and places the
C  returned message in a buffer.  If the desired message is not
C  found, then the message returned so indicates.
C
      COMMON /GMCAT/  IFILE, IMAXLEN, CCATNAME
      CHARACTER*28  CCATNAME
      INTEGER       IFILE, IMAXLEN
C
      CHARACTER*10  CPARML (3)
      INTEGER       IERROR, IMSGLEN
C
      SYSTEM INTRINSIC  FOPEN, GENMESSAGE
C
C     ------------------------------------------------
C     Open message catalog if not open already.
C
      IF (IFILE .NE. 0)  GO TO 10
          IFILE = FOPEN (CCATNAME, %5L, %420L)
          IF (.CC.)  5, 9, 5
5             CMSG = "** Message catalog not open.  Call M.I.S."
              GO TO 99
9         CONTINUE
10    CONTINUE
C
C     ------------------------------------------------
C     Init parms (ASCII NULL (%0) at end of each string).
C     For simplicity, end-of-string is assumed to be the first blank.
C
      CPARML (1) [1:8] = CPARM1
      CPARML (2) [1:8] = CPARM2
      CPARML (3) [1:8] = CPARM3
      DO 20  I = 1, 3
          CPARML (I) [9:2] = "  "
```

```
          CPARML (I) [INDEX(CPARML(I)," ") :1] = %OC
20        CONTINUE
C
C     ------------------------------------------------
C     Actual message retrieval; %O indicates all parms are STRINGS
C     Note:  Since CPARML is type CHARACTER where LOGICAL is
C            expected, there will be warnings (arg type inconsistent)
C            Getting around the warnings is not worth the effort.
C            (EQUIVALENCE to LOGICAL variables does not work.)
C
      IMSGLEN = GENMESSAGE (IFILE, ISET, IMSG, CMSG, IMAXLEN,
     +                      %O, CPARML(1), CPARML(2), CPARML(3),,,,IERROR)
C
      IF (IERROR .EQ. 0)  GO TO 99
C
      CMSG = "** Message retrieval failed.  Call M.I.S."
C
C     ------------------------------------------------
99    continue
      RETURN
      END
```

Appendix D.
FORTRAN '77 source for GETMSG.

```
$standard_level system
$short integers
      BLOCK DATA GETMSGI
*
*  This BLOCK DATA area initializes the COMMON fields for GETMSG,
*  thus simulating a (COBOL) DYNAMIC subprogram.  This way, the
*  subroutine "knows" whether or not it needs to open the message
*  catalog file.  As an aside, the name of the message catalog is
*  given here, as is the maximum length of the retrieved message
*  that GETMSG can handle.
*
      COMMON /gmcat/  ifile, imaxlen, ccatname
      CHARACTER*28  ccatname
      INTEGER       ifile, imaxlen
      DATA ifile /0/, imaxlen /80/, ccatname /'CATFILE.PUB.PROD '/
      END


      SUBROUTINE GETMSG (iset, imsg, cparm1, cparm2, cparm3, cmsg)
      CHARACTER*08  cparm1, cparm2, cparm3
      CHARACTER*80  cmsg
      INTEGER       iset, imsg
*
*  This subroutine is responsible for retrieving the desired message.
*  It opens the message catalog (if needed), delimits the character
*  strings used in parameter substitution (max. 3), and places the
*  returned message in a buffer.  If the desired message is not
*  found, then the message returned so indicates.
*
      COMMON /gmcat/  ifile, imaxlen, ccatname
      CHARACTER*28  ccatname
      INTEGER       ifile, imaxlen
*
      CHARACTER*10  cparm1 (3)
      INTEGER       ierror, imsglen

      SYSTEM INTRINSIC  FOPEN, GENMESSAGE
*
*     ---------------------------------------------------
*     Open message catalog if not open already.
*
      IF (ifile .EQ. 0)  THEN
          ifile = FOPEN (ccatname, o'5', o'420')
          IF (ccode())  5, 9, 5
5             cmsg = '** Message catalog not open.  Call M.I.S.'
              GO TO 99
9         continue
          ENDIF
*
*     ---------------------------------------------------
*     Init parms (ASCII NULL (CHAR(0)) at end of each string).
*     For simplicity, end-of-string is assumed to be the first blank.
*
      cparm1 (1) (1:8) = cparm1
      cparm1 (2) (1:8) = cparm2
      cparm1 (3) (1:8) = cparm3
```

```
      DO  i = 1, 3
          cparml (i) (9:10) = '  '
          iblank  =  INDEX(cparml(i),' ')
          cparml (i) (iblank: iblank) = CHAR(0)
          ENDDO
*
*     ------------------------------------------------
*     Actual message retrieval; 0 indicates all parms are STRINGS
*     Note:  FORTRAN '77 gives a FATAL compile error if we include
*            the 'cparml' variables the same way as in FORTRAN/3000.
*            BADDRESS is a compiler-supplied function.
*
      imsglen = GENMESSAGE (ifile, iset, imsg, cmsg, imaxlen,
     +                      0, BADDRESS(cparml(1)), BADDRESS(cparml(2)),
     +                         BADDRESS(cparml(3)),,,,ierror)
      IF (ierror .NE. 0)
     +    cmsg = '** Message retrieval failed.  Call M.I.S.'
*
*     ------------------------------------------------
99    continue
      RETURN
      END
```

Image Log Files..More than just Recovery Files

Kathy Spanel
GBS Consultants, Inc.
6179 E. Otero Drive
Englewood, Colorado 80112
(303) 721-0770

Steve Utz
Colorado District Attorneys Council
6143 South Willow Drive, Suite 401
Englewood, Colorado 80111
(303) 779-0841

As most HP3000 users are aware, the TurboIMAGE Transaction Logging facility is Hewlett-Packard's solution to data base integrity in the event of a System Failure, Disc Failure, or worse, Disaster. Yet, even though it is the recommended solution and is relatively easy to implement, many users choose not to. That is, until disaster strikes. And, as it typically goes, once implemented the user experiences nothing but good fortune and never has the opportunity to utilize the Transaction Log Files for Recovery! In fact, the Transaction Log Files do nothing but use up vital tape and/or disc space. Is there not something that these Log Files can be used for other than Recovery?

With this thought in mind, one begins to review the TurboIMAGE Reference Manual and notes that the format of these files is, in fact, documented. But, upon closer examination, the terms "WRITELOG", "MPE Log Record Formats", etc. come to view and the manual is quickly set aside. Obviously, the Log Files are only available to those well versed in SPL.

A few knowledgeable people we asked concurred..it can't be done! Don't even try to access the Log Files in any language, period! It can't be done!

Such is not the case. With the implementation of COBOL'85 came the feature "Reference Modification". Reference Modification is a technique which allows one to specify byte positions within a data item. Now armed with this technique and our Record Formats, it has become feasible for even those of us who are COBOL programmers to access these files.

So, what of it? The Log Files give us access to that data which has been added, modified, and/or deleted. Now that we can access these files programmatically, we can generate audit reports. Often times the design of a Data Base is sacrificed to accommodate some reporting requirements. Manual Masters become Details with many paths such as Transaction Date. Why not programmatically access the Transaction Log Files to identify those records processed on a given date?

These are but a few ideas for accessing the Log Files. Let us now take an in-depth look at the algorithms that we can utilize to easily access any Transaction Log File and then a Case Study where we review the implementation of this very technique.

Before we get too far, let us state here and now that we are by no means experts nor do we claim to be. The following information put forth is our attempt at accessing the TurboIMAGE Log Files. The coding examples are just that..examples. If you have a different style or better technique, please use it. Also, our requirements are such that we have focused on accessing the DbPut, DbDelete, and DbUpdate transactions. Our discussion, therefore, will be with respect to these intrinsics. Hopefully, the material presented will enable you to access other transactions if your application so requires. Also, in our environment, we always use an entire Data Set List when posting transactions via DbPut. This paper makes that assumption, as well. With these items in mind, then, lets get started!

As mentioned previously, a review of the TurboIMAGE Reference Manual directs us to the TurboIMAGE Log Record Formats as well as the MPE Log Record Formats (please refer to Appendix E and Appendix F in your TurboIMAGE Reference Manual). Which do we use and where should we start? Lets start by looking at the TurboIMAGE Log Record Formats and perhaps, in particular, the format of a DbPut:

```
Word 0-8    MPE WriteLog Record
Word 9      Log Record Length
Word 10     DbPut Log Record Code ("PU")
Word 11     Data Segment Number
Word 12     Recovery Flag
Word 13     Transaction Number
Word 15     Data Set Number
Word 16     Data Set Type
Word 17     Record Number
Word 19     Mode Parameter
Word 20     Offset to Key Item Value
Word 21     Offset to Item List
Word 22     Offset to Data
Word 23     Begin of Key, Item, List, and Data Buffer
```

Right away in examining this format we are presented with the term MPE WriteLog Record and again are so very tempted to close the book and hang it up. But not so fast! Maybe we should just select a different intrinsic, say DbDelete. But, no such luck. It, too, begins with the MPE WriteLog Record. And, in fact, a quick glance indicates all the formats contain

```
Word 0-8    MPE WriteLog Record
```

Now, there is no question about it...its time to move to another article!

But, wait! Lets not be so hasty! Just what is it about the MPE WriteLog Record and what is it doing in TurboIMAGE Log Records? Its really very simple...TurboIMAGE uses the already existing MPE User Logging Facility which is the facility enabling users and subsystems to log changes. This "change" file can then later be used to recover data lost. Lets see if we can make sense of this now. When TurboIMAGE wishes to post a Log Record, an MPE WriteLog Record is in turn posted. The format of the WriteLog Record is documented in Appendix F of the TurboIMAGE Reference Manual as follows:

Word 0	Record Number
Word 2	CheckSum
Word 3	Subsystem Identifier (1st Byte)
Word 3	Log Record Code (2nd Byte - 2)
Word 4	Time
Word 6	Date
Word 7	Log Number
Word 8	User Buffer Length
Word 9	User Buffer Area

The Word 9 User Buffer Area actually corresponds to the TurboIMAGE portion of the Log Record. So, now that we know this, do we need it? Yes. In our Log File we will find records other than WriteLog Records. There will be OpenLogs, CloseLogs, ChangeLogs, etc. So, we will need to determine some means of identifying which are WriteLogs as these are the ones we are interested in. Is there any other information of value? Yes, the Date and Time may also be of interest. Good. Then, let us start a list of items which we will need to look at in detail:

1..Identify and Select WriteLog Records
2..Access the Date and Time of the WriteLog

Back to the DbPut TurboIMAGE Log Record Format. Now that we're past the MPE WriteLog Record, whats next? Well, really the next item of interest is the "Log Record Code" which in the case of a DbPut is "PU" and again a quick glance at the other intrinsics indicates an "OP" identifies a DbOpen, "UP" identifies a DbUpdate, etc. So, we have a means to identify the intrinsic call.

Proceeding to the next item of interest is the Data Set Number. Yes, "Number" not "Name". No real problem here. If we are looking for intrinsic calls for a particular Data Set we could call DbInfo Mode 201 to return the Data Set Number where we supply the Data Set Name.

Though at this moment the data item Record Number doesn't appear to be of use, we will find later a very good use for it.

Finally, the "Offset to Data". The "Offset to Data" is a Word Offset from the beginning of the TurboIMAGE portion of the Log Record. So what does this mean? Well, because

it is an offset it implies the data portion of the Log Record does not always begin in the same location. Also, since it is an offset, we should be able to massage it in such a way to enable us to use it as a parameter in Cobol's Reference Modification.

At this point, let us add a few additional items to our list:

1..Identify and Select WriteLog Records
2..Access the Date and Time of the WriteLog
3..Develop a Cobol Record Layout for the Log Records
4..Develop and Code the DbInfo Call
5..Review Reference Modification

Tackling our list of five tasks should give us the tools necessary to access the TurboIMAGE Log Records.

First, Identifying and Selecting WriteLog Records. The WriteLog Identifier is maintained in the second byte position of the Subsystem Identifier as a binary 2. One technique to identify these records then is as follows:

```
01  Log-Record.
    05  Filler             Pic X(07).
    05  Log-Id             Pic X(01).
            .
            .
            .

01  Ws-Log-Id              Pic S9(04) Comp.
01  Ws-Log-Idr             Redefines Ws-Log-Id.
    05  Ws-Log-Id-Zero     Pic X(01).
    05  Ws-Log-Id-Value    Pic X(01).
01  Ws-Log-WriteLog        Pic S9(04) Comp Value 2.
            .
            .
            .

     Move 0 to Ws-Log-Id.
     Move Log-Id to Ws-Log-Id-Value.
     If Ws-Log-Id = Ws-Log-WriteLog
*       Record Selected
```

Second, accessing the Date and Time of the WriteLog Record. The Log-Hour and Log-Minute are maintained in binary format in a single byte position. To expand them, we can do as follows:

```
 01  Log-Record.
     05  Filler              Pic X(07).
     05  Log-Id              Pic X(01).
     05  Log-Time.
         10  Log-Hour        Pic X(01).
         10  Log-Minute      Pic X(01).
         10  Filler          Pic X(02).
     05  Log-Date.
         10  Log-Date-Year
                             Pic X(01).
         10  Log-Date-Day    Pic X(01).
             .
             .
             .

 01  Ws-Log-Hour            Pic S9(04) Comp.
 01  Ws-Log-Hourr           Redefines Ws-Log-Hour.
     05  Ws-Log-Hour1       Pic X(01).
     05  Ws-Log-Hour2       Pic X(01).
 01  Ws-Log-Min             Pic S9(04) Comp.
 01  Ws-Log-Minr            Redefines Ws-Log-Min.
     05  Ws-Log-Min1        Pic X(01).
     05  Ws-Log-Min2        Pic X(01).
*
*  Ws-Time will contain the Hour and Minute of the
*  WriteLog
*
 01  Ws-Time.
     05  Ws-Time-Hour       Pic 9(02).
     05  Ws-Time-Minute     Pic 9(02).
*
*  Ws-Log-Year will contain the Year of the WriteLog
*
 01  Ws-Log-Year            Pic S9(04) Comp.
 01  Ws-Log-Yearr           Redefines Ws-Log-Year.
     05  Filler             Pic X(01).
     05  Ws-Log-Yr          Pic X(01).
*
*  Ws-Log-Days will contain the Day of the Year of the
*  WriteLog
*
 01  Ws-Log-Days            Pic S9(04) Comp.
 01  Ws-Log-Daysr           Redefines Ws-Log-Days.
     05  Filler             Pic X(01).
     05  Ws-Log-Day         Pic X(01).
 01  Ws-Days-High-Order-Bit
                            Pic S9(04) Comp.
             .
             .
             .

      Move 0 to Ws-Log-Hour.
      Move Log-Hour to Ws-Log-Hour2.
      Move Ws-Log-Hour to Ws-Time-Hour.
```

```
           Move 0 to Ws-Log-Min.
           Move Log-Minute to Ws-Log-Min2.
           Move Ws-Log-Min to Ws-Time-Minute.

       *   The Date is as returned from the Intrinsic
       *   Calendar.  The first 7 bits are the Year and
       *   the Day of the Year is the remaining 9 bits.
       *   The following code will place the Year in
       *   Ws-Log-Year and the Day of the Year in
       *   Ws-Log-Days.  At that point they can be
       *   concatenated to form a Julian Date or Converted
       *   to yymmdd Format.

           Move 0 to Ws-Log-Year.
           Move Log-Date-Year to Ws-Log-Yr.
           Divide Ws-Log-Year by 2 giving Ws-Log-Year
                  Remainder Ws-Days-High-Order-Bit.

           Move 0 to Ws-Log-Days.
           Move Log-Date-Day to Ws-Log-Day.
           If Ws-Days-High-Order-Bit = 1
               Add 256 to Ws-Log-Days.
```

Third, a Cobol Record Layout for the Log Records. The following format should suffice.

```
       01  Log-Record.
           05  Filler              Pic X(07).
           05  Log-Id              Pic X(01).
           05  Log-Time.
               10  Log-Hour        Pic X(01).
               10  Log-Minute      Pic X(01).
               10  Filler          Pic X(02).
           05  Log-Date.
               10  Log-Date-Year
                                   Pic X(01).
               10  Log-Date-Day    Pic X(01).
           05  Filler              Pic X(04).
       *
       *       End of WriteLog Record
       *
           05  Filler              Pic X(02).
           05  Log-Db-Id           Pic X(02).
           05  Filler              Pic X(08).
           05  Log-Db-Set          Pic S9(04) Comp.
           05  Filler              Pic X(02).
           05  Log-Db-Rec-No       Pic S9(09) Comp.
           05  Filler              Pic X(06).
           05  Log-Db-Offset       Pic S9(04) Comp.
           05  Filler              Pic X(210).
       *
       *       End of User Buffer Area
       *
```

Now for the DbInfo Call. As discussed previously, a call to DbInfo Mode 201 and Mode 202 provide us with the necessary information. The Data Set Length returned by

DbInfo is in Words. Since we will be requiring Bytes, the Length will have to be multiplied by 2. These calls are coded below.

```
01  Ws-Data-Set-01          Pic S9(04) Comp.
01  Ws-Data-Set-01-Info.
    05  Filler              Pic X(18).
    05  Ws-Data-Set-01-Len
                            Pic S9(04) Comp.
    05  Filler              Pic X(14).
        .
        .
        .
    Call "DbInfo" using Base
                        Data-Set-Name-01
                        Mode201
                        Stat
                        Ws-Data-Set-01.
    If C-Word <> 0
        .
        .
        .
    Call "DbInfo" using Base
                        Data-Set-Name
                        Mode202
                        Stat
                        Ws-Data-Set-01-Info.
    If C-Word <> 0
        .
        .
        .
    Multiply 2 by Ws-Data-Set-01-Len.
```

And, finally, a review of Reference Modification. For a thorough discussion, please refer to the CobolII Reference Manual. Basically, Reference Modification is a technique whereby we can specify a starting character position to move from/to and the number of characters. This allows us to use the Byte Offset provided in the Log Record since the starting character position varies. The format of such a Move is

```
Move Data-Item-From(Start:Length) to
     Data-Item-To(Start:).
```

In our case, a specific example may be

```
Move Log-Record(Log-Offset:Length) to
     Data-Set-Buffer(Buffer-Offset:).
```

If, for example, Log-Offset has a value of 21, Length has a value of 40, and Buffer-Offset has a value of 1, then the characters in positions 21 thru 60 of Log-Record would be moved to the Data-Set-Buffer positions 1 thru 40.

With these bits and pieces of code and knowledge, the framework is in place. Now, lets looks at a construction of a record that was posted via DbPut. We will assume that you have previously defined the Data Set Buffer in Working-Storage for each Data Set.

There are four steps in the construction of the record.

1. Determine the number of characters to move from the Log Record to the Data Set Buffer.

   The number of characters to move can be determined based on the Data Set Number and Data Set Lengths previously retrieved via DbInfo. A simple series of If Statements can determine this number.

2. Determine the Offset from the beginning of the Log Record to start moving characters.

   The Offset requires a little bit of computation. Remember, the Offset supplied is from the beginning of the User Buffer Area, not from the beginning of the Log Record. Also, the Offset is in Words, not Bytes.

3. Move the Data to a Buffer Area.

   We found it easier to move the data first to a generic Buffer.

4. Move the Data from the Buffer Area to the appropriate Data Set Buffer.

   Based on the Data Set Number, again a series of If Statements can determine the appropriate Data Set Buffer. Once the data is in the Data Set Buffer, it can be easily accessed with the standard Data Item Names.

Lets see now what this code looks like.

```
01  Ws-Entry-Len          Pic S9(04) Comp.
01  Ws-Buffer             Pic X(512).
01  Ws-Log-Offset         Pic S9(04) Comp.
01  Ws-Buf-Offset         Pic S9(04) Comp.
01  Ws-Tot-To-Move        Pic S9(04) Comp.
    .
    .
    .
```

```
*
*   Determine the number of characters to move.  This
*   number will be placed in Ws-Entry-Len.
*
      If Log-Db-Set = Ws-Data-Set-01
          Move Ws-Data-Set-01-Len to Ws-Entry-Len
        else
          If Log-Db-Set = Ws-Data-Set-02
              Move Ws-Data-Set-02-Len to Ws-Entry-Len
            else
              .
              .
              .
*
*   Determine the offset from the beginning of the Log
*   Record.
*
      Add 9 to Log-Db-Offset giving Ws-Log-Offset.
      Multiply 2 by Ws-Log-Offset.
      Add 1 to Ws-Log-Offset.
      Move 1 to Ws-Buf-Offset.
*
*   Note:  256 is the size of our Log Records.
*
      Compute Ws-Tot-To-Move = 256 - Ws-Log-Offset + 1.
      If Ws-Tot-To-Move > Ws-Entry-Len
          Move Ws-Entry-Len to Ws-Tot-To-Move.
*
*   Move the Data to a Buffer area.
*
      Move Spaces to Ws-Buffer.
      Move Log-Record(Ws-Log-Offset:Ws-Tot-To-Move) to
           Ws-Buffer(Ws-Buf-Offset:).
*
*   Move the data from the Buffer to the appropriate
*   Data Set Buffer.
*
    If Log-Db-Set = Ws-Data-Set-01
        Move Ws-Buffer to Ws-Data-Set-01-Buffer
      else
        If Log-Db-Set = Ws-Data-Set-02
            Move Ws-Buffer to Ws-Data-Set-02-Buffer
          else
            .
            .
            .
```

Well, so far so good. But, as always, there is one small catch. What happens if your Data Set Length is greater than the size of the User Buffer Area? Is that Data available or is it lost? As it turns out, the Data is available and can be found in WriteLog Continuation Records. Sounds nasty! But, we've come this far, so lets take a look.

A WriteLog Continuation Record looks like

```
    Word 0-8    MPE WriteLog Record
    Word 9      User Buffer Area
```

and is identified with a Log-Record-Code of 7 as opposed to 2 as in the case of the WriteLog Records. This means we need to get the data off the initial WriteLog Record and then append to it the data found in the one or more Continuation Records starting in Word Position 9. We will need a few additional data items to accomplish this and some minor modifications to handle these Records.

```
    01  Ws-Entry-Len          Pic S9(04) Comp.
    01  Ws-Buffer             Pic X(512).
    01  Ws-Log-Offset         Pic S9(04) Comp.
    01  Ws-Buf-Offset         Pic S9(04) Comp.
    01  Ws-Tot-To-Move        Pic S9(04) Comp.
    01  Ws-Expected-Move      Pic S9(04) Comp.
    01  Ws-Tot-Moved          Pic S9(04) Comp.
        .
        .
        .
*
*  Determine the number of characters to move.  This
*  number will be placed in Ws-Entry-Len.
*
     If Log-Db-Set = Ws-Data-Set-01
         Move Ws-Data-Set-01-Len to Ws-Entry-Len
       else
         If Log-Db-Set = Ws-Data-Set-02
             Move Ws-Data-Set-02-Len to Ws-Entry-Len
           else
             .
             .
             .
*
*  Determine the offset from the beginning of the Log
*  Record.
*
     Add 9 to Log-Db-Offset giving Ws-Log-Offset.
     Multiply 2 by Ws-Log-Offset.
     Add 1 to Ws-Log-Offset.
     Move 1 to Ws-Buf-Offset.
     Move 0 to Ws-Tot-Moved.
*
* Note:  256 is the size of our Log Records.
*
     Compute Ws-Tot-To-Move = 256 - Ws-Log-Offset + 1.
*
     Add Ws-Tot-To-Move to Ws-Tot-Moved giving
         Ws-Expected-Move.
     If Ws-Expected-Move > Ws-Entry-Len
         Compute Ws-Tot-To-Move = Ws-Entry-Len -
                                  Ws-Tot-Moved.
```

```
*
*  Move the Data to a Buffer area.
*
     Move Spaces to Ws-Buffer.
     Move Log-Record(Ws-Log-Offset:Ws-Tot-To-Move) to
          Ws-Buffer(Ws-Buf-Offset:).
*
     Add Ws-Tot-To-Move to Ws-Tot-Moved.
     Perform A010-Finish-Record thru A019-Exit
         until Ws-Tot-Moved = Ws-Entry-Len.
*
*  Move the data from the Buffer to the appropriate
*  Data Set Buffer.
*
   If Log-Db-Set = Ws-Data-Set-01
       Move Ws-Buffer to Ws-Data-Set-01-Buffer
     else
       If Log-Db-Set = Ws-Data-Set-02
           Move Ws-Buffer to Ws-Data-Set-02-Buffer
         else
           .
           .
           .

 A010-Finish-Record.
*
     Read Log-File
         at end Move "Y" to Ws-Eof.
     Move 0 to Ws-Log-Id.
     Move Log-Id to Ws-Log-Id-Value.
     If Ws-Eof = "Y" or
        Ws-Log-Id <> 7
         Display "Continuation Record Expected and Not
-                "Found!"
         Stop Run.
*
*  Determine the offset from the beginning of the Log
*  Record.
*
     Move 9 to Ws-Log-Offset.
     Multiply 2 by Ws-Log-Offset.
     Add 1 to Ws-Log-Offset.
*
*  Determine the offset in the Buffer.
*
     Compute Ws-Buf-Offset = Ws-Buf-Offset +
                             Ws-Tot-To-Move.
```

```
*
*  Determine the number of characters to move.
*
     Compute Ws-Tot-To-Move = 256 - Ws-Log-Offset + 1.
     Add Ws-Tot-To-Move to Ws-Tot-Moved giving
         Ws-Expected-Move.
*
     If Ws-Expected-Move > Ws-Entry-Len
         Compute Ws-Tot-To-Move = Ws-Entry-Len -
                                  Ws-Tot-Moved.
*
*  Move the data to the Buffer Area appending it to the
*  Data previously moved.
*
     Move Log-Record(Ws-Log-Offset:Ws-Tot-Record) to
          Ws-Buffer(Ws-Buf-Offset:).
     Add Ws-Tot-To-Move to Ws-Tot-Moved.
*
 A019-Exit.
     Exit.
```

At this point in time, we now have in place the means and algorithms to construct records posted via DbPut. We thought we had it all. Until a new requirement presented itself. We needed to also detect DbUpdates. On the surface it did not appear to be too difficult a task. After all, we just got done doing the impossible, right? We could conquer anything at this point!

So, once again we went back to our now familiar Appendix E in the TurboIMAGE Reference Manual to check out the format of the DbUpdate. And we were greeted with some familiar terms that had previously caused us great pain but now were just everyday words in passing! Again, we saw the MPE WriteLog Record, DbUpdate Log Record Code of "UP", Data Set Number, and Record Number. No trouble so far. And, lo and behold, there was an Offset to New Data and an Offset to Old Data. Great! We're in. But, unfortunately, we're not.

Upon further investigation it was found that the DbUpdate New Data and Old Data areas only contained the values for those Data Items whose value did in fact change. In other words, the entire record would not be found in the Log Record! In other words, if the Birth Date changed, thats all that we would have..old and new Birth Date. No Key Value, no Name, no nothing!

It appeared as though we were stumped. But, not willing to accept the phrase "it can't be done" we started to get down to basics. In our case, we were only interested in DbUpdates to Detail Data Sets. This greatly simplified our task..once a record is posted to a Detail Data Set it does not move, unlike Master Data Sets whose records can move due to Migrating Secondaries. Therefore, once a record is posted to a Detail Data Set, it will remain there unless Deleted in which case its record location

will at some point in time be used by another new record. With this in mind then, a solution started to form. While processing the DbPut transactions, could we not also select DbUpdates to the desired Data Sets. Via some means, we needed to track what Record Numbers had been Updated. Also, we needed to select DbDeletes to detect if any of the previously Updated Records had been Deleted. Then, upon completion of reading the Log File, we could go back and process this Update File utilizing the Set Number and Record Number retrieved from the Log Record and issuing DbGets Mode 4 .. Read by Relative Record Number. Admittedly, it sounded a little hazy but perhaps it had some potential.

We thought it was worthy of some further investigation. And, so we developed a plan:

1..Create a Ksam File with the following Record Layout:

```
01  Update-Records.
    05  Filler              Pic X(02).
    05  Update-Key.
        10  Update-Set-No
                            Pic S9(04) Comp.
        10  Update-Record-No
                            Pic S9(09) Comp.
```

2..While processing the Log File, if a DbUpdate is detected, do the following tasks.

..Check to see if the Set Number/Record Number exists in the Update Ksam File.
..If it does exist, then this is merely another Update for a record that was previously Updated and we can bypass it.
..If it does not exist, then add it to the Update Ksam File.

3..While processing the Log File, if a DbDelete is detected, do the following tasks.

..Check to see if the Set Number/Record Number exists in the Update Ksam File.
..If it does exist, then Delete this record from the Update Ksam File as this indicates that a previously Updated Record has since been deleted.
..If it does not exist, then this is merely a Delete for a record that had not been previously Updated in which case we can bypass it.

4..Upon completion of processing the Log File, serially read the Update Ksam File. For each record read, do the following tasks.

..Issue a DbGet Mode 4 using the Update-Set-No and Update-Record-No as retrieved from the Update Ksam File as parameters to the DbGet Mode 4.
..Process Record.

The code to accomplish the above tasks is not included here as it merely involves a number of Ksam Intrinsic Calls and is really independent of the accessing of the Log Files.

We proceeded to implement the above mentioned tasks and surprisingly enough, it worked!

Now granted, you may be saying to yourself..gee, it sure seems like an awful lot of work to go through when we could just add another path here and there or post to a transaction file all changes, etc. And, we did consider such options. But, in our environment, system resources were scarce and anything at all that could be done to alleviate any unnecessary processing would be looked at. And, there was also that sense of mystery and intrigue in being able to make some sense of these Log Files.

So, its a nice technique that works. Does it really have any real life application? We currently have this technique implemented in two major applications. Lets take a quick look at one of these applications.

Case Study

Some background of our application..In our environment it is necessary to maintain massive amounts of current and historical data. Data Sets consisting of 500,000+ Entries are not uncommon. Our first application involved a set of 20+ reports which were strictly driven by New and Updated Entries. And so it was considered to add some paths and/or Data Items with a Transaction Date and Code. But the volume just seemed so unwieldy. A transaction file was considered, but we already were incurring the overhead of Image Transaction Logging which created one Transaction File why incur additional overhead by adding yet another Transaction File? So, the above discussed technique was implemented. I won't go any further in to this application as the dependencies and intricacies of the reporting requirements are quite unique and specific to our environment. Suffice it to say, where previously it could take a full evening shift to complete the reports nightly, the reports now take under 1 hour of processing time. There actually is time available for other processing!

An area which perhaps has more widespread application is that of the maintenance of a Soundex File. We made some inquiries amongst the HP3000 User Community to see how others had implemented Soundex. We found the Soundex was maintained in a Ksam File and typically rather than updating and maintaining the Soundex on-line, the Soundex was recreated on a periodic basis..generally once a week.

Unfortunately, this would not satisfy our needs for two reasons.

1..Our users required a more current Soundex. On-Line maintenance was not required but certainly a one-day lag is all that could be tolerated.

2..Our Soundex File contains close to 1 million records. This requires in excess of 48 hours to recreate. Obviously, this could not be accomplished in an evening.

And so, once again we turned to our technique of accessing the TurboIMAGE Transaction Files.

We found that we could basically clone the Log File Extract Program developed for our Reporting System with the following modifications:

1..Extract only DbPuts, DbUpdates, and DbDeletes for the Data Sets "soundexed".

2..For every DbPut, Post a record to the Soundex.

3..For every DbUpdate, Update a record in the Soundex.

4..For every DbDelete, Delete a record from the Soundex.

The changes were made and implemented. Our Soundex file is now maintained nightly in approximately 15 minutes. We are very pleased.

A few closing remarks...If nothing else, we hope you may have learned a few things in reviewing this article. When the next person tells you it can't be done, think about it...maybe it can! And, finally, a special thanks to Mr. Bob Bussey of Colorado District Attorneys Council whose inspiration and confidence made it all possible.

MANAGING THE RELATIONAL ENVIRONMENT

by Peter Ney
DCE Information Management Consultancy
Prinsengracht 747
1017 JX Amsterdam
The Netherlands

Paper Overview

Many Hewlett-Packard users are suddenly finding themselves faced with an important strategic decision, the move to a relational database environment. Relational databases represent a considerable departure from the established approach to information management for most traditional HP3000 users.

In this presentation I will attempt to answer the two most important questions concerning the migration to the relational environment within the HP3000 world: **WHY?** and **HOW?**.

WHY?

It must be understood that apart from all the technical advantages of RDB over other database or file structures, the relational environment brings several **strategic** benefits. It should therefore not be evaluated merely as a technical solution, but as an integral part of the overall business strategy.

The relational database approach actively supports and promotes the use of information as a corporate strategic tool, it facilitates a much more responsive and efficient business management environment, and can therefore be instrumental in gaining or maintaining a competitive business advantage.

I will discuss the strategic difference between the relational approach and IMAGE, as well as some of the related technical benefits. Some of the topics of consideration when attempting to answer the **Why?** question I will examine are:

The Three Level Schema approach of RDB;
Semantic database manipulation;
Expected productivity increase;
The introduction of End User Computing;
Strategic use of information;
Integrated and mixed environments;
Distributed databases.

I confess that I am an enthusiastic salesman of the relational database approach. All of the companies DCE has helped to advance into the relational

environment are receiving considerable payback for this investment. I hope to convince many others that they should share Hewlett-Packard's commitment to relational technology as a strategic direction, and hope that this neccessarily brief **Why?** section of the presentation will at least prompt them to consider some of the many benefits thereof.

HOW?

There are no shortcuts to establishing an efficient relational database environment that fully supports the business. RDB is not just a replacement for IMAGE, and requires a specific underlying approach and environment framework to be successful. Based on my experience of implementing relational technology, I have published a **Twelve Steps** guide to the process, which I will use as a basis for the **How?** part of this presentation.

These **Twelve Steps** are not intended to form a rigid implementation plan, but act more as a list of neccessary areas to be addressed as part of the overall move to the relational approach.

NEY'S TWELVE STEPS
TO IMPLEMENTING A RELATIONAL DATABASE ENVIRONMENT

1. **SET UP A HIGH LEVEL MANAGEMENT BODY** to plan and coordinate the corporate IS strategy; identify goals, requirements and critical success factors of the company and its information management;

2. **CONDUCT A HIGH LEVEL SISP[1] STUDY** to review the current information management environment; cost-justify the move to relational and establish both the short-term and long-term IS strategy;

3. **ADOPT A FORMAL SOFTWARE DEVELOPMENT CYCLE**, back it up with effective project control and QA discipline; define procedures and documentation standards company-wide; ensure your SDC fully and naturally supports the relational database approach;

4. **ADOPT A FORMAL METHODOLOGY**, effective for the relational environment, for all stages of the SDC; consider supporting it with CASE tools and complementary techniques such as prototyping;

5. **ESTABLISH A CORPORATE DATA DICTIONARY** to support and manage the relational environment, document and control corporate metadata and resource information, and enforce data integrity and standards;

6. **REVERSE ENGINEER EXISTING KNOWLEDGE** to protect current investment; integrate current databases and applications at the business and logical levels (not just physical);

---

[1] Strategic Information Systems Planning (SISP is a DCE Service Mark)

7. ADOPT THE DATABASE APPROACH; don't treat the RDBMS as the database, you need to foster the underlying database philosophy company wide; encourage sharing and manipulation of information and a more sophisticated use thereof;

8. CHOOSE CORRECT RDBMS PRODUCT(S) that support current and future IS requirements in an efficient, natural and flexible way; ensure that the physical implementation can grow and flex with the business;

9. IMPLEMENT SUPPORTING ENVIRONMENT; tools, languages, utilities, etc., that work efficiently with your chosen RDBMS and support minor IS objectives (end user computing, decision support, CASE, etc); the software environment should naturally support the SDC with its underlying methodology, you should never have to change your logical approach for the benefit of a product;

10. ENCOURAGE EXPERTISE AND COMMITMENT from management, users and the IS departmenmt; share goals and successes with users; ensure everyone is aware of the strategic direction;

11. PLAN AND SENSIBLY EXECUTE MOVE TO RELATIONAL; this should follow the overall IS strategy; protect current investment, plan and manage efficient coexistance with current environment, but be strongly committed to long-term advantages of relational; follow visible progress;

12. REVIEW YOUR STRATEGY AND DIRECTION OFTEN; don't shy from objective criticism of your environment and plans; be prepared to change and capitalise on new opportunities; cost justify every step, never follow new develoments for the sake of it.

PETER NEY is a consultant with DCE Information Management Consultancy based in Amsterdam, The Netherlands. He specialises in data management in the Hewlett-Packard environment, including SISP (Strategic Information Systems Planning), IPSE (Integrated Project Support Environment) and relational databases. He is a frequent lecturer and user group speaker.

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

MEETING THE NEEDS OF THE CASUAL USER

Cheryl Van Kirk and John Van Kirk
Van Kirk Consulting
P.O. Box 1204
Port Alberni, British Columbia V9Y 7M1
(604) 724-2080

INTRODUCTION

This paper will discuss some techniques that we found successful in designing and writing an HP3000 computer application for 'casual' users. To help you understand what is meant by a casual user, we will describe the application briefly.

The application is a managed maintenance system developed for our client - Alberni Pulp and Paper Division of MacMillan Bloedel, a large forest products company. Our client wished to have an application that tradesmen and their supervisors could use. Tradesmen (welders, carpenters, pipefitters, etc.) normally have few clerical skills. They usually do not know how to type and there was no plan to train them in typing skills. Their use of the computer was to be occasional and a relatively minor portion of their job duties.

We followed our client's normal programming environment which uses COBOL, TurboImage and VPLUS to build the application. All terminals are in block mode while running the application. There are now about 70 terminals devoted to the maintenance system. Approximately 15 of them are located in shops and used mainly by 300 tradesmen.

When we started this project it was apparent that our users' needs and concerns would differ from the needs of users of office applications. We responded to the special circumstances by building into the system features that make the application easier to use.

This paper will discuss some of the techniques that were effective in creating a successful computer application for casual users. The techniques vary from a simple hardware modification to broad system design principles. However, as a general guideline, making a system easier to use means that the system designers and programmers have to work harder.

PROTECT THE USER FROM HIS MISTAKES

In this category are techniques that make it less likely that the user will do something that makes it difficult for

him to see what to do next. Of particular significance are procedures that keep him from 'locking up' his block mode terminal or taking the terminal out of block mode altogether.

Several of the techniques involve writing escape sequences to the terminal to dynamically change the terminal's configuration. Writing an escape sequence to a terminal can often be done by simply including the escape sequence as a 'COMMENT' line in the UDC that brings up the application. If that is inconvenient, it is also fairly simple to issue an FWRITE intrinsic call using the field 'filen' in the VPLUS communication area as the file number that corresponds to the terminal. An escape character will be represented as <esc> in the rest of this paper.

The techniques are:

1. Disable the STOP key on the terminal. Pressing the STOP key causes the keyboard to 'freeze', which is totally undesirable in a VPLUS application. We disabled the STOP key on a 700 series terminal by popping off the STOP key (a small screwdriver works well) and laying two rubber O-rings in the key well. The correct O-ring size is 7/16" inside diameter by 5/8" outside diameter by 3/32" width.

Then we replaced the STOP key. The keyboard looks exactly the same as before, but the STOP key does nothing. This keyboard 'field enhancement' is completely reversible.

2. Issue an escape sequence which forces CAPS LOCK on if the application does not accept lower case input. This means that the user cannot type lower case input even if he presses the shift key. The following escape sequence will force CAPS LOCK on:

<esc>k1L

3. Disable the BREAK key by setting NOBREAK in the application's UDC. Doing this prevents the users from inadvertently and ungracefully exiting the application.

4. Lock the terminal configuration and other terminal settings as much as possible. We discovered that a terminal that was not logged on was often experimented with by curious trademen to see if they could understand how it worked. Problems were minimized if the terminal was configured so that the tradesmen could do minimal damage as they investigated it.

The following escape sequence is very useful:

<esc>&q1L

This escape sequence locks all configuration menus in addition to the modes: MODIFY ALL, BLOCK, REMOTE, and AUTO LINEFEED. These changes are made in nonvolatile memory and will remain in effect even if the terminal is turned off. Replacing the '1' by a '0' will unlock the terminal configuration and the above modes.

Two other escape sequences are:

<esc>&jB

<esc>&jS

The first escape sequence ensures that the terminal displays the user function key labels. The second escape sequence disables the User/System key. Taken together these escape sequences prevent the user from setting the DISPLAY FUNCTIONS mode on and/or setting the EXT DEV key off which makes a slave printer inoperable. Unfortunately, these escape sequences only affect the volatile memory of the terminal and are reset by a hard reset or by turning the terminal off. They still make a good first line of defense.

5. We suggested that the users not power off their terminals at any time. Powering off a terminal while in a VPLUS application leaves the application in a state that is relatively difficult to recover from. Powering off the terminal was very prevalent initially. Tradesmen are very good at fixing equipment. If they felt that the terminal was not working properly (for example, they received an application error message indicating an invalid response), they tried to fix the terminal. The first step was always to power off the terminal. Some of the shops glued a piece of plastic over the ON/OFF button. Most shops simply taped over the ON/OFF button and wrote 'DO NOT TOUCH' on it.

6. After a couple of users logged off the application when they did not intend to, we made it impossible for users to logoff unintentionally. Initially we had programmed the application so that pressing the f8 function key anywhere within the application always took you back one screen and eventually logged the user off. The application was changed so the final f8 brought the user to a logoff confirmation screen. On this screen he can type BYE and then press f8 to confirm a logoff. Pressing ENTER brings the user to a familiar menu. Pressing f8 without typing BYE, causes an error message.

7. Casual users should not use a microcomputer in terminal emulation mode. This environment is more difficult to control (how do you prevent the user from booting his microcomputer in the middle of your application?) and the CTRL-key and ALT-key combinations are more difficult for the user to learn. On the other hand, some supervisors were already comfortable with a microcomputer and have been very successful using terminal emulation.

AUTOMATICALLY LOG TERMINALS ON AND OFF

For someone who cannot type and is not comfortable with computers, typing the 'HELLO' command to log on can be a daunting and frustrating exercise. A considerable amount of training time is spent initially teaching the user this first step. Often he feels that if it is this difficult just to begin, it can only get worse. Fortunately, MPE now includes the 'STARTSESS' command which allows terminals to be logged on from another session. This MPE command can be issued by any programmer or operator with programmatic session (PS) capability. The format of the command is:

```
STARTSESS ldev [sessionname,]user[/userpass.acct[/acctpass]
          [,groupname[/grouppass]];NOWAIT
```

When it is issued, this command acts just as if the corresponding HELLO command had been typed on the 'ldev' (logical device number) specified. The NOWAIT option is necessary to start the session executing immediately. Occasionally, there may be a pending I/O request for the terminal which should be processed or removed before the STARTSESS command. Preceding the STARTESS command with an ABORTIO command for the same 'ldev' is therefore a good idea.

For our application, a series of STARTSESS commands for specified device numbers that correspond to terminals in the shop areas are done as part of the process of bringing up the computer after daily backups. The sessions are brought up in an inquiry only mode. Clearly, the use of this command has be to consistent with the security restrictions that apply to the application.

Logging off the computer can also be done automatically by a BYE command in the application's logon UDC. This prevents the user from accidentally getting to a colon prompt. Instead, he is logged off when the application finishes. All commands in the UDC that might fail should be preceded by a CONTINUE command. Then the UDC will continue to execute even if the application program aborts. If nested UDC's are used, a CONTINUE command should also precede the invocation of each UDC.

Since the shop terminals run unattended, we also needed a way of logging them off with no user intervention. This means the application program must obtain control of the unattended terminal. Calling VREADFIELDS with a timed read returns control to the application program after a specified time even if no user has pressed the ENTER key. In the maintenance application, we set the time interval to five minutes. The application program then checks to see if the session should be logged off. (In the maintenance application we check to see if the session limit has been set to zero. If it has, the program does a normal termination and displays on the terminal a message that system backups are in progress. The session limit can be determined by calling the JOBINFO intrinsic.)

USE 'LIST' SCREENS

Whenever possible help the user by presenting him with a list of choices to choose from. The most obvious example is the familiar menu screen which has a list of functions to choose from and the user types in the number or letter of his choice. However, it is easier for the user if he does not have to type in the number or letter of his choice, but can TAB down to a one character box in front of each menu selection. He then types any character in the box and presses ENTER. Typing skills needed are minimized since the user does not have to find any specific letter or character on the keyboard.

The list approach should be used throughout a system. Consider a problem from a stock inventory system where each item in inventory has a unique stock code. Instead of requiring the user to know and type in a stock code, lead them through a series of 'list' screens each of which narrows in on the stock code that the user desires. The sequence could be:

A. Show a list of catalog codes to choose from such as Pumps, Electrical, General Spares.

B. Then show a list of section codes within the chosen catalog.

C. Then list the stock codes in this section. This screen shows the most important information about each stock code such as its primary description and quantity on hand.

D. Now allow the user to see more details about a specific stock code by choosing a stock code from the list and then showing up a detail screen for that stock code.

List screens are intuitive to casual users and have the added benefit of minimizing training. It is important to keep the lists relatively short. If the number of choices fills more than two or three screens, then the selection process becomes tedious for the user.

For the list screens to be fully effective, there are a couple of programming techniques that are mandatory. These are:

1. Restrict the user to only valid choices on a partially filled screen. For example, if a page can hold 15 stock codes and there are only ten in a particular section, then the user will see a partially filled screen. It is important that he not be able to tab to the selection box in front of a non-existent stock code. The VCHANGEFIELD intrinsic can be used to turn off the display enhancements and make the field a display only field. This keeps the user from making a nonsensical choice.

2. To be able to bring up a list quickly, it is essential to have fast access paths to the data and frequently the data must also appear in a particular order, such as in date order. IMAGE detail chains (perhaps sorted if the chain length is not too great) work well. However, considerable flexibility can be added by the use of an indexing IMAGE add-on product such as IMSAM and/or OMNIDEX by DISC. We used IMSAM and rapidly grew to rely on the indexed partial key retrieval features that it provided. Some of these features can be obtained using KSAM files, but at considerably more programming effort.

The list approach should be used in conjunction with a forms caching terminal and reasonably fast data communication speeds. Without these facilities the list screens will be slow to appear and user frustration may be unacceptably high.

PROVIDE HELP LISTS

Particularly for the tradesmen doing data entry and occasionally for specific query screens, we provided 'help' lists. A 'help' list for a specific field is merely another screen which can be brought up at the user's discretion which shows him the allowed values for this field. In our application, a help list for a specific field is invoked by typing '?' in the first character of the field and pressing ENTER. Instead of processing the screen, the application traps the '?' and invokes a help facility that brings up a screen with a list of choices for the field in question. As

usual, the user tabs to his choice, selects it and presses ENTER. The application then returns to the original screen with the field in question filled in with the user's choice and the cursor positioned at the field following the 'helped' field.

This feature required a special routine which pre-processed screens and intercepted help requests. In addition to making the system easier to user, help lists reduced typing.

USE DYNAMIC FUNCTION KEY LABELS

Function key labels such as NEXT PAGE and RETURN TO MENU are very helpful in relieving the need for typing and are also intuitive for the user to follow. The labels can be added as global to the application or specific to a particular form through the FORMSPEC utility program. However, even more power is gained by having the function key labels under dynamic programmatic control.

As a simple example, the function keys can be used to give the user a choice of screens that he would likely want to transfer to from the one he is currently on. However, due to security restrictions, he may not be allowed to execute all of the choices suggested in the function key labels. Dynamically removing any function key label that represents a security breach is a clean way to implement this security restriction.

The VPLUS intrinsic for programmatically changing function key labels is VSETKEYLABELS.

PASS KEY DATA BETWEEN SCREENS

A useful technique for minimizing typing and making the system easy to use is to develop a method for passing data between related screens. As an example, let us return to our stock inventory system.

Assume that with each stock item is stored two other 'key' pieces of information:

1. The vendor code for the vendor that we purchase this stock item from.

2. The last purchase order number that this stock item was bought under.

Now assume that a specific stock code's information has been brought up on a screen. This screen has two function key labels of interest to us now:

f3 - VENDOR QUERY

f4 - PO QUERY

If we press f3 (VENDOR QUERY) the system goes to the vendor query screen AND brings up the vendor information for the vendor code associated with the stock item we started from. The odds are good that we will now have the vendor information that we are looking for. A similar process is followed if PO QUERY is chosen.

USE A 'SUSPEND' ENVIRONMENT

Most of us are familiar with a menu environment where you choose something from a menu which leads you to another menu and so forth. Often you can also retrace your steps and work your way back up the menu tree until you reach the first or main menu.

We used a similar technique as users progressed from one screen to another. We called it "suspending" and "returning from a suspend". Looking at the stock browse example again, the steps are:

1. Select STOCK BROWSE from a function menu. Then suspend to:

2. A list of stock catalogs. Then select a catalogue and suspend to:

3. A list of sections within the selected catalog. Then select a section and suspend to:

4. A list of stock codes within the selected section. Then select a stock code and suspend to:

5. A stock detail screen which shows stock information including a vendor. Then press the VENDOR QUERY function key and suspend to:

6. A vendor query screen which has brought up the vendor specified on the previous stock detail screen.

Within the application, we chose f8 as the 'return to previous screen' key. Using dynamic function key labels we were able to always show the user what screen he would return to if he pressed f8. Assume at this point in our example, the user presses f8. He 'returns from the suspend'

to the stock detail screen which still shows the stock information that he just had on the screen.

Pressing the f8 key repeatedly steps the user back through the screens one at a time. The user can start 'suspending' again by selecting a different choice from any of the list screens.

Casual users find this technique to be very intuitive. They seem to expect the computer to behave in this way.

Unfortunately, this technique presents a problem to the programmer. Clearly, he is going to have to save at least a portion of working storage (for a COBOL program) for each suspend. Using normal programming techniques the programmer will soon get a data stack overflow error. This problem can be overcome through the use of extra data segments as discussed in detail in our other paper:

"Writing Your On-line System as One Program"

GENERAL DESIGN GUIDELINES

In addition to the above reasonably detailed techniques, there are several design guidelines that should be kept in mind as you design a system for casual users. These are:

1. Place advanced (non-essential) features of the system in inobtrusive places where the casual user will not notice them. Separate menus are a good idea. Another trick is to position advanced feature input fields at the beginning of a screen and then automatically position the cursor past them. Since casual users expect to progress forward on a screen, they will automatically bypass them. Another place to hide advanced features is on an alternate set of function key labels.

2. Design your security system so that users do not see functions that they are not allowed to execute.

3. Sacrifice speed and flexibility for ease of use. As an example, paging could be done by providing a field where the user types in the page number he wants to go to next. However, function key labels for NEXT PAGE and PREV PAGE should also be provided and are probably the only paging mechanism that the casual user will understand intuitively.

4. Keep your terminology at your users' level. This is a well-known adage, but it is especially important with casual users who will not use the computer frequently enough to learn its particular language. As an example,

the message "NOT AN INTEGER" may be meaningless, while the phrase "USE A NUMBER HERE" is clear. Even words that we feel are in common usage and take for granted may be confusing. A message containing the word "MODIFY" may be inappropriate and the more common word "CHANGE" should be substituted.

5. Sacrifice exactness for simplicity. Perhaps your stock items contain two types of descriptions: a primary description and an extended description. The casual user of the stores system may not understand the distinction and all descriptions should be combined together under the heading "DESCRIPTION".

6. Keep the look and feel of the entire system consistent. While this guideline is good for any computer system, it is perhaps most important for a system serving casual users.

OUTSTANDING PROBLEMS

Unfortunately, there still remain several ways that the casual user may become confused by a VPLUS computer application that we are unable to find a work-around to. The worst ones are:

1. Confusion exists between when to use the RETURN key and when to use the ENTER key for users that have been taught how to do a HELLO command.

2. Users who have been trained on the 'HELLO' command for logging on assume that the cursor movement keys can be used to correct a mistake. They cannot understand why there is a difference in the backspace and back arrow keys now when there is no difference while in the application.

3. Computer crashes (thankfully rare) leave the terminals in block mode and with no obvious sign that there is a problem. Even if the user becomes suspicious that something is wrong, it requires a hard reset to get the terminal in a position to do a log-on once the computer is back up.

4. A local power fail that affects the terminal, but not the computer, will take the terminal out of block mode. Our VPLUS application had difficulties recovering from this situation. (A "REFRESH" only works if the correct procedures are done as the first key strokes to the terminal following the power failure.) A similar problem occurs if a user does a hard reset during the application.

CONCLUSION

We have discussed techniques to make a VPLUS computer application easier to use for the casual user. Many of the techniques would be useful in a normal office environment as well. However, with a casual user community, there will not be a gradual acceptance and understanding of the computer's seemingly strange ways of behaving. To be successful, you should use every reasonable method of making the application easier to use and intuitive. Whenever possible, the application should make it difficult for the user to reach a situation that he does not see a recovery path from. Only then will you meet the needs of the casual user.

Developing a System to Optimize and Control Containerized Cargo Movements

John S. Fusek
& Serena R. Fusek
Virginia International Terminals
P.O. Box 1387
Norfolk, Virginia 23501
(804) 440-7023

Virginia International Terminals(VIT), the company that operates several ocean going shipping terminals for the Virginia Port Authority, is a Hewlett-Packard success story. VIT's innovations have raised the ports of Virginia to second place in number of containers moved on the east coast. Hewlett-Packard equipment allowed VIT's small DP shop to play a large role in this achievement.

At Norfolk, Virginia, their containerized cargo port, VIT innovated a new approach to the old problem of how to load and unload ships quickly. This innovation was needed to give VIT the edge in the highly competitive east coast shipping industry.

In February, 1987, VIT installed two new high speed cranes that move containerized cargo at twice the speed of older models. These new cranes were designed by VIT to use the latest technology along with a dual hoist system that allows the cranes to move two containers at once.

The old system for tracking and moving containers around the yard was too cumbersome to utilize the new cranes at maximum efficiency.

The Norfolk yard covers approximately 700 acres, with containers parked or stacked on any bare space. In 1987, VIT moved 248,208 containers using a 15 year old computer system, the Container Control System, designed before the shipping boom. The system was heavy on paperwork. Moves were ordered with Work Assignment Cards. When a container was scheduled to

be stacked (stored until its ship came in) or moved to be loaded on to a chassis an Assignment Control Card was delivered manually by a messenger who ran the yard's gauntlet among the transtainers, cranes, and hustlers zipping up and down the rows pulling containers.

When a container on a wheeled chassis pulled by a tractor (an 18-wheeler) enters the gate, it is interchanged, then sent to temporary parking in the wheeled area where the tractor is disconnected.

The office key-entered the information on the container into the database: its number, size, weight, port of discharge (destination) and vessel of departure.

To make the most efficient use of storage space, shipping containers are removed from their chassis and placed one on top of the other in the stack. The goal here is to get the container out of temporary parking and into the stacks as quickly as possible. The stack and row should be one from which the container can be removed easily when it is scheduled to be loaded. Under the old system, a foreman assigned a container its place in the stack. Between four and five hundred containers were moved each day.

The Container Control System used the computer to track a container's location in the stacks. But all information on the container was key-entered into the system by clerks who tried, in five, twelve hour days, to keep up with work crews who move containers 24 hours a day, seven days a week. After a weekend's activity in the yard, it sometimes took until the following Wednesday to get the computer updates in line with physical inventory.

As a result, both efficiency and accuracy were lost. Some container movements were never recorded at all.

The Norfolk Terminal services over 50 shiplines which are registered all over the world. Some of these lines are Evergreen, Orient Overseas, Lykes Lines, Torm West Africa, and Yang Ming. Each shipline has it own method of operation, national and international law they must obey and a set of problems unique to that line. Until now, this difficulty has

discouraged the use of a computer system to move and track the containers for a multi-shipline terminal.

Summary of the Problem

Container locations were returning to the computer in a random manner and not all moves were being recorded. The system was cumbersome and too slow to keep up with the needs of the new cranes. There was no way to be sure that the location of the container given by the computer was anywhere near accurate. The possibility for error was enormous. Also, containers that had to be loaded right away were often found to be on the bottom of a stack or in some other difficult, inefficient location.

The new cranes did not provide the edge VIT needed in the competition among east coast terminals if they could not be fed efficiently.

In 1986, VIT's small data processing shop began to search for that edge.

The Shop

At the time, VIT's DP shop had six or less people. These people had to find a solution to mounting chaos while maintaining all existing applications.

What we had on our side was that we were a Hewlett Packard shop. The Container Control System was on a HP 3000, Model 70.

THE SOLUTION

A study by Old Dominion University, in Norfolk, VA, determined that the key to the solution was to get the information about the container location and all moves entered into the system as soon as possible. They recommended that the best time to

enter the information was at the time the work was done. Also, a system was needed that more closely controlled and tracked container movement and kept the container's location in sync with its paperwork.

It was decided that the best solution would be a device in the field that reported to the computer each time a container was moved. The first possibility explored was to use telemetry equipment on the cranes moving the containers to report to the computer, but this equipment could not note the container's number or its height. That was when it was realized there was no way to bypass a human entering the data.

To minimize error, data entry in the field would have to be limited to changing elements only, therefore, the device would have to receive a good deal of its data from the main system. On the other hand, the device would have to be able to operate independently of the main system so work could continue in the field when the main computer was down for maintenance or with problems. The device can best be described as an electronic work order card where the checker (crew chief) just has to fill in the blanks and it is automatically time-stamped and sent to the main computer.

In the environment in which it would work, the device should be rugged. Day hire labor should find it easy to learn and use. It would be most efficient if it had messaging capabilities so that workers were in touch with the office at all times.

To make such a device feasible, the main system would have to assume control of container location assignments and be able to monitor them at all times.

In addition, power to VIT's main computer, the HP3000, was unreliable. Any solution would have to take this into consideration.

All this had to interface quickly with the antique Container Control System.

SEARCHING FOR THE DEVICE

In 1987, once the parameters were laid out describing the device, a search was initiated through available literature for the piece of equipment that came closest to meeting our requirements.

At the first the search turned up two main types of equipment:

1. Dumb terminals that required constant contact with the main system to function. These included LXE and Motorola, which could hold a maximum of 200 bytes.

2. A combination of lap top PC and RF modem attached to a serial port. These were large, bulky and not very rugged, but they could operate independently of the main system if necessary.

We explored different models of each of the two types but none really filled our needs. Yet we could find nothing else, so for a while there was much discussion about which compromise we would make. Meanwhile, software design for the system was carried to the point where it depended on our choice of the equipment in the field. Here design stopped and programming was started to fulfill the design work to that point. We had to make a decision before we could finish the design.

This vacillation went on for a couple of months until, around the end of 1987, a salesman from a local (Tidewater, Virginia) computer seller came in making a cold call on VIT. He turned out to be a VAR for Telxon, which manufactures hand-held smart computer terminals. He showed us the sales literature on the Telxon 750 units, a piece of equipment so new it was still under development at the time.

The 750 had a complete keyboard along with a 16 x 16 character screen. Telxon salesmen routinely bounced demo units off the floor to emphasize their ruggedness. The unit was fully programmable and could store up to a megabyte of programs and data. It was an intelligent computer and could operate independently of the main system--in terms of computing power

it was equivalent to the original IBM PC. This was packaged with a built-in radio frequency modem in a unit the size of an extremely thick walkie-talkie.

It communicated back through a base unit that interfaced to a RS-232 serial port. It could also communicate through a hardwire link. The Telxon 750 Radio Frequency Handheld Terminal was the ideal device for our needs.

Now that we had our device, design could proceed.

SOFTWARE DESIGN

All the basic design was roughed out by March, 1987.

The design was then broken down into several sections that could be worked on independently.

1. The first part was a subsystem that described the yard area physically and allowed yard control personnel to attach loading criteria to the physical areas described.

2. The next subsystem took these criteria and interfaced with the work order and the old container system to generate location assignments.

3. The third subsystem brought this information together and packaged it to send to the handheld units in the field.

4. The fourth brought information back from the field and used it to update locations from the field and provide error correction.

5. Finally a subsystem was created for after-the-fact reporting.

1. The first subsystem was written entirely in Infocenter's Speedware, a fourth generation language. It

consists of a series of screens that allow yard control personnel to enter information on various locations around the the yard; and reports to provide cross referencing and editing. It took one programmer approximately a month to do the main parts of this work after the design was complete.

2. The second subsystem was the most difficult to develop from a design standpoint. Various methodologies were considered and examined, including using Expert Systems and Artificial Intelligence, but these were considered to be a little too unproven in a day-to-day working environment. It was finally decided to attempt to duplicate the foreman's assignment process using the Assignment Variation of the Transportation Method.

After coming up with the algorithm, the main problem was interfacing with the old Container Control System. While most of the needed data was in this system, it had to be edited and converted to a format from which assignments could be made.

The final design on this subsystem first determines the current condition of the yard, then examines the work waiting to be done, performs the necessary editing and conversions and then brings everything together to find the best match from which to generate a container location assignment.

3. Once the assignment locations are generated, they are sorted together in a manner that reflects the order in which the work should be done. Where the old system had broken up the Work Assignment Cards by individual shiplines, the new system lays out the work to reflect the way equipment will travel and attempts to order everything in the most efficient manner. This is all put together in a file and transferred down to the handheld units.

4. The fourth subsystem is mainly a file validation and master file update. When information comes back from the field, it is edited against the current container system and if the information is correct, it is used to update locations and generate billing transactions. If the information is incorrect, it is then brought to someone's attention for correction or deletion.

5. After-action reports are then generated including container and error tracking, production statistics, and billing validation. Historical information is also kept for use in planning operations.

It was decided to write most of the new work in Speedware wherever possible because of the increased productivity inherent in using a fourth generation language. The prototyping ability of Speedware meant we could speed up our usual methods of obtaining user feedback and thus get a tighter development cycle.

To interface with the old Container Control System, however, we had to fall back on COBOL and SPL.

DESIGNING THE SYSTEM

Setting Up Location Assignments

The first parts of the major design was the Location Control and Assignment Subsystems. These were heavily dependent on one another but did not need the handheld to be useful. While we were still searching for the field equipment, we decided we could print the assignment location work orders on cards and get useful feedback from the foreman before we placed the entire system in the field.

We looked at the stacking criteria used for each shipline and found the critical elements needed to arrange the stack for efficient loading. Some of these elements are:

1. Vessel voyage and number
2. Port of discharge
3. Container size and weight

The majority of shiplines use some variation of these elements as loading criteria. Shiplines with smaller vessels are very interested in weight, while larger lines are most interested in container dimensions and port of discharge.

After we had the elements, we set up screens describing all the physical locations on the terminal to tie these loading elements to actual spaces.

Once all the programs were written, live data was entered for one shipline and its use compared with the actual field operations.

When the Location Control System reached a point where space criteria could be controlled easily, we started developing the Location Assignments. The first part matches criteria against actual container data to determine the present state of the yard. From this we are able to show areas where trouble will occur when loading operations begin.

With the current state of the yard ascertained, the location system looks at the work that needs to be done such as export containers and containers in temporary parking that need to be moved into the stack, and import containers that are ready to be loaded on to a chassis and moved out the gate. It examines each container and determines the shipline it is for, then it arranges the container data in the manner the shipline uses to load its vessels. All this information is then loaded into temporary image data sets for easy manipulation by the assignment program.

Using a data set that describes the shipline's preferred method of loading and the yard manager's requirements for keeping order in the stack, the Assignment Program brings both together attempting to find a "best match". If a best match is not found, the assignment algorithms ignore the least significant criteria until a match can be made. When a match is made, both the container and location are examined to make sure there is no non-stacking criteria that may preclude putting the container in that spot. If all is well, the location is attached to the container's work record and updated to the data base.

If, for some reason, the container cannot be placed in that spot, everything backtracks to the matching routines.

DEVELOPING THE SYSTEM

We Are Reviewed

The design of the overall system, which we called the Yard Management System(YMS), was unprecedented. For this reason, some of the company management working with us at the time thought an outside review might be helpful.

Hewlett Packard systems design experts and Telxon systems analysts were called in to review the entire design. Their resulting report codified the system and serves as our users' manual. They made some helpful suggestions but approved the design with few significant changes.

Parking Location Tracking

While the previous study was being finished, the yard manager working with the assignments generated by the first programs found a problem. Information about containers in temporary parking locations was lagging way behind all other information. A container that was ready to be stacked could not be located with any accuracy. This slowed field operations unacceptably.

It was decided to put the rest of the work on hold and develop a system for reporting temporary parking locations. We produced a simple system for the handhelds that allowed the field location checker to enter a container's number and location, then have it update the main container database almost as soon as the container was parked.

This subsystem took a month to write and went on-line in January 1988. It was the first field use of the new handhelds.

From a development standpoint, the Parking Location Tracking subsystem gave the systems analyst and programmer at VIT hands-on experience with the handheld units. It also

introduced the ILA personnel to the units.

There were also immediate benefits in the yard. With Parking Location Tracking, containers had their temporary parking spaces before any paperwork entered the system. Previously the yard had to wait two to three hours for the field checkers logs to be entered and a location to be found.

FITTING THE PIECES TOGETHER

In their report, HP and Telxon suggested the use of multiple base stations for handheld communication with the main computer. When Parking Location Tracking was added to the design, however, it was found that multiple base stations were an unnecessary expense and would require an additional radio license. The multiple stations were replaced by dual HP Vectra PCs tied into a STARLAN local area network, making the single base station a shared resource.

These PCs are connected to a reliable power source and receive information directly from the handhelds even if the main computer is down. This solves our problem with unreliable power to the HP 3000.

After this jog in our road, we found we could follow our map to the end without further complications. The ease of interfacing all necessary new equipment to the shop's existing HP 3000 greatly facilitated our efforts.

Testing and Implementation

With Parking Location Tracking done, we scheduled a three-month intense effort for programming and program testing on the stack management handhelds. Then we planned about a month of field testing to get the units into the hands of the ILA checkers and an August 1 implementation date.

A difficulty arose with printing the container release documents. These are printed when a container, which has passed all customs and shipline requirements and is ready to

go out the gate, is placed on a chassis. The difficulty was that the system was notified that the container was being placed on the chassis while the container was still in the aisle among the stacks and before a temporary parking location had been assigned to it. This caused a clerk to have to look up the container location, after it was parked, and write in that location on the paperwork. The solution was to split the mounting of the container on its chassis and the printing of the container release document into separate operations. Now the field location system triggers the printing of the document when the container reaches temporary parking. If we hadn't had the Parking Location Tracking on-line we would not have been able to solve this problem without many additional steps.

Once this hurdle was leaped, we made our deadline and got the handhelds into the field for systems testing.

The first systems test revealed one design weakness: it caused extra radio traffic between the checker and the hustler drivers. The ILA checker who worked with us on this test devised the solution. He suggested printing a sequencing list similar to those that are used for vessel loading. The list shows the containers in the order in which they are to be moved. The list is passed out to each person on the crew when they assemble for the day's work. Radio traffic has been reduced to sending the sequence numbers off the list back and forth.

Unfortunately, vacation schedules of key yard personnel stretched out the systems testing cycle longer than planned. We advanced to a September 1 live date. Even this date gave us some problems as we would not be able to train all crews in use of the handhelds before they went into the field. It was felt, however, that we should get the system up as soon as possible before the heavy months of October and November. Also, a September 1 date would give us the Labor Day holiday to correct any major problems.

Our only problem was the lack of personnel training, and because the Telxon handhelds are so easy to use, this wasn't serious.

On September 1, 1988, the complete Yard Management System went on-line. This resulted in a 99% decrease in the time needed to update container movements. Work was also much more accurate.

HOW THE YARD MANAGEMENT SYSTEM FITS TOGETHER

Equipment

The Yard Management System uses a distributed network composed of a HP3000 Model 70, HP Vectra PCs, and Telxon 750 Radio Frequency Handheld Terminals(see attached diagram). Each node of the system is intelligent and has storage capacity and alternate methods of transferring data. The PC provides the interface link between the HP3000 and the handheld units. A link from the handhelds directly to the 3000 was possible, but unreliable power to the HP3000 is a problem at VIT. Also, the yard works round the clock 7 days a week. We could not stop work for main computer problems or even routine maintenance. To ensure 100% up time of the handhelds, a Vectra PC connected to a reliable power source is used as an intermediary. Also, it was felt that a dedicated PC would provide the best turnaround time to field.

The Telxon 750 Radio Frequency Handheld Terminal is used by the pad workers for YMS processing of work orders and communication messages. The application consists of menu driven, easy-to-use screens providing all the functions required for the terminals to be the sole source of communication between the office and workers. Information is transmitted to and from the 3000 YMS Application in near "real time" , meaning that the information flows through the system rapidly and the database is updated as work orders are completed, but the system does not provide a truly interactive dialog between the YMS Application and the handhelds.

Operations

When a container enters the gate, it is interchanged, then

sent to the temporary parking in the wheeled area. The office key-enters the information on the container into the database: the container's number, its size and weight, port of discharge (destination), vessel of departure and any other information its shipping company deems important.

A data entry clerk enters the container number into the system the minute it clears the gate so it can be sent out to the Parking Location Tracking system. This eliminates the need for the location man to do anything but pick the number from a list and enter a parking location. This temporary parking position is received back into the main system at this point.

For efficient use of storage space we want to get the container out of temporary parking and into the stacks as quickly as possible. The stack and row should be one from which the container can be removed easily when it is ready to be loaded.

The 3000 YMS Application assigns the stack location overnight in most cases. These locations and any other work orders (such as containers coming out of the stack and being placed on chassis) for the next day are downloaded to the PC each night. All work orders are downloaded from the PC to each handheld through a hard wire link prior to the day workers picking up the units the next day.

Once work orders are downloaded, the workers logon, in their work area location, and begin the day' s work moving containers to their ordered location. Since all work orders are loaded into the handheld, workers can change location during the day without reloading work orders.

As a container is moved and stacked, its new location, including stack, row and row position, is entered into the handheld terminal. Completed work orders are transmitted to the YMS throughout the day, and the YMS database is updated. The office can send new work orders to a specific location any time during the day. Both workers and office can send messages throughout the day to facilitate communication. The terminals also keep track of a work crew's status , knowing when a crew is working, on break, eating a meal or has no work to do. At the end of the day, workers return the Telxon

terminals to the office. Any remaining information in the unit is transmitted to the YMS (via hard wire link) and the terminal is cleared to prepare for the next day' s work order download.

If the information in the handheld is lost or corrupted, the workers can reload the terminal by initiating the download via hard link.

After a day's work is completed, the systems produces various reports that show container movement during the day. Others are produced that track the productivity of the crews and their equipment down to the trucks that move the containers around.

Data Communications

As with most HP shops we have used RS-232 for the large part of our communication needs. The Base station communicates with the PC through serial ports and the PC's interface with the HP3000 using standard asynchronous protocols. The PC's communicate with one another using a STARlan local area network. This was chosen because of our familiarity with twisted pair wiring and low cost of equipment. Western Digital Vianet software was the networking operating system we selected. This simplified installation of the network by people without any PC networking experience. We set up a peer-to-peer network with queue files to facilitate passing transactions between systems. The relative low speed of such a network is not a problem when at most a few hundred bytes of data are being passed in any given second.

The Programs

There are several major programs that make up the system.

The 3000 YMS Application is the heart of the Yard Management System. It consists of a combination of online and batch programs which process the VIT work orders and communicate with the Telxon handheld terminals. It is important to note that only the 3000 YMS Application programs access the YMS database. All interface communication to the handheld terminals is accomplished via two message files. Work orders

and messages transmitted from the 3000 YMS to the handhelds are placed in the HANDHELD-OUT file, and completed work orders and messages transmitted from the handhelds to the 3000 YMS are received in the HANDHELD-IN file. Message files can be configured to activate programs when data is placed in the file, so information received from the handheld terminals is processed
immediately.

The 3000 Controller provides a centralized interface link between the many 3000 YMS Application programs and the the handhelds. The Controller is capable of handling multiple PC Controllers to allow for system growth. This is a non-sophisticated program with no knowledge of YMS Application logic; it simply passes information from the HANDHELD-OUT file to the PC Controller, and from the PC Controller to the HANDHELD-IN file. An audit is kept of all transactions processed and any error conditions encountered.

The PC Controller provides the interface link between the HP3000 and the handheld terminals. The PC Controller functions could have been incorporated into the 3000 Controller but it was decided to use PCs to isolate the crews in the yard from any problems on the main HP3000. The PC Controller is a sophisticated program functioning as an integral part of the YMS Application logic. The PC controller maintains work crew status information, and passes work orders and messages between the HP3000 and the handhelds. In case of HP3000 power interruption, the PC Controller will continue handheld terminal processing monitoring work crew status and storing information until the HP3000 is ready to receive data again, and provide PC keyboard input of new work orders and messages. An audit is kept of all transactions and any error conditions encountered.

There are presently three controllers running now that communicate with each other over a local area net work or LAN. One PC controller is attached to the base station directly and provides rapid response to the day-hire-labor. Another is for the Parking Location Tracking subsystem and the final one manages the Work Order Management subsystem. These last two interface directly with the HP3000. A fourth PC is used to speed downloading of work without tying up the other two and

can substitute when one other the others is down. This fourth one also handles reporting in case of HP3000 failures.

The Telxon Base interfaces with the PC Controller to provide the actual communication link to the radio frequency handheld units. This is a non- sophisticated processor with no knowledge of the YMS application logic; it simply passes information between the PC Controller and the Telxon terminals using radio transmissions.

There is also a background program that runs continuously, 24 hours, a day to receive messages and completed work orders from the handheld terminals, and update work order details in the YMS Database. Database update include updates to existing work orders and adding new work orders generated by checker moves and the PC Controller entry when the HP3000 was unavailable.

CONCLUSION

In January, 1988, the Parking Location Tracking subsystem went on-line and in September, 1988, the entire Yard Management System went live. In 1988, VIT handled 323,201 containers, versus the 248,208 in 1987, a 30% increase. This increase in containers propelled VIT past its number one east-coast competitor, Baltimore.

The YMS is designed for expansion, so that it will not become obsolete as the port continues to grow. Eventually Virginia Port Authority's other three terminals (in Portsmouth, Newport News and Front Royal) will go on the system. It will include vessel operations (tracking and moving containers off-loaded from ships) and rail movements.

With the Yard Management System, Virginia port operations are ready for the twenty-first century.

GLOSSARY

Container - The box like upper portion of a trailer in a tractor trailer rig(18 Wheeler), that has cargo packed inside for shipment. They are usually 20 or 40 feet in length and can come in various types such as open topped and flat units.

Stack - The most efficient use of storage space. The containers are placed on top of one another right up against others.

Transtainer - Traveling Cranes that move containers into and out of the stack and from chassis to chassis.

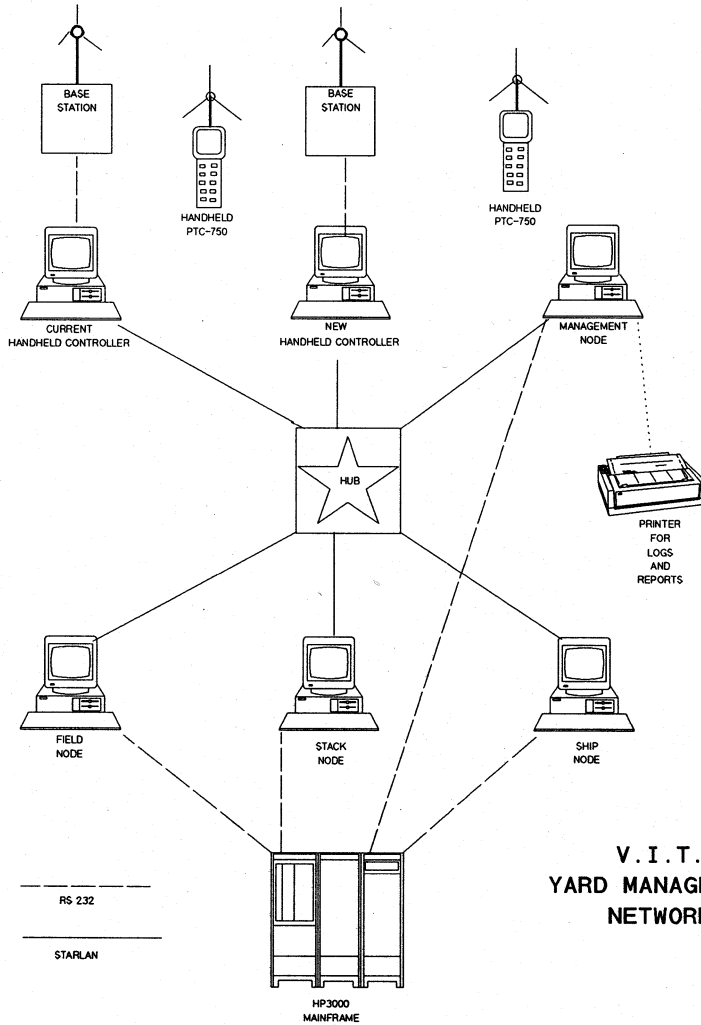
Chassis - The lower portion of the trailer consisting of the bottom frame and wheels.

Crane - Portainer, the device that moves containers on and off vessels.

Hustler - Small tractors/trucks that move containers around the yard, not for road use.

Checker - Chief of a work crew moving containers. He handles the Telxon unit.

ILA - International Longshoreman's Association, the main union for port operations.



V.I.T.
YARD MANAGEMENT NETWORK

Control Container Movements

5808-19

WIP Tracking in an Automated Wafer Fab

Jay Zimmett
International Rectifier
41915 Business Park Drive
Rancho California, CA. 92390
(714)-676-7500

Overview

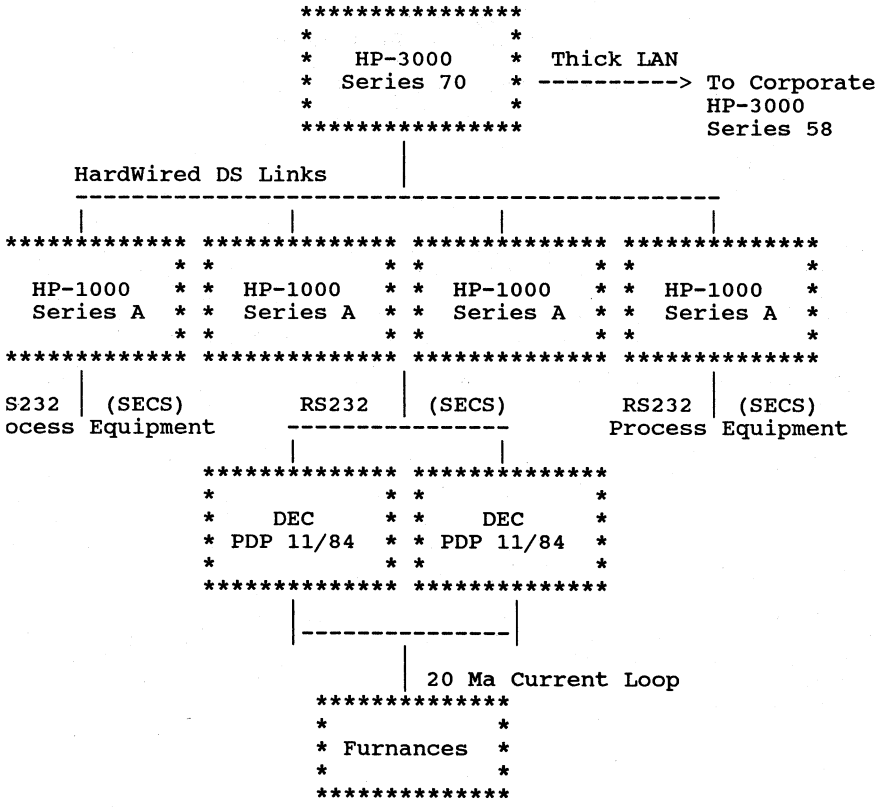
Hexfet America, a division of International Rectifier located in Rancho California, is a new 285,000 square foot semi-conductor facility that currently is producing more than 5 million power MOS-FET's each month. This is partially accomplished with the use of computer systems for process control. The computer software helps manufacturing manage the equipment, processes and bottlenecks of product in the wafer fab.

This facility produces assembled product from raw wafers in less than two weeks. With a fast cycle time, and the volume of WIP (Work in Progress) being processed how do you manage the processes, equipment and product being built? A system is needed to manage the production line and report on yields, inventories and machine problems.

In September of 1986 I joined International Rectifier to assist in the implementation and setup of the SPN (Semi-Conductor Productivity Network) system and the M.I.S department. This paper describes how this project was implemented.

Computer Systems

Hexfet America is using an HP-3000 Series 70 with 16 megabytes of main memory and over 2 gigabytes of disc storage (four 7935's and one 7937 disc drives). Connected to the series 70 are four HP-1000 A-Series computers, and two DEC PDP 11/84 computer systems. Research and Development at Corporate uses an HP-3000 Series 58, one HP-1000 A-Series and one DEC PDP 11/84 computer. Corporate R & D and Hexam are linked together via a Local Area Network (LAN). Approximately 30 personal computers and more than 80 terminals are direct connected to the Series 70.



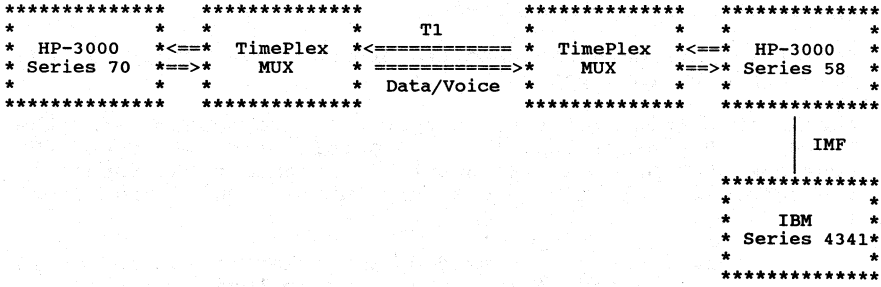
Data Communications Links

Data communications reliability is achieved by using established industry standards. The HP-3000 Series 70 is connected to the HP-1000's using distributed systems (hardwired DS) at 56kb. The DEC's are connected to one of the HP-1000's using RS-232, 25 pin serial communications. The process equipment is connected to the HP-1000's using RS-232 and the SECS protocol. The PC's are connected using RS-232 to the serial ports of the PC's and Walker Richer & Quinn's Reflection terminal emulator software.

Hexam and Corporate are connected using HP's thick LAN (IEEE 802.3) over a T1 data communications link with Applitek LAN

gateways. As a backup to the T1 data link there is also a 9600 baud leased line that is connected with HP's distributed systems (leased line DS). Inventory updates are transfered daily to Corporate as production is completed. Daily shipment data is transfered to Hexam based on the prior days output.

HP's IBM 3270 terminal emulator (IMF) is used to connect the Corporate HP Series 58 to an IBM 4341 where Corporate Order Management and company inventories are located.



FAB Equipment

The types of equipment that are located in wafer fab and managed by the SPN system are standard wafer fab machines with a SECS interface for connection to an external computer system.

The equipment managed by SPN is:

Furnaces

Acid Etchers

3180 Back Metal Sputterers

ION Implantors

Probers

SPN System

SPN is the software system used to manage the WIP and equipment at HEXAM. We currently process more than 50 lots (100 wafers per lot) on a weekly basis. Process reliability

and equipment setup instructions are managed and maintained by the computer software.

SPN is used only as a WIP and process control system with MRP functions accomplished using MSA's AMAPS system. Several crossload programs strip data from the SPN system and update the AMAPS manufacturing system on a daily basis.

The SPN system is a collection of real-time sub-systems that include; Inventory Control (IC-10), Process Control (PC-10), Engineering Analysis (EA-10) and Cost Accounting (CA-10). Pick any two letters of the alphabet, add a dash 10 and there probably is a sub-system for it.

Formerly a standard HP supported product, now modified and maintained by Internation Rectifier.

The major portions of the SPN that we are using are the IC-10/PC-10 sub-systems. They are a collection of over 400 programs that provide a rich set of transaction functions and reports, including several IR unique Quiz and Cobol reports.

The other two sub-systems used on a regular basis are CA-10 Cost Accounting, and EA-10 Engineering Analysis.

Where does it Start

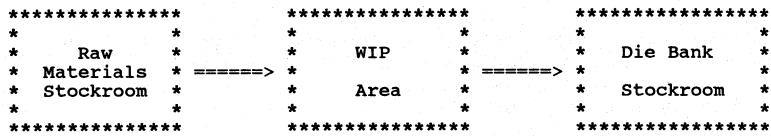
IC-10 is implemented using, stockrooms and WIP areas. A stockroom is a raw materials inventory location and the WIP areas are where the product flows through.

Raw material is received into a stockroom by part number as it arrives into the receiving department and passes incoming Quality Assurance. At this point the raw materials are ready to be used in production and are assigned by the production control department to starts for that week. Production control then issues the raw material to wafer fab for processing.

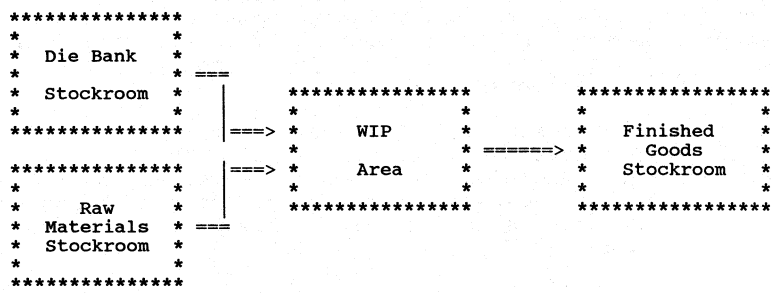
Raw material inventories consist mainly of raw silicon wafers, and photolithography masks for wafer fab.

Once a lot is created (a single wafer fab lot consists of 100 wafers in 4 groups of 25) by issuing material from the stockroom into the WIP area, stockroom inventories are decremented and WIP inventories are incremented. The new lot is now visible to the SPN system and ready for its journey through the wafer fab and into the die bank stockroom for Assembly.

Wafer Fab Product Flow



Assembly Product Flow



Product Flow through Wafer Fab

Once a lot has been issued to wafer fab for processing, it is located at the first process location. A WIP area consists of many process locations and a location consists of many operations. IC-10 can track a lot at the high level process locations or the detail level of an operation. The PC-10 system is used to actually process the lot. IC-10/PC-10 are integrated together and the operator is not aware where the two systems merge.

The SPN system consists of verbs that perform the actual operator request. The PC-10 verb TRACK is used to move the lot from one operation to the next and handles the transfer from PC-10 to IC-10 on location level transfers.

When the lot is created, a part number is assigned to the product which determines what type of wafer we are producing. From the part number, the SPN system knows what process recipes to use and what the lot flow from operation to operation will be.

The TRACK verb requires only the lot number and the station

ID that will be processing this lot as input. The TRACK verb then verifies that the station can process this lot and is not busy. It then displays the in-going operator instructions and downloads the correct process recipe for this part number and station to the fab equipment. The operator is then informed whether the download was successful and instructions on how to process the lot. The TRACK verb determines whether you are tracking into a station or out of a station.

Once the lot has completed its current operation, the operator again uses the TRACK verb to track out of the current station and has the opportunity to adjust the lot's quantity if some of the wafers where damaged or broken at that process step. After accounting for all the wafers, additional instructions are displayed on how to prepare the lot for the next operation.

Once the lot has completed its journey through wafer fab and all wafers have been accounted for, the lot is then closed into the die bank stockroom. Inventories in WIP and the stockroom are adjusted accordingly. Lot are stored until production control issues the die to Assembly for making the finished product.

Where is that Lot?

We now have an idea on how the SPN system works:

1. Receive material into a raw materials stockroom.
2. Issue that material from the stockroom to the line.
3. Move and process the lot at each operation.
4. Close the lot from WIP into the die bank stockroom.

But how do we keep track of all the product flow? SPN has more than 50 different reports, online and offline, that will report in real time what the current inventories in stockrooms and WIP areas; by part number, location or operation.

These reports let management view how much product is moving through the production line and where possible bottlenecks may be occurring.

Examples of some of the screens and reports are:

CCLOT Screen

Show a particular lot its current status and history.

```
TD-10 SCREEN IMAGE                          05/22/89  10:20:34      PAGE: 1

    SPN SEMICONDUCTOR PRODUCTIVITY NETWORK

VERB: CCLOT     RETAIN?        SEE A WIP LOT'S LOCATION AND OPER INFO

     LOT: 203167                                              CUR LOCN: 2280
    PART: 005230                                            START QTY:      100
 PROCESS: 00523000          OPERATION:                      CURRENT QTY:     3776
START DT: 03/23/89 16:04:41  STATUS: CLOSED                 OPER STATUS:
SCHED CLOSE: 04/01/89          TYPE: PRODUCTN                 PRIORITY: 0

VERB     TRANS DATE  EMPL     LOCN PR TY  COMPL   SCRAP   RECYL S

BIN LOT NUMBER      GOOD  SCRAP   RECYCLELOCN NEW PART
  1                 3776                      005248

BINCLS   04/01 06:46 91963    2280 0  P    3776       0       0
                                            GOOD WFRS:      95 ZERO WFRS: 1
$TRKMV   04/01 06:45 91963    2260 0  P      96       0       0
LOSS     03/31 13:07 81447    2240 0  P    1010/1
$TRKMV   03/31 13:07 81447    2240 0  P      96       1       0
COMMENT  03/31 09:56 81447    2240 0  P BROKE UNDERNEATH CASSETTE...
COMMENT  03/31 09:54 81447    2240 0  P BARRY WAS FIXING CARRIAGE, IT

                    *** ENTER C TO CONTINUE ***
```

CSTAHST Screen

View the lot and processing history of a particular piece of equipment.

```
TD-10 SCREEN IMAGE                          05/22/89  10:01:22      PAGE: 2

    SPN SEMICONDUCTOR PRODUCTIVITY NETWORK

VERB: CSTAHST  RETAIN? Y       SEE STATION HISTORY

         AREA: 2002-HEXAM WAFR FAB                          DATE: 05/22/89
  STATION ID: 102        -FURNACE (BORON DOPE)              CLASS:
EQUIPMENT ID: 01A         TYPE:           MAX ACTIVE WAFERS:     0 MODE: A

  DATE      TIME    ACTION   HISTORY
05/22/89 08:36:28 IN       LOT:203636     OPR:FN652     EMP:81479     QTY: 99
05/22/89 06:18:46 ALARM    Binary Alarm is CLEARED.                   CNT: 1 CLR
05/22/89 06:15:36 ALARM    Binary Alarm.                              CNT: 1 SET
05/20/89 11:47:32 OUT      LOT:203628     OPR:FN652     EMP:81479
05/20/89 08:02:55 IN       LOT:203628     OPR:FN652     EMP:81479     QTY: 99
05/20/89 03:33:23 OUT      LOT:203627     OPR:FN652     EMP:81490
05/19/89 20:56:21 IN       LOT:203627     OPR:FN652     EMP:81490     QTY: 100
05/19/89 20:50:26 OUT      LOT:203616     OPR:FN652     EMP:81490
05/19/89 12:53:13 IN       LOT:203616     OPR:FN652     EMP:91963     QTY: 100
05/19/89 12:36:17 OUT      LOT:203599     OPR:FN652     EMP:91963
05/19/89 08:43:09 IN       LOT:203599     OPR:FN652     EMP:91963     QTY:  99
05/17/89 10:47:39 ALIVE
05/17/89 09:15:18 DEAD
05/16/89 18:11:57 OPLOAD                  OPR:FN652     EMP:81449

                  *** PRESS C TO CONTINUE ***
```

CLOTQ Screen

View what stations are available to process a lot.

```
TD-10 SCREEN IMAGE                          05/22/89  10:03:49      PAGE: 3

    SPN SEMICONDUCTOR PRODUCTIVITY NETWORK

VERB: CLOTQ    RETAIN? Y      SEE STATIONS WHICH CAN PROCESS A LOT

AREA: 2002 - HEXAM WAFR FAB  LOT: 203599       SCHED CLOSE: 05/20/89 PRI: 0

CURRENT QTY:     98       PROCESS: 1078     - HF,G3,P,0100    LOCATION: 2240
  START QTY:    100     OPERATION: M167    -PROCESS  SELECT STATION:
   NEXT LOC: 2260       NEXT OPER: ETCH207 -PROCESS    SAME STATION:
  LAST MOVE: 05/22/89 07:22 PART: 126627                        AISLE: 1
 START TIME: 05/13/89 07:42     LOT STATUS: OPENED     OPER STATUS: QUEUED
                               QUEUE STATUS:          QUEUE STATION:

                    CURRENT  COMPLETE QUEUE SEL
STATION ID STATUS   OPER-NO    TIME    CNT  CNT  SELECTED LOTS
301        IDLE               00:00:00   0    0
302        IDLE    25-0082    13:49:30   0    0
303        BUSY    M123       10:03:06   0    0
304        IDLE    25-0036    00:32:41   0    0
305        IDLE    M134       06:44:20   0    0
306        BUSY    25-0092    10:15:31   0    0
307        BUSY    M145       10:46:33   0    0

                  *** ENTER C TO CONTINUE ***
```

WIP Tracking in an Automated Wafer Fab

PCR 210 Report

Show current inventories and yields for all locations
in the wafer fab.

PCR210.A (A.07.04) RUN 22 MAY 1989 09:54
AREA: 2002-HEXAM WAFR FAB MONTH START DATE 05/01/89 00:00
REWORK TYPE: ALL ACTIVITY WEEK START DATE 05/22/89 00:00
DAY START DATE 05/22/89 00:00
END DATE 05/22/89 09:21

INT'L RECTIFIER HEXFET AMERICA
FOR ALL LOT TYPES
PRODUCTION / INVENTORY AND YIELD SUMMARY - GRAND TOTALS

LOCN	LOTS PROC	CURR INV	BOH	PRCS'D	DAY GOOD OUTS	YLD %
2020 STEP 1	159	0	0	500	500	100.0
2040 STEP 2	170	699	399	200	198	99.0
2060 STEP 3	164	100	0	98	98	100.0
2080 STEP 4	174	394	296	0	0	100.0
2100 STEP 5	177	648	727	79	79	100.0
2120 STEP 6	175	403	700	376	376	100.0
2140 STEP 7	174	970	693	99	99	100.0
2150 STEP 8	167	488	755	366	366	100.0
2160 STEP 9	163	464	488	390	390	100.0
2180 STEP 10	162	390	197	197	197	100.0
2190 STEP 11	165	197	0	0	0	100.0
2200 STEP 12	164	685	980	295	294	99.7
2210 STEP 13	6	176	176	0	0	100.0
2220 STEP 14	153	196	0	98	98	100.0
2240 STEP 15	155	169	430	359	352	98.1
2260 STEP 16	163	1235	1224	341	244	71.6
2280 WAFER FAB CLOSE LOCN	149	0	0	244	213	87.3
LOCN	LOTS PROC	CURR INV	BOH	PRCS'D	DAY GOOD OUTS	YLD %
TOTAL 1ST LINE	640			3642		69.3
TOTAL 2ND LINE		7214	7065		3504	

DATA ON THE PCR 210 REPORT IS FICTICIOUS

Future Automation Projects

CIM projects that are under development are focused on our Assembly operations. One major problem in assembly is the use of the wrong test programs at final test.

The interface from the SPN system to the testers for the downloading of test programs will help alleviate the problem of wrong test programs being used on the wrong parts. Using a test program for a 500 volt product on a 100 volt product will blow out all the gates. The proven technology used to manage the wafer fab equipment is being used to start the automation of Assembly.

MIS System Resources

What does it take to support a 24 hour, 7 day per week operation?

The SPN system is supported by two programmer/analysts who remain on call when not in the plant. Operations fields all initial calls and performs first level problem determination and if necessary then notifies the appropriate programmer to handle the problem.

Operations is only supported on first shift and are on call after hours, with all personnel having terminals at home to dial into the system.

To help operations manage the system, selected third-party software packages were purchased. ONLINE-3000 by Orbit software is used as our backup software to minimize downtime due to backups. JOBRESCUE by NSD is used to manage the many report jobs that run and only prints the JCL of jobs that have not completed normally. SPEEDEDIT by Bradford is used for programming to get the job done faster and better. ADAGER by Rego is used to manage capacities of our databases.

Conclusion

For the needs of International Rectifier, SPN is the right choice. We are able, on a consistent basis, to process all lots of indentical parts using predefined computer based controls without adding any operator error associated with equipment setup. This allows the engineers to concentrate on improving the process recipes and not worry about equipment setups and operator error associated with this.

The additional third-party software products make the job of running and maintaining the system much easier.

AUTOMATIC IDENTIFICATION GETS BOTTOM-LINE RESULTS

Thomas R. Swanson - Office & Materials Control Manager
David N. Thompson - Data Processing Supervisor
James E. Halverson - Senior Systems Analyst

3M
Corporate Quality & Manufacturing Services
3M Center, Bldg 427-2-01
St. Paul, MN 55144-1000
612/736-0554

Introduction

In today's business climate, reducing costs, improving accuracy, and increasing productivity are a must. Our way to achieve this was through the computerization of manual, time-consuming, paper-intensive processes. Using the HP3000 Series 70 as the host, we have designed and implemented inventory control systems that integrate radio frequency (RF), voice recognition, bar codes, laser scanners and hand-held auto ID terminals. The benefits of these systems are significant: annual savings of over $150,000, accurate, real-time data, a 35% increase in productivity and improvement in job satisfaction and quality.

Rack & Core System

The Rack And Core System (RACS) was designed to process orders and track inventories of tape racks and cores. RACS inventory is contained in over 99 plants and warehouses nation-wide, including the one million square foot warehouse at the St. Paul Distribution Center (DC). There are over 20 different sizes of racks and cores, and in most cases the difference between sizes are a few inches. Inventories for each item run into the thousands, and every day there are as many as 500 items in transit. The DC acts as the "home warehouse," with most shipments coming into or going out of this facility.

Originally, there was almost no tracking of inventories. Plants and warehouses would contact the DC when racks or cores were needed, and the DC would ship them (if they had them), not keeping track of what was being shipped. Many of the plants and warehouses were in the habit of "hoarding" inventory, not letting the DC know they had them, so as to keep a hidden inventory on hand. As you can imagine, there were many problems that needed to be addressed.

To solve these problems, we designed RACS. The system is housed on an HP3000 Series 70, and input is through HP terminals, a Vectra PC and radio frequency (RF) hand-held terminals. The RF terminals are unique, in that they combine the best aspects of both batch and on-line processing. To accomplish this, we took a batch hand-held terminal with a built-in laser scanner and attached it to a portable radio. The radio transmits the entered data to a receiver, which passes the information on to the HP. If the receiver or system are busy when the hand-held attempts to send, or if the user takes the RF units out of receiving

range, the hand-held stores the data internally, automatically transmitting when possible.

The system itself is relatively simple. All the racks and cores are bar coded with its size and the unique 11 digit stock number. When a rack or core is shipped, the employee scans the rack or core with the RF terminal's scanner and enters the quantity. The system then compares what was entered to the order and checks for possible discrepancies or back orders. Receivals are treated in a similar fashion. The data is entered/scanned into the RF terminals and the system does instantaneous comparisons and updates. When cycle inventories are taken, the stock number and quantity are scanned and entered on the RF terminal, which in turn checks with the data base inventory and either directs the employee to proceed (if inventories match) or to recount the item (if the inventories don't match). When inventory is completed, a detailed report of non-matches is printed for reconciliation.

Not only has RACS allowed us to greatly improve the accuracy of the inventories and shipments, but it has reduced trailer loading time and eliminated the need for an annual physical inventory. We are seeing a dollar savings of over $30,000 annually based on reduced man hours. RACS has worked so well in fact, that the various warehouses no longer feel it necessary to keep extra inventory on hand, thus reducing the overall system inventory.

Reship Order Control System

The Reship Order Control System (ROCS), which is used at the St. Paul Distribution Center (DC), is designed to handle the reship operation. "Reship" is defined as receiving freight from other 3M sites and outside vendors, consolidating the loads and then shipping full trailer loads of material to various 3M operations, thereby cutting shipping costs. In a typical month, 10,000 pallet loads, over six million pounds of freight, are handled. The system previously in place to process this volume of material and accompanying paperwork had a number of problems which needed to be overcome. These included transposition errors, inaccurate and incomplete information, as well as data entry errors. These problems, plus the eight to sixteen hours lag time in entering data, produced problems in tracking both information and material through the system.

The need for a new system was obvious, one that could accumulate all the data on the receiving documents, minimize manual effort and interpretation and input errors. We needed to be able to answer questions on receiving loads in a minimum amount of time, and reduce the time delay in reporting shipping information. Often, local vendors would want to know if we received their load 30 minutes after the trailer entered the site. ROCS solved all these problems and at a relatively inexpensive price.

The equipment used for ROCS includes the HP3000 Series 70, three hand-held batch terminals, a voice recognition system, a Speech Aided Modem (SAM), a bar code printer and assorted HP terminals and scanners. The hand-helds and voice system are used to enter shipping and receiving data into the computer. The hand-helds are uploaded via phone lines into the SAM, while the voice system works via RF. Upon receiving the information, the system updates the data bases and, for receivals, prints out a bar coded label, which is applied to the load. These labels have both bar codes and human-readable data, including destination,

invoice number, date shipped, multi-order information, etc. We also assign a unique pallet or PRO number to each shipment coming through the DC. We use this number to track the loads, from staging to shipment.

Our receiving and shipping information gets into the system so fast we can honestly say that if it's not on the computer, it hasn't been received yet. Not only did we save significantly in productivity, accuracy and quality, we also showed first year savings of $75,000 on total spendings of $26,000. These savings were accomplished by reducing man hours through effective utilization of personnel.

Employee Store Inventory Control System

Because of our previous successes in system implementation, the staff group overseeing the 3M employee stores network came to us looking for in inexpensive yet accurate method for tracking inventory and orders. The employee stores sell 3M products and various non-3M merchandise to employees at "cost." Products are stored at a central warehouse and sold through two stores and a mobile trailer in the St. Paul region. Outside of St. Paul, over 50 branch locations participate via catalog sales, which are handled at the warehouse.

The receiving, stocking, order filling and physical inventorying required a mammoth paper-driven process. Every procedure and transaction involved either new or modification of existing paperwork. This resulted in slow turnaround on goods, inaccurate inventories and misplaced orders. The computer system, which we designed to replace this cumbersome manual system, is called the Employee Store Inventory Control (ESIC) system. ESIC incorporates bar code technology, auto ID hand-held terminals and HP hardware.

ESIC was implemented at the warehouse with a Vectra PC, printer, modem and three hand-held terminals. These data terminals are unique in that they have a built in bar code scanner allowing for one-handed operation. The equipment at the two St. Paul stores, and certain branch locations, consists of HP terminals, integral printers and modems. All these work stations tie into our HP3000 Series 70, which houses the background programs and data bases.

The warehouse has two methods for receivals: Full pallets are placed in rack storage, where every location has a unique bar coded number. These pallets are followed from location to location using a magnetic backed label, on which a bar code of the catalog number is printed. These tags are placed on the rack next to the location label. This assists in tracking the product from receiving to shipping, and helps to verify the accuracy of the loads. The second method for receivals involves fast moving items such as video and audio tapes, sandpaper and Post-it notes. These items are stored in bin locations, which are assigned specifically for that product. This allows the catalog number to double as the location number. As items are received into the warehouse, catalog number, location and quantity are all scanned or entered into the hand-held. The hand-held is uploaded to the HP3000 via the PC at various times during the day.

Order filling requires a different procedure. Orders from the stores and catalog sales locations, which have terminals and modems, are entered on-line at the branch location, and is available for processing at the warehouse immediately after being entered. Non-inputed catalog orders are sent via mail to the ware-

house and are entered onto the system upon receival. A pick-list is then printed, and the employee fills the order using the hand-held to record the catalog number, location filled from and quantity. When the order is filled, the data is uploaded and a packing list is printed detailing the items filled and any back orders. The data base is automatically updated to reflect the new inventories at each location.

The third procedure in ESIC is completing the monthly physical inventories. To accomplish this, the catalog numbers, locations and quantities are downloaded into the hand-held. The employee then takes the hand-held into the warehouse, scans the catalog number and location and enters the physical count. The hand-held compares what was entered to what was downloaded. If they match, the employee is instructed to proceed to the next item. If the quantities differ, a recount is requested. When all items have been counted and entered, the information is uploaded to the PC, where a reconciliation report is produced.

A final step, not yet installed, is point-of-sales inventory control. This will call for Vectras, with cash drawers attached, to be used as registers. Bar code scanners will be attached to each PC/register so that all sales will immediately update the data base.

Conclusion

RACS, ROCS and ESIC are all relatively simple systems. Their complexities lie within the hardware and software that is available to you, as it was to us. Our job (and possibly yours) was to put the pieces of the "puzzle" together to provide cohesive, organized and effective data gathering systems.

WRITING YOUR ON-LINE SYSTEM AS ONE PROGRAM

John Van Kirk and Cheryl Van Kirk
Van Kirk Consulting
P.O. Box 1204
Port Alberni, British Columbia V9Y 7M1
(604) 724-2080

OVERVIEW

We are a small consulting firm which obtained a contract to write a management maintenance system for Alberni Pulp & Paper, a division of MacMillan Bloedel Ltd, on an HP3000 series 70 using COBOL, IMAGE, and VPLUS. This type of system has extremely varied usage by each person. A typical sequence might be to look up information for:

A work order
The equipment to be repaired for that work order
The spare parts list for that equipment
The status in mill stores for the parts required
History on work done on this equipment in the past
Which workers are trained to do this work

In this environment, the ability to move quickly between functions is critical to its success.

The flow in this system also required a frequent cascading of functions. For example,

A function to list all purchase orders originated by a user might be executed.

The user may select one of these purchase orders and execute a function to list the related purchase order information.

He may then execute a function to display the vendor information for the selected purchase order.

The flow down and back up such an inquiry sequence had to be natural and easy for the user.

Finally, the completed system will have more than 400 functions that the user may want to use. Therefore a procedure had to be developed to manage this large number of functions which allowed for quick and natural movement from one function to another.

WHAT IS WRONG WITH PROCESS HANDLING

On the HP3000 it is common to use process handling to move between various functions in an on-line system. This can be successful but has several problems:

1. Process initiations are extremely resource intensive and inherently slow.

2. In a VPLUS application, the terminal environment must be closed and reopened each time a process change is made.

3. In a database environment, the database is continually being closed and reopened.

4. Passing information between processes can be awkward.

Some of these problems can be partially alleviated by suspending from a process and reactivating it. However in a large system, the number of processes can become high.

PROGRAM DESIGN

In our opinion the best solution is one program which contains all on-line functions. On a classic HP3000 using a traditional program structure such a program would quickly run into data stack overflow. Therefore the program must be structured in a different way to avoid this problem.

Let us look at the purchase order search example posed in the overview. The following subroutines might be written to implement a solution to this problem:

Subroutine A: List all purchase orders for a user.

Subroutine B: Display the information for a specific purchase order.

Subroutine C: Display the information for a specific vendor.

A traditional structure would normally have

Mainline program calls subroutine A. Then,

Subroutine A calls subroutine B passing the selected purchase order number. Then,

Subroutine B calls subroutine C passing the associated vendor code.

Soon the data stack on the HP3000 is exceeded.

An alternate structure would be

Mainline program calls subroutine A. Then,

Subroutine A saves the selected purchase order number and returns to the mainline. Then,

Mainline program calls subroutine B. Then,

Subroutine B retrieves the selected purchase order number, displays the purchase order information, saves the associated vendor code and returns to the mainline. Then,

Mainline program calls subroutine C. Then,

Subroutine C retrieves the selected vendor code and displays the vendor information.

For the alternate structure to work and avoid data stack overflow, the following conditions must be imposed:

1. All subroutines must be dynamic.

2. Information must be passed between subroutines by some mechanism other than standard parameters.

EXTRA DATA SEGMENTS

The classic HP3000 has a limited data stack size of 31,232 words. This data stack contains all data directly accessible by the program. For example in a COBOL program, the working storage fields, file record buffers, and data fields (such as literals) which are created by the COBOL compiler must all be stored in the data stack. In addition, VPLUS and other system facilities take data stack space. However, there exist extra data segments which can allow a program to have more memory space for data. Each extra data segment can be up to 32,768 words long and a process can have multiple extra data segments.

Unknown to many programmers is the fact that they use extra data segments in almost all programs. When a file is opened, an extra data segment is created to handle the blocks of data being read and/or written.

The restriction on extra data segments is that the program can not directly access the information stored in the extra data segment. For example in the case of file input, the system must transfer the record to be read from the block of records that has been read into the extra data segment to the program's data stack.

MPE FACILITIES TO USE EXTRA DATA SEGMENTS

MPE allows a user to create and manage extra data segments if he has data segment management (DS) capability. This capability must be associated with the program when it is PREPed using ';CAP=DS' on the PREP command. Five intrinsics are provided to manipulate extra data segments as described below.

GETDSEG

This intrinsic normally creates a new extra data segment. The call to this intrinsic passes the maximum size of the extra data segment and returns the 'logical index number' which is used in all other intrinsics to identify the extra data segment. The newly created extra data segment will remain until the process terminates or the FREEDSEG intrinsic is called.

DMOVOUT

This intrinsic moves a specified number of words from the normal program data stack (working storage in a COBOL program) to the extra data segment. An offset is passed to allow the information to be moved anywhere within the extra data segment, not always starting at the beginning.

DMOVIN

This intrinsic moves a specified number of words from the extra data segment to the normal program data stack (working storage in a COBOL program). Again an offset is passed to determine where the information starts in the extra data segment.

ALTDSEG

This intrinsic increases or decreases the size of the extra data segment. It is used for efficiency reasons only and can never expand the extra data segment to a size greater than that passed to GETDSEG.

FREEDSEG

This intrinsic releases an extra data segment when it is no longer required.

USE OF EXTRA DATA SEGMENTS

Extra data segments were used to solve both of the requirements mentioned for the alternate solution in the Program Design section above. Many subroutines can not be

dynamic since they must retain information from one call to the next call. In this case, the information can be moved to an extra data segment just before the return from the subroutine and then moved back to the program's data stack after it is called again. Often it is only a small subset of the program's variables which must be saved.

Information can be passed between subroutines using extra data segments as well. One subroutine can later move information to an extra data segment from its working storage. A different subroutine can move the same information from the extra data segment to its working storage.

This method of passing information is not precisely what extra data segments were designed for. There is a system limitation for the number of extra segments that can be created (at most 4096 data segments can be created in total for all processes running on a system). They are normally used for relatively large blocks of data such as their use by the file system. In the use described above, there will typically be numerous small blocks of data which must be stored. Taking the obvious solution of using a different extra data segment for each case where information must be saved will not work since it would require too many different extra data segments.

To get around this problem, we wrote two subroutines, WRITE-COMAREA and READ-COMAREA, to handle what we refer to as a communication area. A communication area is a named block of data which is stored in an extra data segment.

The calling sequence for WRITE-COMAREA is

```
CALL "WRITE-COMAREA" USING COMAREA-ID COMAREA-NAME
                           COMAREA-LENGTH COMAREA-DATA
```

Initially COMAREA-ID is set to zero. On the first call, an extra data segment is created and its identifying "index" (returned by the GETDSEG intrinsic to uniquely identify the extra data segment) is returned in the COMAREA-ID field. A table is created at the beginning of this extra data segment to form an index of communication areas. The name of the block of data to be stored is passed as a six character COMAREA-NAME. Each call to WRITE-COMAREA searches the index of communication areas for the passed COMAREA-NAME. If it is found, the information in COMAREA-DATA is moved to the previously allocated area in the extra data segment for this communication area. If it is not found, the next unused portion of the extra data segment is reserved for the new COMAREA-NAME, an index entry is created, and the passed COMAREA-DATA is moved to it. COMAREA-LENGTH is the number of bytes of data passed in COMAREA-DATA.

The calling sequence for READ-COMAREA is similar:

```
CALL "READ-COMAREA" USING COMAREA-ID COMAREA-NAME
                          COMAREA-LENGTH COMAREA-DATA
```

In this case, COMAREA-NAME should be found in the index. The information previously saved in the allocated area of the extra data segment is returned to the calling program in COMAREA-DATA. By convention we returned HIGH-VALUES in COMAREA-DATA if the name was not found in the index. In order to protect against incompatibilities, the COMAREA-LENGTH is passed on all calls and verified for consistency with previous calls for the same communication area. This verification proved helpful in program maintenance to insure all subprograms using the same communication area are changed if new fields are added to it.

These two routines make it very simple for programs to use extra data segments for any reasonable amount of data. In some cases, a communication area will only be a few words long. In other cases, a communication area will be thousands of words long. If the first extra data segment fills, additional extra data segments are created automatically. A junior programmer needs to know nothing about extra data segments to use them effectively.

There may be concern about the overhead of using extra data segments in this way. Due to our client's programming standards, the READ-COMAREA and WRITE-COMAREA routines had to be written in COBOL so some improvement could be obtained by rewriting them in PASCAL or SPL. We ran tests on the COBOL routines on an idle series 70 to calculate the average CPU time for calls to READ-COMAREA and WRITE-COMAREA. Depending on the block size and number of different communication areas, each call took 1.5 to 3 milliseconds. In our experience, as long as the usage is kept reasonable (usually only a few calls each time the user hits the ENTER key), system response time is not noticeably affected.

TRANSFER MECHANISMS

A traditional menu system allows a user to transfer between the basic functions in the system. We perceived the requirement to support transfer mechanisms similar to menus within the application routines themselves as discussed in the purchase order example above. To support this facility, we developed the concept of one function suspending to a second function. Using this facility, one function would present a list of items from the database and then allow the user to select one. That function would then suspend to a function which processes the selected item.

In the suspend operation, a substantial portion of the current working storage is saved using the WRITE-COMAREA routine above. Control is then transferred to the function which processes the selected item. When the user exits that function, the saved working storage area will be restored and control will be returned to the initial function. The user then will see the same list as he made his initial selection from. He can make another selection, go to another page of that list, etc. Within reason, this process can be repeated several levels deep and can be applied to other situations not involving lists. We arbitrarily set a maximum of 20 levels. The resultant screen flow has been well received by the users and the main cost on the system is a somewhat higher virtual and real memory usage.

The suspend facility is a very controlled facility where the programmer determines which functions a user can suspend to at any point. This control insures that lower level functions do not alter communication areas required by higher level functions. There was also a requirement for an uncontrolled facility. The best example is a telephone call requiring a query when the user is in the middle of some involved operation. Ideally, the system should allow the user to make the query and then resume the involved operation where he was. For this case, a similar facility to the suspend called an interrupt was developed.

For the interrupt, the initial processing to save the current environment is similar to that for a suspend. However, in this case the COMAREA-ID is zeroed to create new copies of any communication areas and the databases are opened a second time to preserve current chain positions. The main menu is then displayed to allow the user to do any function in the system. When the user terminates the interrupt, the databases opened at the time of the interrupt will be closed and the extra data segment created for communication areas used during the interrupt will be released. Again the user is presented with the screen he had been working on when he initiated the interrupt. The overhead in the case of the interrupt is clearly greater than that for the suspend and the interrupt was of secondary importance to the system.

FLOW CONTROL PROGRAM

This alternate program structure design will allow a large program to be written. However care must be taken to insure the resulting program is manageable. Essentially the main program must be a traffic cop to direct the user inputs to the appropriate subprogram which will process them.

For example in our system, the main program (called the flow control program) did the following work:

1. Handled almost all of the VPLUS work.

2. Did the required processing to transfer between functions.

3. Did the required processing to handle the suspend and interrupt facilities.

4. Processed function keys except those requiring a specific application action such as next page.

5. Handled the security to determine which user can execute which function.

6. Handled the display of help information.

FLOW CONTROL DATABASE

A well designed system should be composed of modules which can be organized in ways that will satisfy the user requirements. Since user requirements are always changing, flexibility is very desirable in the design of a system.

We decided that the flow between modules should be kept in a database which we called the flow control database. In this way the flow control program could be written to handle menus and other flow mechanisms in general. Then as the system developed, new subroutines could be written and put into the group SL. The instructions for processing the new subroutine would be put into the flow control database. This structure has a minor problem with dynamically invoking subroutines on the HP3000. We will discuss this problem later with our solution.

The flow control database is composed of the following types of information:

1. Function definitions
2. Menu definitions
3. Function key definitions
4. Users who can access the system
5. Function security rules by user

Let us look at the first three of these in turn. The last two are mainly concerned with security which is beyond the scope of this paper.

1. Function definitions

In most cases, a function was a VPLUS form and the subroutine which did the processing for that form. The flow control database contained the following information for each user function:

- The function name as known to the user.
- The function name as known to the program (this was required since in several cases one subroutine was used to process multiple VPLUS forms and appeared as different functions to the user)
- The VPLUS form name.
- The default field where the cursor should be positioned.
- The description of the function if it appears on a menu.
- Other information depending upon application requirements

2. Menu definitions

A menu definition was viewed as just being a another function. This is one case where a single program function processes many user functions. In the flow control database, a menu function had a list of the functions associated with it and a special screen title field to help customize the look of the form since we also used a generic form. This approach made it very easy to create custom menus with minimal work on our part.

3. Function key definitions

Function keys were extensively used throughout the system to move from one function to another function. For each user function, the flow control database contains the function key labels to be displayed and what action is to be taken by the flow control program if the user depresses the associated key. The following list of actions are the most commonly used:

ACTION	DISCUSSION
Transfer	Unconditionally transfer to the specified user function.
Suspend	Suspend to the specified user function.

Interrupt	Interrupt the current processing and transfer to the specified user function.
Process	Processing not done by the flow control program.
Exit	Return to previous function if suspended, return to previous menu if not suspended, or exit if no more menus.
Undefined	Function key is undefined.

SIDE BENEFITS

The above structure led to a single call to VREADFIELDS in the flow control program which received any input from the user. This provided some unexpected opportunities. For example:

1. The system had many unattended inquiry terminals which could not be logged off in the normal way. If the program detected that the session limit had been set to zero, the program would terminate and the logon UDC would do a BYE. By putting a timeout on the call to VREADFIELDS, data centre operations could set the session limit to zero, wait five minutes, and all terminals would be logged off for backups.

2. Usage statistics could be collected by function including average elapsed processing time (i.e. the time from the return from VREADFIELDS to its next call). This is not true response time since it does not include data communications time, but does give a good indication of program bottlenecks and provides future statistics to monitor response time degradation if the system becomes overloaded.

PROBLEMS ENCOUNTERED

This structure achieved the basic design goals, but as always there were some problem areas. Some of these were:

1. A VPLUS forms file has an absolute limit of 32,767 records. At this time, it appears this limit may be encountered with large programs. For example, we currently have 229 forms and are using 15814 records. Using multiple forms files in a program could be considered to get around this limitation.

2. Dynamically invoking subroutines on the HP3000 can be slow and resource intensive. Therefore we had an application specific subroutine which used a CASE

construction to check each function and call the appropriate subroutine. Therefore whenever a new program function was added to the flow control database, this routine had to be changed. This was a nuisance, but the information was very static so we rarely had to change this routine for an existing function. Any changeable data was in the flow control database.

3. A COBOL subroutine can not be made dynamic using the procedures discussed above if both of the following conditions apply:

   A. The subroutine uses COBOL input/output statements such as OPEN, READ, etc.

   B. The subroutine must be called multiple times while the file is kept open.

   In a database environment, this problem does not occur frequently. In some cases, the system intrinsics FOPEN, FREAD, etc. can be used instead of COBOL's OPEN, READ, etc. to allow specific subroutines to be dynamic.

CONCLUSION

Overall, this structure has worked well for us. Compared to a system with process handling, changing functions is quite rapid. The system has also been able to adapt to the user's needs.

The use of extra data segments has been almost transparent to the programmer but allowed far more freedom than normal in data usage on the HP3000. The limited data stack problem is not eliminated but is greatly alleviated.

We feel the development effort has been less than a more traditional approach since most of the routines which do the application oriented processing are not concerned directly with VPLUS (except for accessing subroutines which set fields to an error status and change field enhancements). The application routines are passed a screen image from the flow control program, do their processing on it, and return it to the flow control program (not dissimilar to the work required in a batch system). Extra effort was required for some of our list processing, but this functionality was beyond what is often included in an on-line system.

In the ultimate test, the system has been well received by the users who appreciate its flexibility, speed of changing functions, and intuitive flow between functions.

Contributed Educational Software on the HP3000

Elisabeth M. Craig, Ed.D.

Center of Excellence for Computer Applications
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, Tennessee 37403
615-755-4389

Abstract

The HP2000 system, discontinued some years ago, contained a rich library of educational software. The programs were contributed from a number of sources and were accompanied by documentation on each program plus test runs. When the HP2000 was phased out in 1982, Hewlett-Packard and the University of Tennessee at Chattanooga (UTC) jointly agreed that the latter should be given the support to convert these programs to HP3000 systems. This process was completed in 1984 and the approximately 400 programs have been available on the INTEREX Contributed Software Library since that time.

UTC decided several years ago to take the software one step further by linking the best or most commonly used programs through a menu. The programs are currently adapted for UTC's classic Series 44/58 and the new Series 950 Precision Architecture system. The menu manages approximately 374 separate BASIC programs pertaining to seventeen areas of study (such as mathematics, chemistry, engineering, social studies, etc.).

This paper describes this educational software as well the organization of the menu, and some of the most popular programs. The paper also provides information on the supporting documentation. It is hoped that this software will be the beginning of a library for educational users of the HP3000 systems. Such a library would not only benefit these users, but would provide a valuable resource for others since many of the programs are for generic applications such as linear programming, data handling, statistics and mathematics (there are also a few games).

Introduction

From 1974 to 1982 The University of Tennessee at Chattanooga (UTC) had a HP2000 system which was made available, without charge to faculty, staff and students. Dr. Lloyd Davis became Director of Academic Computing when that department was created in 1974.

As part of the HP2000 system, a contributed library of over 400 programs was included in the software that UTC received. The programs were written in BASIC and covered nine areas:

Data Handling
Testing, Debugging and Programming Aids
Math and Numerical Analysis
Probability and Statistics
Scientific and Engineering Applications
Management Sciences and Operations Research
Business and Manufacturing Applications
Education
Miscellaneous (Games)

The documentation consisted of five volumes. There was a minimum of two pages per program, showing a test run and providing general instructions about each program. HP updated this BASIC Library, as it was called, every six months. The programs selected for inclusion were deemed to have wide-spread applications and were to be used with HP time-sharing systems.

When the HP2000 minicomputer was dropped from maintenance by Hewlett-Packard, UTC's Academic Computing department suggested to HP that the educational programs should be adapted for use on the 'classic' HP3000 system (such as the Series 44 or 58). This software had been an important contribution to educational institutions and as President of SIGED (Special Interest Group for Educational Users), Dr. Davis had received numerous calls from educational users regarding this software. HP agreed to Dr. Davis's idea and in 1983 work was begun on this project.

Some of the HP2000 were not highly functional or did not have a wide appeal; these programs were culled from the orginal HP2000 contributed library. Software was developed at UTC to convert the BASIC Library from an HP2000 format to one that was compatible with the HP3000. Since the file structure on the HP3000 was different from that of the HP2000, some programs were rejected because they were too file dependent and would not work well in the HP3000 environment. Others were rejected because they designed for handling peripherals (such as paper tape) which were no longer in common use. By mid-1984 the project had been completed though each of the 374 programs and/or systems was still in a stand-alone format.

Since documentation is an important aspect of any contributed library, the pages relating to the HP2000 programs were reproduced in microfiche format. This method proved to be an inexpensive way of recording the docmentation. Copies of the microfische are stored at The University of Tennessee at Chattanooga and several copies were sent to Hewlett-Packard. One copy of the original document is

still stored in the Center for Computer Applications (CECA); this copy is used for testing purposes since it is easier to read than microfiche.

One problem in using the programs on the HP3000 was that they were not linked and were located in different groups and accounts. We decided to produce a BASIC program which would provide such connection and make the programs easier to access. To make the process 'user-friendly' the BASIC program provided a series of screens which would allow users to go directly to a specific program or to browse through all the offerings. The organization and operation of this program, or menu, are described in the next section.

Overview of Menu Program

'MENU' is a large menu-driven BASIC program which links the approximately 374 separate programs. Because of the large number of programs, the original nine categories or topics in the HP2000 organization were increased to seventeen. These topics were arranged in alphabetical order.

The menu program is organized to display a main screen (shown below) which provides access to submenus. The main menu lists the 17 broad subject areas and users are requested to make a selection. Based on the number which the user enters the program links to the appropriate submenu screen.

****

MAIN MENU

1. BUSINESS & MANUFACTURING APPLICATIONS
2. DATA HANDLING
3. ECONOMICS
4. EDUCATIONAL ADMINISTRATION
5. ENGINEERING
6. ENGLISH
7. FINE ARTS
8. FOREIGN LANGUAGES
9. HEALTH SCIENCES
10. HISTORY
11. MANAGEMENT SCIENCES & OPERATIONS RESEARCH
12. MATH & NUMERICAL ANALYSIS
13. MATHEMATICS (EDUCATION)
14. PROBABILITY & STATISTICS
15. PROGRAMMING & COMPUTER SCIENCE
16. SCIENCE (EDUCATION)
17. SOCIAL SCIENCE
18. *** EXIT PROGRAM

In some categories, there were so many programs (mathematics is a good example) that the category screen had to be broken into several sub-levels each accessed by a screen. In this situation, the usual method of determining the level was the degree of difficulty of the material presented in the individual program (e.g. beginning, intermediate, advanced). All subscreens show not only the name of the individual programs, but also provide a brief description of that program.

There are a total of 51 screens and each has one or more control options listed at the bottom. The control options are identified by three asterisks which preceded them. Typical control options included the ability to exit the menu program, to return to the main menu or to link to submenus (the latter for categories containing several levels of difficulty).

Program Refinements

Because the programs orginally were written as stand alones, and were developed in the early 1970s, they contained features which required changes, such as the following:

1. Programs often contained more than one STOP statement in which case the program would not chain back to the menu. These statements were replaced with GOTOs which passed control to a single END statement.

2. Some programs printed output then immediately chained back to the menu without giving the user time to view the output. This problem was corrected by inserting code at the end of the program which asked the user if he/she wanted to run the program again or read the output before continuing the program.

3. The selected program often started printing on the screen while the menu was still displayed. This problem was overcome by inserting a PRINT statement to clear the screen. For example:

```
390 PRINT '27"H"'27"J"
```

4. The HP2000 programs were written at a time when terminals handled only upper case characters. Most programs had to be altered so that INPUT statements would accept either upper or lower case letters and then translate them to upper case for comparsions within the program.

5. Many programs were document dependent and expected the user to have access to the pages of test runs. In converting programs, it was deemed important to overcome this dependence so PRINT statements were added to give the user instructions on what information to enter. (Since the work area

for BASIC programs was more extensive than that on the HP2000 it was possible to add material without exceeding the limits.)

The testing of these programs was extensive. Each program had to be checked against the original run to ensure that it still ran correctly and then tested to ensure chaining from the menu program worked correctly. Even at the time of writing this paper, the process is continuing since all software, including the menu, were migrated to a Series 950 in late 1988.

Since the programs now contain internal instructions to the user, the amount of documentation is greatly reduced. A manual was produced which contains a description of each program, organized by category. The manual also contains indexes for cross-ferencing the programs and categories.

To provide some idea of the diversity of the software, some of the more widely-used programs are listed below:

Name	Description
CPATH	Critical Path Evaluation
LINPRO	Linear Programming Model
ANNUIT	Annuity Analysis
MKBUY	Make-Buy Decision Analysis
TUTOR	Series of lessons on BASIC language
PRINT	Large letters generator (for banners etc.)
CRVFT	Least Squares Curve-fitting
SIMPLX	Solves Linear Problems
ANOVA	Factorial Analysis of Variance
CURFIT	Performs Least-squares fit
SUNSET	Sunrise-Sunset Predictor
CYCLES	Plots biorhythms

The creation of the MENU system has created a worthwhile resource for UTC users; it is anticipated that this software will be an enhancement to classroom instruction. For example, computer science majors can use the TUTOR series to learn the BASIC programming language which presently is not included in the curriculum for those majors. In addition, it is hoped that these programs will be the basis of a contributed library available to all educational institutions which have an HP3000.

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

NOTES

HOW TO DEVELOP AN AUTOMATED PROCUREMENT SYSTEM

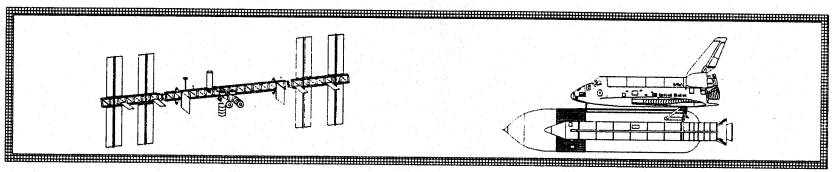
J. L. HILL

LOCKHEED ENGINEERING & SCIENCES CO

2400 NASA ROAD 1

HOUSTON, TEXAS 77058

713 333-7257



PROJECT RECAP

HOW TO DEVELOP AN AUTOMATED PROCUREMENT SYSTEM
5815 - 2

PROJECT RECAP
CHARTER

- PROCUREMENT DATA TEAM CHARTER - 4/20/88
- TEAM MEMBERS
  - GERRY ALLEN
  - BILL HERREN
  - DICK PRINGLE
  - SCOTT SEALING
- OBJECTIVES
  - STUDY AND RECOMMEND APPROACH

PROJECT RECAP
RECOMMENDATIONS

- PROCUREMENT DATA TEAM RECOMMENDATIONS
  - EVALUATE LSOC PROCUREMENT SYSTEM
  - CONTINUE 3RD PARTY REVIEW
  - RELUCTANCE TO 3RD PARTY SOFTWARE
    - CURRENT CRITICAL OPERATIONS
    - HIGH RISK
    - PRODUCTIVITY RECOVERY

PROJECT RECAP
APPROACH

- ASSIGN/HIRE FULL TIME ANALYST
- INITIATE IWT TO LSOC FOR SYSTEM EVALUATION
- INTERVIEW LESC P/O, RECEIVING AND A/P AREAS
- PRESENT LESC REQUIREMENT OVERVIEWS
- EVALUATE LSOC SYSTEM TO LESC REQUIREMENTS
- IDENTIFY THIRD PARTY SOFTWARE CANDIDATES
- DETERMINE HARDWARE REQUIREMENTS
- DIRECTION DECISION
- DEVELOP PROJECT CONTROL DOCUMENT
- IMPLEMENTATION PLAN

L E S C REQUIREMENTS

HOW TO DEVELOP AN AUTOMATED PROCUREMENT SYSTEM
5815 - 6

MODULES

- PROCUREMENT AUTOMATION REQUIREMENT
  - PURCHASE REQUEST
  - PURCHASE ORDER
  - MATERIAL RECEIVING
  - ACCOUNTS PAYABLE

A P P R O A C H

- INTERVIEW LESC PERSONNEL
- OBSERVE DAILY OPERATION
- EVALUATE WORKSHEETS
- OVERVIEW OF CURRENT PROCESS
- IDENTIFY BOTTLENECKS
- DEVELOP AUTOMATION REQUIREMENTS
- REVIEW AUTOMATION ALTERNATIVES

PURCHASE REQUEST REQUIREMENTS

	MATRIX
• READILY AVAILABLE TO USER	
- ORIGINATE SCPR/POR	N
- REVIEW SCPR/POR	P
- APPROVE SCRP/POR	N
• REAL-TIME UPDATING	
- SCPR/POR RECORDS	Y
- P/O RECORDS	Y
- MANAGER'S CHECKBOOK RECORDS	N
• ABILITY TO PRODUCE DOCUMENTS	
- SCPR/POR	N
- DEBIT MEMO'S	N
• SCPR PRIORITY SYSTEM	Y

PURCHASE REQUEST REQUIREMENTS - 2

	MATRIX
• ABILITY TO PROCESS MULTIPLE RECORDS	
- MULTIPLE PWO'S ON AN SCPR	N
- P/O WITH MULTIPLE SCPR'S	Y
- MULTIPLE SCPR/PWO COMBINED	N
- MULTIPLE P/O'S FROM AN SCPR	Y
• CURRENT INFORMATION AVAILABLE	
- SCPR/POR	Y
- HISTORY FOR 2 YEARS	Y
- SPEAR SYSTEM	N
- VENDOR MASTER-STORE RPT INFO	Y
▪ $ TOTAL OF ORDERS	Y
▪ DATE LAST ORDER	Y
▪ OWNER CATEGORY	Y

PURCHASE REQUEST REQUIREMENTS - 3

	MATRIX
• SR & QA CONTROLLED STORAGE INDICATOR	P
• SYSTEM TO INCLUDE SCPR/POR	Y
• BLANKET PURCHASE ORDER/AGREEMENTS	Y
• DEFECTIVE PRODUCT/DELAYED DLVY ALERT	N
• TOTAL ELECTRONIC APPROVAL	N

PURCHASE ORDER REQUIREMENTS

	MATRIX
• READILY AVAILABLE TO USER	Y
• REAL-TIME UPDATING	
- P/O INFORMATION	Y
• SYSTEM INTERFACES	
- HQMIS	N
- IPMIS	Y
• CURRENT INFORMATION AVAILABLE ON-LINE	
- SCPR/POR	Y
- PURCHASE ORDER	Y
- MATERIAL RECEIVING	Y
- ACCOUNTS PAYABLE	Y
- BUYER/SCA ACTIVITY	P
- VENDOR MASTER	Y
- HISTORY	Y
• ABILITY TO PRINT PURCHASE ORDERS	Y
• ABILITY TO PROCESS MULTIPLE RECORDS	Y

MATERIAL RECEIVING REQUIREMENTS

	MATRIX
• INTERACTIVE SYSTEM	Y
• ENTER INFORMATION REAL-TIME	Y
• EITHER OF TWO RECEIVING DOCUMENTS - DIRECT RECEIVING DOCUMENT NO P.O. REQUIRED - LESC RECEIVING DOCUMENT P.O. REQUIRED	Y
• ON-LINE CAPABILITIES - INQUIRE ON PRIOR SHIPMENT - MODIFY INFORMATION ON PRIOR SHIPMENT - REPRINT DOCUMENT FOR PRIOR SHIPMENT - PRINT DOCUMENT FOR CURRENT SHIPMENT	Y
• ONE(1) PART RECEIVING DOCUMENT	Y
• BATCH REPORTS DAILY/WEEKLY/MONTHLY	Y
• MICROFILM & STORE DOCUMENTS	Y

ACCOUNTS PAYABLE REQUIREMENTS

	MATRIX
• CURRENT INFORMATION AVAILABLE - REAL-TIME UPDATING OF RECORDS - INTEGRATED SUB-SYSTEMS - ON-LINE INQUIRES - EASY ACCESS TO HISTORICAL DATA	Y
• PAY ON RECEIPT OF INVOICE - VERIFY P.O. AND RECEIPT ON-LINE	Y
• PRINT EMPLOYEE TRAVEL ADVANCE CHECKS AT CASHIERS DESK	Y
• PRINT RECEIVING DOCUMENT IN A/P - MAINTENANCE/TIME & MATERIAL	P
• VENDOR MASTER AVAILABLE ON-LINE	Y

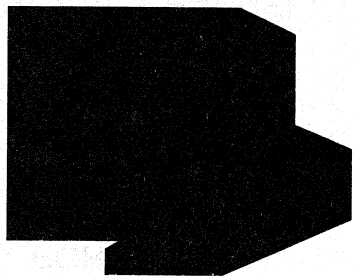
AUTOMATION ALTERNATIVES

- L S O C SOFTWARE
  - INSTALLATION TURNAROUND
  - SIMILAR REPORTING REQUIREMENTS
  - DEVELOPED FROM HQMIS SOFTWARE
  - ON-LINE, REAL-TIME SOFTWARE
  - LESC PERSONNEL KNOW LSOC SYSTEM
- THIRD PARTY SOFTWARE
  - NOT FULLY EVALUATED/LESC REQUIREMENTS
  - SMITH, DENNIS & GAYLORD
    USED BY L T O C
  - M C B A HAS 24% OF HP MARKET

LSOC PROCUREMENT SYSTEM
REQUIREMENTS EVALUATION

- ANALYZE ESCAPS REQUIREMENTS TO LSOC SYSTEM
  - RESULTS OF P/O, M/R, A/P COMPARISON
    - 98% MET OR MET WITH MINOR MODIFICATION
    - 2% WILL NOT MEET REQUIREMENTS
  - RESULTS OF P/R COMPARISON
    - 60% MET OR MET WITH MINOR MODIFICATION
    - 40% WILL NOT MEET REQUIREMENTS

E S C A P S
PROJECT STATUS



HOW TO DEVELP AN AUTOMATED PROCUREMENT SYSTEM
5815 - 17

ENVIRONMENT COST
LSOC/THIRD PARTY

• ONE(1) 625 MB DISK DRIVE	$16,000
• 24 ADDITIONAL PORTS DEPENDING ON WORKSTATIONS ON-LINE/SHARE	$13,000
• IMB (INTERMODULE BUSBOARD)	$7,000
✓ 2 GIC'S (GENERAL INTERFACE CHANNEL)	
✓ COBOL COMPILER-RIGHT/COPY	

T O T A L	$36,000

THIRD PARTY
ALTERNATIVE

• PROCUREMENT SYSTEM/LTOC	$100,000
• INSTALLATION LTOC VERSION	TBD
• MODIFICATIONS	TBD
T O T A L	$100,000

LSOC
ALTERNATIVE

✓ INSTALL LSOC PR/PO/MR/AP VERSION W/O MODS

- E-TEAM MODIFICATIONS 650 HRS $9,600 (SEE ATTACHED 1)
- INTERFACES/MODIFICATIONS REQUIRED LESC ENVIRONMENT 1450 HRS (SEE ATTACHED 2) $23,200

T O T A L $32,800

- Note: The above modifications do not include any modifications to the existing SCPR system.

COST SUMMARY

• THIRD PARTY SOFTWARE	
- VENDOR T B D ??	$100,000
- ENVIRONMENT COST	$25,000
- INSTALL/MODIFICATIONS	TBD
T O T A L	$125,000
• L S O C SOFTWARE	
- LSOC	$20,000
- LESC INSTALLATION 200 HRS	$3,200
- ENVIRONMENT COST	$25,000
- MODIFICATIONS 2,100 HRS	$33,600
T O T A L	$81,800

E S C A P S
PLANNED EVENTS

- LSOC BUILD ESC SYSTEM AT LSOC; SEND TO LESC ON 11/4/88
- INSTALL LSOC SYSTEM AS IS
- MINIMAL MODIFICATION
- SELECTED USER TRAINING
- USED BY EACH DEPARTMENT
- USERS TO ACCEPT/REJECT
  - ACCEPT PERMANENT/INTERIM
- MAJOR MODIFICATIONS/THIRD PARTY

Symmetry in Graphic Representations as an Aid to Database Design

Mark W. da Silva
University of Hawai'i Cancer Research Center
1236 Lauhala Street #402
Honolulu, Hawai'i 96813
(808) 521-0054

Abstract

The importance of an optimal database design cannot be overstated. Designs that efficiently model critical user views of the data will result in tremendous productivity benefits for the Data Processing component as well as other departments in the organization.

Using the standard graphic representations for IMAGE databases, this paper explores the use of symmetry as a design aid. It is shown that clear, understandable graphic depictions of a data model can stimulate the thought processes of the designer and result in new designs that better model the data.

First, the graphic representation for a sample database is presented, then modified using a nine-step process. These modifications result in a series of new graphic designs, each of which evolves from visual cues present in the diagram of its predecessor. Finally, suggestions are provided for the IMAGE database designer to utilize the described techniques.

Introduction

Ideally, the database designer will produce well-documented specifications for three models of the data: conceptual, logical, and physical.

The conceptual specification (end users' view of the organization) is usually presented in terms of entities and relationships.

The logical model follows. This is a mapping of the conceptual design onto the best-suited database model available in the application environment (hierarchical, network, or relational). The logical model is independent of any particular hardware or software systems.

The physical model is created using the data definition language specific to the chosen computing environment. For our purposes, this is the IMAGE schema data base language.

This paper concerns itself with the initial definition and re-design of the physical model for the IMAGE database management system.

The primary tool used to represent this definition will be the standard IMAGE graphic representations: triangles for master datasets, and trapezoids for detail datasets (see figure 1). A connecting arrow represents the path between a master and its associated detail dataset(s). This means there is a search item for which a chain of detail entries may exist.

The key idea behind this paper is simply that clear diagrams enable the designer to think clearly. Conversely, confused disorderly diagrams impede the thought processes.

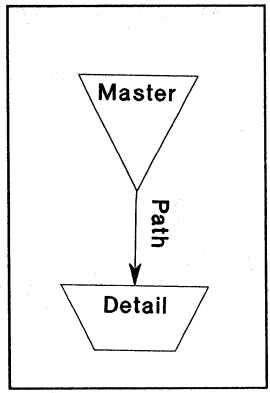


Figure 1 Standard IMAGE Graphic Representations

The Sample Database

The sample database is taken from the example in HP's Turbo IMAGE DBMS Reference Manual (see bibliography). This database represents a simplified order/entry system for a fictional business. The system tracks sales of products to customers and maintains an inventory and list of suppliers of these products.

The sample database consists of four master, and two detail datasets as shown in figure 2. These diagrams and their subsequent revisions are presented solely as a means of illustrating a process. The database design, therefore, should not be construed as representing an ideal model of an order/entry system.

The CUSTOMER master contains the customer/account number, and other attributes unique to a given customer. The PRODUCT master contains stock/product numbers and descriptions for each item sold by the company. The SUPPLIER master contains information about the suppliers of each of the aforementioned items. The DATE master simply tracks each date an event occurs, thereby enabling the retrieval of chains of detail entries by date.

The SALES detail contains line-item information about a single product ordered by a customer (e.g. product number, price, etc.). The INVENTORY detail contains information for a given product such as quantity on hand, the supplier, actual cost, etc.

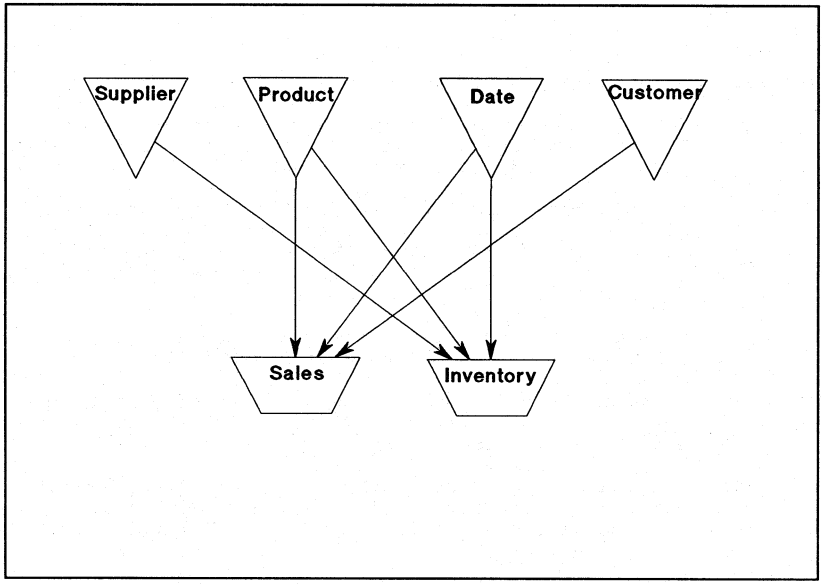


Figure 2 Initial Diagram of the Orders Database

The above diagram is confused deliberately to better portray the revision process. The imposed disorder, however, is not beyond the realm of real-world possibility. For example, the author inherited documentation produced by a now-defunct consulting firm. One of their diagrams depicted a portion of an IMAGE database that consisted of six master datasets, four detail datasets, and eleven paths. All but one of the paths in their diagrams crossed one or more other paths.

The Revision Process

To increase both the usefulness and readability of the graphic representation, a nine-step revision process is suggested. Using this process, we will both add utility and complexity to the sample database, and at the same time increase the clarity and functionality of the diagram.

The nine suggested steps for revision are:

1) Untangle the mess. If you've inherited or created a Medusa, comb out her hair. As much as possible, keep paths isolated (no crossing lines).

2) Add all the paths and datasets that might have even the slightest relevance to your ultimate data model. Don't worry about overdoing it. You will edit and cut in later steps. This is a quick-and-dirty brainstorming opportunity.

3) Untangle it once more. This time keep in mind the relative importance of each path and dataset.

4) Eliminate superfluous datasets. Carefully study each dataset in your diagram. Consider its relevance to the data model you are trying to represent.

5) Eliminate superfluous paths. Consider the possible methods of data retrieval. Decide which detail entries will require chained retrieval, and eliminate any unnecessary paths.

6) Impose symmetry on the diagram. Keeping in mind the relative importance of each dataset, try to center those you consider most significant. The number of paths allocated to a dataset may give you a clue to its relative importance in the model.

7) Consolidate logically related portions of the design. Locate datasets with strong relationships next to each other. Consider diagramming datasets with weak or no relationships separately, or dropping them completely from your design.

8) Evaluate the asymmetric portions of the diagram. Sometimes these sections reveal the lack of a potentially useful dataset or path; or indicate that another organization of the diagram will increase its clarity.

9) Start over. Like the system development life-cycle, this can be a never-ending process. Refine your design until you're completely satisfied.

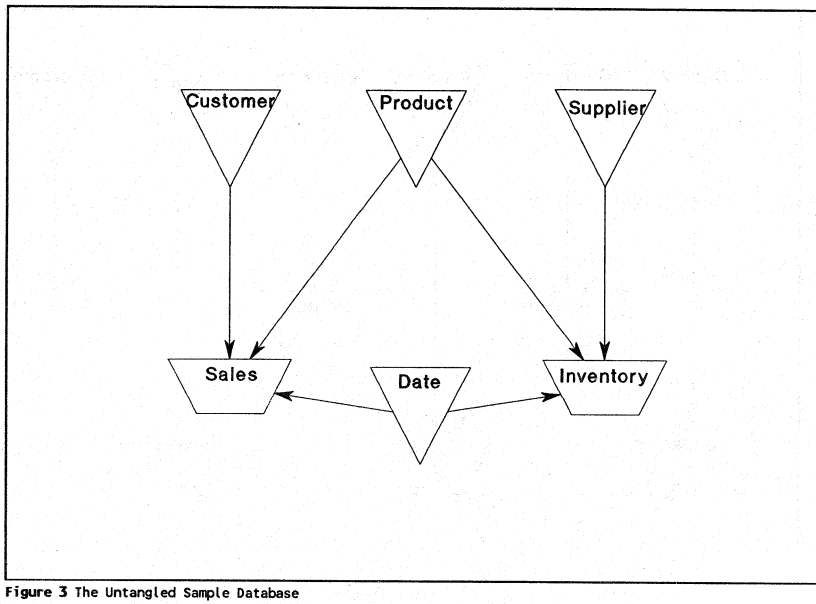


Figure 3 The Untangled Sample Database

Figure 3 shows the sample database after the initial untangling of the paths. Notice that the relationships are much more apparent. This representation is, by the way, identical to that given in the previously mentioned Turbo IMAGE reference.

Once some initial clarity has been established, the second step may be applied. Adding extra paths and datasets will initially complicate matters, but this will be corrected later.

The first revised and enhanced diagram is given in figure 4. Note that an attempt has been made even at this early stage to maintain some degree of order and symmetry.

In figure 4, the sample database has been enhanced two ways: First, the capability to retrieve customers, products, and suppliers by name in addition to code number has been added. Second, a method of grouping several individual sales into an order has been provided.

Retrieval by name is accomplished by placing the attributes of each of the original three master datasets into three new detail datasets: CUSTOMER, PRODUCT, and SUPPLIER.

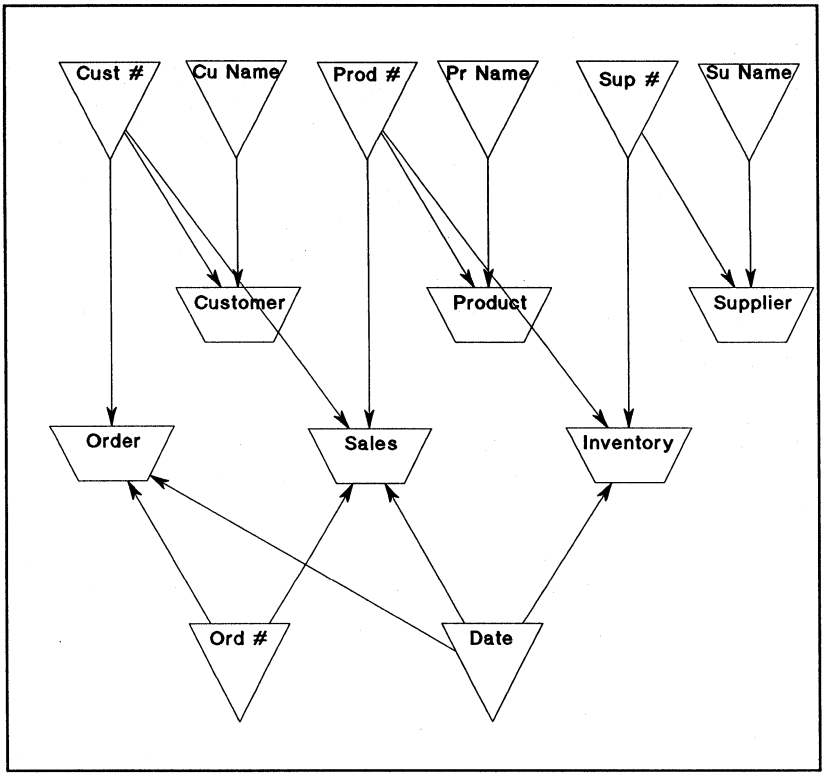


Figure 4 First Revision of the Enhanced Sample Database

The search item numbers are placed in three new master datasets CUST #, PROD #, and SUP #. The actual name retrieval is accomplished by adding the CU NAME, PR NAME, and SU NAME datasets. These will contain a word-indexed search item (e.g. a SOUNDEX string). A chain in the detail dataset may then be searched when a name is given to the system.

The ability to group a customer's sales into order transactions is accomplished by adding a master dataset, ORD #, which will contain a unique number assigned to each order by the system. A detail set called ORDER has also been added to contain data that applies to the entire order (e.g. a reference to the customer's purchase order, the customer number, etc.).

A path was added from the CUST # master to the ORDER detail so that an order or orders may be easily retrieved for a given customer. Another path from the ORD # master enables retrieval by order number. Finally, entries in the SALES detail are chained together by a path from the ORD # master.

Note that the new datasets do not all have valid IMAGE names. These names were selected for illustrative purposes only.

After the necessary additions have been made to the new design, another untangling of the diagram may occur (step 3). This time, however, consider the relationships between, and importance of, each dataset and path in the design. Figure 5 illustrates this revision. Note that groupings similar to the simple design in figure 3 are beginning to reappear. Customers, products, and suppliers each have their own section of the diagram.

Because of the simplicity of this sample database, steps 5 through 7 have been applied and illustrated in a single diagram: figure 6.

Note that the CUSTOMER, PRODUCT, and SUPPLIER details are centered between their respectively associated master datasets. This enables one to mentally isolate and condense these nine datasets into the three entities they represent.

Also notice the prominence of the SALES detail in the diagram. This datasets has the most paths and will probably be the most frequently accessed.

Finally, notice the lack of symmetry in the diagram. This may suggest further enhancements to the design (see steps 8 and 9). Perhaps a payables system to the supplier should be added, or a more detailed tracking of orders placed with suppliers would be beneficial.

Suggested Tools and Techniques

Nothing more than a pencil and some paper are required to achieve the re-design objectives. There are, however, an unlimited number of tools and techniques you may find useful. These can range from an expanse of wet sand at the beach to sophisticated CAD programs.

The logical extension to the idea of pencil and paper is the blackboard. If you have a magnetic board, it may be helpful to print out a >FORM SETS with QUERY. Cut out the individual datasets and stick them on the board, paths can then be drawn wherever necessary. The cut-outs will function just as well on your desk-top.

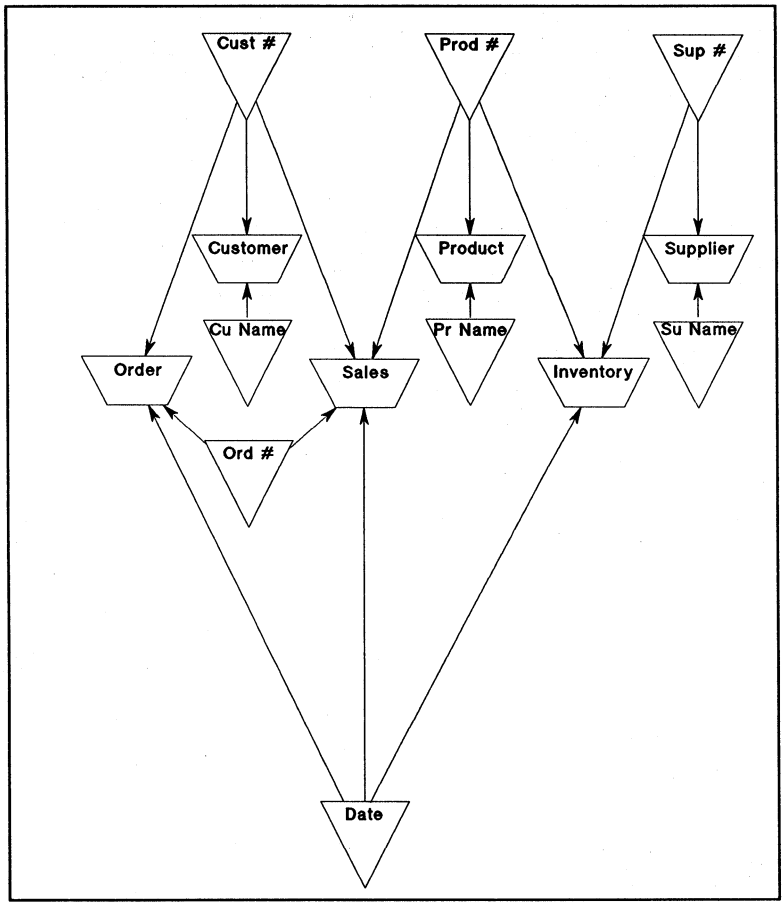


Figure 5 The Untangled Revision

Since the diagram ultimately ends up on paper in your documentation, your thinking is limited to two-dimensional representations. A child's toy (like the one with those colored wooden pieces) can be used to extend the modelling process to three dimensions. Please note that this technique is best utilized when upper management is not present.

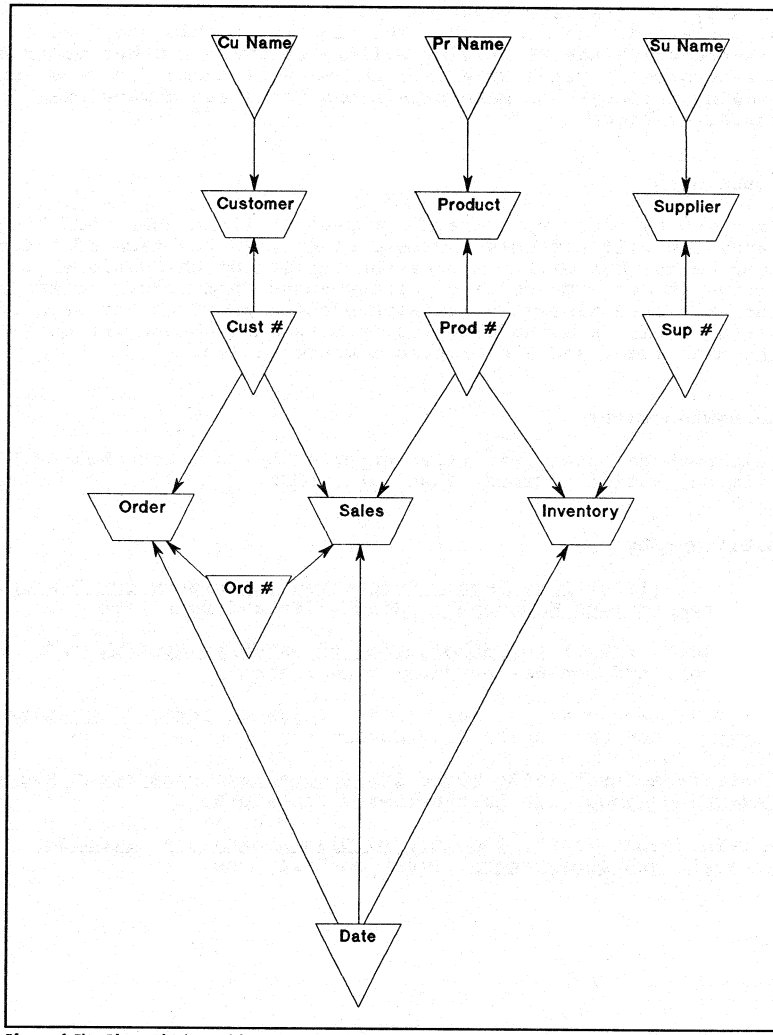


Figure 6 The First Final Graphic Representation

Finally, PC graphics programs can effectively be used to draw and manipulate design graphics. The figures in this paper were created using the HP Drawing Gallery software. Other programs to create similar graphics exist; and many of these programs--unlike Drawing Gallery-- do make provisions for three dimensional representations.

Conclusion

Imposing symmetry upon graphic representations when modelling databases will not only increase clarity in the face of added complexity, but will increase the ability of the designer to arrive at correct, concise, easily-understood models of the data. The suggested nine-step revision process provides the database designer with a methodology to improve and enhance the quality of the data model and its graphic representation.

Acknowledgement

This work has been partially supported by contract #N01-CN-55424 from the National Cancer Institute, DHHS.

Bibliography

Atre, S. (1980) Data Base: Structured Techniques for Design, Performance, and Management, John Wiley and Sons, Inc.

Date, C.J. (1977) An Introduction to Database Systems (2nd Edition), Addison-Wesley Publishing Company.

Fong, Elizabeth N., et. al. (1985) Guide on Logical Database Design, NBS/SP-500/122 U.S. Government Printing Office.

Hewlett-Packard (1985) Turbo IMAGE Data Base Management System Reference Manual, HP part number 32215-90050.

Martin, James (1987) Recommended Diagramming Standards for Analysts and Programmers, Prentice-Hall, Inc.

Migration Made Easy

Victoria Shoemaker
Taurus Software, Inc.
770 Welch Road Suite 3A
Palo Alto, CA 94304
(415) 853-6893

So you have purchased a Hewlett-Packard Precision Architechture machine (HPPA/Spectrum/Series 900). Congratulations! So you are going to migrate. What does migration mean, exactly? How does one migrate anyhow? Oh...you haven't thought that all the way through yet. Hmm, maybe you should start planning your migration now.

This paper will take you through the steps you will need to take to make your migration effort to the HPPA machines complete and successful. There are seven steps:

- Education
- Analysis of Existing Applications
- Developing of a Migration Plan
- MPE/V Conversions
- Installation of HPPA machines
- Compatibility Mode Operation
- Migration to Native Mode Operation

Each of these steps will be discussed in detail.

Education

The HPPA machine is completely different from the classic HP3000. There are very few similarities in their hardware architectures. HP has gone to great pains to ensure that old job streams and programs will run on the HPPA machines, but all of the classic HP3000 hardware emulation is done via software on the HPPA machine. Underneath the software are two different machines. For all of the differences between the two computers, they might have been put out by two different manufacturers.

The point I am trying to make, is that what makes the classic HP3000 hum, does not make the HPPA machine hum. In order to take advantage of the performance gains projected by HP, you are going to have to migrate your software to native mode. All the knowledge you have gained about stacks, PCBs, MPE tables and such are of little help. You must learn about this new machine from scratch. So, step one, get educated.

There are a number of different ways to learn about the HPPA machines: user groups, manuals, FASTLANE consulting, HP classes and books. Some of the topics you are going to want to make yourself familiar with are as follows:

- Native Mode vs Compatibility Mode. Actually this is a very simple lesson. Native mode operation means that it is using the native instruction set (RISC) of the HPPA machine. Compatibility mode means that software is emulating the native mode operation of a classic or MPE/V based machine. In order to take advantage of the performance gains of the HPPA machines over the classic HP3000s, you are going to have to convert your applications to native mode.

- System Management. The HPPA machines don't use SYSDUMP. They have a new configuration manager called SYSGEN. It is very different from SYSDUMP. The HPPA machines don't use OPT or any of the other performance monitoring tools you are used to. The HPPA machines handle terminal connection differently than you are used to.

- Operational Changes. From the simplest thing, bringing up the machine to private volumes, the HPPA machine is different. You are going to have to learn how to do everything over again.

- New/Changed Commands. There a number of new commands in MPE which is now called MPE XL. The command resolution and syntax have changed dramatically. All the changes are very positive. You are going to want to take advantages of the positive changes immediately.

- Data Migration. The classic HP3000 is a 16 bit machine. The HPPA machine is a 32 bit machine. This spells data migration. You need to know how this will affect your data files and program's internal data structures.

- Programming Environment. There are a number of new programming tools available including a symbolic debugger. You are going to want familiarize yourself with its features before you begin conversion to native mode.

- Programming Languages Issues. Each programming language has its own set of challenges when moving to native mode that you are going to have to be familiar with before you begin. We will discuss some of those issues later in this paper.

- New/Changed Intrinsics. A number of intrinsics have changed. In addition, a number of new intrinsics have been added to take advantage of some of the new features in MPE XL. I am sure that you are going to want to take advantage of these when converting to native mode.

As you can see the list is fairly impressive of some of the basic things you are going to need to know before you begin your migration. So, how do you go about this? Easy, there are number of different ways, some even free!

1. User's Groups. There are a number of different speakers talking at this conference alone about migration issues. Attend their talks. Read their papers. Ask questions. Some of the users have already gone through migration experiences. Learn from their successes and mistakes. HP has released early bird sites and FASTSTART companies from their confidentiality agreements. These companies are a wealth of knowledge.

2. Manuals. HP has published a number of manuals to help companies have a successful migration experience. Some of the titles are as follows: *Migration Process Guide*, *Programmer's Skills Migration Guide*, and *COBOL Migration Guide*. I found the first two listed very helpful in getting an overall picture of what is involved in migration. Remember when you read these manuals, read them as Alfredo Rego recommends, like love letters.

3. Consulting. Both HP and independent consultants are offering migration assistance. I have not used either one. HP's consulting is called FASTLANE consulting. It includes a one day class taught at your site and a migration planning meeting. Independent consulting seems to be varied. As with any consulting, make sure you are dealing with experts.

4. Classes. HP offers two classes to help you in migration: system management, and programmer's class. The system management class is three days long and deals with the new SYSGEN and other system utility software. The programmer's class is seven days long and is for programmers who plan to assist in the migration effort. It covers language specific information and the symbolic debugger.

5. Books. There is a book covering Spectrum issues called *Beyond RISC! An Essential Guide To Hewlett-Packard Precision Architecture*. I am sure it is worth reading. It is available through Software Research Northwest, Inc.

Application Analysis

Once you are educated, you are now ready to review your applications for any known migration issues. Some obvious gotchas are:

- FORTRAN. There is no FORTRAN/66 native mode compiler on the HPPA machines. You must first convert your FORTRAN/66 to FORTRAN/77. I don't code in FORTRAN, but from what I understand this is not a trivial task.
- COBOL. All COBOL/66 programs need to be converted to COBOL/II.
- BASIC. There is no BASIC/3000 native mode compiler on the HPPA machines. Business Basic has a native mode compiler.
- SPL. There is no SPL native mode compiler on HPPA. A third-party compiler, SPLASH compiles SPL to native mode.
- Privilege Mode. Remember when HP said "Don't use privilege mode." Well now you know why. MPE XL is a completely different operating system internally from MPE/V. This means there is 50-50 chance that your privileged mode programs will need to be changed before they will run in native mode.
- Floating Point. The HPPA machines uses IEEE floating point arithmetic. The classic HP3000s used their own brand of floating point arithmetic. If you use real numbers this could be an issue for you. For your information, COBOL does not use floating point, PASCAL and SPL have floating point facilities, and FORTRAN uses floating point extensively.

Other than these obvious issues, there a number of specific language and intrinsic related issues. Luckily for you, HP has put together a migration tool package. The price is right, $100. The package includes two programs: RTM and OCA.

RTM (Run Time Monitor) is a utility program designed to help you identify areas within your MPE/V programs that could be a problem when ported to MPE XL. It logs calls to MPE/V intrinsics which have been changed or are not supported on MPE XL as they happen. The logging is controlled by RTMSYS.PUB.SYS. The logging reports are printed using RTMREP.PUB.SYS.

OCA (Object Code Analyzer) is a utility program which scans program and SL files for potential problems. OCA scans a program for "problem" intrinsic calls. The advantage to using OCA is that you are able to obtain immediate results without the need for logging. OCA may not be as accurate as RTM because it cannot always tell what parameters an intrinsic is called with.

Both of these migration tools run on MPE/V based systems. You can run them before beginning migration. Don't forget to analyze both internally written applications and third-party solutions. Your migration effort will be affected by both types of applications.

At the end of this phase of migration, you should have a list of applications and any potential migration problems. With this list, you will be able to begin the next phase of migration, planning.

Planning

Planning is the single most important element of your migration. Regardless of how many applications run in your shop, how many machines you have, how much third party software you run – your migration's success depends on how well you have planned it out. Spend the time to plan. It will pay off.

The first thing you will need to decide is which of your applications are worth migrating to native mode operation. Some factors that may enter into your decision are:

- How often does this application run? If the application only runs once a year, does it really need the performance gains provided by converting it to native mode?

- How much trouble will it be to convert this application to native mode? If you have a frequently called FORTRAN/66 routine, is it really worth converting it first to FORTRAN/77 and then to native mode?

- Do you have the source code for this application? Obviously for third-party solutions, you probably are going to have rely on the vendor for native mode solutions. For contributed routines, you may have to let them run in compatibility mode for lack of a better solution.

- Is this a high volume application? If the application processes a great deal of transactions, it may be worth converting for the increase in transaction throughput.

- Do you have to maintain compatibility with MPE/V based machines? If this is true, then you may not want to take advantage of the native mode compilers. Native mode object code will not run on MPE/V based machines. There is a middle ground for such applications. OCT (Object Code Translator) converts an MPE/V object module and adds native mode instructions to the end of the program file. This way the program can run on both MPE/V and MPE XL programs. OCTed programs do not run quite as fast as native mode programs, but it is a good compromise for programs which need to run on both types of systems.

Once you have determined which of your applications will be migrated to native mode, you must develop a migration plan for each application. HP can help with the development of this plan through their FASTLANE consulting.

Along with your application specific plans, you will need an overall strategy for handling privilege mode programs/routines. Some of your privilege mode programs and routines will have to be rewritten entirely. Some of the functions can easily be replaced with complimentary functions on the HPPA machines. Regardless of the situation, you must decide if you wish to migrate privilege mode programs to native mode or let them run in compatibility mode.

The last consideration is site planning. During your migration, there will be a time during which both your classic and HPPA machine will share the same

room. This means you need to prepare your computer room for two scenarios: parallel operation (HPPA and classic) and HPPA. During parallel operations, you will need power, air conditioning, and peripherals for two machines. You must plan the additional strains on your computer room. To help in your planning, if you were to just move your existing applications directly onto a HPPA machines, you would need an additional 20% disc space. Make sure you have enough disc ordered for your machine.

So to review, you are going to need plans for the following:

- Application specific migration
- Privilege mode strategy
- Site planning.

MPE/V Migration

Prior to receiving your HPPA machine, there are a number of tasks that can and should be completed on the MPE/V base machine. A checklist of these tasks follows.

1. Upgrade to the latest release of MPE/V. MPE XL's base release was UBdelta4. This is the starting point for MPE XL. It is best it you get on this release of MPE/V before migrating.

2. Get on the latest release of your programming languages. If you are using BASIC/3000 convert your programs to Business BASIC. If you are using FORTRAN/66 convert your programs to FORTRAN/77. If you are using COBOL/66 convert your programs to COBOL/II. If you are using SPL, either convert your programs to another language or buy SPLASH. For all compilers, get to the current version of the compiler.

3. Call INTRINSIC. For all system intrinsics, the new format of the call on MPE XL is CALL INTRINSIC. Change all COBOL programs to use this format.

4. Block Mode. If you developed your own block mode terminal routines, convert them to use the new standard block mode routines: VTURNON/OFF, VPRINTSCREEN, and VBLOCKREAD/WRITE.

5. FOPENs of LDEVs are not supported. The DTC does not assigned a fixed LDEV numbers for devices. Opening a specific LDEV may not produce the same result on MPE XL as it does on MPE/V. It is possible to assign fixed LDEV numbers, but this is not the default configuration.

6. UDC Conversion. You will want to change UDCs which have the same name as the new MPE XL commands. You will also be able to omit 70% of your system-wide logon UDCs which run programs in PUB.SYS because of the new implied run and HPPATH variable.

7. SYSDUMP does not exist. You will want to convert your job streams which use SYSDUMP to use STORE. The commands FULLBACKUP and PARTBACKUP are not supported on MPE XL.

8. Peripherals. Not all the peripherals that are supported on MPE/V based machines are supported on HPPA machines including paper tape, and cartridge tape drives. Get a list from your HP CE.

Installation

The awaited day finally arrives! Your HPPA machine finally arrives! Well, here is the bad news, you are probably going to need some help getting everything configured and setup. As you recall, SYSDUMP does not exist on the HPPA machines. It has been replaced by SYSGEN.

When migrating from MPE/V to MPE XL you will use DIRMIG to create your directory, accounting structure, user logging parameters, RIN table, and private volume information. SYSGEN is the device configuration dialogue. In addition, there is another utility, NMCONFIG, used to configure your LAN and DTCs.

Well, if you are like me, these are all new and it would be nice if someone was there to help you.

Now the good news, once your accounting structure and other internal structures are in place, you can just restore your applications and they will just run! Compatibility mode operation just works! So let's look at our next step compatibility mode operation.

Compatibility Mode

As you recall compatibility mode is the term HP has chosen to mean running a program that contains instructions for the classic HP3000. When a compatibility mode program runs on MPE XL, the instructions are actually emulated by the emulation software.

The purpose of running your programs in compatibility mode is to ensure that the operation of these programs on MPE XL produces the exact same results as it did on MPE/V. Almost all of programs work exactly the same.

Typically compatibility mode testing continues through one complete accounting cycle. During this testing, your programmers can begin familiarizing themselves with the new features of MPE XL and the feel of the HPPA machines.

Migration to Native Mode Operation

During this phase of your migration, you are going to implement the application migration plans. This is an appropriate time for your programmers to attend the programmer training course and to learn the new debugger. They will also need to familiarize themselves with switch-stubs, which is a technique for switching between native mode and compatibility modewithin a program.

Once the conversion to native mode is complete. You can address yourself to optimizing the performance of your programs. There are several things which can affect performance of your applications:

- Mixed mode applications. Mixed mode applications run slower than native mode applications because of time required to go through switch-stubs.
- OCT applications. Programs which have been run through OCT will not run as fast as those which have been recompiled using a native mode compiler. This is due to the literal translation of MPE/V instructions.
- Extra data segments. Extra data segments are an MPE/V data structure and should be converted to mapped files for extra performance.
- Use of KSAM/RIO/CIR/MSG files. These file structures are supported using compatibility mode file intrinsics only. Because the intrinsics are supported in compatibility mode only, they will be inherently slower than other file structures.
- Word alignment. You can expect increased performance for programs that align their internal data structures to 32 bit word boundaries.

As with the classic HP3000, we are going to have to experiment with what exactly makes this machine perform. I am sure that over the next few years we will all be learning about the performance tuning techniques for the HPPA machines.

Summary

The most important lesson from this paper is two-fold: get educated and plan your migration. I believe that if you learn as much as you can before you get started and then plan your migration, you cannot have anything but a successful migration. Good luck to you!

I want to express my gratitude to John Bria of Hewlett-Packard for his help in developing this paper. It is his talk that provided me with the basis for this paper. Without his help, this paper would still be blank.

Designing Large Databases with IMAGE

by
Eric Savage

Dynamic Information Systems Corporation
652 Bair Island Road, Suite 101
Redwood City, CA 94063
(415) 367-9696

Large IMAGE database applications can either perform well and meet the user's expectations, or they can cripple your HP-3000. It is possible to design and manage your applications so that the performance is satisfactory. This paper will explain what causes poor performance around large IMAGE databases, and how you can achieve acceptable performance.

For the purpose of this paper, a large database is defined as containing any dataset with more than 500,000 records. Databases can easily grow larger than this. My personal experience centered on a database with six million records. The severe impact to performance seems to start around the 500,000 record level.

What causes poor performance with large databases?

There is one major cause for poor performance with large databases -- accessing the database consumes too much I/O. There are three leading causes for this: a) an inefficient approach to retrievals, b) poor database design, and c) insufficient database maintenance. In all of these situations, steps can be taken to improve performance.

It is commonly agreed that I/O is the slowest performer on both the classic and Spectrum HP-3000 machines. It follows that if large databases consume enormous amounts of I/O, and I/O is the slowest performer, then this bottleneck is the source of most performance problems with large databases.

Another factor makes these problems worse. As a database grows, retrievals tend to take longer. While the increase in time is proportional to the increase in records, the same is not true for the user's patience. You can explain to a user that a report should take all day, but waiting all day is often beyond the user's level of tolerance. Because of this, we need large databases to perform better than smaller databases.

Upgrading your CPU probably isn't the answer

Most people think that upgrading their CPU will improve performance. Usually it does, but only in certain situations. With large databases, the improvement is not dramatic. Any upgrade in the HP-3000 family will cause an incremental improvement in performance. Upgrading to a Spectrum will also make serial reads faster. But none of these are dramatic enough to handle the problems of a large database.

The reason for this is that the CPU is not at fault in the performance problem. Upgrading your CPU makes your computer "think" faster, but a large database is suffering from a different ailment. The dominant activity with large databases is accessing the disc drives. Therefore, what a large database needs is faster disc drives.

Faster disc drives are not currently available. In fact, the speed of disc drives has barely increased during the past ten years while the CPU's have become increasingly fast. This is illustrated in Figure 1 on the next page.

Two new technologies, RAM disc drives and fiber optic interfaces, offer some promise for the future, but are limited today. The RAM disc drives available today cannot accommodate large databases. Fiber optic interfaces improve serial reads, but don't really improve the other types of access. These technologies do not present a significant solution to performance problems with large databases.

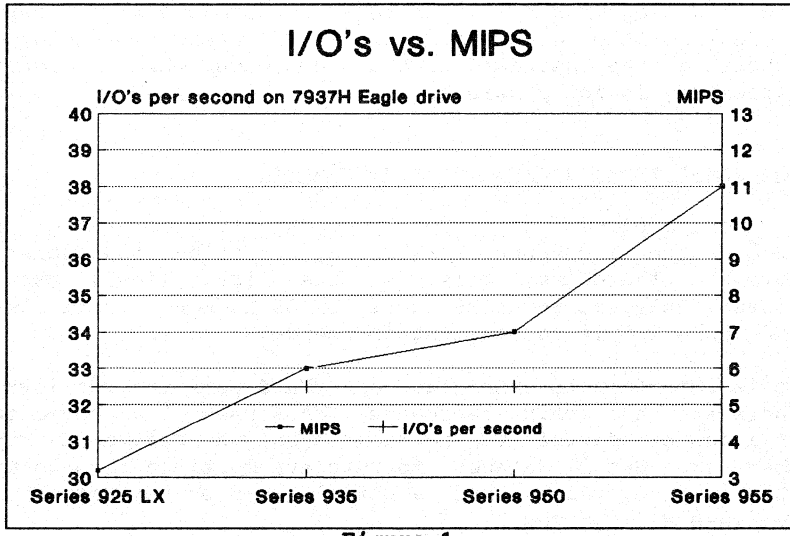


Figure 1

The Solution

There is only one effective solution: reduce the I/O needed by the application. There is both the opportunity and technology to do this today.

The solution is dramatic. The I/O bottleneck in IMAGE is so severe that response time can be accurately predicted based on the amount of I/O to be used. Any reduction in I/O means a proportional reduction in elapsed time. It is frequently possible to reduce the I/O consumed by IMAGE by 75%. This means a improvement in retrieval time of up to 400%.

The opportunity to reduce I/O exists because IMAGE is inefficient with many of its operations. In fact, if IMAGE was efficient in its use of I/O, large databases would rarely cause performance problems.

The three major steps to improve performance will focus on increasing the efficiency of IMAGE where possible, and bypassing the standard IMAGE approach when increasing efficiency is not possible.

The first step: analyze your retrievals

In most applications, the I/O spent for retrieving data exceeds the I/O spent for adding, deleting, and updating data. Therefore, this is the first likely target. Conveniently, it is also where IMAGE is most inefficient, and where I/O is the easiest to reduce.

It is important to understand how IMAGE performs retrievals. There are three main approaches IMAGE uses -- a keyed read on a master dataset to retrieve a unique record, a keyed read on a detail dataset to retrieve multiple records with the same key value, and a serial read on either set to retrieve all records.

The keyed read on the master dataset is an excellent performer. Except in the rarest cases, it cannot be improved upon. It rarely takes more than one I/O to retrieve the record, regardless of the dataset size.

The keyed read on the detail dataset can be an excellent performer, and it can also be a poor performer. Each detail dataset has one key -- the primary path -- which is designed to be efficient. The remaining keys in a detail set vary in efficiency. The more a key uniquely qualifies records, the better its efficiency.

Lastly, there is the serial read. The serial read deservedly has a reputation as the most costly and most inefficient retrieval. The serial read can always be improved upon.

The first step to improving your application performance is to analyze your retrievals. Then, improve the efficiency of the retrievals using the techniques described in this paper.

The serial read

The serial read is a costly act, especially with large databases. Performing a serial read on a 500,000 record dataset can take more than an hour. In nearly every case, the time taken for a serial read can be reduced by at least a third. In some situations, it can be reduced much more.

There are two reasons in IMAGE for doing a serial read. The first is that you want all of the records in the dataset. The best way to improve the performance of this type of serial read is to use a technique called MR-NOBUF. This technique may be written into your programs, or can be performed using third-party packages.

The second reason in IMAGE for doing a serial read is that your selection criteria cannot use any of the available IMAGE keys. In this case, there are two solutions. If you want to retrieve more than around 5% of the records in the dataset, use a MR-NOBUF serial read.

If you want to retrieve less than 5%, and this retrieval is performed frequently, then install a key to support the retrieval. There are several types of keys you can use. If you want to search based on the values in a single field, then use an IMAGE key. It is critical, though, that the search only involve one field, and that you are allowed by IMAGE to add a key to the dataset. This is because IMAGE only allows you to use one key at a time, and restricts the number of keys per dataset.

If you cannot add another IMAGE key, or if you want to search based on the contents of more than one field, use an alternative indexing method available through a third-party package. These packages allow the addition of keys outside of the limitation of IMAGE, and allow multiple keys to be used simultaneously.

Both of these approaches will greatly reduce the I/O required to search the database. They do this by consuming I/O only to retrieve records that you will use. This differs from the serial read where I/O is spent retrieving every record, and then the unwanted records are discarded.

Improving the serial read using MR-NOBUF

An understanding of how IMAGE performs a serial read will reveal how it can be improved.

IMAGE always retrieves records from datasets a block at a time. It does this regardless of the number of records you request. A block of records is a group of contiguous records in a dataset. The number of records in a block is determined by the blocking factor, which can be seen in the listing from a QUERY FORM SETS command. The typical blocking factor ranges from 5 to 30.

One I/O is required for each block of records retrieved. Thus, a dataset with 500,000 records and a blocking factor of 10 will require 50,000 I/O's (500,000 / 10 = 50,000).

It is possible to retrieve more than one block of records per I/O using MR-NOBUF. With this approach, the HP-3000 allows a program to specify the number of records to retrieve with each I/O. Frequently, the program can retrieve 5-10 times the number of records with each I/O, thus improving performance dramatically.

Here is a simple analogy which illustrates this technique. Imagine trying to read a book, but finding that each sentence is on a separate page. You would obviously spend a lot of time turning pages. This is analogous to performing a serial read. Each time you turn the page represents an I/O. If you were to increase the number of sentences on each page to 20, you would spend less time turning pages. This is analogous to having a blocking factor of 20. Just as each turn of the page returns 20 sentences, each I/O returns 20 records. Now imagine being able to put 200 sentences on the page. Very little time would be spent turning the pages, and most of the time would be spent reading. This is analogous to using MR-NOBUF to increase the number of records retrieved with each I/O.

This technique is somewhat difficult to incorporate into your programs. A simpler approach is to use one of several third-party programs which will perform MR-NOBUF retrievals for you.

When to use MR-NOBUF to improve serial reads

There is a simple formula for determining when the serial read using MR-NOBUF is the best choice to use. To use the formula, complete the following form:

FORMULA FOR WHEN TO USE A MR-NOBUF SERIAL READ

1. Enter the blocking factor of the dataset: _____

2. Multiply this number by 5: (this is a conservative estimated gain from MR-NOBUF) _____

3. Enter the capacity if a master dataset, or the number of entries if a detail dataset: _____

4. Divide line 3 by line 2: _____

IF YOU WANT MORE THAN THE NUMBER OF RECORDS IN LINE 4, THEN USE A MR-NOBUF SERIAL READ.

Figure 2

The final result from Figure 2 represents the number of I/O's needed to perform the retrieval. It is also the "break-even point" for using a MR-NOBUF serial read.

Improving the serial read using a key

If you want fewer records than calculated in Figure 2, then consider installing a key. Consider this only for frequent retrievals, though, since installing a key will increase I/O

during the update phases (add, modify and delete). In fact, it is appropriate to compare the number of I/O consumed updating the key with the number of I/O saved by using the key. Only install the key if the savings exceeds the cost by a very healthy margin. As a general rule, a key that eliminates a frequently run serial read is easy to justify.

There are two types of keys that can be used -- IMAGE keys and third-party indexing packages. IMAGE keys are efficient but limited. First, IMAGE limits you to one key in a master set and 16 keys in a detail. You must not have exceeded these limits to choose an IMAGE key. Secondly, IMAGE requires an exact, full-key value. In some searches, this is not readily available. Lastly, IMAGE only allows use of one key at a time. IMAGE is inefficient if your retrievals require evaluating several fields. In this case, IMAGE requires you to pick one key and retrieve all records for that key. The evaluation of the other fields must be done after the record is retrieved. This may lead to the retrieval of many unwanted records, which will waste I/O.

If your requirements exceed the limits of the IMAGE key, then consider a third-party alternative indexing system. The keys provided by these software products have much broader limitations and provide solutions to the problems of IMAGE keys. They often provide features that go well beyond the limits of IMAGE, adding flexibility to your database.

With large databases, it is especially important to avoid standard IMAGE serial reads. They alone can cripple your HP-3000. With smaller databases, a standard serial read can be tolerated. With large databases, they cannot be afforded. Using the three techniques described above -- MR-NOBUF serial reads, certain IMAGE keys, and alternative indexing systems -- the standard serial read can be eliminated.

Improving the keyed read to a detail dataset

As mentioned earlier, there are two kinds of keys in detail datasets. One key in the detail dataset, called the primary path, is structured to be more efficient that the others.

IMAGE keys usually require one I/O to retrieve one record. The primary path uses the IMAGE blocking factor to improve efficiency the same way as a standard serial read. This allows IMAGE to retrieve multiple records with the same key value with each I/O.

The primary path relies on a maintenance function called the single set reload to achieve its performance gain. A single set reload sorts the records in the detail dataset in the order of the primary path. This insures that when IMAGE retrieves a block of records at the price of one I/O, many (if not all) of the records in the block contain the same key value.

The impact of the primary path is illustrated in Figure 3 below:

Detail Dataset After Single Set Reload
Blocking Factor = 4

ORDER-NO (primary path)	PRODUCT-NUMBER (standard path)	QUANTITY	
.			
0001	A00479	1	┐ 1 block
0001	A00334	12	| equals
0001	A09443	1	| 1 I/O
0001	A00023	1	┘
0002	A00479	4	
0002	A09443	1	
.			

Figure 3

In this example, the single set reload has sorted the records in order of the primary path. This allows IMAGE's use of blocks and blocking factors to retrieve multiple records sharing the same key value with each I/O. Notice that because ORDER-NO is the primary path, and a single set reload was performed, only one I/O is needed to retrieve the

four records with ORDER-NO equal to 1. If ORDER-NO was not the primary path, then the dataset would not be sorted by ORDER-NO. This means that each I/O, while retrieving four records, would be likely to retrieve only one record with the key value we desire. Thus the same retrieval would consume four I/O's.

The primary path can be assigned to strategically improve certain retrievals. Assign the primary path to the most frequently used, non-unique key. It is important that a non-unique key be chosen, even if this means not choosing the key most frequently used. The primary path is only effective at retrieving multiple records with the same key value.

Another common use of IMAGE keys in detail datasets can also be improved. As mentioned earlier, IMAGE only allows the use of one key at a time. When users want a small subset of the records in a database, they must choose the single most useful key available. But this doesn't insure an efficient retrieval. When users want to further restrict the search by selecting against other fields, they can only do so after the record has been retrieved. I/O is spent for unwanted records, making the retrieval inefficient.

An example of this is retrieving all people in San Francisco, California. The state field is an IMAGE key, so a standard serial read can be avoided. But the further restriction by city cannot be done using an IMAGE key. I/O is spent retrieving the records for all people in California. The program must then further select the people in San Francisco.

This situation is also improved using third-party indexing systems. Again, these methods allow simultaneous use of multiple keys. This eliminates the unnecessary I/O spent retrieving records which will be immediately thrown away.

The impact of efficient retrievals

After analyzing your retrievals for efficiency, and then incorporating the methods described above, you will have made your retrievals about as efficient as possible. It is

common that these approaches reduce the I/O spent by an application by 2-4 times. This alone can add life to your HP-3000.

The benefits are even more dramatic with large databases. Remember that we expect large databases to perform more efficiently than small databases, given our natural impatience. This also means that an improvement in performance on a large database is more appreciated. The user may not thank you for reducing a retrieve from four minutes to one minute, but they will if you reduce it from four hours to one hour.

The second step: perfect the database design

Nothing substitutes for a well designed database. A poorly designed database is a poor performer and is difficult to manage. A complete discussion of database design is beyond the scope of this paper. However, performance problems with large databases can be attributed to a bad database design.

The rules for proper database design apply to both small and large databases. The main difference is that bad design is more visible in the performance of the large database.
If you are not certain that your database is well designed, you should perform a design review. I recommend two useful articles on the subject:

1. Practical IMAGE Guidelines by David J. Greer
   Published in the IMAGE/3000 Handbook by WORDWARE.

2. Improving Database Performance
   HP 3000 Application Note #54, available through the HP Response Center (Document #5959-9251)

There are two aspects of database design which are useful for tuning the performance of a large database - the size of the BLOCKMAX and the choices of primary paths.

BLOCKMAX is a value set at the creation of the database determining the physical size of the IMAGE block. Changes

to the BLOCKMAX correspondingly change the number of records retrieved with each I/O. Thus a higher BLOCKMAX improves the standard IMAGE serial read and keyed reads using the primary path. Changes to this can be made using most of the database maintenance utilities, but changes should be made carefully. Increasing the BLOCKMAX in databases with a large number of interactive users can degrade performance. A BLOCKMAX of 1,024 words is recommended for most databases, but for databases with few interactive users, a higher value can be considered.

The choice of the primary path was discussed earlier. The best choice for the primary path is the most frequently used, non-unique key in a detail dataset. You should note that if you do not directly specify a primary path, IMAGE selects it for you. Unfortunately, IMAGE's choice is often the worst.

The default choice for the primary path is the first unsorted path in the detail dataset. Most database designers organize their fields so that the most unique identifiers, which are usually IMAGE keys, are located at the top of the list. The keys are rarely sorted, since they tend to be unique identifiers of the records. Therefore, these keys tend to be selected as the primary path.

Since the best choice for a primary path is the most frequently used, non-unique key in the dataset, the default choice is often a poor choice. Review each detail dataset and insure that the primary path is strategically placed.

The third step: perform regular database maintenance

Nothing substitutes for database maintenance, also. This is especially true with large databases. Ironically, database maintenance is usually performed less frequently with large databases.

The time required for database maintenance is usually proportional to the number of records. Therefore, maintenance on large databases can take a long time. A single set reload on a 500,000 record detail dataset can

take an hour or more. Performing a single set reload on all of the large datasets in your database can require a weekend. Because of this, maintenance is often postponed. Avoiding maintenance worsens performance. This creates more demand for the available system time, further reducing the likelihood of database maintenance. This causes database performance to spiral downward.

There are two important database maintenance functions which affect the performance of large databases -- single set reloads and capacity management.

Single set reloads are important for two reasons. They eliminate empty slots in a detail dataset that were created by deleted entries, and they sort the dataset in order of the primary path. Both of these make retrievals more efficient. The frequency of single set reloads depends on your application. If your application adds and deletes many records each month, then a single set reload is needed more often. The more you add and delete records, the more frequently you must perform single set reloads. Datasets that remain static do not need repeated single set reloads.

Capacity management in master datasets is an effective way to keep the dataset efficient. The capacity is used as part of the algorithm for placing records in a master dataset. Prime numbers produce more efficient placement in the dataset, so these should be used when choosing a capacity. Additionally, a master dataset with an ASCII key should never be allowed to fill beyond 80%. A severe performance problem can occur with migrating secondaries when a master dataset fills beyond 80%.

The rules for capacity management are different with binary keys. This is because the approach taken by IMAGE with binary is substantially different. It is not necessary to use prime numbers, and it is not necessary to maintain 20% buffer space in the dataset. It is only necessary to keep the capacity greater than the difference between the highest key value and the lowest key value. For example, if your highest key value is 10,000 and the lowest key value is 6,000, then you must set the capacity greater than 4,000 (10,000 minus 6,000 equals 4,000).

Binary keys have a reputation as bad choices for IMAGE keys. This is because people don't always follow the rule stated above. In fact, if this rule is followed, binary keys are the best choice. The consequence of not following this rule on large databases is fatal -- the time needed to add a single record can be the same as a standard IMAGE serial read of the dataset.

Maintaining your database may take time, but it is critical to sustain performance. A regular maintenance routine must be scheduled in from the start.

Conclusion

A large database will always attract attention. It can either be the reason for poor performance on an HP-3000, or it can be the pride of the company. Nothing seems to aggravate users more than having to wait for a computer. Likewise, nothing seems to amaze users more than seeing data come from a computer faster than they thought possible.

It is possible to have a large IMAGE database that meets the user's expectations, and the tools and techniques for doing this exist today.

NATURAL LANGUAGE: THE ULTIMATE USER-FRIENDLY INTERFACE

Roger Lawson, Chris Hinde and John Connolly
Proactive Systems Inc.
339 South San Antonio
Los Altos
CA 94022
(Tel: 415-941-9316)

ABSTRACT

An overview of the problems associated with the interpretation of English language by computers is given. Possible commercial applications of the technology and the state-of-the-art are covered including a brief description of the various techniques that have been used.

The authors argue that for certain applications (e.g. data base query systems) most of the problems in this field have now been solved. Reference will be made to a prototype system developed by the authors in PROLOG for the interrogation of SQL data bases. Examples of the kind of dialogue possible with such a system will be given.

INTRODUCTION

There are many possible uses of natural language computer processing - for example automated language translation, text scanning to pick up information for further computer processing, improved word processing systems that will correct grammar, and data base querying. Some of the applications are already being commercially marketed (e.g. machine translation and data base inquiry) and there are certainly practical implementations of the technology in use.

As regards the conventional method of retrieving information from a data base it is currently normal to use a data base query language such as SQL. Like ordinary programming languages, query languages have a rigidly defined lexicon and syntax, mainly to ensure unambiguous queries which can easily be interpreted by a computer system. For example, the English sentence, "Get all employees who receive more than $10,000 a year" could be translated into SQL as:

```
select (employee.name,employee.wage)
   from employee
   where (employee.wage > 10000)
```

Although the use of SQL may represent a reasonably satisfactory means of communication with a data base by a computer expert, it is not without its disadvantages. Query languages are artificial languages and consequently they have to be learnt by those who employ them. This learning process involves not only committing to memory the specific vocabulary of the particular query language concerned, but users also require familiarity with the relevant syntax.

Learning the lexicon and syntax of a query language takes time and effort, and remembering them subsequently is not easy, especially for people who do not use them frequently. Moreover, most people do not find it straightforward to formulate relatively simple English queries such as "Get part numbers for parts supplied by a supplier in London to a project in London" in a query language. This query requires knowing the internal name of the relation relating suppliers, parts and projects and to realise that it is not the one relating suppliers to the list of parts they could supply. In other words, from a purely human-centered point of view, query languages are not the optimum mode of information retrieval from a data base.

One solution to some of these problems that has been implemented is to provide a menu based user interface (eg. HP VISOR for SQL data bases, or HP INFORM for IMAGE network data bases). However, in practice users require training in the content of the data base and the way to define an inquiry. Such Systems are also slow to use as you have to step through several screens to define even the most basic query. By comparison natural language is concise and to the point - for example "give me the total sales last week broken down by region".

These observations raise the question of whether interaction between the user and the data base might be conducted instead through using a natural language such as English. If this were possible it would mean the user could employ the mode of communication most familiar to him or her. In this sense, a natural language interface to a data base would constitute the ultimate user-friendly technology for this application.

Incidentally it should be noted that in this paper we have assumed written (ie. keyboard entered) language rather than spoken language. The problems of interpreting continuous speech are great and are still in the process of being solved - they are therefore outside the scope of this paper although the ultimate aim would certainly be to complement the data base interface described in this paper with a voice-input component.

THE OBJECTIONS TO NATURAL LANGUAGE

Three main objections are usually levelled at natural language as a means of human-computer interaction. The first is that natural language is inherently ambiguous. Curiously enough, this objection is always expressed in a natural language statement which neither the speaker nor the listener appears to find ambiguous at all! In order to appreciate what is at issue here, consider the utterance "I wonder who is the most highly paid member of Frug-Frug Records". This utterance is a statement, and yet a person who heard it and was in possession of the relevant information would be at liberty to interpret it not as an assertion but as a request for information. What this example illustrates is that the interpretation of natural language utterances is not merely a straightforward process of decoding its meaning from its form (i.e. from the structured sequence of words of which it consists). Rather, the utterance constitutes a piece of evidence as to the meaning which the speaker intends to convey. It may, of course, be an indispensable piece of evidence, but it still needs to be interpreted in the light of the context in which it appears, given the conventions of the ambient culture. In the absence of such a context, utterances of natural language may indeed be ambiguous. But utterances are always produced in some context, which serves to constrain their possible range of interpretation, so that in practice natural language is far less ambiguous than is often supposed.

What is however true is that in order to reproduce in a computer the human ability to interpret natural language utterances of all forms, it would be necessary to encode into the system a vast amount of knowledge, encompassing the lexicon of the language, its morphology and syntax, its semantics and pragmatics. The lexicon of a language is its vocabulary; morphology relates to the internal forms of words (for example whether a noun is singular or plural); syntax relates to the structure of phrases and sentences; semantics is concerned with the range of possible meanings of words and sentences; and pragmatics deals with the interpretation of utterances

in relation to the context in which they are produced. The enormity of such a task constitutes the second of the usual objections to natural language computer processing. Admittedly, if the aim were to develop a system capable of interpreting any English sentence in any context, this objection would have considerable force. On the other hand, data base query is a limited task domain, requiring of the system the capacity to deal with only a subset of a natural language in relation to a highly specific context. Consequently the idea of providing a natural language interface to a relational data base is not unrealistic.

The third objection to natural language is that it is verbose. Compared with statements in ordinary programming languages, it is true that natural language sentences are often (though not always) more lengthy. However, compared with data base query languages, the difference is perhaps less striking, and in any case the extra terseness associated with a query language has to be weighed against the overhead of having to learn it. Thus, the three standard objections to natural language based systems do not apply with any great force to the data base interface application we have in mind here.

THE TECHNOLOGY OF NATURAL LANGUAGE PROCESSING

There are numerous different ways of constructing a natural language interpretation system. However, two particular techniques for analysing sentences have proved to be popular. One of these involves the use of Augmented Transition Networks (ATNs), while the other is based on an algorithm known as an Active Chart Parser (ACP).

ATNs were introduced by Woods (1970). As their name suggests, they are based on a grammatical description of the target language in the form of a series of networks. Traversal of the appropriate networks is the central process involved in parsing sentences using an ATN. The results of the parse are stored in a series of special-purpose registers.

ACPs derive from the work of Kay (1967), Earley (1970) and Kaplan (1973). They are so called because they make use of a graph-like data structure known as a chart to build up the analysis of a sentence. They operate in association with a grammar in the form of a context-free production system, and offer a particularly efficient means of natural language parsing.

Natural language interfaces to data bases have been produced by others in the past. Probably the best known is the Intellect system which is based on ATN technology. The Intellect system has several hundred installations, mainly on IBM and DEC systems - they claim the ability to interpret correctly over 95% of user requests and their demonstrations are quite impressive. However system performance impact seems high which may be one reason why most users are computer professionals -

the poor end-users still don't get a look in. There is also a natural language interface for the RAMIS products (called ENGLISH would you believe!), SPOCK and NATURAL LANGUAGE for ORACLE data bases, and several PC based products. The latter tend to work on a simple pattern matching approach - typically looking for item names and key words (or synonyms thereof). However they are technically trivial and are likely to be caught out by anything other than a simple request - the moral here being that you get what you pay for.

THE TECHNIQUES WE CHOSE

When we commenced to look at development of a natural language data base enquiry system we chose not to take a simple parsing approach for three reasons.

Firstly parsing based systems are computationally quite heavy, owing to the combinatorial explosion problem. Secondly they have difficulties with poorly formed or ungrammatical phrases - which are of course exceedingly common in normal speech. Thirdly from a review of the development of such systems in the past, and their capabilities, it was obvious that no single method will suffice to quickly and easily interpret sentences that no human would have any difficulty with, ie. different techniques must be combined and used as appropriate.

For example consider the following sentences:

1. "Ann took the cat to the vet because she had injured her tail".

2. "The robber fell off the bank".

3. "The dusting of the new cleaner was not very thorough".

Any human can interpret all three without difficulty because of semantic knowledge. A computer system may find all three ambiguous.

Similar problems are exhibited in phrases such as "How many of our staff are based in London and New York". A literal translation of this always gives the answer zero! A human can interpret it correctly because he knows that staff are only attached to one office at a time.

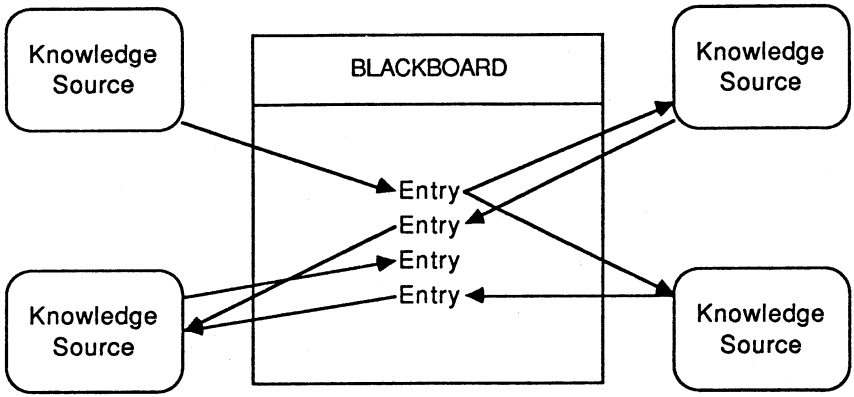
As regards an interface to the data bases to be interrogated we chose to generate SQL rather than have a direct interface so as to ensure portability. SQL is fast becoming an industry standard and is supported by most data base suppliers (e.g. on the HP3000 you can use it with HP SQL, ORACLE, INGRES and even IMAGE soon).

We chose PROLOG because of its power and simplicity for this application. Although PROLOG may not be the most machine efficient language, our aim was to translate queries to SQL in a fraction of the time necessary to actually process the query. On many real commercial data bases this is not a difficult requirement to meet as the processing time can be hours rather than minutes!. In practice we are looking for interpretation in seconds.

We started the development in early 1987 based on the above ideas and the project is now named "ECHO".

ECHO ARCHITECTURE

The ECHO system is based around a ranked-bid truth maintained blackboard system which is unique in its combination of features so as to enable us to incorporate various techniques (i.e. information sources). There is a precedent for using blackboard systems in natural language understanding with the Hearsay systems developed at Stanford University; however the use of a truth maintenance system is novel in this application and the ranked-bid system is a further development of blackboard architectures.



Blackboard systems use knowledge sources of various types to solve problems in a collaborative manner. Each knowledge source examines the blackboard for interesting entries that it can do something with. There is also a set of "facts" which if true could allow a rule or knowledge source to "fire". It is also possible to maintain several interpretations concurrently - for example of ambiguous sentences. A considerable amount of the work in developing the ECHO system has gone into designing the Blackboard management system and implementing the system's

matching algorithm in a way to maintain efficiency.

Our representation of partial data base queries involves storing a term representing the amount of the sentence which is explained by the current interpretation. The knowledge source which waits for data base queries to be formulated has two bids into the blackboard system: the highest ranking bid expects all the English sentence to be explained and so if such a sentence is formed by the other knowledge sources it will "fire" and the most obvious interpretation which explains the English will be presented to the user. The most obvious is defined by the one which has invoked the "most credible" rules and which starts from the most credible English tokens. For example the English word "name" could mean "employee name", "manager name", "supplier name" or any number of other interpretations. The presence of the word "employee" makes the interpretation "employee name" more likely or credible than any other.

The knowledge sources may offer many different types of knowledge or information and it is this aspect which we now address. Many previous natural language understanding systems have been based on a syntactical analysis of the sentence followed by a throrough semantic analysis and finally the required action is formulated via consideration of the pragmatics. It is often the case though that humans can make a syntactically invalid statement which is clearly understood by the receiver; the statement "sales forecast" has no verb and so is syntactically invalid but is interpretable by almost anyone. In any particular situation the sales forecast of the assumed objects would be drawn from a previous context. Syntactic and semantic knowledge is important in determining the required action, but in the context of a data base query system, nothing is relevant unless it helps to resolve the required action. We have therefore adopted an approach primarily based on the content and structure of the data base to guide the interpretation; and although the system incorporates an ACP, syntax is used only as and where necessary to reduce ambiguity.

THE DICTIONARY

This brings us to the dictionary requirements. A large part of our work has been to study why a particular interpretation is correct for a particular sentence; this has suggested that the presence of key words can be critical in determining the meaning. If a particular relation is indicated together with one or more fields or attributes then it is generally a good assumption that it is this relation with the indicated fields that is required and with an indicated task such as "print" or "retrieve" the correct formulation can be achieved. The dictionary should therefore contain indicators for actions, relations and fields with their various synonyms. Syntactic information would be held separately as accepted English syntax is fairly universal. Where there are entries of fields present, the problem of which field is intended arises. Where there is only one field with a suitable domain then there is no ambiguity; however

where there are at least two attributes which take entries from the same domain, some method of resolution is required. This is generally resolved by syntactic or morphological examination. Where there is a relation with such an ambiguity then the information required to resolve the ambiguity must be present in the dictionary.

It is also possible to find that the required relation is not present but can be obtained by suitable join, project and select operations. This is contrasted with an unnormalised data base where all relations might be present in their required state. We do not advocate dispensing with normalisation as this does not solve the problem. It is relatively simple to form the required relations automatically where the graph formed by join operations is acylic as there is a unique join path between any two relations. Where this is not the case, there is an ambiguity unless there are default join paths to form new relations from those existing in the data base. We see the need for these to be specified somewhere and the dictionary seems to be the relevant place. Where there are no default join paths specified then the system will present the user with suitable alternatives.

The dictionary is stored as a set of PROLOG clauses which are consulted as required. We see this as desirable from the point of view of translation efficiency but are concerned should the dictionary become very large. This set of PROLOG clauses is augmented and updated from the data base at appropriate intervals. Each data base accessible by the system contains a set of relations describing the data base's structure and possible contents. The definition of this set of relations is held by the system on startup and so interrogation of any data base to ascertain its probable contents before accessing it is also possible.

One remaining problem requires resolution and this is the the recognition of domain entries. For example, in the question "How many Smiths are there?" a person will recognize Smith as being a name which therefore limits the search domain. Any natural language computer system may therefore also have to know the possible contents of each field. Two solutions present themselves: firstly, all possible entries could be stored along with their domain name and so every entry in the data base would exist in a subsidiary dictionary; or secondly, domain entries could be recognized by their context, i.e. by syntactic or pragmatic means. An example of such an enquiry would be "wage of Smith?". As wage is an indicator of a field in the employee relation, an interpretation of Smith could be an entry in the employee relation also. If the fields are arranged in the order of selectivity then the data base could be interrogated intelligently. With our architecture as described above we have implemented both strategies for domain recognition with a higher priority going to a dictionary entry than for an assumed field. The effect of this is that common entries could be stored explicitly and uncommon ones derived. The uncommon entries would usually be slower to interpret.

Finally relational operators should be represented. In particular: "more than", "exceeded", etc, are built into the system but others are contained in the dictionary.

We see all information that is required being contained in the dictionary and it being customised to the application by the user, but with "core" dictionaries being supplied with the software to facilitate set-up.

KNOWLEDGE SOURCES

There are four major knowledge sources currently implemented in the ECHO system:

- User questions.
- Lexical analysis.
- Formation of data base queries.
- Invocation of the data base.

A description of each follows.

User Questions Knowledge Source

The subsystem for questioning the user is normally given a low ranking and therefore only asks the user for another command when it has nothing better to do. As the Blackboard is empty on start-up, the user query subsystem is the only one capable of being activated as it does not require any other entries to be present. The result is a system prompt of "QUERY >". The user then enters a query which is subsequently processed. The Blackboard is split into many sections corresponding to the various knowledge classes. The users query is entered as an entry of type "phrase" with an appropriate assumption number and corresponding consequence number. The entry is a PROLOG term of the form:

```
consequence (number,type,entry,assumption bases).
```

such as:

```
consequence (1,phrase,[print,employee,names],[[1]]).
```

Lexical Analysis Knowledge Source

The lexical analysis stage is not strictly a lexical analyser as it associates English words with the pragmatics of the data base instead of with their "conventional" lexical classes although some conventional lexical analysis is also carried out. A more complete lexical analysis would allow greater latitude in assuming domain type for unknown words, although this would be at the expense of some fault tolerance. It is this knowledge source which contains knowledge about the structure

of the data base and the clues which the English sentence gives as to the intentions of the user. It is therefore separated out from the system as the data base changes. We have separated the conventional lexical analysis from the data base pragmatics and so potentially eased the task of data base administrators.

The result of this phase is a set of entries which correspond to the interpretations placed on the words in the user query. From the query above we may obtain entries of the form:

```
consequence (number,lex,
  lex (type of word,
        meaning to back end dbms,
        number of preceeding words,
        word(s) used in interpretation,
        original phrase),
        assumption bases).
```

Such as:

```
consequence (2,lex,
  lex (action,select,0,[print,_,_],
        [print,employee,names]),
        [[1,2]]).
```

The above tells us that the system has interpreted the word "print" in the phrase "print employee names" as an action meaning select (in SQL).

```
consequence (3,lex,
  lex (relation,fonts,0,[print,_,_],
        [print,employee,names]),
        [[1,3]]).
```

The above entry tells us that the system has interpreted the word "print" in the phrase "print employee names" as a relation containing information about "fonts" or type faces. The query "print print" would therefore be interpretable under suitable circumstances as a request to list the type faces available.

```
consequence (4,lex,
  lex (relation,employee,1,[_,employee,_],
        [print,employee,names]),
        [[1,4]]).
```

In the above the word "employee" is interpreted as a relation.

```
consequence (5,lex,
  lex (field,name of employee,1,[_,employee,names],
       [print,employee,names]),
       [[1,5]]).
```

In the above, two words together make up a definite field.

```
consequence (6,lex,
  lex (field,name,2,[_,_,names],
       [print,employee,names]),
       [[1,6]]).
```

The above entry means that names could be interpreted separately from employee. A subsequent query involving "supplier" could then link into the interpretation of "names" on its own to form the field "name of supplier".

Data Base Query Formation Knowledge Source

This knowledge source is the most complex and has involved the most work in its construction. It is separate from the pragmatics of the data base and also separate from the lexical analysis system. It has been built in this manner to enable different data base query languages to be implemented without changing the basic structure of the data base. This provides for portability of the software.

Following through from the example above we find this knowledge source combines the second consequence with the fifth consequence to form:

```
consequence (7,sql,
  sql([select(name of employee)],
   [print,employee,names],
   [print,employee,names]),
   [[1,2,5]]).
```

Other entries would be formed, however this entry is now acceptable to the "Invocation of the data base" knowledge source as it uses all the words in the user query. As the presence of such an entry will trigger the system to present this as a possible interpretation to the user we now leave this section.

Data Base Invocation Knowledge Source

The data base is invoked when the ranking of the invocation system is higher than any other knowledge source's bid. In this way we can formulate all reasonable or highly supported queries before invoking the data base or let the data base

invocation take place as soon as any data base query has been formed.

This knowledge source makes two bids to the system, one at a relatively high level which requires any data base query formulated from the English to "explain" all the sentence; the other at a lower level accepts partial explanations for the initial sentence. In this way if the English is fairly tightly written with no extraneous or unrecognised words then the system is immediate. On the other hand, if extra words are inserted which have no bearing on the actual query then the system tries to make sense of them, fails and then looks for other interpretations which ignore the unexplained words.

Continuing with our example we find that this knowledge source presents this in a form suitable to the user as:

```
Do you wish to execute this command?

select employee.name
from employee

|:yes.

The SQL command for database empdb is ...

select employee.name
from employee

/g
INGRES is now being invoked ...

|name                |
|--------------------|
|Smith               |
|Jones               |
|Blake               |
|Clarke              |
|Adams               |
|--------------------|
(5 rows)

Do you wish to input another phrase?
|:
```

The system is now ready to accept another phrase or will search for another interpretation of the user query if requested. The ranking system helps to focus on the MOST REASONABLE environment.

The user request processed above could be quickly processed using a single context truth maintenance system, however should the next query be merely "supplier" then this would be interpreted as the relation supplier. One interpretation would be to select the whole of the supplier relation but another would be to select the "name of supplier" field using the previous context. This clearly requires multiple contexts, but such are introduced only after the "obvious" solution has been found.

Should the next query be "address" then the system will give, before but amongst other interpretations, the option of selecting employee names and addresses.

CONSIDERATION OF THE USER

As stated earlier, the natural language interpretation system described in this paper is designed to operate in relation to a highly restricted domain, namely data base query. It has also been our policy to assume a co-operative and reasonably well-motivated user, who may be expected in practice to employ a somewhat stereotyped range of sentence structures. These pragmatic constraints have made it possible to utilise a more limited syntactic and semantic analysis component than would be necessary in a more general purpose natural language processor. Moreover, they have made possible the construction of an interface that does not demand strict grammatical correctness in the input supplied by the user. With the addition of a spelling checking /correcting mechanism, the system will be tolerant of most user input errors.

EXAMPLES OF SYSTEM IN OPERATION

The following are some simple examples of the kind of translation from english language to SQL that is achievable by the software:

```
QUERY: list product descriptions.

SQL COMMANDS:     select
products.descript,products.stock_code
                  from products,products

OUTPUT:           |descript      |stock_code  |
                  |--------------|------------|
                  |Sugar         |FDD31224345|
                  |Beans         |FDD13133333|
                  |Toothpaste    |FDG23233222|
                  |---------------------------|
```

Note the display of record identifier rather than just the named field.

```
QUERY: print sales.

SQL COMMANDS:     select *
                  from sales

OUTPUT: |account |stock-code  |quantity |price  |value  |
        |--------|-----------|---------|-------|-------|
        |463134  |FDD31224345|        5| $14.80| $74.00|
        |487943  |FDD31224345|        1| $14.80| $14.80|
        |478531  |FGG34343333|        2|  $9.60| $19.20|
        |454555  |FGG33433444|       10| $20.50|$205.00|
        |-----------------------------------------------|
```

```
QUERY: show the parts with a weight less than 17

SQL COMMANDS:      select parts.num,parts.weight
                   from parts
                   where parts.weight < 17

OUTPUT:          |pnum     |weight   |
                 |---------|---------|
                 |P1       |       12|
                 |P3       |       14|
                 |P4       |       15|
                 |-------------------|
```

```
QUERY: give me the total sales for supplier 30

SQL COMMANDS:      select total=sum(supplier.sales_val)
                   from supplier
                   where supplier.supp_num = 30

OUTPUT:          |Total        |
                 |-------------|
                 |           60|
                 |-------------|
```

```
QUERY: I'd like a list of all the sales after 23 January 87

SQL COMMANDS:      select sales,*
                   from sales
                   where sales.deliv_dat > '1/23/87'


QUERY: How many sales are more than $3 in value

SQL COMMANDS: select total = count (distinct sales.value)
              from sales
              where sales.value > 3


QUERY: Which sales are between 3.50 and 4.50 in value

SQL COMMANDS:      select sales,*
                   from sales
                   where sales.value < 4.5
                     and sales.value > 3.5

QUERY: Print total value and quantity of all sales

SQL COMMANDS:      select total = sum (sales.quantity),
                   total = sum (sales.value)
                   from sales


QUERY: Print a list of sales sequenced on stock code within
       supplier and showing supplier name and product name

SQL COMMANDS: select distinct products.descript,
                 sales.account, sales.stock_cde,
                 suppliers.supplier
              from inventory, products, sales, suppliers
              where products.stock_cde = sales.stock_cde
              and sales-stock_cde = inventory.stock_cde
              and inventory.supplier = suppliers.supplier
              order by supplier asc, stock_cde asc
```

POTENTIAL FUTURE DEVELOPMENTS

There are several areas where an approach such as ours scores over more conventional parsing based systems. Most significantly it uses clues or evidence upon which to base its interpretations; in this way the use of mixed input systems including menu and pointer devices with limited keyboard input becomes feasible, so that communication with the system comes to include more than simply the use of informal "English", embracing in addition any other form of information input which is found to be natural to the user. Humans point and gesture in order to communicate effectively and to eliminate ambiguity rapidly. We wish our system eventually to make sense of any input, keyboard or otherwise, that the user might feel natural. The Ultimate User Friendly Interface would then be realised in a natural language environment in the fullest sense, enriched by the appropriate gestural accompaniments which naturally occur in human communication.

REFERENCES

Allen, J (1987). Natural Language Understanding. Menlo Park, Ca: Benjamin/Cummings.

Boddington, R & Elleby, P. (1988). Justification and Assumption based truth maintenance systems: when and how to use them. Workshop on Reason Maintenance Systems and their applications, University of Leeds.

Date, C.J. (1987). A Guide to the SQL standard. London: Addison Wesley.

de Kleer, J. (1986). An Assumption-based TMS. Artificial Intelligence Journal 28,127-162.

Doyle, J. (1979). A Truth Maintenance System. Artificial Intelligence Journal 12,231-272.

Earley, J. (1970). An efficient context-free parsing algorithm. Communications of the Association of Computing Machinery 6,451-55.

Engelmore, R. & Morgan, A. (1988). Blackboard Systems. London: Addison-Wesley.

Forgy, C.L. (1982). RETE: A Fast Algorithm for the many Pattern/Many Object Pattern Match Problem. Artificial Intelligence Journal 19,17-37.

Harris, L.R. (1977). User-orientated data base query with the Robot natural language query system. International Journal of Man-Machine Studies 9, 697-713.

Kaplan, R. (1973). A general syntactic processor. In Rustin, R. (ed.), Natural Language Processing. New York: Algorithmics Press. 193-241.

Kay, M. (1967). Experiments with a powerful parser. In Proceedings of the Second International COLING Conference, 10.

Lesser, V.R., Fennel, R.D., Erman, L.D. & Reddy, D.R. (1977). Organisation of the Hearsay II speech understanding system. IEEE Transactions ASSP 23, 11-23.

Provan, G. (1987). Efficiency Analysis of Multiple-Context TMSs in Scene Representation. Technical report OU-RRG-87-9, Robotics Research Group University of Oxford.

Shneiderman, B. (1981). A note on human factors issues of natural language interaction of database systems. Information Systems 6,125-29

Shneiderman, B. (1987). Designing the User Interface: Strategies for Effective Human-Computer Interaction. London: Addison-Wesley.

Sperber, D. & Wilson, D. (1986). Relevance: Communication and Cognition. Oxford: Blackwell.

TurboIMAGE/XL and the Integration with MPE XL Transaction Management Services

The Foundation of High Performance Database Management

Anna Zara and Rich Friedrich

Commercial Systems Division
Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino, CA 95014

ABSTRACT

This paper discusses the innovative merger of database and operating system functions that result in the highest performing TurboIMAGE database environment ever offered on an HP3000. We will discuss the integration of TurboIMAGE with MPE XL Transaction Manager services. This paper provides an overview of transaction concepts and benefits, the structure of Transaction Manager services, the integration of database and operating system services, and the improvements for system performance.

INTRODUCTION

In the migration of TurboIMAGE to native mode on MPE XL, new features of the MPE XL operating system were leveraged in order to achieve the highest possible level of performance. The MPE XL Transaction Manager (XM) was one of these operating system features. This integration along with its benefits will be discussed in the sections that follow. But before that some background will be provided on the underlying concepts behind transactions and transaction management.

TRANSACTIONS

Basically a transaction can be described as a set of modifications to data that must be viewed as a unit in order to maintain the correctness of that data. A classic example of this is the transfer of funds from one bank account to another. In transferring these funds the balance in one account must be decreased while the balance of the other account is increased. These modifications must be treated as a unit, the subtraction of funds from the first account cannot be allowed to stand without the addition of the same amount to the second account.

More formally, the set of modifications that together define a transaction must have the properties of atomicity, consistency, isolation and durability also known as the ACID properties [GREY 88]. Together these properties define the underlying assumptions of transactions. A transaction is atomic by nature since either all related modifications are present in the data or none of them are. A transaction by its atomic nature always maintains consistent logical relationships between related data. In order to maintain these logical relationships, data which is part of an as yet incomplete transaction is isolated (i.e. not usable from other transactions). Finally, once a transaction has been completed, its modifications will be durable and usable in other transactions. Figure 1 summarizes these properties.

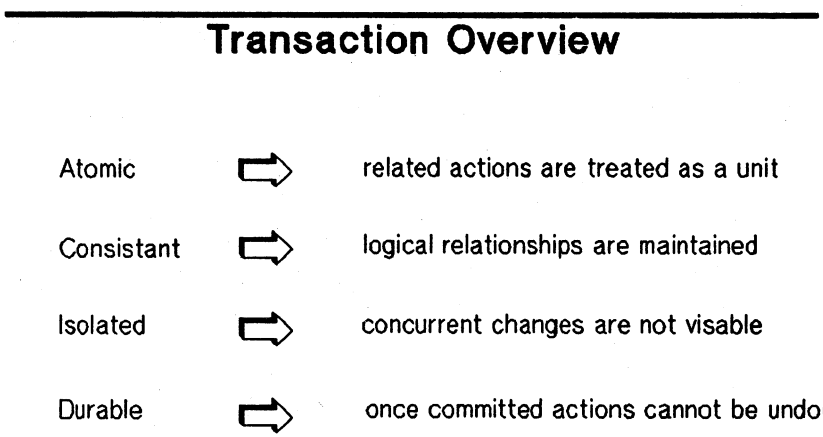


FIGURE 1. TRANSACTION PROPERTIES

In the above discussion of the properties of transactions, no specific mention was made about the type of data that is modified in a transaction. Traditionally, transactions have been thought of as being the modifications that applications make to databases as in the example given above describing the transfer of funds. With the implementation of a transaction manager in MPE XL the transaction concept has become an integral part of the operating system and can be used by the operating system and its subsystems to assure that modifications to all types of data, including internal data structures, have the ACID property.

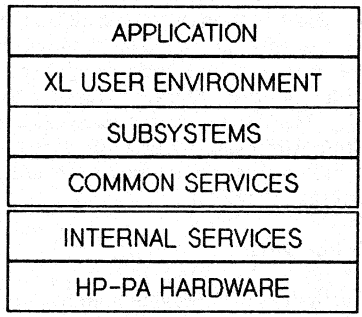
MPE TRANSACTION MANAGEMENT SERVICES

Overview of the MPE XL Operating System

The MPE XL operating system's design objectives included increased functionality, flexibility, and improved ease of use, while providing the foundation for future HP Precision Architecture (HP-PA) hardware platforms. A key goal was to provide the mechanisms in system software that would allow customer applications to take advantage of the potential from high performance processors that can be built with HP-PA [BIRN85] [FOT87]. Transaction Management was designed to provide the services necessary to meet the demanding requirements of

large, on-line transaction processing environments, especially those that are data access intensive like TurboIMAGE/XL.

MPE XL SOFTWARE ARCHITECTURE



Commercial Systems Performance
OSARCH RJF 3/22/89

HEWLETT PACKARD

FIGURE 2. MPE XL SOFTWARE ARCHITECTURE

The basic structure of the MPE XL operating system and its environment is shown in figure 2. The highest level of this structure consists of the application programs developed by the customer or an independent software vendor using industry standard languages such as COBOL, Pascal, FORTRAN, or 4GLs such as TRANSACT. The MPE XL user environment contains those services that support session and job activity like the command interpreter. It also contains an intrinsic interface which provides process, file and other common service access. In addition, database, networking and system management utilities also use these interfaces to support their activities.

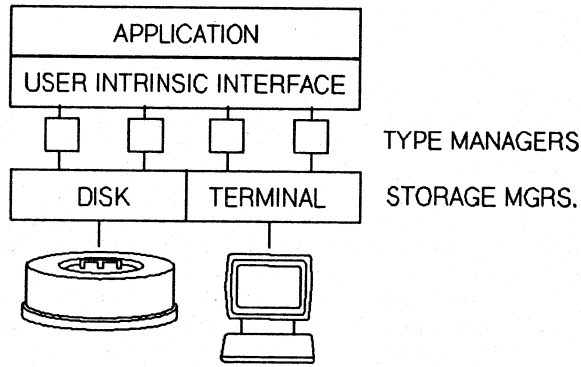
MPE XL provides a set of privileged common services that include job and session control to support process management, standard table management to synchronize access to internal system tables, ports for interprocess communication, file system and directory services to access and manage external disk data, and transaction management services for concurrency control and recovery. We will explore the file system and transaction management services in more detail later.

The core services supported by MPE XL are internal services for sharing the hardware resources of processor, memory and secondary storage (disk). Virtual Space Management provides the mechanisms for translating virtual addresses to disk addresses. VSM was designed specifically to take advantage of the large addressability of the HP-PA architecture. VSM uses B*trees to efficiently handle the 2^48 (2 to the power 48) bytes of addressability (the equivalent of 470,000 7937 disk drives) with the minimum of allocation, search and update overhead. Memory Management is responsible for the effective use of main memory based on current, global process demands and has been designed to take advantage of the large main memories available on HP-PA systems. Memory management allows the memory resident portion of a file to grow to fill all of the available memory on a lightly loaded system or utilize only its equitable amount based on current, global memory demands. (There is no explicit disk caching as in MPE V). Memory management interacts with the disk Storage Manager to determine optimal disk I/O prefetching (prepaging) block sizes when a page fault occurs or is predicted to occur, and supports posting strategies (writing of pages to disk) that allow the gathering of writes in contiguous virtual space to minimize the number of physical disk I/O requests [BUSCH87].

Overview of the MPE XL File System

A primary goal of the File System is to provide high performance access to file data to meet the requirements of higher performing commercial systems. Building on the successful foundation of MPE V disk caching, the MPE XL file system designed and implemented a structure to eliminate file buffers and disk cache domains, replacing them with disk files that are mapped directly into virtual space (i.e. mapped files). This eliminates the need for explicit disk file buffer management and significantly reduced the file location (search) overhead by taking advantage of HP-PA virtual to physical translation hardware. It also eliminated the wasteful CPU consumption caused by copying data from one buffer area to another.

MPE XL FILE SYSTEM ARCHITECTURE



Commercial Systems Performance
FSARCH RJF 3/22/89

HEWLETT PACKARD

FIGURE 3. MPE XL FILE SYSTEM ARCHITECTURE

The structure of the MPE XL file system is shown in figure 3. Applications access the file system via the user intrinsic interface layer. A set of type managers, each unique to the particular file type and storage device, are available. Finally, separate disk and terminal storage managers exist that are optimized for the particular characteristics of the hardware devices they support. The file system's structure allows the flexibility and extensibility to take advantage of new storage technologies with minimal impact on current file system structures. In addition, transaction management is an integrated mechanism that addresses systematic problems of commercial data processing systems like concurrency, data availability and recoverability, and ever increasing performance demands due to disk I/O traffic.

Overview of Transaction Management Services

A need for high performance concurrency control and transaction management services has been discussed in academic circles for years. The crux of the problem is the proliferation of file system, database system, and application software providing some aspect of data access control, consistency, recovery and auditing. Each supplies it own unique solution to the problem resulting in commercial data processing systems that are complex and less reliable; compromising performance potential and adding to administrative headaches. MPE XL has become the first

widely available commercial operating system that offers integrated transaction management services designed for high performance data management systems. This was achieved by tightly coupling transaction management, memory management and HP-PA supplied mechanisms. The result is a consistent and efficient facility for use by the file system and other privileged subsystems in providing concurrency control and recovery mechanisms for all permanent system objects and user files that have the transaction management property.

Files are said to have a transaction management property when they are attached to an XM log file. Once attached, all modifications to those files will be done under the control of XM transactions which are logged. This provides an automatic and transparent recovery mechanism in case of transaction abort, application abort or system failure.

XM services are an integral component of the MPE XL operating system. Not only do they provide support for the TurboIMAGE subsystem, but they are also used by the OS to provide concurrent and recoverable access to the file system directory structures, file label management, and disk free space allocation (Secondary Storage Management). These services also provide the basic mechanisms to support the serial write queue for User Logging and KSAM.

XM provides logging and recovery managers to guarantee that any updates made by a transaction are atomic. This is done through image logging of modifications made to objects which have the XM recovery property. Using TurboIMAGE/XL as an example, when an application calls DBPUT to insert an entry into a detail dataset, XM first copies the data currently in that portion of the database file (binary zeros for the detail, the current chain count for the master sets) to the XM log file in memory; this is known as the before image. Then the database files is modified required by the TurboIMAGE intrinsic. Once the file has been modified, the modification is copied to the XM log file in memory; this is known as the after image. If the XM transaction in question cannot be completed, the transaction will be recovered by XM through the application of the before images back to the affected files.

To optimize disk access, the log file is written to disk based on an algorithm which is determined by a timer pop (currently one second), the amount of data logged, the number of data pages frozen in memory or when block on commit is requested.

XM uses a write ahead log protocol which requires the log file to be durable on disk before a transaction's modified data pages are allowed to be durable on disk. Because of this protocol, after a failure the recovery manager reads the log file on disk and rewrites all changes to data pages that were not yet on disk. This protocol also requires that memory manager and XM communicate about the status of a particular page preventing dirty pages from being swapped to disk before their corresponding log images are made durable on disk. Physical logging insures that file data and internal data structures are consistent after a system or application failure or a transaction abort.

To reduce the amount of time it takes to recover committed data after a system failure, XM uses a checkpoint process to periodically synchronize the log file and all committed data on disk. This process is triggered by the amount of logged data. XM's log file is divided into two halves, both of which are circular. When one half of the log file is full, a checkpoint occurs. The checkpoint process schedules I/Os for all dirty pages still in main memory that have not yet been written to disk

and have images in that half of the log. The writing of these data pages occurs in the background to minimize impact on system resource usage. During the checkpoint, XM logging continues using the other half of the log.

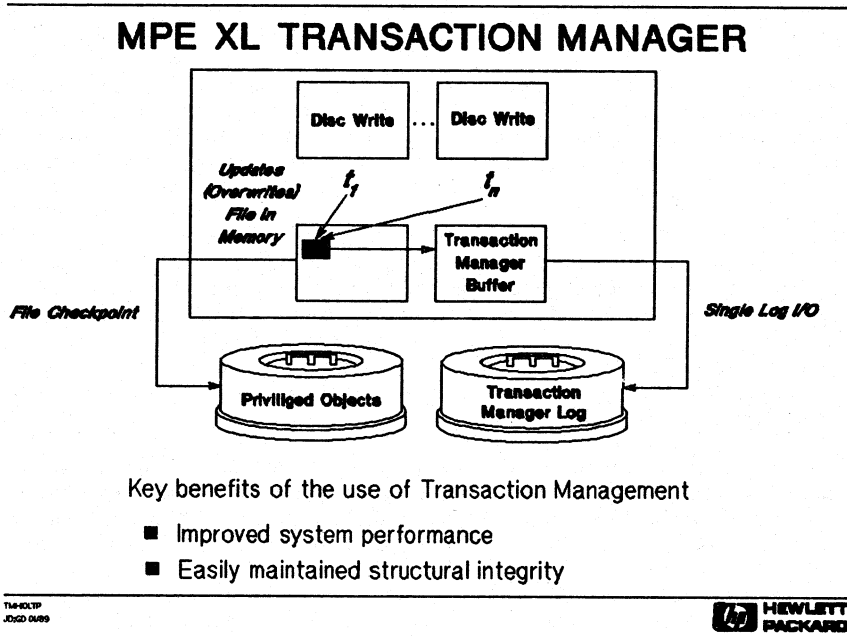


FIGURE 4. DYNAMIC OPERATION OF TRANSACTION MANAGEMENT

Figure 4 summarizes the operations of TurboIMAGE/XL during steady state execution when only the XM log buffer must be posted to disk. This post takes a single I/O. At checkpoint time, the XM log buffer is posted and then all dirty pages are also posted. The result is that the XM log file and the corresponding database pages are synchronized on disk and a subsequent recovery operation, if needed, would take minimal time.

TURBOIMAGE/XL HOOKUP TO MPE XL TRANSACTION MANAGEMENT

The primary benefits of the integration of XM into the internals of TurboIMAGE/XL have been in performance and integrity of database structures (physical integrity). Integrating with XM has allowed TurboIMAGE/XL to eliminate the serialization points associated with waiting on disk write completions for database modifications. Since all changes to database files are logged by XM, only the XM log file needs to be written to disk for those changes to be durable. When the database pages are finally written out to disk

during a checkpoint of the XM log, effects of multiple modifications can be written to disk through fewer I/O requests.

Aside from increasing the efficiency of writes to disk and reducing write related serialization, the XM hookup has meant that intrinsic level recovery is the default mode of TurboIMAGE operation. All modifications associated with a given TurboIMAGE intrinsic are part of an XM transaction. The ACID property of XM transactions mean that either all modifications associated with an intrinsic are made or none of them are. In other words if a process is interrupted during a TurboIMAGE intrinsic on MPE XL, the effects of the incomplete intrinsic will be rolled back so that all chains and the associated data will be left in a consistent state.

In using the MPE XL XM services, TurboIMAGE/XL uses the same trusted interface that is used by other subsystems that take advantage of XM. (A trusted interface is a series of access procedures which assume that the caller is critical, in system code and is using the correct calling sequence. Incorrectly calling a trusted interface procedure will result in either process or system abort.) The first step taken in using XM is to give database files the XM recovery property. This is done by attaching all files within the database to the system level XM log for the volume set in which the database resides.

Once a database file has the XM recovery property, all modifications to that file will be logged to the XM log file by the Disc Storage Manager (DSM). If a given write is not part of an already existing XM transaction, it will be an XM transaction in and of itself; in this case DSM will start an end transaction on behalf of the writer.

As mentioned previously all modifications associated with a TurboIMAGE intrinsic are part of the same XM transaction. Before DBPUT, DBDELETE or DBUPDATE make a change to a database file, the XM procedure to start a transaction will be called. This procedure will begin a transaction that will contain all the writes associated with the intrinsic. When the last modification has been made the intrinsic will be committed (ended). This activity is illustrated by Figure 5 for the actions taken by DBPUT when a record is to be inserted into a detail dataset with one automatic master path.

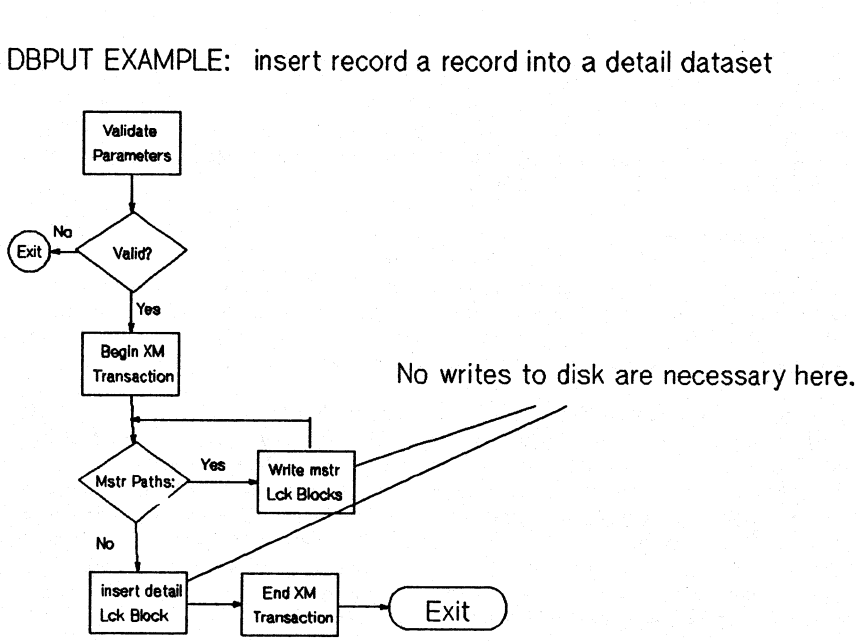


FIGURE 5. FLOWCHART OF A DBPUT

The sequence of events described above are the same as those performed by TurboIMAGE/V, the XM transactions being the main difference. If an error were to result that prevented the above intrinsic from completing when using TurboIMAGE/V, chains would be broken. TurboIMAGE/XL, on the other hand, will automatically return to a consistent state before any other intrinsic is allowed to access the affected data.

TURBOIMAGE/XL RECOVERY OPTIONS

The use of XM has also changed some of the details of the recovery options available in TurboIMAGE/XL from those available on TurboIMAGE/V. The following is a brief discussion on the differences between the TurboIMAGE/XL and TurboIMAGE/V recovery options.

Default Mode

On TurboIMAGE/V default mode provides minimal recovery protection. The serial write queue is used to maintain the ordering of writes to the database, but it is possible for broken chains to result in the event of a system failure.

TurboIMAGE/XL's default mode provides intrinsic level recovery for all modifications associated with an intrinsic through XM. XM also maintains the ordering of the writes to the database through its log file. The XM log will be written to disk at least every second. In the event of a system abort it is possible to loose one second's modifications, however, because of the ACID properties of XM transactions, no chains will be broken.

Output Deferred Mode

On TurboIMAGE/V blocks in the database control block (DBB) buffer pool are not written to disk until a least recently used mechanism requests that the block be written. (MPE V global files opens make this possible.) If the system fails while a database is being accessed in this mode, the database will be corrupted.

Using output deferred on TurboIMAGE/XL means that the database files (with the exception of the root file) do not have the XM recovery property and that modifications to the files will not be protected with XM transactions. Since on MPE XL, main memory is basically a write through cache, modifications to database pages will be written to disk only when the Memory Mangers decides that a page is no longer hot and can be overlaid after posting. This is similar to what happens in the DBB when output deferred is enabled on MPE V, the amount of data involved however is greater. As on MPE V if the system aborts while a database is being accessed in this mode, the database will be corrupted.

Intrinsic Level Recovery (ILR)

TurboIMAGE/V maintained the physical integrity of the database by logging DBPUT and DBDELETE intrinsics to a log file through the a control block (ILCB). If the system failed during database access, the first open of the database after the failure would trigger the recovery of any incomplete intrinsics. The resulting database would not have any broken chains.

As mentioned previously intrinsic level recovery is the default mode for TurboIMAGE/XL. The only difference between default mode and ILR for TurboIMAGE/XL is that the XM log file is posted after every DBPUT and DBDELETE. This reduces the exposure to system abort of DBPUT and DBDELETE from one second to zero. However, using this feature will degrade application performance considerably so it is not recommended.

Rollforward Recovery (User Logging Enabled)

This mode is basically the same for both TurboIMAGE/V and TurboIMAGE/XL. Log records are written to MPE User Logging (UL) log files for each DBOPEN, DBCLOSE, DBBEGIN, DBEND, DBMEMO, DBPUT, DBDELETE and DBUPDATE intrinsic. In the event that the database requires recovery, the database is restored from and archive tape and DBRECOV is used to reapply the changes recorded in the UL log file.

On MPE XL, UL is a compatibility mode subsystem. It has also taken advantage of XM features to reduce some serialization on its data structures. Also, because UL log files are attached to XM the sequence of UL log file writes and database file writes are maintained if the UL log and the database are in the same volume set.

Rollback Recovery

Since this mode also uses UL it has not changed very much in the migration of TurboIMAGE to MPE XL. However, the use of XM has resulted a decrease in the amount of I/O's needed to make a modify intrinsic durable. If DBBEGIN/DBEND pairs are used to mark logical transactions in an application, even more writes can be avoided since it is only at DBEND time that an I/O must be forced to assure the durability of a logical transaction.

PERFORMANCE ANALYSIS OF TURBOIMAGE/XM

During the early development of MPE XL and TurboIMAGE/XL, system models were used to predict expected performance gains of migrating TurboIMAGE code (in SPL) into native mode and hooking up a subset of its services with XM. Experimental systems were developed and benchmarked to verify the model's predictions and provide additional tuning insights. One of these benchmarks was based on an on-line transaction processing application that used TurboIMAGE and file system access methods. A summary of the results from this experimental is depicted in figure 6. (This benchmark was chosen as representative of customer environments after an analysis of MPE V workloads [PERR86])

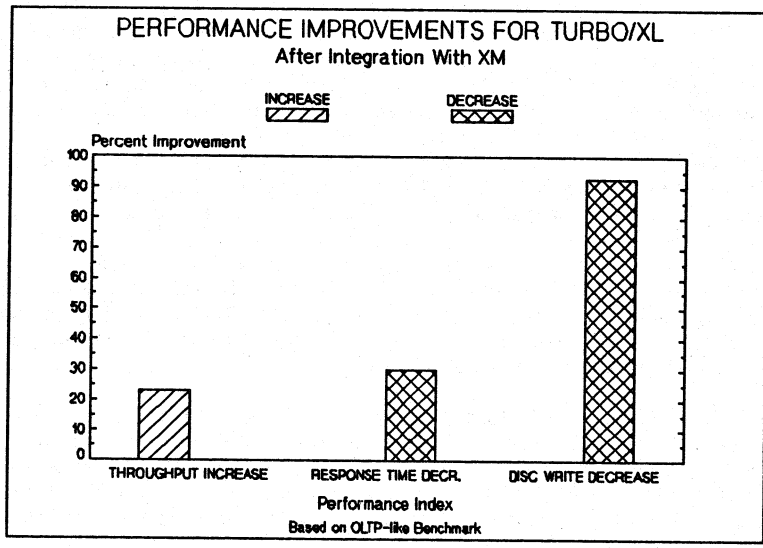


FIGURE 6. PERFORMANCE IMPROVEMENTS OF TURBOIMAGE/XL USING D

The on-line transaction processing benchmark demonstrated a 23% improvement in throughput for the TurboIMAGE/XM experimental as compared to the regular TurboIMAGE port. Response time was also reduced by 30%. Most of these improvements were a result of the elimination of unnecessary database and operating system physical disk I/Os and their resulting CPU consumption. The total number of physical disk write I/Os was reduced by 93% and there was an 88% reduction in the number of physical disk I/Os per transaction. The multiprogramming level (i.e. the amount of concurrency in the system) also increased as a result in the reduction of process stops due to disk I/O [SPL86].

In addition, a database write intensive batch job was benchmarked. Although its CPU utilization was only reduced by 28% over the TurboIMAGE without XM, the elapsed time was reduced by 69%. This was due to the elimination of unnecessary process stops due to database disk writes.

TurboIMAGE/XM's ability to take advantage of faster CPUs and large memories with the resultant reduction in the number of disk I/Os minimizes one of the stickiest bottlenecks that plagues high end commercial computing systems: becoming I/O bound on the disk subsystem. TurboIMAGE/XM has provided the high performance basis that can meet the computing requirements for HP's high end customers into the next century.

TurboIMAGE/XL application Tuning Tips

If Rollback recovery is used, performance can be improved if logical transactions are marked with DBBEGIN/DBEND. Rollback recovery requires that the current transaction be made durable before the next transaction can be started. If DBBEGIN/DBEND are not used to mark logical transaction groupings of modify intrinsics, each modify intrinsic is assumed to be a transaction, so the XM log is posted due to the request for block on commit after each modify intrinsic. When DBBEGIN/DBEND are used, the XM log is only posted when DBEND is called because it is at that point that the transactions must be made durable.

Do not use ILR if the one second exposure to system abort is not an issue. TurboIMAGE/XL does not require ILR for rollback. In fact if ILR is enabled along with rollback recovery, the benefits of using DBBEGIN/DBEND for logical transactions is voided since ILR will post the XM log after each DBPUT and DBDELETE because it requests a block on commit.

Keep the number of buffers in the DBB constant. Since MPE XL is a paged system with larger main memory configurations then MPE V, the size of the TurboIMAGE/XL control blocks are not an issue. The expansion and contraction of the DBB on MPE XL will cause serialization on the DBB which will impact performance of applications which open and close databases frequently.

If a dataset has few entries and will be accessed often (i.e. the account number dataset in some financial applications) better performance will be seen if the blocking factor of that dataset is small. TurboIMAGE/XL works at the block level, since TurboIMAGE/XL has improved concurrency few entries in a block will allow more users to access the data simultaneously.

REFERENCES

[BIRN85] J.S. Birnbaum and W.S. Worley
Beyond RISC: High Precision Architecture
HP Journal, August 1985.

[BUSCH87] J.R. Busch, A.J. Kondoff, and D. Ouye
MPE XL: The Operating System for HP's Next Generation
of Commercial Computer Systems, HP Journal, Dec. 1987.

[DEMM88] W. Demmel
MPE XL Release 1.0 and 1.1 TurboIMAGE/XL SE Training Materials
Hewlett Packard SE Course 289, 1988.

[FOT87] D.A. Fotland, et al
Hardware Design of the First HP Precision Architecture
Computers, HP Journal, March 1987.

[GREY88] J. Grey and A. Reuter
"Transaction Processing Lecture Notes", V3
Summer 1988.

[PERR86] D. Perry and K. Mitra
"Representative Benchmarks"
Hewlett-Packard Internal Technical Report,
October, 1986

[SPL86] Systems Performance Laboratory
"Series 930 and 950 Performance Report"
Hewlett-Packard Internal Technical Report,
December 1986.

[ZARA87] A. Zara, D. Fu and K. Montgomery
TurboIMAGE/XL: XM Integration Design Document and Prototype
Hewlett-Packard Internal Document
Spring 1987.

P.A. Bernstein, V. Hadzidocos and N. Goodman
Concurrency Control and Recovery in Database Systems
Addison-Wesley, 1987.

C.J. Date
An Introduction to Database Systems Volume 2
Addison-Wesley, 1983.

J. Grey
"Notes on Database Operating Systems"
I.B.M. Research Report RJ2188, I.B.M San Jose Research
Laboratories, 1978.

HP EDIT: The Design of a New Interactive Editor

Michael Suckow

Hewlett-Packard
Data and Languages Division
19111 Pruneridge Avenue
Cupertino, California 95014

"For every complex problem, there is a simple elegant solution which will not work"

Henri Louis Mencken

Introduction

The design of a highly interactive editor for MPE systems presents certain unique technical challenges. Developers who have attempted to port applications from other operating systems such as UNIX or DEC's VMS have found that although MPE offers great functionality, it also requires a change of mind-set.

The purpose of this paper is to examine the use and potential misuse of MPE character-mode I/O by focusing on a case study: the development of HP EDIT, a new full-screen editor.

HP EDIT Background

HP EDIT is a full-screen programmer's editor with an extensive command set. It offers relatively unrestricted full-screen editing, including the use of the cursor keys and other terminal keys such as Insert Line and Delete Line. Other features include a multi-level undo facility, support for COBOL-numbered files, and a comprehensive on-line help system. It is intended for use with direct-connect terminals.

In the early days of the project, a two-person design team was assembled to investigate our customer's needs and possible solutions. We were both knowledgeable about editors, but inexperienced with MPE I/O. As we were about to find out, this would prove to be a disadvantage.

The HP EDIT investigation

The goal of the HP EDIT project was to create a powerful and highly interactive full-screen programmer's editor for MPE/V and MPE/XL systems. We wanted to provide an alternative to line editors such as EDIT/3000 and block-mode editors such as TDP/3000 screen-mode, neither of which we considered optimal for programming. We also felt that releasing a new block-mode editor would not greatly improve on our current offering.

The design team examined editors available from HP and third-parties. We did an evaluation of not only HP3000 products, but also of editors running on PC's, HP-UX, and other manufacturer's minicomputers.

An early solution which appeared promising was to develop or port a modeless editor such as ones used on HP-UX systems. A modeless editor has only one mode of operation. Text and commands may be entered at any time typically using the alphanumeric keys for text and control keys for commands. An example of an editor of this type is Emacs, originally developed by Richard M. Stallman at MIT. Since an Emacs-like editor was already available from HP on HP-UX systems, a port to MPE would have made it possible for us to offer a single editor across computer platforms.

We decided to use character-mode I/O to achieve the highest level of interactiveness. It was felt that programmers would be more productive if the editor allowed an action to be performed with a minimal number of keystrokes. The use of character mode also made it possible to implement a granular undo feature capable of undoing small changes to the screen. We hoped to provide the functionality of a PC editor, which we saw as the best solution, on a multi-user terminal-based system.

The MicroEmacs port

The next step was to prototype the user interface, and to show it to a few programmers. We decided to port a version of MicroEmacs to MPE. MicroEmacs is a public-domain editor which had already been ported to various machines.

The port itself went uneventfully due to the modular design of MicroEmacs, and to the quality of the Corporate Computer Systems' C compiler on MPE/V. Once the prototype was up and running, though, we began noticing that MicroEmacs couldn't always keep up with a person entering text. The problem wasn't immediately apparent because the editor seemed to perform adequately when used by lab engineers who were not proficient typists. When confronted by our able secretary, the prototype quickly surrendered - it began losing keystrokes. The prototype was also unable to read the terminal cursor keys which transmit a two-character sequence, or function keys programmed with more than one character.

Why was MicroEmacs losing characters ?

The problem stemmed from Emacs' reliance on single-character reads. Emacs allows the arbitrary remapping of any key to any command. To provide this functionality, Emacs has to be able to react to every keystroke; therefore requiring single-character reads. The solution to this problem, if there was one, was not intuitively obvious.

Why was the MPE version of MicroEmacs losing characters? Surely, this was not the case on other systems on which MicroEmacs was running. What was different about MPE?

The primary reason for the single-character read problem is the lack of typeahead on MPE. Typeahead is an operating system facility which allows buffering of the user input data in preparation for future read requests. In order to capture user input on MPE, an application needs to post a read against the terminal prior to the user striking a key. If the read is not posted, the keystrokes will be lost. Also, if the user enters more characters than were expected by the read, the excess text will be discarded.

Attempts to improve the prototype

The MicroEmacs prototype convinced us that we would not be able to rely on ordinary single-character reads. Still, we weren't quite ready to abandon hope for our prototype. Our next step was to explore several techniques which we hoped would yield the desired results. The techniques we tried were:

No-wait I/O

We had determined that we weren't able to post a second read fast enough to capture the next character in a cursor key sequence. This lead to the idea of posting two no-wait I/O reads against the terminal in the hope that each read would capture one of the desired characters. A no-wait I/O read is one in which the system returns control to the calling program before the I/O operation has completed. Having opened the terminal twice, it is possible to post two no-wait I/O reads and have them both be active at the same time. This method did improve our ability to capture a sequence of characters, but not sufficiently for us to read the cursor keys. Another drawback to this technique, aside from the fact that it does not work, is that it requires that a portion of the program run in Privileged Mode.

Multiple processes reading from the same terminal

In this attempt, we had a program spawn another process which, in conjunction with its parent, posted reads against the terminal. The hope was that with two processes reading from the terminal simultaneously, it would be possible to capture all the input reliably.

Unfortunately, this technique did not solve the problem, even when used in conjunction with no-wait I/O reads. In fact, this approach is functionally identical to opening the terminal twice as we did in the no-wait I/O case. Using a second process did not buy us any additional performance.

Using high priority queues for the read

We came up with the idea of raising the priority of the process during I/O operations in the hope that the reads would be posted fast enough to capture all the terminal input. Once again, our test was unsuccessful. We did not raise the priority of the process to a level where it would interfere with the operating system. Still, we have no reason to believe that this approach would ever have worked reliably.

Changing the terminal's baud rate

None of the techniques we had tried up to this point made more than a nominal difference in our ability to capture keystrokes. Our next, and somewhat desperate attempt, was to programatically lower the terminal's baud rate. We reasoned that we could use the terminal's handshaking protocol to slow down the data transmission sufficiently to allow the capture of character sequences such as cursor keys. Once again, we were able to improve our performance, but not substantially. Even if this approach had succeeded, it probably would have been undesirable since an abort of the editor would have left the terminal at an unexpected baud rate.

We tried one more test combining no-wait I/O, multiple processes, high priority reads, and reduced terminal baud rate, but to no avail. We finally came to the conclusion that there was no reliable way to keep up with the terminal input using single-character reads.

Multi-character reads with a time-out

The core of our problem was determining if the user would be entering a single character or a sequence of characters (i.e: a cursor key). We could always capture two characters, but how could we force a two-character read to terminate if a single character had been entered. We attempted to get around the problem by posting a two-character read with a time-out. The idea was that if a single character had been entered, the read would terminate anyway because of the time-out. There were two major problems with this approach: First, the smallest time-out value allowable is one second, a relatively long time period for a user who is waiting for a command to be executed. Secondly, and more importantly, the file system will not transfer any data to the user's program if a read is terminated due to a time-out. The system considers a time-out to be the result of an unsuccessful read attempt; therefore, it throws the data away.

Non-destructive reads

The idea was to post a single-character non-destructive read. If the returned character was an Escape, we would assume that the user had pressed a cursor key (e.g.: Escape-A which is equivalent to Cursor Up). We would then attempt to re-read the terminal buffer, but for two characters this time, thus retrieving the original escape sequence. Our attempt was unsuccessful because the ATP/DTC only transfers to the terminal drivers the number of characters in the original read request; in this case: one. The second character was lost.

Non-destructive reads are intended for message file I/O. They are not useful for terminal I/O.

A time for reflection

Ultimately, we came to the conclusion that although we could develop an editor relying exclusively on single-character reads, we could not guarantee that it would be able to keep up with the terminal input, even under ordinary circumstances. Data integrity being of paramount importance, we had no choice but to eliminate Emacs as a possible solution.

The reader may wonder if our conclusion will remain valid in the future. Indeed, the unsuitability of Emacs is predicated on the absence of typeahead on MPE systems. This current restriction will soon disappear as HP begins shipping MPE/XL 1.2 which incorporates an optional typeahead facility. One would be tempted to ask: couldn't we implement a reliable Emacs editor on MPE/XL by turning on typeahead. The answer, technically, is yes.

There is however another consideration which one should consider: the impact of an application such as Emacs on system throughput. To illustrate this, let us take a closer look at MPE character-mode I/O.

Character-mode I/O on MPE systems

Character mode, in the MPE world, is a method of I/O interaction in which every keypress on the terminal keyboard is transmitted to the system. The HP3000 uses intelligent terminal controllers (the ATP or Advanced Terminal Processor on MPE/V systems and the DTC or Distributed Terminal Controller on MPE/XL systems) to handle the grunt work of capturing keystrokes. The controller places keystrokes into a buffer until the requested number have been received or until the user terminates the read, usually by pressing the Return key. The string of characters is then sent as a whole to the system.

In contrast, block mode refers to an I/O interaction method in which the terminal interprets all keystrokes locally, and only transmits a screen of text in its final form.

The fundamental design of MPE I/O is particularly well suited for transaction processing applications. In fact, it is partially because of its I/O structure that MPE is able to deliver excellent throughput with a large number of terminal connections. MPE achieves its performance by offloading much of the I/O work to the ATP (on MPE/V systems), the DTC (on MPE/XL systems), and the terminal (when used in block mode).

MPE I/O is optimized for large block-mode transfers and has a higher overhead per character for small character-mode reads. MPE achieves its best I/O performance when used with block-mode applications. In block mode, the systems can process whole screens of information at one time. In a character-mode application where reads are posted for each line on the terminal, the system needs to be awakened every time a read has completed. Even worse, if the application is reading one character at a time, the system needs to process each character as soon as it is transmitted by the terminal. In other words, single-character reads are expensive in terms of CPU usage. The total overhead becomes more significant as the number of reads increases.

Assuming the availability of typeahead, there is no reason why an application under MPE couldn't rely exclusively on single-character reads. Such an application, however, could reduce or completely eliminate MPE's inherent advantage for on-line transaction processing by tying up the CPU with constant I/O operations.

Because of the lack of typeahead on MPE/V and the impact on system throughput, we abandoned our prototype, Emacs, and editors using a similar I/O model.

The genesis of HP EDIT

It had become necessary to re-think our strategy. We still wanted to use character-mode I/O, but in a more efficient way. We then took a closer look at the internal editor VOODOO, which appeared to be making good use of character-mode I/O. The editor had been used extensively within HP for several years, and had received positive reviews. It became the basis for the HP EDIT product. HP EDIT was developed by "bootstrapping" itself - the source code was written and maintained using HP EDIT from the very start.

We knew that we would have to enhance the editor considerably to make it into a product, but now at least, we had a solid foundation for HP EDIT.

The HP EDIT compromise

We believe HP EDIT to be a good compromise for MPE systems offering a high level of interactiveness with optimized character-mode I/O performance. The editor uses a combination of single and multiple-character reads depending on the needs of the interface.

Single-character reads in Command Mode

HP EDIT uses single-character reads in command mode. Users may issue commands by pressing single keys. For instance, when the 'd' key is pressed, the character over the cursor is deleted. The Delete Character command is executed immediately; there is no need to press either the Return or Enter key. The use of single-character reads in command mode is reasonable because the user is not expected to be entering a large number of commands over a short period of time. The risk of losing commands is minimal since a user typically will wait for visual feedback from the editor between keypresses.

If the user were to keep the space bar (the Forward Character command) depressed, the editor would ultimately lose a few instances of the key. This is not as serious a problem as it would seem. The user who is depending on a key's auto-repeat feature can hardly be expected to know that the editor has only executed 25 of the 30 Forward Character commands transmitted by the terminal. The loss of characters is acceptable here because it is not perceived by the user.

Single character on a function key

HP EDIT has an optional two-level function key interface. The function keys are programmed with single characters so that they may be read in command mode where single-character reads are the norm.

Use of cursor sensing for each command

HP EDIT, in command mode, cannot read a two-character cursor key sequence. Instead, the editor queries the terminal for the cursor position before executing a command. Cursor sensing is also quite useful when the editor is used over a slow modem connection. The user may use the cursor keys to move the cursor around the screen as fast as the terminal allows. In contrast, cursor movement through the use of the space bar would require considerable data traffic on the communication line.

Large buffer read in Text Entry Mode

HP EDIT's second mode, Text Entry, is invoked whenever a user needs to input or modify text. A user is likely to spend a great amount of time in Text Entry Mode while creating a new file. When switching from Command Mode to Text Entry Mode, HP EDIT performs the followings actions:

1) It straps the terminals so that terminal keys, such as Insert Line and the cursor keys, are transmitted to the system.
2) It posts a large (4K byte) read against the terminal which captures all the keystrokes.

The user may enter and modify text, move around freely on the screen, and take advantage of the terminal editing keys without any intervention from the host system. This design also guarantees that the editor won't lose any user data in Text Entry, irrespective of the user's typing abilities.

When the user presses the Return key, the read terminates and HP EDIT begins processing the buffer. The editor scans through the buffer and emulates the various escape sequences corresponding to the terminal editing keys in order to update the internal data structure properly. Once the buffer has been processed, another read is posted, and the sequence repeats itself until the user exits Text Entry by entering a special key followed by Return.

The Backspace key plays a special role in Text Entry Mode. In an ordinary MPE terminal read, the Backspace never reaches the application. Instead, it is intercepted by the terminal controller which removes the last character from the buffer, but not from the screen. This was not acceptable for our product since we didn't want to have a "ghost" character left on the screen after a Backspace. "Ghost" characters would ruin the "What You See Is What You Get" feel of the editor. HP EDIT solves the problem by using unedited I/O (also called transparent I/O) which allows the Backspace key to be treated like any other key. HP EDIT also declares the Backspace to be the alternate terminating character, thus allowing the editor to regain control when Backspace is pressed. When a read is terminated by a Backspace, the editor will remove the last character from the screen and post another read. In this fashion, HP EDIT achieves a true destructive Backspace.

The function keys operate identically in Command and Text Entry Mode. In Text Entry, the function keys terminate the read after having inserted their single character into the buffer. HP EDIT recognizes the unique character assigned to each function key, executes the corresponding command, and returns to Text Entry Mode. Each function key is programmed with a control character which isn't displayed on the screen when the key is pressed.

Buffered terminal output

HP EDIT attempts to reduce the number of writes to the terminal by buffering up text and terminal control sequences. This is achieved through the use of a central output facility which is called throughout the product. The next section has a few more details about this optimization.

Intelligent use of the DTC LAN controller on MPE/XL systems

HP EDIT I/O is structured so that the number of LAN packets sent to the DTC on an MPE/XL system is minimized. LAN performance can be very important to character-mode applications. Please see the next section for more details.

Thoughts and recommendations for software designers

The development of HP EDIT was a challenging experience which taught us a lot about the proper use of character-mode I/O. Our initial lack of understanding of the MPE I/O structure led us to some fairly creative experiments. Nevertheless, there is definitely something to be gained by making one's own mistakes.

Our original design, based on single-character reads, would have led to an unreliable product; clearly not an acceptable option. It must be emphasized that the lack of typeahead is not the only significant consideration here. Typeahead would solve the apparent problem: the loss of characters, but it would do nothing to improve the overall system throughput.

I find it ironic that the lack of typeahead has often "encouraged" developers to create applications with efficient I/O. Let me make it clear that I am not speaking out against typeahead, but against the abuse of typeahead. Systems such as UNIX and DEC's VMS have to process every keystroke from every terminal, which is quite expensive in terms of CPU overhead. Because of this, these systems cannot handle as large a number of terminals as an MPE system of the same class. A large number of users running keystroke-interactive applications will bring just about any system to its knees, including an HP3000.

Here is a list of thoughts and recommendations for the software designer:

Character mode versus block mode

Developers should carefully evaluate their application's needs before choosing an I/O strategy. In many cases, block-mode I/O will prove a better fit than character-mode I/O, typically yielding better overall performance.

If an application is going to require a substantial quantity of input from the user, and if a function key interface is considered adequate, then the developer should look into block-mode I/O.

If the developer decides that block mode is inadequate, then character mode should be used, but only after considering the recommendations below.

In cases where the amount of terminal I/O is small, either block mode or character mode may be used since the choice of an I/O strategy is less critical. However, user interface considerations can still dictate which I/O strategy the developer adopts.

"Robust" I/O

Developers should strive to achieve "robust" I/O in their applications. One cannot assume that a series of single characters reads will perform reliably in all environments, even though they may work perfectly well on a development machine.

Use multiple-character reads

Character-mode applications should use multiple-character reads whenever possible. In general, single-character reads should be avoided when the user is likely to be entering several significant characters in rapid succession. The use of the alternate terminating character can often provide the additional interactiveness needed. For example, if a program posts a multiple-character read and defines the question mark as the alternate terminating character, the user would be able the press the '?' key at any time to receive appropriate assistance.

Buffering terminal output

Output to the terminal can be optimized by buffering strings. Buffering up character strings and terminal control sequences can be especially beneficial on MPE/V systems. On MPE/XL systems, the DTC will perform automatic concatenation of FWRITEs into as small a number of LAN packets as possible.

FCONTROLs and the DTC

On MPE/XL systems, FCONTROLs should not be inserted between FWRITEs and FREADs. More specifically, FCONTROLs should not be inserted between a prompt and its associated read. The DTC has the ability to combine a series of writes followed by a read into a single LAN packet, assuming that they are sent within a 750 millisecond window. If an FCONTROL is inserted between the prompt and the read, the system will end up sending out three LAN packets: one for the writes, one for the FCONTROL, and one for the read. The prompt would appear on the screen, but the read, which would still be two LAN packets away, wouldn't be ready to accept the user's input.

The recommended sequence is: FCONTROL, FWRITEs, and finally the FREAD. If the FCONTROLs can be isolated at the beginning of the program, so much the better.

These recommendations are offered in the hope that you will not have to repeat our mistakes. Do not fear; there are still plenty of good mistakes to make!

Designing an Image Management System

Lynda Pickering

Hewlett-Packard
Office Productivity Division
Nine Mile Ride
Wokingham, Berkshire RG11 3LL
UK

The 'Paperless Office' has been spoken about since computers were first invented. However, recent reports indicate that the Paperless Office is more of a myth than a reality. A Study conducted by Coopers and Lybrand highlighted that 95% of information that a company holds is still in paper format. Every day, American businesses alone produce more than one billion pieces of paper, and this is expected to double every four years!

Managing both electronic and paper based filing systems can be more difficult than managing a 100% paper filing system. Information is not easily accessible and difficult to locate and retrieve. In today's information retrieval world this is a big issue since customers demand access to more information. The challenge facing companies is therefore to improve the efficiency of information management by integrating 100 percent of information into a single environment.

The 100 percent solution is a Document Image Management (DIM) System. DIM is the electronic capture, manipulation, storage, retrieval and communication of hardcopy documents, whatever their content, to be integrated with existing data processing and office automation applications.

With a DIM Solution paper based information could be duplicated electronically from the start. Using optical storage, databases and local area networks all of a company's information can be filed, stored, retrieved and communicated in seconds. For maximum benefits, these systems should be integrated with existing data processing and office automation applications.

The benefits of a DIM System are unlimited: efficient distribution of all documents, instantly available whenever they are required; immediate access to information company wide, not just in department files; the ability to pull together vital marketing and sales information from all sources - the advantages and applications can clearly be seen.

Identifying the need for an Image Management System is the easy part, selecting and developing a system to best fit user needs is the critical step in ensuring the system brings maximum efficiency. To do this, the various components of Document Image Management System need to be reviewed in turn. These components fall into two main categories: Hardware components and software components.

THE HARDWARE COMPONENTS

At the heart of all DIM systems is the 'control' computer which may either be a 80286/80386 PC, other microcomputers, a mini or mainframe. DIM systems are often characterized as being one of three types:

(i) Single host - these systems consists of a central control computer, either a mini or a mainframe. All image processing tasks are performed by the control computer, which also controls the monitors, scanners, printers and optical disk peripherals.

(ii) Client Server - these DIP systems comprise multiple workstations connected to a centralized file server, mini or microcomputer, which controls the database and an optical jukebox. The workstations control dedicated peripherals for scanning, printing and image display. Multi-tasking is much easier in these systems as the peripherals are independent and each workstation can be optimized for its task.

(iii) Standalone PC Systems - A 286/386 PC acts as the control computer which performs all the image processing tasks and also controls the peripherals.

Input

In any type of image system the hardcopy documents must first be captured. Scanners form the front end to DIM systems, converting documents that exist either on paper or film into electronic form.

The dots that compose a scanned image are called pixels. The greater the density of pixels the higher the resolution of the images. The resolution of an image is expressed in terms of dots per inch (dpi). The appropriate resolution to scan images will depend on the application for which it is being used. Generally, resolutions of 200 dpi to 400 dpi are appropriate to business applications.

For convenience scanners can be divided into four main categories: office or convenience, production or high volume, large format and microform. Office or convenience scanners are

typically restricted to A4 documents or smaller, do not support an automatic document feed and are generally flat bed. Scanning resolutions supported are generally 200-400 dpi. Scanning times are typically 5-10 seconds per page.

Production scanners are designed for high volume, heavy duty use. They are typically flat bed and support automatic document feeds. Some scanners can process both sides of a document simultaneously. Paper size can be up to A3 and resolutions variable from 150-400 dpi. Scan time ranges from 1-3 seconds per page.

Large format scanners are used for engineering drawings. They can handle documents up to the width of A0 and, being typically slot fed, can handle almost arbitrary length. Resolutions of up to 1000 dpi are available. Speeds can be in the order of 4 minutes for an A0 drawing.

For microform, scanners are now available for aperture cards, jacket fiche and 16 mm roll film.

The most appropriate scanner to implement and the most appropriate image scanning resolution will depend on the application for which the Document Image Management System is being used. For example, if the objective is to eliminate hardcopy documents then the highest quality images will need to be generated. On the other hand, if the objective is to place hardcopy information in an electronic filing system to support quick access and retrieval, high performace scanning will probably be a pre-requisite and some sacrifice in image quality acceptable.

Image Compression

Images, having been scanned into the system, are compressed to reduce the number of bytes required for image storage and transmission. Image compression is achieved by 'removing' unused white space from the image. It is normally performed by a hardware board which resides in the workstation. It may however, also be performed by software. Using CCITT Group 4 algorithms ratios of 20:1 are achieved reducing a one page document from 1Mb to 50K.

Storage

Most DIM systems require the use of a complete hierarchy of storage devices, including: RAM in the workstations; magnetic disc drives for storage and the database, system software and temporary storage of image data; and, magnetic tape drives to backup the magnetic discs and optical discs for storage of the image data.

For image storage the basic options provided are either optical disc, magnetic disc or microfilm. Of these, optical discs provide the ideal image storage medium due to its high volume storage capacity, fast access and cost effectiveness.

Unlike microfilm, magnetic and optical disk provide easy on-line access to images once they have been stored. However, although magnetic discs offer the fastest retrieval rates for images its high cost as a storage device prohibits its use in the majority of DIM applications.

Microfilm is the cheapest method of storage and an ideal medium for archival data demanding infrequent access. Its slow speed of retrieval preclude it as an on line storage device.

Optical disks are the most widely used storage media in DIM systems. There are a range of optical storage media in the marketplace today, all of which use laser beams to record and read back information - both analogue and digital. They fall into three categories: read only, write once and rewritable. The read only optical discs, referred to as CD-ROM, are used for database and technical publishing. Since DIM applications support the filing and retrieval of information read only technology is not used. Write Once Read Many (WORM) disks are the most widely used. This will undoubtedly change however as we are now seeing the emergence of rewritable optical discs.

Optical discs offer significant benefits over other storage types. They are capable of very high volume storage - up to 5 Gbytes on one 12 inch disk. This would allow for approximately 100,000 A4 image pages to be stored - the equivalent to 10 large filing cabinets. The cost of this storage is very cheap compared with magnetic media. Optical discs can be used individually on a stand-alone drive or in a jukebox where several optical discs, up to 64 platters with one or more drives, can operate together.

Output devices

Monitors - the availability of high resolution monitors capable of displaying a document image at a high resolution, is key to the widespread acceptance and use of Document Image Processing Systems.

Image display can take place on a variety of different screens, each manufacturer offering different sizes and resolutions. Some manufacturers provide specialized image screens based on Sun or Appollo workstations. The most popular workstations are those provided as a simple upgrade to existing IBM PC ATs, XTs or PS/2s or compatibles. These upgrades involve exchanging the existing PC screen for a higher resolution monitor (with controller and display) suitable for image display.

The screen display allows multiple windows to be used simultaneously, each window allowing the user to perform a different function - for example, one for data processing and one for image manipulation.

Printers - The primary output device for Document Image Management Systems are laser printers. These printers can simultaneously print both text and images. The speed of printing will depend on a number of factors including hardware configurations and image resolution.

SOFTWARE COMPONENTS

There are, of course, a number of levels of software supported in all DIM systems. The two main software components are the operating systems and the application software.

The Operating System Software

The operating system software is generally hardware specific, the majority therefore being proprietary. Proprietary operating systems are often very inflexible, being constrained to the architecture of the computer. Two operating systems not restricted to specific equipment are MS DOS and UNIX. Both of these operating systems are finding very widespread use in Document Image Management Systems.

Unix has proved to be very popular for multiuser DIM systems as it offers a wide range of peripheral support. Unix also offers multitasking and multiuser operation and is therefore well suited to large networks. MS DOS has achieved wide spread acceptance as the workstation operating system offering support a wide range of compatible software.

Application Software

The most important aspect to remember about DIM systems is that they represent a technology and nothing more. The enabling software application of the technology is what is important. This is confirmed by a quote in the Computer Consultants Magazine "Imaging technology is not as important as the applications".

Different companies and even different departments within the same company file and process paper in different ways, creating the need for many types of diversified system applications. For example, in the financial services industries and insurance companies Document Management applications are needed for claims, policy and credit card voucher processing. Government agencies are tendering for systems to handle maintenance manuals, correspondence, documents requiring public access and archives.

Deciding on the most appropriate system software application needs careful analysis of the paper process being automated - what is the document flow of information, how is information indexed and who needs access to the information once filed?

Due to the complexity of these applications, which are totally unique to the environment in which they exist, there is no such thing as a perfect application. At the very best software applications should offer the versatility to be 'customized' to meet individual user needs.

Systems offering the greatest potential benefits are those that offer software application 'development tools'. With these tools applications can be designed to meet the specific needs of the users complementing fully paper information flow and filing needs.

INTEGRATION

In general, the DIM solutions available today are standalone systems and can not be integrated with a company's existing data processing and office automation applications. However, in order to solve business problems effectively, it is important for DIM systems to be integrated with other applications.

A fully integrated DIM solution can substantially improve the efficiency with which a business operates. From the moment a document arrives in the mailroom, it can be captured as an image and passed through the company. This information can then be used alongside other text and data stored electronically.

SUMMARY

Designing and implementing an Image Management Solution to best suit users needs requires a full understanding of the paper process being automated. Once this has been achieved the various hardware and software components of the system need to be evaluated in turn. The ultimate question that needs to be addressed is how the system is going to 'fit in' with the company's existing data processing and office automation applications.

HEWLETT-PACKARD'S DOCUMENT IMAGE MANAGEMENT SOLUTION

Hewlett-Packard recognize the area of Document Image Management to be one of strategic importance in which we can make a significant impact. Currently under investigation we have an Image Management product.

To differentiate ourselves from current competitive products we are planning to develop a one vendor open architecture solution based on industry standards. Initially our solution will be

based on an HP 9000 or a HP Vectra RS dataserver and connected via Officeshare to Vectra workstations. It will be possible to integrate this solution with the HP 3000. Also, our solution will be scalable enabling a customer's Image Management System to grow as their needs expand.

An extensible architecture will let images to be supported as a standard data type and enable the support of new data types in the future, including voice and video. Initially the user interface will be based on MS Windows however will migrate to New Wave in the future for Office Automation integration. The system's open architecture will also enable integration with existing Data Processing Applications.

Information Retrieval in Business

Chris Marshall

Hewlett Packard Ltd
Nine Mile Ride
Wokingham
Berks,
UK.

Abstract:

For some time now Office Systems have allowed users to create text and transmit it rapidly to any part of the world. But until recently the systems to manage, store and retrieve this vital business information have been largely confined to specialist areas.

Now we are seeing the emergence of information retrieval systems in the office extending this service to a wide range of users. Advantages include a reduction in the duplication of information, but the most significant benefits derive from better decision making through timely access to relevant business information.

This paper presents what an information retrieval system can offer to a business and discusses the potential benefits of such systems and the kinds of functionality that will become more common in future.

1.0 Introduction

It is increasingly recognised that information is a commercial commodity. Recent years have witnessed a good deal of activity amongst providers and users of electronic information preparation systems (e.g. word processors, graphics packages, spreadsheets) and electronic information distribution systems (e.g. electronic mail, voice mail, fax). Thus we are well able to produce and distribute increasingly large amounts of the information commodity.

To date there has been less evidence that we have the need or the will to find solutions to assist in the filing, management, and retrieval of the information commodity. Thus we currently experience something of a paradox. We generate and circulate masses of electronic information, but we are poorly placed to handle and effectively utilise that electronic information.

Electronic information filing, management, and retrieval solutions have lagged behind up until now but, in future, the effectiveness of systems which help prepare and distribute

electronic information will be increasingly dependent on the improvement of filing and retrieval solutions to back them up.

A large part of the reason for information filing and retrieval solutions lagging behind has been the relatively weak market pull. End users have perhaps only recently begun to appreciate the full actual and potential value of information. Paper piled on desks, crammed into drawers and filing cabinets, or hidden in boxes, is often regarded by office workers and managers more as a hindrance to work than as a precious raw material.

This paper starts from the proposition that strategically and practically, information must be seen as a precious raw material. The benefits of electronic filing and retrieval systems are becoming apparent and we will touch on a few here. We will also review some of the basic functionality that electronic text filing and retrieval products are starting to demonstrate and review a case study of the cost benefits of implementing such systems. The paper finishes with a look at some emerging future trends.

2.0 Historical Perspective

As companies have recognised that information is a valuable asset they have searched for efficient ways to catalog and retrieve it. In previous decades and centuries, systems were paper based. Even today, paper is still a dominant medium and indeed other physical media such as microfiche and video are also being increasingly used to store information. Text retrieval systems are a response to the problem of managing information and as more electronic information has appeared it followed logically that systems would evolve for electronic filing and retrieval. The first electronic text retrieval solutions were introduced in the early 1970s and were mainframe based. They were largely confined to specialist areas such as libraries, information services, managing research and development information and by lawyers for litigation support.

As more and more information is generated electronically the problem of timely retrieval of relevant information has become increasingly acute and the range of applications for text retrieval systems has grown considerably, since the problems of document management are not unique to any particular industry. As a result there is now a wide range of text retrieval packages available on the PC, minicomputer and mainframe offering a wide variety of functionality and integration. The applications of these products are many and varied and include:

* Correspondence control
* Customer files
* Indexing physical libraries
* Patient records
* Standards & regulations
* Legal information
* Personnel records
* Technical information
* Market intelligence
* Government records

3.0 Problems with Conventional Filing and Retrieval Systems

In order to understand the future opportunities for electronic filing and retrieval products it is helpful to briefly review some of the problems and inefficiencies that arise from conventional paper-based filing and retrieval systems and practice. To do this, it is useful to consider shortcomings in each of the three areas of conventional information filing, management, and retrieval.

FILING:

Sharing - with conventional approaches it is difficult to file information so that it is available to many people and can be efficiently utilised by them all. It often requires the security and expense of human custodians.

Security - securing physical files can be expensive and difficult.

Common rules - if common rules are applied to the filing of information, it generally much easier to find that information again. However, individuals often find it difficult to be consistent, and this problem is much worse if several people share a filing system.

Where to file? - nearly all documents can be filed using a multitude of keywords and attributes. In physical files this leads to difficulty in deciding which department, drawer or folder to file a document in. Often files end up being put in an inappropriate place or are copied and stored in several places.

MANAGEMENT:

Duplication - often many individuals deal with identical information and keep copies of it in their files. This is a wasteful duplication of time, effort and resources.

Accumulation - over time, old and redundant information tends to accumulate. With conventional systems this causes clutter and makes retrieval more difficult yet it requires a large investment of time and energy to clear out or tidy old files.

Version control - keeping track of which is the latest update of a document, and knowing if someone else is currently revising it is very difficult in paper-based systems.

RETRIEVAL:

Sheer volume - finding specific items in larger document stores is difficult because of the sheer amount of information that has to be searched and browsed. There are few shortcuts to take you directly to the relevant article(s).

Laborious repetition - in conventional systems, when a search for information has to be carried out regularly in order to catch updates and new relevant material, this involves repetition of the same laborious information trawl. There is little opportunity to automate the process.

Cross relevance - stored information becomes a really important commodity when it can be used to highlight cross-relationships between data and important new opportunities. With manual systems it is very difficult to relate important pieces of information that are in separate folders and drawers.

Where is it? - when it comes to tracking down a vital report, it is the moment of truth. Can the correct folder be located easily? Is the report there or is it mis-filed, or has someone else got it, or has it been "lost"? These are common problems with conventional systems.

Electronic filing and retrieval solutions can bring the power of computers to bear on all these "conventional" problems, often reducing them significantly, and in some cases eradicating them entirely, especially when the stored information is predominantly electronic. However, we must be careful not to view electronic systems as a panacea. They cannot in themselves solve all problems and still require careful planning and management if they are to be maximally effective. There is an nice quote from Wilson (1986) which illustrates this point:

> *"If an information store is efficiently organised and indexed, additional information results in an increase in knowledge. But adding information to an inefficiently organised store can result in greater difficulty in accessing existing information and therefore may actually reduce the effective knowledge level."*

4.0 Benefits of Electronic Text Retrieval

Text retrieval packages provide users with a means to quickly locate the information they need and so save time spent manually searching for relevant information. Not only does this reduce costs but it helps make users more productive and effective in their decision making by ensuring they have timely access to all the available information. The indexing options available allow users to be sure that they have access to all the relevant information.

Indexing documents using a text retrieval solution will prevent the "loss" of information which occurs when someone removes a document from its usual place or as team members change and allow new members to be brought up to date quickly. Text retrieval solutions offer a form of structured filing so that even users who are poor at planning information retrieval will be able to retrieve their data at a later date.

Text retrieval packages also facilitate the sharing of information within teams through the central storage of information and help make the right information available to everyone in the workgroup. This eliminates the need for individual copies of information and so reduces duplication and wastage of resources. Updating information becomes a much simpler task when there is only one central copy to update.

The problems of information management are not unique to any particular industry and some typical applications would be:

* Sales and Marketing - A text retrieval system could be used to store product and pricing information. Then a sales or marketing person could use the system to answer any general query on product X. This would mean that others in the company who were not product specialists would be able to answer general enquiries. The benefits of such a system would be the improved turnaround of queries and the increased availability of information.

* Correspondence - All client correspondence could be stored in a text retrieval system allowing the user to quickly locate a letter sent to a client, without having to know when it was sent or which department generated it. The benefits would be improved turnaround on queries leading to greater customer satisfaction.

* Research and Development - With a text retrieval system a company with extensive R&D activity could store all their research activities and results. Then anyone in the company could use the system to find out if any work had been carried out on their area of interest so eliminating duplication of effort.

In summary, the major benefits of electronic filing and retrieval packages over conventional paper-based systems come down to displaced costs, opportunity hours, and productivity gains as follows:

DISPLACED COSTS:

* reduced need for, and use of, paper

* reduced need for photocopy machines

* reduced need for filing cabinets, drawers and folders

* reduced need for space for holding paper-based materials

* reduced labour costs in filing and finding information

OPPORTUNITY HOURS

* less time spent in manual filing of information

* less time spent in manual searching of information

* less time spent photocopying

* less time spent distributing information

* less duplication of filing and retrieval effort

PRODUCTIVITY GAINS

* better decision-making through timely access to relevant information

* reduced interruptions since people need less help to find information

* better control over the management and access of information

* better teamwork resulting from "community" access to group files

5.0 Functionality and Needs Assessment

Retrieval of information using a text retrieval system is based on the content of a document and data recorded in specific fields at the time of indexing. Wide and powerful functionality is now available with the better electronic filing and retrieval systems. Some of the more powerful features include:

* Full text retrieval - this powerful search method allows the user to search for words in the content of a document. Users do not have to remember a document name but can search on any word or phrase and the system will tell the user how many documents contain the search string. Some systems have a keyword in context capability which provides the highlighting of words searched on in the text of a document to allow the user to quickly find the information they need.

* Stop and go lists - Indexing using stop and go lists, allows you to control which words in a document are indexed so helping efficient use of disc space. A stop list is a list of words that are not useful for retrieving information, for example, "then", "the". Words in a document that are in a stop list will not be indexed, so reducing disc space requirements. Alternatively you may wish to use a go list, which specifies only those words that are to be indexed in the text of a document.

* Keyword searching - Keyword systems allow you to manually add indexing information when storing a document such as document name, author and any specific words which indicate the content of the document. The success of this system relies on selecting good keywords initially so that users will be able retrieve information later. Some systems have a synonym

capability to allow users to build a list to help reduce inaccuracies due to subjectivity, for example if a user searches on "LAN" the system will also return occurrences of "Local Area Network".

* Wildcard and truncation - Wildcard and truncation capabilities allow you to search for variations of a particular root of a word, for example searching on "tele*" would return telephone, television and so on.

* Boolean operators and nesting - Boolean operators (and, or, not) allow the user to be quite precise about what information they need and nesting provides the use of brackets to facilitate complex searches.

* Proximity searching - Proximity searches allow the user to define a search for two terms occurring within a specified distance of each other. This capability is especially useful in the area of litigation support for legal departments.

In deciding on a text retrieval system the following questions should be asked:

* how much information will be stored and what overhead will be incurred in terms of disc space?

* how flexible are the indexing and retrieving methods and what sort of response times for searching can be expected?

* who needs access to the system and how can sensitive information be protected?

* how easy is it to print and edit retrieved information and is the system integrated with any other applications?

* how will the user's information management needs change in the future?

Typical facilities offered by sophisticated electronic filing and retrieval systems are probably best understood by focusing on an example system. It is worthwhile briefly considering an example of a text retrieval system offered by Hewlett Packard. This system is called HP File/Library, and is a bolt-on product to HP Deskmanager. It is not the most powerful or basic system available, and it has its own particular strengths and weaknesses as a product. It does however serve as a useful example for our purposes.

HP File/Library consists of a number of flat catalogs, created as necessary by a system administrator. On entering the Library the user is presented with a list of these catalogs. Each catalog contains the items themselves, they have no hierachical structure to them (like DOS directories for example). This has the distinct advantage that users do not need to know where to file an item other than the appropriate catalog. The division and

retrieval of information within the catalog is achieved through the search mechanism.

The system can be used to index any item held online (electronic information) or offline (papers, microfiche, video, etc.).

When an item is filed into a catalog the user is prompted for three of the fixed attributes common to every item held in the library. These attributes are Author, Keywords and Comments and allow you to uniquely identify each item. The system supplies a further 5 attributes for each item indexed : Subject, Creator, File Type, Status and Create Date.

Items are retrieved from a catalog by searching on one or more of the attributes. Each catalog has an automatically maintained key-word dictionary to help find suitable keywords for searching. Searching on the "Comments" attribute will perform a "full-text-search" on the Comments on every item indexed in the catalog.

HP File/Library offers security at three levels (to the Library, the catalogs and the items within the them). This allows you to determine what a user sees and how they can manipulate the items that they have access to.

To prevent two or more users trying to simultaneously update the same item there is a checkout mechanism which lets a user copy a document to his area, edit it and check it back in. While a document is checked out no one else can change it.

There is an archiving facility to allow items to be stored off to magnetic tape. When an item has been archived, references to it are still maintained in the catalog indexes for searching.

A number of extensions to the HP File/Library are currently in development and when completed, these will give the system some additional powerful facilities. Some systems already offer some of these facilities and new developments are taking place all the time. Shortly however, these more powerful facilities will come to represent the basic set of functionality that all users of electronic filing and retrieval solutions will need and demand:

* Full text indexing and searching of the content of items including support for stop and go lists and provision of a keyword in context (KWIC) facility. KWIC highlights the relevant keywords in retrieved documents allowing them to be quickly browsed.

* Validation of the attributes like "keywords" and "author" on filing (so that indexing rules can be enforced) and on retrieval (so that time is not wasted searching for attributes that have not been used to index information).

* Simplified searching methods including a powerful, easy-to-use search command language, and searching directly from keyword and author listings (e.g. return all documents by author "5").

* A PC interface to provide host filing services for PC users.

* Automatic archiving allowing a time to be specified in months, after which, items that have not been accessed will be automatically archived off the system thus reducing accumulation of old and infrequently used documents.

6.0 A Brief Case Study of Benefits of Electronic Filing and Retrieval

An example of how electronic filing and retrieval was used to share information more effectively and efficiently, is at a site of a major car manufacturer. (They actually made use of the HP system mentioned earlier - HP File/Library).

This site has four departments generating varied information for their dealers, for example, current promotions, leasing rates. This information was copied to all the departments by photocopying and manual distribution. All four departments receive calls from dealers and are expected to deal with queries relating to any area.

Finding the information relating to the dealer's query was often a lengthy process as the information may have been mis-filed or not yet circulated or was being used by someone else to answer a different dealer query. The dealers were dissatisfied because of the time taken to resolve the query. The departmental costs associated with photocopying, storing and retrieving the information were high.

This site decided to store and retrieve all information relating to dealers, centrally with an electronic text filing and retrieval system.

Sharing information between the departments using HP File/Library has reduced photocopying and storage costs. Now that all information is stored in one central place, the time taken to retrieve information has been reduced with a saving in personnel costs. The departments have improved their turnaround on dealer queries and offer a more efficient service. The dealers are pleased with the improved turnaround in their queries which allows them to offer a better service to their customers.

7.0 Future Trends

Looking at the use of text retrieval systems today shows that there are a number of areas where text retrieval systems are likely to develop considerably:

* Use of optical disk technology for the storage and rapid retrieval of vast amounts of information will start to grow significantly.

* As optical character recognition systems continue to improve, an increasing amount of information currently held on paper will be transferred to the electronic media.

* An increasing number of the more successful systems will have to offer the sophisticated functionality discussed in section 5.0 of this paper as standard.

* Today, sophisticated text retrieval systems and image management systems are an both areas of significant growth. However the document management system of tomorrow will need to pull together the elements of both these systems to handle compound documents, that is documents containing, text, graphics, scanned images, voice and maybe even video. Document management systems will not only have to retrieve these compound documents but also deal with the issues associated with displaying the information.

* The interfaces to document management systems will become easier to use, allowing you to search for information using powerful yet simple mechanisms and using Artificial Intelligence techniques to help the user retrieve information. There may be a proliferation of hyper-text based filing and retrieval mechanisms.

* Increased integration with other applications and better networking to help users share information.

To remain competitive and productive, companies must start to view their information as a precious commodity. In future, it will be increasingly important to be able to file, manage and retrieve this information using the powerful, flexible facilities offered by electronic filing and retrieval systems.

References:

Wilson. P.A., 1986
"Introducing Electronic Filing"
National Computing Centre Publications Ltd, UK.
ISBN 0-85012-477-8

Taking Advantage of Mapped Files

Rachel Schwab
Hewlett-Packard
Commercial Systems Division
19483 Pruneridge Avenue
Cupertino, CA 95014

INTRODUCTION

The MPE XL mapped file feature can improve application performance. However, many programmers do not take advantage of this feature because they are uncertain about how to access mapped files from their applications. In particular, programmers who use COBOL and other languages that do not support pointers, may find mapped files confusing.

This paper strives to eliminate this confusion by presenting a "how to" approach to writing programs that do mapped access. It reviews what mapped files are, discusses the necessary steps to include in a program that accesses mapped files, and examines an example program.

WHAT ARE MAPPED FILES

The term "mapped file" makes you think that a mapped file is a special kind of file. Actually, a mapped file is not a special kind of file, but rather a special way of accessing a file. Mapped files provide a means of reading from and writing to files without using the file system.

By bypassing the file system, mapped access cuts out a step in getting to the file. Therefore, this type of access is almost always quicker than file system access. However, since the file system is bypassed, the services that the file system provides, including prefetching, are not available.

On MPE XL, the memory manager handles all file requests. When you use mapped files, your program must communicate directly with the memory manager. It does this by using system pointers, which contain virtual addresses where the files reside. At this time, the only languages that support this type of pointer are HP Pascal and HP C.

Do not conclude, however, that if you have a COBOL application, you cannot use mapped files. Your COBOL application could call a subprogram written in HP Pascal or HP C to read from and write to the file. The example program included in this paper will illustrate how this is done.

HOW TO WRITE PROGRAMS THAT USE MAPPED ACCESS

Suppose you have an application that currently uses the file system and you want it to use mapped files instead. The first decision to make is which file to convert to mapped access. Identify the most heavily accessed file in the application; this is the file you should convert into a mapped file. Once you have successfully converted one file, you can decide whether or not to convert the other files the application accesses. An application can access some files using mapped access and others using the file system.

The second decision to make is whether to access the file using the short mapped option or the long mapped option. These terms refer to the size of the pointer used to access the file. Short mapped uses a short pointer (32 bits) and long mapped uses a long pointer (64 bits).

In order to make this decision, consider how the file will be used and how large it may grow. A short mapped file can have a maximum size of four megabytes, and a process can have a maximum of six megabytes of short mapped files open at any time. The file size is determined by the file limit, not by the actual number of records in the file. Long mapped files may be up to two gigabytes long. There is no practical limit on the amount of virtual address space that a process can access with long mapped files.

A STEP BY STEP APPROACH

Lets take a look at the steps involved in changing a file which uses the file system to use mapped access instead:

1) Convert the OPEN statement (or call to FOPEN) to a HPFOPEN intrinsic call. HPFOPEN is the only way to open a file for mapped access.

2) If reading the file, call the FFILEINFO intrinsic to find out how many records are in the file.

3) Replace the READ and WRITE statements with simple assignment statements which access the pointer originally returned from the HPFOPEN call. This pointer will be modified as different records in the file are accessed.

   Since system pointers only exist in HP Pascal and HP C, this is where an application written in another language would need to call an HP Pascal or HP C subprogram to do the reading or writing.

4) If writing to the file, before closing it, add calls to FPOINT and FCONTROL to set the end of file marker.

Lets examine each of these steps.

STEP 1: USING HPFOPEN TO OPEN A MAPPED FILE

On MPE XL systems, there are two intrinsics which open a file, FOPEN and HPFOPEN. FOPEN on MPE XL is identical to FOPEN on MPE V; the parameter list and functionality are exactly the same. Therefore, FOPEN only supports file system features available on both systems; it cannot be used to open a file for mapped access.

In order to use mapped access, the file must be opened with the HPFOPEN intrinsic, which returns a file pointer. This pointer holds the virtual address of the file on the system.

Here is a COBOL program segment that calls HPFOPEN to open a file to write to using mapped access. Following the code segment is a description of each HPFOPEN parameter referenced.

```
01    FILENUM2                PIC  9(9) COMP.
01    STAT.
      05    INFO              PIC  9(4) COMP.
      05    SUBSYS            PIC  9(4) COMP.
01    FILENAME-OPTION         PIC  9(9) COMP VALUE 2.
01    FILENAME2               PIC  X(9) VALUE "%EXAMPLE%".
01    DOMAIN-OPTION           PIC  9(9) COMP VALUE 3.
01    DOMAIN-NEW              PIC  9(9) COMP VALUE 0.
01    ACCESS-TYPE-OPTION      PIC  9(9) COMP VALUE 11.
01    WRITE-ACCESS            PIC  9(9) COMP VALUE 1.
01    LONGMAP-OPTION          PIC  9(9) COMP VALUE 21.
01    LONGPTR2                PIC  9(18)COMP.


CALL INTRINSIC "HPFOPEN" USING FILENUM2, STAT,
     FILENAME-OPTION, FILENAME2,
     DOMAIN-OPTION, DOMAIN-NEW,
     ACCESS-TYPE-OPTION, WRITE-ACCESS,
     LONGMAP-OPTION, LONGPTR2.
```

FILENUM2

A unique file number returned by HPFOPEN.

STAT

A status array consisting of two 16-bit words. If HPFOPEN is successful, zeros are returned in the status array; otherwise, an error number is returned in the first status word and the file system subsystem code (143) is return in the second status word. A positive error number indicates that the file was opened with a warning; a negative error number indicates the file was not opened because of some problem.

PAIRED PARAMETERS

The remaining HPFOPEN parameters are arranged in "item number, item" pairs. Each item number tells HPFOPEN what kind of data to expect in the following item. The MPE XL Intrinsics Manual contains a table of all the valid item numbers for HPFOPEN. Following is a description of the item pairs used in the example.

FILENAME-OPTION and FILENAME2

The FILENAME-OPTION is item number two. Its corresponding item, FILENAME2, contains the ASCII file name delimited on both sides with a percent sign.

DOMAIN-OPTION and DOMAIN-NEW

The DOMAIN-OPTION is item number three. Its corresponding item, DOMAIN-NEW, has value zero which tells HPFOPEN to open a new file.

ACCESS-TYPE-OPTION and WRITE-ACCESS

The ACCESS-TYPE-OPTION is item number 11. Its corresponding item, WRITE-ACCESS, has value one, which tells HPFOPEN to open the file for write access only.

LONGMAP-OPTION and LONGPTR2

The LONGMAP-OPTION is item number 21. Its corresponding item, LONGPTR2, is a 64-bit integer variable, which HPFOPEN will use to hold the virtual address of the file.

STEP 2: USING FFILEINFO TO FIND THE NUMBER OF RECORDS

If you are going to read records from a mapped file, after you open the file with HPFOPEN, you need to call the FFILEINFO intrinsic to find out how many records are in the file. When you read the file, you must make sure that you stay within the file limits. By finding out the file's end of file value you can make sure you never move the file pointer past it.

The following COBOL code segment calls FFILEINFO to find out the number of records in the file.

```
01  EOF-ITEM                    PIC 9(4) COMP VALUE 10.
01  EOF-IN-RECORDS              PIC S9(9) COMP.

CALL INTRINSIC "FFILEINFO" USING FILENUM1, EOF-ITEM,
     EOF-IN-RECORDS.
```

The first FFILEINFO parameter is a file number returned from HPFOPEN. The remaining FFILEINFO parameters are arranged in "item number, item" pairs, similar to HPFOPEN's paired parameters.

Only one item number is shown in this example, EOF-ITEM. EOF-ITEM is item number 10; its corresponding variable, EOF-IN-RECORDS, will hold the number of records in the file. FFILEINFO reads this information from the file label.

STEP 3: USING POINTERS TO READ AND WRITE A FILE

Ordinarily, once you open a file, your program reads from and writes to it using file system intrinsics. With mapped files you don't use intrinsics to read and write to the file. Instead, your program deals with the file as if it were one huge array. It does this by using the file pointer.

HPFOPEN returns the address of the first record in the file. So you can access the first record in the file by referencing the address returned from HPFOPEN. If you want to access other records in the file, you need to increment the pointer by the necessary offset.

Suppose you want to read record number N in the file. You would use the following equation, where N equals the record number, to compute the address of the record in the file. All addresses are byte addresses.

```
RECORD-ADDRESS = POINTER-RETURNED-FROM-HPFOPEN +
                 (RECORD-NUMBER - 1) * RECORD-LENGTH.
```

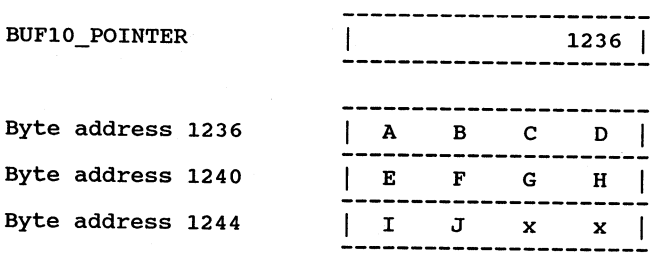
Once you have computed the correct address, you can do the actual reading or writing. The correct address points to the location in the file to read or write. To "read" from the file, you assign the data at the address referenced by this pointer to a program variable. To "write" to the file, you assign a program variable as the contents of the address referenced by this pointer. The pointer manipulation must be done in HP Pascal or HP C.

Assume you have declared a pointer variable called BUF10_POINTER, that points to a character array containing ten characters. Using Pascal, BUF10_POINTER is declared as:

```
TYPE
    BUF10 = ARRAY[1..10] OF CHAR;

VAR
    BUF10_POINTER : ^BUF10;
```

BUF10_POINTER is a pointer variable; therefore, its contents is an address. Using Pascal notation, BUF10_POINTER^ is the contents of BUF10_POINTER's address. This is illustrated in the following diagram:



The value contained in BUF10_POINTER is 1236, a byte address. BUF10_POINTER^ refers to the contents of this address; that is, the contents of byte address 1236. Since BUF10_POINTER points to a ten character buffer, BUF10_POINTER^ equals the ten characters beginning at byte address 1236, "ABCDEFGHIJ".

Now suppose that BUF10_POINTER points to the location in the mapped file you want to read. Also, suppose that MYRECORD has been declared to be of type BUF10, that is, a ten-byte buffer.

In order to read from the file, you store the contents of what BUF10_POINTER is pointing to into MYRECORD. The statement:

```
MYRECORD := BUF10_POINTER^
```

accomplishes this. The length of the data type BUF10_POINTER points to (in this case, ten bytes) determines how many bytes are moved.

To write to the file, you store the data in variable MYRECORD as the contents of the address referenced by BUF10_POINTER. The following statement does this.

```
BUF10_POINTER^ := MYRECORD
```

STEP 4: USING FPOINT AND FCONTROL TO RESET THE END OF FILE

When you use mapped access to write to a file, before you close the file, you must use the FPOINT and FCONTROL intrinsics to tell the file system how many records you have written. If you neglect this step, the file system will close the file with an end of file (EOF) count of zero and all records written will be lost.

A program that writes to a mapped file must keep track of the highest record number written, and pass this number to FPOINT. FPOINT will use this number to set the logical record pointer for the file. After calling FPOINT, the program must call FCONTROL, passing it a control code of six. This tells the file system to copy the file's logical record pointer, which has just been set with FPOINT, to the file's end of file marker. After the call to FCONTROL completes, the file can be closed with FCLOSE.

Following is a COBOL code segment that makes these three intrinsic calls:

```
CALL INTRINSIC "FPOINT" USING FILENUM2, REC-NUMBER.

CALL INTRINSIC "FCONTROL" USING FILENUM2, 6, DUMMY.

CALL INTRINSIC "FCLOSE" USING FILENUM2, 1, DUMMY.
```

EXAMPLE PROGRAM

Lets now examine a COBOL program that uses mapped files. For example purposes, this program calls an HP C subprogram to do mapped reads from an old file and an HP Pascal subprogram to do mapped writes to a new file. The code for the program and subprograms is appended to the end of this paper.

The program begins by opening two files for long mapped access. It first opens CICAT.PUB.SYS for reading. This file contains the command interpreter help text; it was chosen simply because it is a large text file that resides on every MPE XL system. After opening CICAT.PUB.SYS, the program then opens a new file called EXAMPLE for writing.

Once both files are opened, FFILEINFO is called to determine the number of records in CICAT.PUB.SYS. The program then loops reading each record in the file and writing it to EXAMPLE. The loop terminates when there are no more records to read. At that point, EXAMPLE's end of file is marked and set, and the two files are closed.

The loop code is the interesting section of this program. This loop consists of five statements, two for reading and three for writing each record. Statement one reads a record from CICAT.PUB.SYS by calling the HP C routine, CMAP-READ, discussed below (notice that hyphens are converted to underscores in HP C and HP Pascal). After the record is read, statement two increments the file pointer to position it at the next record to read. Statement three writes a record to EXAMPLE by calling the HP Pascal routine, PMAP-WRITE, also discussed below. Statements four and five increment EXAMPLE's record number and position its file pointer at the location of the next record to write.

CMAP-READ

The CMAP-READ routine is shown below. The statement numbers shown on the right are used in the description that follows.

```
struct buf {
   char buf80[80];                    [1]
   };
cmap_read(long_pointer,myrecord)      [2]
struct buf ^long_pointer;             [3]
struct buf *myrecord;                 [4]
{
   *myrecord = *long_pointer;         [5]
}
```

The COBOL program passes this routine two parameters: CICAT.PUB.SYS's file pointer (LONGPTR1), and a buffer to hold the data read (DATA-BUFFER). These formal parameters correspond to CMAP-READ's actual parameters, LONG_POINTER and MYRECORD [2]. Both of these parameters are declared to be of type "struct buf"; structures are how records are declared in C. The "buf" structure type consists of one field, a character array containing 80 bytes [1].

The first parameter, LONG_POINTER, is declared to be a long pointer to a data structure of type buf [3]. Long pointers are declared in HP C using the "^" symbol; avoid confusing this "^" with Pascal's pointer notation described earlier. The second parameter, MYRECORD, is declared to be a short pointer to a data structure of type buf [4]. Short pointers are preceded by a "*".

The only executable statement in the CMAP_READ routine "reads" a record by taking the contents that LONG_POINTER points to and storing it into the memory location referenced by MYRECORD [5]. The asterisks in front of LONG_POINTER and MYRECORD dereference the variables. Dereferencing LONG_POINTER is necessary in order to get the contents of the address LONG_POINTER points to. Since MYRECORD is already a pointer (note that in the COBOL routine, DATA-BUFFER is passed by reference), by dereferencing it, data is actually stored into the address referenced by the pointer.

PMAP-WRITE

The PMAP-WRITE routine is shown below. The statement numbers shown on the right are used in the description that follows.

```
$standard_level 'ext_modcal'$                [1]
$subprogram$                                 [2]
program dummyname;                           [3]
type
   buf80=packed array[1..80] of char;        [4]
   buf80ptr = ^$extnaddr$ buf80;             [5]
```

```
procedure pmap_write(long_pointer : buf80ptr;[6]
                     var myrecord : buf80);
begin
   long_pointer^ := myrecord                    [7]
end;

begin                                           [8]
end.
```

The first statement in the routine is necessary since manipulating system pointers is not part of the industry standard for Pascal [1]. Standard Pascal only understands heap pointers. Therefore, in order for PMAP_WRITE to use system short and long pointers, the compiler must be told to expect them.

Even though PMAP_WRITE is not a program, in order for the source file to compile correctly, there must be a dummy main program. This dummy main program is made up of the program statement at the beginning and the begin/end block at the end [2, 3, 8].

PMAP_WRITE's two actual parameters, LONG_POINTER and MYRECORD, correspond to the COBOL main program's formal parameters, LONGPTR2 (EXAMPLE's file pointer) and DATA-BUFFER [6].

LONG_POINTER's data type (BUF80PTR) specifies a pointer to an eighty byte buffer [5]. The "$EXTNADDR$" compiler option tells HP Pascal that this particular variable will have an extended address, that is, 64 bits instead of 32 bits. The "^" indicates that this is a pointer variable. Used together, "^$EXTNADDR$" tells the compiler that this variable is a 64-bit pointer.

MYRECORD's data type, BUF80, is an eighty character buffer [4]; notice the preceding VAR in the procedure declaration indicating that this parameter is passed by reference.

The body of PMAP_WRITE consists of one statement [7]. The data contained in MYRECORD is stored as the contents of the address referenced by LONG_POINTER.

SUMMARY

The approach discussed in this paper is a straight forward method for writing programs that access mapped files. The program presented reviews each necessary step. Even though this program is simple, you should be able to study its code and apply what you learn when you convert your own applications to use mapped access. Your efforts will be rewarded with improved application performance.

COBOL PROGRAM

```
 IDENTIFICATION DIVISION.
 PROGRAM-ID.      SAMPLE.
 REMARKS.         THIS PROGRAM DEMONSTRATES HOW TO USE MAPPED
                  FILES FROM A COBOL PROGRAM.
 ENVIRONMENT DIVISION.
 CONFIGURATION SECTION.
 SPECIAL-NAMES.
    CONDITION-CODE IS COND-CODE.

 DATA DIVISION.
 WORKING-STORAGE SECTION.

**************************
*  HPFOPEN VARIABLES     *
**************************
 01  FILENUM1                      PIC 9(9) COMP.
 01  FILENUM2                      PIC 9(9) COMP.
 01  STAT.
     05  INFO                      PIC 9(4) COMP.
     05  SUBSYS                    PIC 9(4) COMP.
 01  FILENAME-OPTION               PIC 9(9) COMP VALUE 2.
 01  FILENAME1                     PIC X(15) VALUE
                                       "%CICAT.PUB.SYS%".
 01  FILENAME2                     PIC X(9) VALUE
                                       "%EXAMPLE%".
 01  DOMAIN-OPTION                 PIC 9(9) COMP VALUE 3.
 01  DOMAIN-OLD                    PIC 9(9) COMP VALUE 1.
 01  DOMAIN-NEW                    PIC 9(9) COMP VALUE 0.
 01  ACCESS-TYPE-OPTION            PIC 9(9) COMP VALUE 11.
 01  READ-ACCESS                   PIC 9(9) COMP VALUE 0.
 01  WRITE-ACCESS                  PIC 9(9) COMP VALUE 1.
 01  LONGMAP-OPTION                PIC 9(9) COMP VALUE 21.
 01  LONGPTR1                      PIC 9(18) COMP.
 01  LONGPTR2                      PIC 9(18) COMP.
**************************
*  FFILEINFO VARIABLES   *
**************************
 01  EOF-ITEM                      PIC 9(4) COMP VALUE 10.
 01  EOF-IN-RECORDS                PIC S9(9) COMP.
**********************
*  OTHER VARIABLES   *
**********************
 01  DATA-BUFFER                   PIC X(80).
 01  REC-NUMBER                    PIC 9(9) COMP VALUE 0.
 01  DUMMY                         PIC 9(4) COMP VALUE 0.

 PROCEDURE DIVISION.
 MAIN-ROUTINE.

*   Open CICAT.PUB.SYS using mapped read access.

      CALL INTRINSIC "HPFOPEN" USING FILENUM1, STAT,
        FILENAME-OPTION, FILENAME1, DOMAIN-OPTION,
        DOMAIN-OLD, ACCESS-TYPE-OPTION, READ-ACCESS,
        LONGMAP-OPTION, LONGPTR1.
```

```
      IF INFO <> 0 THEN
        DISPLAY "HPFOPEN ERROR, INFO = ", INFO
        CALL INTRINSIC "QUIT" USING 11.

*     Open EXAMPLE as a new file for mapped write access.

      CALL INTRINSIC "HPFOPEN" USING FILENUM2, STAT,
        FILENAME-OPTION, FILENAME2, DOMAIN-OPTION,
        DOMAIN-NEW, ACCESS-TYPE-OPTION, WRITE-ACCESS,
        LONGMAP-OPTION, LONGPTR2.
      IF INFO <> 0 THEN
        DISPLAY "HPFOPEN ERROR, INFO = ", INFO
        CALL INTRINSIC "QUIT" USING 12.

*     Call FFILEINFO to determine CICAT's EOF.

      CALL INTRINSIC "FFILEINFO" USING FILENUM1, EOF-ITEM,
        EOF-IN-RECORDS.
      IF COND-CODE <> 0 CALL INTRINSIC "QUIT" USING 21.

*     Copy records from CICAT.PUB.SYS to EXAMPLE.

      PERFORM 100-MAP THROUGH 199-MAPEXIT UNTIL
        REC-NUMBER = EOF-IN-RECORDS.

      DISPLAY REC-NUMBER, " WRITES MADE TO THE MAPPED FILE.".

*     Mark and set EOF on EXAMPLE.

      CALL INTRINSIC "FPOINT" USING FILENUM2, REC-NUMBER.
      IF COND-CODE <> 0 CALL INTRINSIC "QUIT" USING 31.
      CALL INTRINSIC "FCONTROL" USING FILENUM2, 6, DUMMY.
      IF COND-CODE <> 0 CALL INTRINSIC "QUIT" USING 32.

*     Close both files.

      CALL INTRINSIC "FCLOSE" USING FILENUM2, 1, DUMMY.
      IF COND-CODE <> 0 CALL INTRINSIC "QUIT" USING 33.

      CALL INTRINSIC "FCLOSE" USING FILENUM1, DUMMY, DUMMY.
      IF COND-CODE <> 0 CALL INTRINSIC "QUIT" USING 41.

      STOP RUN.

 100-MAP.
*     Call the HP C subprogram to read from CICAT.PUB.SYS.

      CALL "CMAP-READ" USING \LONGPTR1\, DATA-BUFFER.
      ADD 80 TO LONGPTR1.

*     Call the HP Pascal subprogram to write to EXAMPLE.

      CALL "PMAP-WRITE" USING \LONGPTR2\, DATA-BUFFER.
      ADD 1 TO REC-NUMBER.
      ADD 80 TO LONGPTR2.

199-MAPEXIT.   EXIT.
```

HP C ROUTINES

```
struct buf {
   char buf80[80];
   };

cmap_read(long_pointer,myrecord)

struct buf ^long_pointer;
struct buf *myrecord;

{
   *myrecord = *long_pointer;
}


cmap_write(long_pointer,myrecord)

struct buf ^long_pointer;
struct buf *myrecord;

{
   *long_pointer = *myrecord;
}
```

HP PASCAL ROUTINES

```
$standard_level 'ext_modcal'$
$subprogram$

program dummyname;

type

   buf80=packed array[1..80] of char;
   buf80ptr = ^$extnaddr$ buf80;


procedure pmap_read(long_pointer : buf80ptr;
                    var myrecord : buf80);
begin
    myrecord :=  long_pointer^
end;

procedure pmap_write(long_pointer : buf80ptr;
                     var myrecord : buf80);
begin
    long_pointer^ := myrecord
end;

begin
end.
```

TITLE: Information Management Update

AUTHOR: Pat Adamiak

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 5870

Using the Alternative Loan Method to Make the Lease/Buy Decision

Gary F. Rea

Capitol Health Care
PO Box 12625
Salem, OR 97309

Introduction

The choice of leasing a capital asset to acquire its services has been available to decision makers for over 20 years. While leases can have many different terms, features and conditions, the types of leases that a computer professional would normally be confronted with fall into two general categories: financial leases and operating leases. To help in understanding the analysis that is presented here, it is helpful to understand some of the basic similarities and the differences between the two types of leases.

With both types of leases, ownership of the asset remains with the lessor (seller). With a financial lease, the lessee (buyer) is normally committed to a legally enforceable and noncancelable contract that specifies a fixed number of payments at a specified amount. Should the lessee default on this contract the lessor could sue for the value of the remaining lease obligation. In addition, with a financial lease, the lessee is normally required to maintain the asset in good condition at his or her own expense. Generally Accepted Accounting Principles (GAAP) require that this type of lease be accounted for as a "capital lease" and recorded on the books of the firm as the acquisition of an asset and the incurrence of a liability, similar to the financing of a purchase.

An operating lease differs from a financial lease in two ways. First, the lease may be canceled by the lessee. This cancellation may have some restrictions such as giving notice within a certain period before termination or termination only after a certain limited amount of time has elapsed from the start of the lease. In either case the lessee is not obligated for the full term of the lease. Second, the lessor is typically responsible for the maintenance of the asset, the cost of this maintenance being included in the lease amount. For financial accounting the lease payment is considered the same as rent.

Various so-called advantages of financial leases over a purchase have been put forth, including:

- Leasing allows for 100% financing without up-front equity payments; thus, cash and working capital can be conserved and freed for use in other projects.

While it is true that a lease may allow 100% financing, the same may be available through other financing means. It could be argued, however, that the alleged advantage of conserving cash and working capital and freeing them for use in

other projects is more likely the result of sub-optimal managerial constraints placed on the capital budget. Whether a project should be undertaken is a function of the return from the project at a given cost of capital, not whether it can be leased. A financial leasing arrangement might be viewed as being similar to secured debt financing and, as stated before, is required by GAAP to be recorded in a similar manner. With a lease the firm has entered into an agreement to make a series of fixed payments. This agreement is difficult to cancel, and financial distress can accompany any violation of the agreement. Leasing also produces financial leverage. Each of these are similar to secured debt and as such has a similar effect on the firm's capital structure. The decision to lease should be based on the lowest financial cost to the firm, not on preserving working capital.

- Leasing can preserve lines of credit or avoid violating restrictive loan covenants, which prohibit the issuance of additional debt.

As just stated, leases look much like secured debt. Because of the required fixed payments, it could be that the payments to other debtors may be at higher risk. This higher risk will be reflected in future higher risk premiums paid by the borrower. It may be true that the firm may get a one-time gain by leasing, but it is likely that they will pay for it in the future through higher interest rates on borrowing.

- Leasing reduces the risk of obsolescence present during periods of rapid technological changes, allowing for flexibility in changing equipment.

It would seem that someone would have to bear the financial cost of obsolescence. If both the lessee and the lessor foresee the same cost of obsolescence the lease rate would reflect that cost, and the lessee would be no better off leasing than buying.

- Leasing terms themselves are often more flexible than those of other debt agreements.[1]

Typically the financing terms available to a firm are a result of the credit status of the firm and the nature of the lending institution that the firm is dealing with. The flexibility of financing may be more a function of shopping for rates and terms rather than being an inherent feature of leasing.

The point is that leasing does not in itself carry clear advantages in all cases and that each case must be examined to determine the most advantageous method to acquire an asset. This paper discusses one method of making that evaluation.

[1] These advantages were presented in an article entitled "The Growing Popularity of Leasing" in the Controllers Quarterly, 1988. The comments are the authors.

Background

The equivalent loan method depends on the differential cash flows between leasing and purchasing. It also requires that the purchase alternative be separately evaluated to determine if the asset is worth purchasing. This evaluation of the purchase alternative looks at the NPV of the cash flows from the acquired asset, discounted at the appropriate cost of capital that has been adjusted for the risk associated with acquiring the asset. The details of this analysis is beyond the scope of this article and the assumption will be made that it is to the benefit of the firm to acquire the asset. The only outstanding question is what method to use to acquire the asset, Lease or Purchase?

The analysis is based on a computer system that was recently acquired by Capitol Health Care that consists of an HP-950 processor, 4.5GB of disk, line printer, tape subsystem and various terminal controllers and interfaces. The purchase price was $466,700. Pertinent terms of the lease specify 60 monthly payments of $9,841.42, which included personal property tax. The payments are due at the end of each period. There is also a 10% purchase option at the end of the lease term. Other assumptions include a corporate tax rate of 35% and that all deductions will be usable by the firm. To reduce the size of the illustrations the analysis has been done using annual figures over the five-year period. It should be noted, however, that the accuracy of the analysis will suffer to the point of giving erroneous results if this assumption is used in practice.

Differential Cash Flows

As mentioned, the analysis is based on the differential cash flows that result from leasing as compared to the purchase option. The first step in the analysis is to identify these differential cash flows. For this example, five cash flows were identified: 1) the cash outlay for the equipment, 2) the tax deduction resulting from depreciation, 3) the Personal property tax on the equipment, 4) the Lease payments, and 5) the salvage value of the equipment at the end of the lease. The actual cash flows that will result from a lease are a function of the transaction and will vary with each lease arrangement. Also note that we are interested in the cash flows, not necessarily the accounting entries that result.

The differential cash flows from leasing are shown on lines 1 through 4 and line 6 in Figure 4. We need to examine the rationale, behind and the computation for each, of these differential cash flows.

A. The cash outlay for equipment:

The cash outlay for the equipment is considered a positive cash flow that results because the firm need not expend the cash to acquire the equipment. The value is simply the purchase price of the equipment.

B. The tax deduction resulting from depreciation.

The depreciation tax subsidy that results from the tax deductibility of depreciation will be lost if the equipment is leased. Because of this "loss" the

resultant tax subsidy foregone is considered a negative cash flow. The computation of the lost depreciation tax subsidy is shown in Figure 1.

Figure 1.

Calculation of Depreciation Tax Subsidy

Total Depreciable Outlay: $466,700
Tax Rate: 35.0%

YEAR	MACRS Rate	Depreciation Charge	Depreciation Tax Subsidy
t = 1	20.00%	93,340	32,669
t = 2	32.00%	149,344	52,270
t = 3	19.20%	89,606	31,362
t = 4	11.52%	53,764	18,817
t = 5	11.52%	53,764	18,817

Because it is normally to a firm's advantage to use accelerated cost recovery methods to compute depreciation for tax purposes, the current Modified Accelerated Cost Recovery System (MACRS) method specified by the IRS has been used in this example. The MACRS method specifies that five-year life property be recovered using the 200% declining-balance method over the life of the property with a half-year convention, with a switch to straight-line method in order to maximize the deduction. This all sounds complicated, but you just use the tables supplied by the IRS.

The depreciation charge is computed by multiplying the MACRS rate for each period times the total depreciable outlay. To compute the lost depreciation tax subsidy, the depreciation charge is multiplied by the corporate tax rate, in this example 35%. The result (depreciation tax subsidy) is the additional tax the firm will pay as a result of not having the tax deductible depreciation. This cash flow is shown as a negative value on line 2 of figure 4.

C. Personal property Tax:

The Hewlett-Packard lease payment includes personal property tax and as a result the lease option produces a positive cash flow. This cash flow results from the fact that the personal property tax does not need to be paid directly under the lease option but would occur if the equipment were purchased.

Figure 2.

Calculation of Personal Property Tax Savings

Value of Personal Property $466,700
Personal Property Tax Rate $33.75 Per $1000
Tax Rate 35.0%

Year	Factor	Taxable Value	Property Tax	After Tax Savings
t = 1	75.0%	350,025	11,813	7,678
t = 2	50.0%	233,350	7,876	5,119
t = 3	30.0%	140,010	4,725	3,071
t = 4	20.0%	93,340	3,150	2,048
t = 5	10.0%	46,670	1,575	1,024

Personal property tax is based on a tax rate per $1000 of assessed value. The trick is to determine what the assessed value of the equipment will be in future years. In this case a call to the local county tax assessor determined that the county used the percentages of original purchase price shown in Figure 2 to determine taxable value. The tax saved was then computed using the tax rate per $1000 times the taxable value for each period. This tax savings must be adjusted to reflect its after-tax value. This adjustment is required because the firm will not have the tax deduction that would have resulted if the property tax had been paid. The after-tax savings is computed by multiplying the property tax by one minus the corporate tax rate. The result is shown in Figure 2 and on line 3 of Figure 4.

D. After-tax Lease Payments:

The annual lease payments are probably the easiest of the differential cash flows to identify, they only result if the equipment is leased. The important thing to remember is that the analysis is interested in after-tax lease payment cost. Lease payments are tax deductible, and as such the resulting cash flows needs to account for the tax savings resulting from the tax deductibility. The computation of the after-tax lease payment cash flow is shown in Figure 3 and is one minus the corporate tax rate times the lease payments for the period.

Figure 3.

Calculation of After-tax Lease Payments

Annual Lease Payments: $118,097
Tax Rate: 35.0%

YEAR	After-tax Lease Payment
t = 1	76,763
t = 2	76,763
t = 3	76,763
t = 4	76,763
t = 5	76,763

The Equivalent Loan

When the differential cash flows for each period are totaled, the result is the differential stream of net cash flows resulting from leasing as opposed to purchase. Those flows are, implicitly, the differential after-tax financing flows associated with the lease transaction[2]. Hewlett-Packard is implicitly lending $466,700 to the firm at t=0 and assuming responsibility for property taxes in exchange for promised lease payments and the depreciation tax subsidy. The equivalent loan looks at what the firm might borrow if the cash flows beginning at t=1 and ending at t=5 on line 5 in Figure 4 were promised to a lender. The equivalent loan amount can then be compared to the implicit loan in the lease. If the equivalent loan exceeds the implicit loan in the lease then purchase is superior to a lease. If the implicit loan in the lease is greater then the equivalent loan then the lease is superior.

To compute the equivalent loan the differential cash flows for t=1 through t=5 are discounted using the after-tax, risk-adjusted discount rate. This rate should reflect the risk associated in the cash flows. The best approximation would normally be the lessee's secured borrowing rate. This computation is shown on lines 7 through 10 of Figure 4.

As can be seen in Figure 4, the equivalent loan resulting from the differential cash flows from leasing using a discount rate of 8.125%[3] is $415,618. Using the decision guidelines defined above, the lease would be superior to purchase because

[2] Archer, Stephen H., Choate, G Marc, Racette, George; Financial Management, New York: John Wiley & Sons, 1983;

[3] A secured borrowing rate of 12.5% (Prime +1) has been assumed along with the corporate tax rate of 35%.

the implicit loan in the lease, $466,700, exceeds the equivalent loan, $415,618. One other thing needs to be considered however, before the final decision is made and that is the effect of salvage value.

Because the lease calls for an optional 10% purchase option at the end of the lease term, the lessee is either forgoing a benefit in the future through the lost salvage value or is going to incur an additional expense to acquire ownership of the equipment. Either of these events produce a negative differential cash flow that must be considered. The amount of this cash flow can be difficult to determine. The upper limit of the cash flow is set by the 10% purchase option provision in the lease. The lease however, allows the lessee to selectively purchase the pieces of equipment under lease that he or she may wish to acquire. Good business practice would suggest that the pieces with market prices that exceed the 10% purchase option price would be acquired from the lessor and the other pieces would be acquired on the open market. The cost on the open market should of course include the cost of shipping, installation and other associated costs. If the lease did not include a purchase option provision, the lessee would need to make a "best estimate" of what the future value might be.

In this example with a 10% purchase option provision, the conservative approach would be to consider the full 10% as the differential cash flow. To compute the present value of this cash flow, a higher after-tax discount rate to reflect the greater uncertainty, i.e. higher risk, associated with this cash flow is used. This example uses an after-tax discount rate for the salvage value of 15% which produces a present value of the purchase option of $23,203. The net equivalent loan is shown on line 13 in Figure 4 and is the sum of the equivalent loan value and the purchase option value. As can be seen, the net equivalent loan of $438,822 still exceeds the implied loan of the lease, and as such the lease is superior to purchase. It should be noted that the consideration of salvage value will only make the equivalent loan value higher. For that reason, the salvage value computation need only be made if the implied loan in the lease exceeds the equivalent loan after the first analysis.

SUMMARY

It is important to remember that the circumstances of the example are what resulted in the lease being superior to purchase in this case. The particulars of each equipment acquisition need to be analyzed to make the correct decision. The analysis of the leasing and purchasing options can be done through the following steps:

1. The differential cash flows (after-tax) from the proposed lease need to be determined and discounted at the appropriate after-tax discount rate.

2. The salvage value must be estimated and also discounted at an appropriately higher after-tax rate to reflect the higher risk in that part of the analysis.

3. If the resultant equivalent loan exceeds the implicit loan of the lease, purchase is superior to lease. If the implicit loan in the lease exceeds the equivalent loan then the lease is superior.

As a final note, the Hewlett-Packard lease terms in this example are somewhat different from a large number of leases that you will encounter:

1) Payments are due at the end of each period. Many leases call for payment at the beginning of each period. 2) No additional up-front costs, "first and last" months payments. 3) The payment includes property taxes that are based on national averages which favor us in Oregon. 4) Although not considered in this analysis, the selective purchase option is attractive.

All of these features tend to favor the lease option over the purchase. The point to remember is that the terms of the lease and the resultant cash flows must be understood.

Differential Cash Flows from Leasing

		t = 0	t = 1	t = 2	t = 3	t = 4	t = 5
1. Outlay if Computer is Purchased		$466,700					
2. Depreciation Tax Subsidy			(32,669)	(52,270)	(31,362)	(18,817)	(18,817)
3. Personal Property Tax Savings			7,678	5,119	3,071	2,048	1,024
4. After-tax Lease Payments			(76,763)	(76,763)	(76,763)	(76,763)	(76,763)
5. Net Cash Flow from Leasing		466,700	(101,754)	(123,914)	(105,054)	(93,532)	(94,556)
6. Purchase Option							(46,670)
7.	Tax Rate	35.0%					
8.	Secured Borrowing Rate	12.5%					
9.	After-tax Discount Rate	8.125%					
10.	Equivalent Loan		415,618				
11. Purchase Option Discount Rate		15.000%					
12.	Purchase Option		23,203				
13.			438,822				

Figure 4.

DIRECT CONTRACTORS VERSUS AGENCIES, THE NEW TAX LAW IN PERSPECTIVE

Carl E. Melchior
Independent Contractor's Group
P.O. Box 820816
Dallas, Texas 75382
(214) 690-7847

The following information comes in part directly from the Media Department of the Internal Revenue Service, Washington, D.C. and the House, Ways and Means Committee on Capitol Hill in Washington, D.C.

So you're a company looking to hire contract computer personnel. If you're like most companies then you are probably so confused by the new tax laws and have heard so many stories that you don't know where to start or what to believe. Well it is my task to enlighten you on what the Tax Reform Act of 1986 really does mean in the case of contract personnel. Some frequently asked questions are:

1). *Can I still hire a contractor to work directly for my company?*

2). *Do I have to use an agency to hire a contractor?*

3). *What are my tax liabilities if I hire a direct contractor?*

In recent history probably no other Tax Reform Act has been as misinterpreted as Section 1706 of the Tax Reform Act of 1986. In just a few basic steps I will try to clarify what all of the implications of this section are. After reviewing the IRS information I will then attempt to clear up the many misconceptions involving agencies and direct contractors.

The main question that looms in the DP manager's mind is, can I hire a contractor directly or does the new tax law force me to hire through an agency? Well the IRS clarification of the new tax law specifically states that any company may continue to hire direct contractors without having to go through an agency as long as the same IRS criteria for contractors that existed before the new tax law was passed are still met.

On January 1, 1987 the Department of the Treasury Internal Revenue Service released their clarification of Section 1706 of the Tax Reform Act of 1986 and it reads:

"WASHINGTON -- The Internal Revenue Service announced today that Section 1706 of the Tax Reform Act of 1986 does not automatically convert certain technical service specialists from independent contractors to employees for income and employment tax purposes. The specialists are to be classified as independent contractors or employees under generally applicable common law standards, the IRS added, responding to misunderstanding over this matter.

Thus, for example, a technical service specialist who was treated as an independent contractor for employment tax purposes before the Tax Reform Act will continue to be recognized as an independent contractor for such purposes after the fact if the specialist is an independent contractor under the common law standards."

And it goes on further to say: "By its literal terms, Section 530 (d) does not technically apply to a "two-party" situation. Thus if a taxpayer directly contracts with a worker (the specialist) to provide the specified services not for another person but for itself, then Section 1706 of the 1986 Act does not alter the extent, if any, to which Section 530 of the 1978 Act is available to that taxpayer."

After reading the above clarification from the IRS, the next obvious question is what are these common law standards that were mentioned. Well, what the IRS has done is issued a series of guidelines commonly referred to by contractors and agencies as the "20 Questions". It is the answers to these questions that determine if the specialist is really an independent contractor or an employee. Since the questions can be rather involved let me just give you an overview of the kinds of topics the IRS looks at. What they really want to know are things like does the employer have control over the specialist as to how, where and when the work is to be done. Can the employer for example tell the specialist that he can not provide services to any other companies while on the employer's project or that he can not take days off in the middle of the project or that he must follow a certain order in which the project is to be completed? Does the employer provide tools and training or is he contracting for the specialist's specific training already?

In most situations the contractor is really his own company and it is his own company rules that he abides by, not the company he is contracting to.

Now lets look at one other area of interest. You are probably saying to yourself right about now that there probably are a few assignments around your DP shop that would really be appropriate for some outside direct contract help but is it really safe from the IRS point of view to hire a direct contractor? Maybe the following will put your mind at ease.

Supposedly there is a story floating around about a man who wanted to open a take-out chicken stand. After doing some basic business research he found that he would have to hire a consulting company to do the demographics research to see where to place his stand at a cost of about $100,000. Well needless to say that anybody who is starting out in the chicken business knows that if you layout that kind of cash for research, then you won't have any money left to open your stand with. What to do? Well thanks to some ingenuity (I think this guy used to be an old SPL programmer) our chicken man called the franchise office of Mc Donalds and asked about their demographic requirements. Finding that their minimum requirements were much higher than his, he did the only thing he could. That's right, he opened his stand right across the street from Mc Donalds and now he doesn't own just one stand but hundreds. Somehow I feel that this story is true but even if it isn't it still holds a valuable lesson to be learned.

Look to the large corporations first and you just might find your answer.

We will assume that the large multinational corporations of the world have some of the finest tax and legal consultants that money can buy. I think we're safe with that assumption. It also stands to reason that if these large corporations thought that they were flagrantly violating any IRS laws that they would not risk their corporate name or image by doing so. So maybe by looking at these corporate giants who have researched the tax laws we too can learn something just like our chicken man. Oh, I know what your asking now. Names. You want some names of these giants so you'll feel at ease. Well have you ever heard of Boeing Electronics, or maybe Exxon Chemicals. While we're at it you can always rely on the old favorite; yes, that's right, Hewlett-Packard. All of these companies have found no problem with the new tax law and that's because there is no problem with the new tax law.

Now that the question of hiring direct contractors has been cleared up it is time to try to clear up the misconceptions surrounding agencies and the steps needed to insure protection of your company. What are the advantages of hiring a direct contractor instead of going through an agency?

To answer these questions let's first start with a little background material on the way things used to be. Before the Tax Reform Act of 1986 everything was quite simple. In the case of companies seeking

contract personnel all they had to do was call a contract house and ask for a contractor to be sent out. The agency then provided the specific contractor and billed the company twice per month for the total amount of hours times rate. The agency would then in turn issue a check to the contractor for the amount of hours times rate, minus the agency fee of course. Simple. No taxes, no W-2 forms, no year end accounting hassles. Quite simply put, the contractor had the worry and headache of how much tax to pay and when to pay it.

Let's look at the advantages of hiring through an agency first. Normally the reason that most DP managers hire through an agency is that they don't have the time to run advertisements in the job classifications of the newspaper and then search through a stack of resumes, most of which are unqualified, and then interview fifteen candidates for one position. It is much easier to let a third party source do the hunting for you and just send three top candidates to your office. The other reason is that DP managers seem to feel that if for some reason the contractor should leave in the middle of a project, that the agency could then supply another contractor on short notice.

There is one fatal flaw in this logic though. Let's analyze the above scenario and see what it really holds in the case of direct contractor versus agency. We will take three contractors, A, B and C and will suppose that you have a one year contract to be filled. Contractor A will be a direct contractor and contractors B and C will be agency contractors. For the sake of argument we will also assume that contractor C will represent any qualified contractor that could fill your position at a future date.

Case #1. Agency Contractors:

Your contract starts January 1st and everything is going well until on June 1st your agency Contractor (B) up and moves to Silicon Valley (supposedly to start his own software company). The first thing as a DP manager you do is call the agency up and ask them to quickly send you a contractor who is qualified and can come into the middle of your project and perform a miracle. They kindly send you contractor(C).

Case #2. Direct Contractors:

Your contract starts January 1st and everything is going well until on August 1st your direct contractor(A) up and moves to Silicon Valley (supposedly to help save contractor(B) before he has to file bankruptcy). The first thing you do as a manager is call the agency and ask them to quickly send you a contractor who is qualified and can come to your project and perform a miracle. They kindly send you Contractor C.

Now as a manager of your data processing center what have you accomplished. Well in either case it doesn't make a bit of difference if you go through the agency or go direct because hiring a direct contractor as opposed to an agency contractor does not affect what contractors are available at any point in the future. You will always be able to get Contractor C.

But if you look closely at the second case you will notice something that the first case didn't offer. In the second case you should have saved the agency fee since it was a direct contractor you hired. Now a days the standard agency fee is 25 % of the billing rate. You don't have to be an Einstein to figure that the savings on contract rates of 25% over six months can add up to thousands of dollars. Naturally not all contractors charge the same rates but usually they can give you a better rate than an agency by splitting that 25% with your company.

Another aspect of agencies to look at is that it is usually standard industry practice to try to get your company to sign a short term agreement, maybe three months with the possibility of extension, so that if you have a six to nine month assignment a rate increase can be requested at the end of each three month agreement. But beware. Normally the bulk of the increase you give to the agency does not get passed on to the contractor but stays with the agency itself. It is generally a good practice on all rate increases to check with your contractor to make sure all parties are aware of such changes in agreements.

Still another misconception in dealing with agencies is that the contract you sign with an agency is the same as the contract the agency signs with the contractor. This is usually never the case. Many contracts between company and agency may spell out a thirty to sixty day Notice of Intent to end the contract by the agency, however the agency's contract with the contractor may only spell out two weeks. This means that should your contractor decide to leave with only two weeks notice in the middle of your project that there is probably nothing legally you can do to force him to stay or to recoup damages from him. Remember your contract is with the agency not the contractor. When dealing with direct contractors this problem is alleviated as there is only one contract in question.

Still another problem that may arise out of a dual contract situation is the following. Beware of non-compete agreements!!! If there is one thing that can not be stressed strongly enough it is the area of the contract that deals with non-compete clauses. The basic premise of this idea started out with good intentions but has turned many companies' plans into nightmares. Normally you will always find a clause in your contract with the agency specifying that once the contract has ended you will not hire the agency contractor for any more projects for a specified period of time.

This insures that you won't ask an agency to furnish a person for six months and then decide after one month to cancel the agency contract and hire the contractor directly. This situation seems reasonable. But what has started out as reasonable has been taken too far by many agencies. The real question that needs to be answered is not what is the agency's role in the placement but rather at what point has the agency received total compensation for their work. Most agencies feel that as long as the contractor works continuously for the client company that they are entitled to their fee, no matter how many extensions to the original contract. Now most companies are all for being able to protect their interests but then again they're also for fair play too. It would seem that if a company hires a contractor for a three month assignment through an agency and that if they decide to keep him on for additional assignments it is because they like the quality of that particular contractor's work, not because of the job the agency is doing.

Normally when you work with an agency you need to decide up front what terms are agreeable should you decide to use the contractor for additional assignments. One year non-compete agreements are absurd. Six month non-compete agreements are not much better. A fair rule of thumb would be an agreement to hire the agency contractor for a certain period of time (i.e. three to six months). If after the initial contract is over your company wants to extend the contract further then you would be willing to extend for the same term as the original with the maximum time period of six months. This means that if you hired the contractor for three months you would be willing to extend for three months. If you hired for six months the maximum extension would be six months. Then again if you hired for nine months the maximum extension would still be for six months. Remember that as the DP manager it is you who should be making the rules, not the agency. If both sides are reasonable usually a very fair settlement can be arranged.

Before signing any contractual agreement with an agency, make sure that both you and the contractor receive a release stipulating at what point in the future both you and the contractor may work freely together. This release should state that not only is your company free to hire the contractor directly but that they are also releasing the contractor from his non-compete agreement with the agency. The release also needs to be for all company sites, not just your site in particular. You need to make sure that the agency sends a release to the contractor before signing any agreements. If you don't you could get stung. Remember your dealing not only with a client-agency contract but also with a contractor-agency contract.

Now you're probably asking, "How many times does this situation ever really arise?" Let me just say that this is the rule not the exception. I have seen many times during my career major problems stemming from non-compete agreements. It's your right as a DP manager to be entitled to a fair and complete understanding of all contingencies before signing any contract.

Now don't get me wrong. There are many agencies that are completely above board and these are the kind the industry appreciates. But as in any industry there are also those that are not. To be safe many times agencies will tell you that they prefer not to discuss the details of the contractor's contract with you and that they will not discuss your company's contract with the contractor. All this does is lead to confusion. If an agency-contractor contract can affect your company in any way (i.e. non-compete agreements, etc.) then you have a right to see the contractor's contract before you sign your agency-company contract. Insist on it. Any reliable agency will be glad to show it to you. Many direct contractors today are willing to work without a formal contract as long as they can obtain a letter from your company on company stationary listing the major points of the agreement (i.e. rate per hour, length of contract, procedures for termination of contract).

Now assuming that the previous information has made some basic common sense, let's look at how we can apply those lessons to your DP shop in order to save you a lot of headache and agony.

The basic decision most DP managers face when it comes down to contractors versus agencies is usually when do they need the contractor to start and how long is the assignment to be. Naturally the longer the assignment the more money it is going to cost for agency fees. In a short term scenario (three months or less) an agency contractor might be your best bet. But what about those longer term assignments. In most cases a direct contractor would probably be your best bet. What about those cases where an assignment has popped up all of sudden and you need a contractor who could have started yesterday. Once again an agency is probably your best bet. But, if you have a 30-day or more lead time before your project starts then a direct contractor might serve your needs better.

Now I know you are probably wondering what is the quickest, easiest, and least painful way to go about bringing in contractors to your project sight. Well as with anything, planning is the answer. The time to start lining up outside contractors is not the day before the project is to start, just as the day to order your computer equipment is not the day before you're ready to start programming. What many companies have found is that the best way to handle consulting needs is to start a little early in establishing a pool of future contractors from which to choose. Probably the best way to insure that contractors you pick are qualified is word of mouth. There are numerous sources from which to receive information on qualified contractors. Just attending one users group conference can introduce you to many companies who could pass along names of contractors they have used in the past.

As a DP manager you should start using some outside contractors on some very short term assignments perhaps of one to four weeks. If

you use this method and the contractor is qualified you can then put his name into the contractor pool. If for some reason he is not qualified then you have only invested a minimal amount of money and time and will not have the misfortune in the future of a possibly ruined project on your hands. To be quite honest I have seen more than one project in my time ruined by both direct and agency contractors because management took them at their word that they knew their craft when they really didn't. Once you have created a small pool of qualified contractors there is only one other thing to take care of and that is you need to give your contractors good advance notice of projects that will be coming down the line. Remember contractors have to make a living too. You just can't call them up the day before the project and expect them to be ready tomorrow. If however you know that sometime in the next two to three months you will probably have a project coming up, then most contractors will be able to arrange their schedules to fit your project in. Advance notice is no problem but no advance notice can be a killer. If you follow these two steps of forming a small pool of contractors to call and giving them enough advance notice then your results should be a project that results in success. At the worst, even if your project gets scrapped you have lost nothing but one phone call to the contractor. A small price to pay considering the larger costs of not having a qualified contractor available if the project starts according to plan.

There is one other thing you should always do as a DP manager with any contractor. Remember this is your project. It is your responsibility to manage it and guide it along. Many times I have seen a company hire a contractor and show them the system they desire and the final reports they want and then they let the contractor design and code the whole system without ever checking any of his work. Many times they wind up with the final product they specified but not designed the way they wanted. Contractors are not mind readers. If final results are all you care about that's fine. But if you want the designs, programs and documentation to follow certain company standards then you need to periodically check any contractor's work. On any project it is always management's responsibility to make sure the project is going according to plan.

Now in summary let's look at the best way to successful projects through contract personnel.

If your a shop where budgetary constraints are not a problem and you need contractors in a hurry then agencies are probably the best way for you to go. If however, you do have a budget to keep and you have some lead time for projects then probably a skilled pool of a few qualified direct contractors would be the route to go. If you decide to bring direct contractors into your shop then the best results can be achieved if you:

1.) *Provide enough advance notice to have a contractor available when the project starts.*

2.) *Keep the contract or letter of contract short and simple.*

3.) *Periodically check the progress of the project to make sure both parties are agreed on the methods being used.*

Remember that contractors are a valuable source of productivity to any company. But then again they are only as good as the management and the tools that they have to work with. With these few simple steps you should be able to have a very productive shop and many project successes.

What Every Manager Should Know About Consultants

Daryl A. Frame
Discreet Computer Service, Inc.
16161 Nordhoff St., Suite 21
Sepulveda, Ca 91343
(818) 366-2681

As a manager you know how many times in a single day you are called upon to make decisions. Technology is moving forward so rapidly that it is not always wise to use yesterdays solutions to solve todays problems. A "classic" example of this is the introduction of the new Hewlett Packard Precision Architecture (HPPA) machines. They offer not only greater performance characteristics, but a whole plethora of new techniques for solving todays problems. For some shops the day of reckoning has already begun. For others, the opportunity to capitalize on these new opportunities is still future. Suffice it to say that if your shop is even a little bit progressive, a Precision Architecture machine is probably in your future.

What that means for you as a manager is that you are either currently faced with the task of converting to run on one of these new machines or you soon will be. The issue of how to accomplish the task could well include the use of consultants.

Of course, there are many other circumstances that might make sense for you to consider using a consultant. As a long time consultant who has worked in various management positions, I will endeavor to shed some light on how best to utilize the services of a professional consultant.

It is apparently true that various regions of the United States tolerate Data Processing consultants dif-

ferently. In Southern California where I live, consultants are used extensively. On the other hand, I understand that on the East Coast the situation is quite different; the use of consultants is the exception rather than the rule. Whatever the situation is that you are most familiar with, I would like to show you under what circumstances it makes sense to use a consultant and how you can do so safely.

FACTS ABOUT SECTION 1706

Like so many other managers, you have probably heard a lot in the trade press about the legal aspects of using consultants in the wake the 1986 Tax Reform Act. Of course, I'm referring to the infamous Section 1706.

This little publicized section of the 1986 Tax law began to surface as a real problem for Data Processing consultants in the late fall of 1986. Companies that used such specialists stood to loose as the Wall Street Journal pointed out. It said, "companies that continue to treat regularly employed technical specialists as independent contractors will run the risk of having the IRS declare them employees and levy payroll taxes plus penalties on the employers".[1]

Now that sounds like a real potential for liability on the part of companies that use consultants and it's not to difficult to figure out what happened next. Lot's of objections were raised as to how the section got into the tax law without public hearings, etc. Consultants associations like those listed in table 1 began to cry foul. Soon they all began to organize in an effort to repeal Section 1706.

On January 21, 1987 the IRS issued an official clarification of Section 1706. "The clarification made it clear that Section 1706 made no change in the relationship between these contractors and their clients, whether or not a broker acts as a middleman. . . Notice 87-19, attached to the IRS News Release, affirmed that Section 530 of the Revenue Act of 1978, known as the 'safe harbor' continues to apply to the consultant-client relationship, whether the consultant personally arranged for the contract or the contract was negotiated with the participation of a broker or

any other 'third party'. . . The release also indicated that in a three way [brokered] arrangement, that the IRS common law tests of independence would be applied to the broker's treatment of the worker. The client will be no more at risk than he/she would under pre-existing law".[2]

PROFESSIONAL CONSULTANT ORGANIZATIONS

Independent Computer Consultants Association (ICCA)
Data Processing Management Association (DPMA)
Institute for Electronic and Electrical Engineers (IEEE)
Software Services Association (SSA)
American Consulting Engineers Council (ACEC)
National Association of Computer Consultant Brokers (NACCB)
National Society of Professional Engineers (NSPE)
Professional and Technical Consultants Association (PATCA)
Professional Services Council (PSC)

Table 1

In simpler terms, you the client, continue "to have Safe Harbor protection under Section 1706, still being covered by section 530 of the Revenue Act of 1978. As the client, you are still protected". The IRS has gone on record that you are not liable for employee taxes if the consultant in question is later found to be an employee. "Only three party contracts are affected by Section 1706, and then only the broker-consultant relationship is examined. If an independent contractor in a three party relationship is found to be an employee for failure to conform to the Common Law standards, then the Technical Services Firm or Broker is held liable, not the client".[3]

During the many months that have elapsed since the initial IRS ruling there have been a number of abortive attempts to clarify or repeal Section 1706. From the governments position, the matter is very complex.

At issue here is the viability of an individual or company reporting income as a contractor. A truly independent

business would obviously serve various clients, assume risk and control their own work. Later in this paper, I will discuss what can be done from a practical stand point to insure that there are no problems with the IRS. However, at this point there is a more pressing question to consider.

WHY A CONSULTANT?

Why would you ever want to bring in an outsider, a consultant for other than a dire emergency; particularly if you are not used to working with consultants? It may be difficult to convince yourself that there is any major difference between the people on your staff and those that would recommend themselves as a consultant. I have heard it said that a computer consultant is any professional in the industry who has been out of work for three or more months. While that may be true in some cases, the fact remains that there are numerous individuals and organizations that are completely competent and worthy of your consideration. They are available to assist you to complete projects in your organization and they come in many flavors.

Most often, though, consultants come into an organization as Analysts or Implementers. By that I mean they are skilled at designing and building systems. Computerworld[4] said that companies generally use consultants "on software development projects to secure skills that the permanent staff lacks or when they must build up the staff for a limited time". Furthermore, International Data Corp., a market research firm in Framingham, Mass., expects the demand for contract systems development to increase about 20% a year in the next five years. "Much of the demand will stem from a growing willingness of companies to contract out complex chores such as systems design and project management in conjunction with systems integration projects".[5]

Think of the valuable services that are available from consultants; services that may not be available on your staff when you need it most. In addition to Programmers and Analysts, there are resources in the areas of system management, security, data base administration, system performance, conversion, personal computers, data communications, documentation, training, etc. The list is almost endless.

Consulting services are available directly to the management level as well. Corporate policy, planning, feasibility studies and proposals are all services which can be contracted. Remember that a consultant from the outside can often see matters objectively while those on the inside may have difficulty.

And there is yet another good reason for bringing a consultant in to help you. Let's say that you have a large, complex task that needs to be done. If you hire an employee, you are obligating yourself to a long term commitment. When the project is done you will need to find other work to keep that highly skilled employee motivated. On the OTHER HAND, a consultant can be phased in and out as required. In this era of tight budgets and cost conscious management your skill in manipulating resources, including consultants, could be a big feather in your cap.

Plus, unless you work in a much more liberal place than the ones I generally consult to, adding additional personnel is always a problem. Bringing in a consultant with just the right skills for a short term project may be just the ticket. It's cost effective and cost conscious!

JUSTIFICATION

As you contemplate how you are going to "sell" the idea of having a consultant do some work for the company, bear in mind that cost savings and other benefits come in many forms. Direct costs are not to be ignored, of course. But it is the astute manager who recognizes these extra benefits and is able to use them in the justification process. What do I mean by "extra benefits"? Let's take a look at some examples and you'll see what I mean.

First, consider the case where a manager is thinking of contracting with a programmer to augment staff on an overdue project. If properly chosen, the consultant can (at no additional charge to the company) provide training in the form of tips and techniques that have been gleaned from years of experience in the industry and at numerous other companies.

The use of these tips and techniques at your company will continue long after the consultant is gone.

In an organization where budgets are tight and additional full time personnel are out of the question, a consultant may be just the ticket when it comes to getting an emergency project done. But how can you justify the expense? There is no provision in your departments budget for outside services. Why not get the user department that needs the emergency project done to pay for it? After all, if the project is going to save that department money, their budget can probably support the cost.

Has system security become an issue in your shop yet? Everyone these days is aware of the hacker problem. There are several software packages on the market that are designed to help with this growing menace to your vital company information. Have you taken adequate steps to counter this threat?

Horror stories abound of hackers causing millions of dollars in damages because they were able to penetrate a computers security measures. At one of California's largest banks, a trusted employee is implicated in a $21 million computer scam. More recently, world attention has been drawn to the cases of hackers causing serious problems on the ARPA network. According to one industry expert, "more than 90 percent of the nation's businesses depend on computer technology, and it is this dependence on computers that is creating serious problems. . . At present, the best concept managers can apply to protect their computers is simply good security".[6]

True systems security is a specialty in itself. Do you have the technical skills on staff to meet the demand? Most smaller to medium sized shops do not. Nor do they need this type of sophistication on a full time basis. It may be possible to contract for a complete security audit by a qualified expert in the area. This is a perfect application for a consultant. The additional technical resource only needs to be applied long enough for you to get an acceptable security program installed and working to your satisfaction. Then a periodic review can be made to insure that you are

getting the maximum benefit. In this case then, an ounce of prevention may well be worth a pound of cure.

SELECTING A CONSULTANT

Given all of the above considerations, it is clear the selection of a consultant to assist you on an important project should not be taken lightly. Unless they have personally worked with a consultant that they know can do the job and is available, most managers wisely endeavor to obtain qualified consultants by means of referrals. In my opinion, this is always the best way to get a qualified consultant. However, that may not always be possible since you may be looking for a special skill that is not generally available.

Selecting a consultant under these circumstances can be more difficult. The real challenge though is locating consultants to choose from. Several consultants associations have been mentioned throughout this paper, but one that is intimately familiar with the HP environment is SIGCONSULT. There are both national and regional chapters of SIGCONSULT, so it may be a good starting point for qualified HP talent.

The advantage of going to an association for candidates is that they all have certain qualifications that all of their members must meet. SIGCONSULT of Southern California for example requires that all of its members have five years of data processing experience; at least one year of which must be on HP in the last three years. Each member must have spent at least 50% of their time consulting in the last year. Also each member must supply at least three letters of recommendation from clients. You can see then that those who qualify for SIGCONSULT are tried individuals with a proven record of performance.

Here are come suggestions that were put out by the Independent Computer Consultants Association[7] that can serve as a guide to selecting a qualified consultant. If followed they will assure that no problems arise later with regard to employer/employee relationships as defined the IRS.

* Is the consultant incorporated and/or have a registered business name? This would lend credibility to the fact that the company is in fact a viable stand alone company.

* Does the consultant have his or her own letterhead and business cards? Again, it would be difficult to argue that the company was autonomous if it had no business identity.

* Does the consultant advertise? (A professional association's Directory of Consultants would suffice).

* Does the consultant have an Employer Identification Number? A Social Security Number is not as desirable.

* Does the consultant belong to any associations of similarly skilled persons? This would support a special skill determination.

TYPE OF AGREEMENT

Not to be overlooked also is the matter of adequate documentation of the relationship of a client and consultant. This is where you as a manager would be concerned. Some companies prefer to establish this relationship by means of a purchase order. If your company works that way be sure that the following items are clearly spelled out and adhered to:

* The complete task to be performed and any deliverables

* Terms of payment

* The period of performance

* No relationship of employer-employee is created by the agreement.

* Your company is not obligated to provide Workmen's Compensation Insurance covering the consultant or his/her personnel.

* Under what circumstances the agreement may be terminated

Bear in mind that most, if not all, Purchase Orders do not contain language effecting the above provisions. It may take several pages to adequately spell out all of the necessary agreements that exist between you and the consultant. Take the time to do it right. It may save you a lot of grief in the future.

Of course, the best way to address the relationship between a company and a consultant is by means of a contract! That may seem obvious, but it is surprising how often this little matter is overlooked. If you use consultants a lot or are planning to, you would be well off to have a contract drawn up by your legal counsel. The benefit is that it's worded to protect your best interests.

Most consultants have a contract that can be used. Be sure to read it carefully and have it examined by your legal counsel to insure that you can live with everything that is in it. That's just good common sense. Don't be scared away by a multi-page document. There are many things that should properly be covered by a contract.

Naturally, all of the items discussed above which would be included on a purchase order should be covered in a contract also. In addition, a contract might deal with these items:

* What, if any, working facilities will be provided by the company for the consultant to use in the performance of the tasks outlined.

* An anti-competition clause.

* The disposition of materials or files supplied to the consultant at the expense of your company.

* That use of the companies equipment or communications facilities will be for the purposes of completing the project assigned.

* How disputes that may arise between the consultant and the company will be resolved and under what legal jurisdiction.

MAINTAIN ARMS LENGHT RELATIONSHIP

When you enter into an agreement with a consultant be sure that you understand the implications of your on site management practices. In order for there to be an arms length relationship, the working relations between your company and the consultant must also fit within the Common Law standards set out by the IRS. The most important of these Common Law standards is the matter of control and right to control. The IRS National Office Technical Advice Memorandum of July 17, 1985 states: "Independent contractor should be subject to the control and direction of another merely as to the result to be accomplished by the work, and not as to the means and methods of accomplishing the result". What this means is that it should be possible to distinguish the work of consultants from that of employees who are routinely assigned to whatever tasks are at hand.

Specifically there are several areas of special concern. The Independent Computer Consultants Association makes the following recommendations[8]:

* Facilities, Equipment and Workspace: Client firms should inform independent contractors of the resources available for their use and of any requirements for security and confidentiality of data. The consultants should then make use of these resources as they deem necessary for fulfillment of the contract.

* Code Walkthroughs: Walkthroughs have been mistaken by auditors as a management control tool, when in fact they are tools for product control and client training. Walkthroughs are also an ef-

fective method of detecting unintended technical errors. Problems in the statement of the end result may also be detected. Managers should not participate except as technicians interested in ensuring an error-free product.

* Staff Meetings: Consultants should attend meetings and participate only to facilitate the design, modification/enhancement or implementation of the product to be delivered. Consultants generally participate in meetings germane to the conditions or their contracts, and should not be required to attend general meetings held for the benefit of the client's employees.

* Length of Service: On long contracts, make sure the work is segmented. Each piece should be considered a separate deliverable, fully documented and turned over to the client's staff before any work is done on the next part. The client should consider requesting consultants to submit an updated statement of work for client approval for each assignment. It is perfectly legitimate for a consultant to have a multiple year assignment, provided other tests of independence are met.

* Sources of Income: If a consultant has several substantial sources of income within a calendar year, it is difficult to press a case for employee status. It is therefore to everyone's benefit for consultants to have time for marketing, serving other clients, and attending to the operation of their own businesses.

* At Will Discharge: If you can terminate a contract "at will" without cause and/or liability, this implies an employer-employee relationship. If you can terminate a contract only when some contracted-for contingency is met, this supports the independence assertion.

By following these general guidelines, it is possible to use consultants in your company with a minimum of side affects. You will be able to identify those consultants

with the appropriate experience level from those who would lack the necessary depth and background. This will prove to be an important factor when it comes time for the project to come to a close. No one can afford to find out when it comes down to the wire that the consultant they selected just didn't have what it takes to come through.

REFERENCES

1. Wall Street Journal, December 9, 1986, p. 6
2. Software Services Association Newsletter, Winter 1987, p. 7
3. Independent Computer Consultants Association, Letter to all members dated March 1987, p. 1
4. Computerworld, June 20, 1988, p. 79
5. Ibid.
6. CFCA Communicator, May 1988, p. 4
7. Independent Computer Consultants Association Northern California Chapter Newsletter, March 1987
8. Ibid.

PAPER #6010

HOW TO OPERATE A SUCCESSFUL INFORMATION SYSTEMS DEPARTMENT IN AN ENTREPRENEURAL ENVIRONMENT

By William D. Krupinski
BASICOMPUTER CO.
1585 Frederick Blvd.
Akron, Ohio 44320
(216)873-1000

BACKGROUND

In order to understand how to operate a successful Information Systems department in an entrepreneural environment, you must first recognize what characteristics make up an entrepreneural company and how such a company goes about running its business. Let me build a scenario of a company which falls into this category. First of all, it is a company which has experienced tremendous growth over the years, doubling its annual sales in less than a three year time span and, have probably achieved this more than once since the company's inception. The company has been in existence less than fifteen years and is either privately held or has just recently gone public on the "Over-the-Counter" market. Its business is that of non-manufacturing type, usually distributional company and,

most likely that of a "value-added" distributor. The "value-added" makes it unique in the business, and gives the company its niche. The company's major focus to date, has been in the Sales and Marketing area. Most likely, the President was at one time the Executive Vice President of Sales and Marketing for the corporation. Because of his phenomenal sales record he has risen to the rank of Chief Executive Officer and is now in the position of running the entire company from sales to finance, and administration. The company has reached a point in time where it has grown so much that no one person can "keep his arms around it".

DEFINING THE ENVIRONMENT

Now that the company has grown exponentially over the years and the main focus has been on sales and marketing, without a doubt the administrative areas and the various controls are almost nonexistent. Functionally, the personnel in these areas are either not very knowlegdeable or have grown up with the company. Either way they lack the expertise necessary to operate in a tightly controlled environment needed for the larger company. In addition to this fact, there also exists a limited number of people to do the job overall, resulting

in many overlooked processes and some not performed at all.

The Information Systems are most likely the weakest of all areas. The Computer hardware is about 7 to 10 years old. The Company outgrew the system two years ago, and wants to expand its operation even more. Because the equipment is so weathered and being pushed to its limit, the company frequently experiences computer down time, usually at the most critical time periods, such as a month end closing. The Applications Software is totally outdated and no longer fits the way the company presently does business. It has been modified many times with "bandaid" solutions, rendering installation of the latest software release impossible. Also, because the M.I.S. department is understaffed and spends the majority of its day "fighting fires", no new development takes place. The head of M.I.S. constantly hears about how much money the Information Systems department costs, and how little it provides for the company. Further development of this scenario, this gives way to the proliferation of Personal Computers (P/Cs). These P/Cs will run a myriad of different applications, anywhere from "off the shelf" spreadsheets to custom design software systems which handle a certain segment of the company's business. The thriving use of P/Cs creates an environment of the

stand-alone P/C applications, known as "P/C Islands". These systems usually do not connect to any other system within the organization, and most likely the data is hand entered from the mainframe reports into these P/C applications. This gives way to a whole new set of problems, among them data integrity errors and dual sets of reports. This environment incurs high costs (cost of keeping obsolete equipment running and multifarious number of "high-powered" P/Cs), commits multiple data integrity errors, houses an unknowlegdeable staff, and uses data that basically running amok.

As the company moves forward in time, it may or may not hit the point where growth and/or profits stabilize, but is beginning to see signs of it occurring. The "Corporate Fathers" have, all of a sudden, realized good strong information systems with tight controls are necessary in order for the company to remain an entity in its market.

GETTING ORGANIZED

As the head of the M.I.S. department, you may have inherited this environment previously described, or have been a part of

it since its advent. Whatever the case, it is paramount to you and most advantageous to the company that you get organized. Of even more importance is the need to start organizing the future direction of the Information Systems effort. Some things you can do rather quickly are:

- Establish a formal procedure to identify the amount of backlog work and publish it to the corporate officers.
- Establish procedures for processing critical applications in order to reduce errors.
- Determine and attack the most visible problem facing the M.I.S. department today.

Keep in mind, though, that the most visible problem may require another "bandaid" solution or can not be resolved within the scope of the Information System department. You will also notice, I said attack the most **visible** problem not the most **important.** I will expound on that subject later in this paper.

Some other things to do that are more of a continuing effort or of a long-range nature are:

- Establish a written departmental procedures manual

describing how you will be running your department.

- Perform a hardware projection study, estimating how long it will be before you outgrow the current computer system, and make recommendations based upon your findings.
- Establish a major project list with time estimates to completion. Build a case to hire additional personnel and/or contract programming.
- Build a Local Area Network (LAN) to connect the "P/C Islands" to the main transaction processor.
- Create a M.I.S. Steering Council, made up of the decision makers of the company or the head of the functional areas serviced by Information Systems department.

A LAN will provide an immediate solution to connecting the P/Cs together, eliminating a lot of problems discussed previously. The creation of a M.I.S. Steering Council is the most essential thing needed to be successful in restructuring the M.I.S. effort. For the Council to be effective, it must contain the proper members and address the appropriate issues. Consider the following as a suggest list of members:

- The President

- Chief Financial Officer
- Vice President of Sales/Marketing
- Vice President of Operations/Inventory Control/Purchasing
- Controller
- Head of M.I.S. Department

Some important guidelines to follow are to:

1) Have your immediate manager head up this Council.
2) Let him approve the agenda items and schedule the times of the meeting (I suggest meeting monthly or every other month at first, then once established meet quarterly).
3) Lastly, if are the head of M.I.S. do not give yourself any voting privileges on decision matters, act as a consultant to this decision making group.

If you have voting rights, your viewpoints may be considered to myopic or parochial in nature. Stay out of the politics of the company and let the committee decide the future M.I.S. direction, you just need to execute it. This doesn't mean

you can not have an influence on what the committee decides. You should provide them with a ostensible direction along with a variety of alternatives. Tailor the discussions in the meetings it to the level of your audience. If you are talking about specific individual computer programs and what to change in them, you have the wrong people on the Committee. My advice is to have in-depth preparation in order to discuss specific applications, but try to stay on the long range items as much as possible. The things the M.I.S. Steering Council should decide consist of:

- Issue directives for the Business Systems Plan
- Approval of MIS budgets and addendums
- Approval of all major computer hardware acquisitions
- Establishing and defining priorities
- Approving and authorizing major systems designs (set the M.I.S. manpower effort for major projects)
- Approving manpower levels to project teams
- Approving adhoc teams for various projects, ie. a Software Evaluation Committee
- Determining whether to continue or terminate systems efforts in various environments
- Performing post implementation evaluations to

verify proposed cost benefits

As you can see, this committee is really using the foresight of the company's three to five year Business Plan and providing this intuitiveness to the Information System department. In other words it is predicting what systems will be needed for the future. The last things to remember when dealing with this Council are to always be prepared, have alternatives, keep the meetings under two hours long, have complete cost information for all endeavors being evaluated, and offer recommendations where appropriate.

MAINTAINING THE PROPER PROFILE

As the Head of the Information System department it takes more than just doing the "right job" all of the time. In the environment previously described, it becomes almost mandatory to maintain a high visible profile, a profile which can be observed by the officers of the company. Keep in mind, that at first it will be like being viewed under a microscope, but in the long run it will be worthwhile. One of the difficulties in operating this way is that you have to be above reproach. You can not afford to make any political

enemies and you have to create the impression that you maintain full control over whatever situation arises in the Information Systems department. Remember one thing: PERCEPTION IS REALITY!!!

Because you are dealing mostly with sales and marketing personalities who only buy off on things that are properly packaged, make your presentations colorful and full of flair. Always consider your audience. If you find it difficult to do this, get a book on effective presentations and study it. The benefit of this is that if you can sell and then implement it, you become accepted in the general sphere of management, and you are consulted with beforehand for your opinion. How many times have you been on the end of the "whiplash" of a new corporate direction, finding out about it a short time before it is to be implemented. The only way that you will become successful is if you are treated with the proper respect of the upper management group. Unfortunately, in an organization such as this, the question that quickly arises of "What have you done for me lately?" I will tackle that topic in the area of "Look For The Early Win".

WHAT SYSTEMS TO IMPLEMENT FIRST?

Chosing the sequence of implementation of systems may be an easy or difficult decision. An easy decision would be to implement a system which is critical to the survival of the company. A difficult one, would be to implement multiple systems critical to the operation of one necessary system. That later decision lengthens the lead time for your department and puts more pressure on you. If your company has selected a packaged applications software system then, by all means, regardless what anyone else wants, do not deviate from the vendor's recommended implementation sequence. The vendor 's recommendation of an implementation sequence is based on a good reason. Too often, I have seen software implementations take three to five years because someone thought they had devised a better sequence than that which was recommended.

Here is a suggestion of a simple system to implement first. Design a non-complicated product configurator tied to a Retail Pricing system. Capture the quotes on the mainframe and use them as a base to feed your Order Entry system. You can design something simple and elementary. For faster

implementation use one of the "4Gls" on the market. Unless you have a complex product to design, you can have this system working in less than a month. The Sales and Marketing people, who probably make up the majority of your company, will love it. The system will save time and effort and allow them to standardize their pricing. It also provides for faster order entry and invoice distribution.

LOOK FOR THE EARLY WIN

As I said before, in an organization I have described PERCEPTION IS REALITY. If you are able to solve an ongoing problem very quickly, the cognition or the image you will project will be a positive one. Look for something simple, installing a Local Area Network to connect the P/Cs is a "no-brainer". You can install a P/C based LAN, and use proven "off the shelf" software to accomplish this task. Even though the cost is somewhat expensive, it can be cost-justified by the amount of data entry saved and the improvement of data integrity. Also, look for something that has the shortest amount of lead-time to install, something that is quick and has a lot of visibility. When you look for that "early win" make sure it is something that affects a lot

of people in the organization or something everyone knows has been a chronic problem for a long time.

What you are trying to establish with the quick win is credibility and the perception that you can provide the quick solutions to various problems. After a couple of victories, Upper Management starts to rely on your decision making capabilities. After a while you become a member of the sphere of management, providing for more rapid future decisions.

GETTING YOUR WAY

Now that you have this new found credibility, you will start getting your way more often. This can be both good and bad. The good part is that the decision making process is done in a shorter time frame. You can obtain faster decisions on various matters allowing you to react to different problems quicker. Upper Management respects your decision making process and tends to give you "Carte Blanche" on all capital expenditures you request, with little or no justification. This also works in your favor because you can continue your problem solving methodology faster helping to increase your

credibility even more.

The down side of getting your way is that you may tend to make decisions and apply solutions too quickly. You make decisions without thinking them through or without researching other alternatives. If you are never challenged on your decisions, you start to become complacent. You will begin to make decisions based on past experiences and "gut feel". In a high tech environment you must be aware of alternatives needed to solve some of the new problems that arise everyday. My advice to you, is to find someone within your organization or a colleague for you to sound your decisions off of. This person should be knowledgeable in the problem area you are trying to resolve and be willing to challenge your decisions.

The ideal situation is to have a "happy medium". In the cases where there are large and high dollar costs, try to use your own perseverance to find multiple alternatives. For simple low cost decisions make a quick selection and move on. Too often small decisions are procrastinated over to a point where any decision is academic. Sometimes making a wrong decision, if of a lesser nature, is better than making no

decision at all. If possible, try to establish a policy to allow you to sign off on any purchase of $1000 or less, provided it is budgeted for. Lastly, don't start making poor or lengthy decisions. If a decision is perceived as a bad one, even though it isn't, then it will be received as a bad decision regardless of how much you try to change it. As quickly as you have established your credibility, you will see it disappear. Regaining it will take a much longer time period than before.

SUCCESSFUL IMPLEMENTATION

There are a lot of factors which go into a successful implemtation. I have touched on most of them in this paper. The important thing to remember no matter what happens during these processes, is to remain in control or in charge. Never give up the control factor to anyone else, especially a Non-M.I.S. person. The project will be doomed to failure. If the project fails, then you will forever take the blame. Always stay organized or at least give the impression that you are organized, and keep upper management informed of progress or lack of it. Try to keep to the schedules and budgets established. If the project is too big, break it

down into more manageable sections. Always make sure you have the proper personnel working on the project, both from the end-user and the M.I.S. department. If your staff is not adequately trained get them the proper training and allow for the schedule to slip because of this. Obtain the properly trained staff through outside services and/or new hires. One thing which will hold up a system implementation is if the person from the end-user department who has all of the responsibility is not allowed to make any decisions regarding departmental policy. This person spends every minute running back and forth to the "decision-maker" for resolutions doubling the time effort involved which will once again it be perceived as a M.I.S. delay.

Look for these traps and beware of them as they will cause a project to slip every time. Remember that once you have built a case to management recommending a certain direction, "you have got to deliver the goods". The selling of the concept is only half of the effort. Implementation is the rest, it is truly what you were hired for. If you have never implemented a large-scale system, start off with a small one. That way, if you make mistakes they will be less noticeable or will not have such a large impact.

In an entrepreneural company, the head of the M.I.S.

department has to be more than just a technician, a system implementer. He has to be a salesman, a marketer, an accountant, and a leader.

BIOGRAPHY:

WILLIAM D. KRUPINSKI
DIRECTOR OF M.I.S.
BASICOMPUTER CO.
AKRON, OHIO

BASICOMPUTER COMPANY IS A PRIVATELY HELD COMPANY OPERATING IN SIX STATES AND FOURTEEN CITIES. THIER BUSINESS IS A VALUE-ADDED DISTRIBUTOR OF MICROCOMPUTERS.

SEVENTEEN YEARS OF M.I.S. EXPERIENCE WITH OVER EIGHT YEARS IN THE MANAGERIAL RANKS.

INSTALL MANY HP/3000 COMPUTER SYSTEMS OVER THE LAST SEVEN YEARS, ONE THE EARLY IMPLEMENTERS OF THE MPE-XL COMPUTER SYSTEMS

BACHELOR OF SCIENCE IN BUSINESS ADMINISTRATION FROM LAKE ERIE COLLEGE, PAINESVILLE, OHIO.

MEMBER OF INTEREX FOR FIVE YEARS

ATTENDED WASHINGTON D.C., DETROIT, LAS VEGAS, ORLANDO CONFERENCES

A Step-By-Step Approach to Selecting Accounting Software

Robert E. Shelley
Highgate Financial Systems
2000 Powell St., Suite 1200
Emeryville, Calif. 94608
(415)596-1707

We've all heard the war stories about the accounting department that bought the "wrong" software and, after a year of wasted money, wasted time, and utter frustration, had to throw the product away and begin again. But in fact, the selection of accounting software is just another business decision. Like any other important choice, it requires serious thought and planning, but you don't need a crystal ball.

Instead, a step-by-step approach to selecting accounting software will help you reach your goal: a quality software product that meets your organization's particular needs, is backed up by effective service, and carries a price tag you can afford.

Who Is Responsible?

The first step in this process -- and it is crucial -- is to assign responsibility for the overall selection process. Without a clear designation of responsibility, the process can take longer than necessary.

Since it is the eventual user of the system who will have to live with it, the user should have primary responsibility for selecting the system. This assignment always should be made to someone with accounting responsibility, not to the MIS department. Candidates would include the CFO, the VP of Finance, the corporate controller, a regional controller, and the accounting manager; the appropriate choice depends on the size and nature of the business, as well as the individual talents and capabilities of the person given the responsibility.

But whoever is chosen should be generally knowledgeable about the accounting needs of the organization, the current problems confronting the company, and the general plans concerning the future of the company (plans to expand, contract, move into new markets, and so on).

Depending on the circumstances, it may be best to choose a selection committee rather than one individual. Particularly in larger organizations, this can ensure that all aspects and dimensions of the accounting process are

considered. Committee members should come from accounting, data processing, the line organization, and other areas of the organization directly affected by accounting data.

Once the responsibility for the selection process has been assigned, you should put together a general project plan, including milestone dates, individual time commitments, and project budget. This will help to set realistic expectations throughout the organization about what is going on and how long it will take.

Identify Your Requirements

The next big step is to answer the question, What do we need from this software? To determine the scope of your search, create a summary list of your requirements. Decide what accounting modules are required -- GL, accounts payable, accounts receivable, order entry, inventory control, fixed assets, and so on.

Briefly document the manual and computer-based systems that currently address these requirements. This will ensure that everyone recognizes the current situation and the need to change it. Include a summary of your current problem areas, noting which ones must be solved with new software and which ones it would be nice to have solved.

Gather summary statistics showing how much accounting activity takes place in your organization. Such statistics include the numbers of:

- Accounts in your chart of accounts
- Departments
- Invoices sent out per month
- Vouchers for payment per month
- Orders taken per month
- GL transactions per month.

You need to determine these figures relative to your current activities and also to your projected work load three years out. This will provide your data processing group with a sense of data volume changes and will assist the vendors in evaluating your requirements.

Once you have done these preliminary analyses, you can create a detailed listing of your requirements. This can be organized in many ways, but I suggest the following approach:

1. Make a listing by subledger module, and within that by general areas, such as data entry, control mechanisms, reporting, etc.

2. Indicate next to each requirement the degree of

need, such as "critical," "want to have," "nice to have."

3. Keep the list in a word processor because it will change frequently during the selection process.

A detailed listing of possible requirements for a general ledger system is contained in Appendix A. You can use it as a starting point in building your own detailed requirements listing -- deleting, modifying, or adding whatever requirements reflect your circumstances. A similar listing can be creating for any other subledgers you require.

You also need to make a preliminary budget for how much you expect to spend for the packaged software. Be sure to include the initial cost of the software, the cost to convert your present data to the new system, the cost of installation and training, and the annual cost of software maintenance by the vendor.

Approaching the Vendors

Once you have created your requirements list, you are ready to contact accounting software vendors. Start by making a list of vendors. There are several sources for this list:

- Trade magazine advertisements
- HP Chronicle directory of software products (The Software Source)
- DataSources, a directory of software products
- HP sales representative; HP Business Systems Solutions directory
- Companies in a similar line of business to yours
- Appendix B to this paper, which contains a listing of vendors.

Call the vendor candidates on the phone to do prescreening. At this early stage, you will be telling them what business you are in and giving them an overview of what you are looking for. What you want to find out from the vendors is what software packages they have to offer and what installations they have that are similar to your organization.

If you find a vendor's offering of possible interest, request literature. Tell them that after you have reviewed their brochures you will contact them again to determine if a meeting should be arranged.

Based on the vendor data you have at this point and your known requirements, pick four to six vendors to invite in for further discussions. Send them a copy of your detailed requirements document, including statistics on item counts --

current and future. They should review the document and indicate where their product does and does not meet your requirements.

Working with the Vendors

Your next task is to work effectively -- and efficiently -- with the vendors to determine which one best meets your needs. The range of four to six vendors is somewhat arbitrary, but keep in mind that it is very time-consuming to work on a detailed level with many more vendors than this.

Once you have your short list, arrange the initial meetings and let each vendor know that you expect to hear about their company and about their products that are of interest to you. You also expect to review any questions the vendor has, based on their analysis of your detailed requirements document. And you want pricing information on the software packages, conversion of data, training, and annual maintenance.

It is very important that the meeting be what the diplomats call a "frank and open exchange of views." By discussing such areas as your proposed budget for this project and the other vendors invited to bid, you will help uncover any inconsistencies or misunderstandings on both sides. If not discovered early in the process, these failures of communication can cause endless troubles later.

If possible, see a detailed demonstration -- on your own computer -- of each system you are considering. Before taking the time for this, however, you may want to review the information you have gathered up to this point, in order to eliminate vendors who do not offer enough of a solution to warrant further investigation. At the demonstration, be sure to ask the vendor to show you clearly how their package will solve the most important problems you are having with your present system.

Comparing the Alternatives

Comparing the capabilities of the vendors is the next step in the selection process. Using the detailed requirements form filled out by each vendor, score their responses relative to your requirements.

The scoring can be done initially on each vendor's response to your listing of requirements, or it can be done on separate worksheets. In any case, the following example will help identify the purpose of such a comparison. In this example, your requirements are scored as follows:

3 = A critical need
2 = Want to have
1 = Nice to have.

The vendors' responses are scored in a similar way:

3 = Offers complete solution
2 = Offers acceptable solution
1 = Offers partial solution
0 = Not available from the vendor.

Now let's see how this is brought together for a sample group of questions about two vendors' general ledger packages:

Figure 1

Question	Your Requirement	--Vendor A-- Response	Var	--Vendor B-- Response	Var
1. Batch controls	3	3	0	2	-1
2. On-line account analysis	3	2	-1	3	0
3. Financial report writer	3	2	-1	3	0
4. Complex allocations	2	2	0	3	1
Totals			-2		0
Total of negatives			-2		-1

As noted in Figure 1 above, for each vendor a variance is calculated by subtracting your requirement from their response. This number highlights differences between your needs and the vendors' products.

Notice also that two totals are calculated. The simple arithmetic total gives you an overall sense of how the vendors' capabilities compare to your needs. The higher the score, the better the vendor. But, as the example shows, this figure by itself can be misleading, since vendor B has a score of 0, implying that they meet all the requirements. Totaling the negatives is a check to ensure that you overlook neither a vendor's shortcomings nor their strengths.

You may also want to make a list of just those features that are different among the vendors. This helps to better differentiate each vendor from the others and may uncover some meaningful comparisons for you.

At this point it is especially important to contact two or three references supplied by each vendor. Ask specific questions, about features that are important to you as well as about the overall level of satisfaction with the product and the vendor.

In addition to comparing the features of different vendors' offerings, you also need to compare costs. To begin, bring together all of the initial costs for each vendor. This includes the initial license to use the software, the cost of converting your present data into their package, and the cost of installation and training. Be sure to ask each vendor if there are any other front-end costs that you will incur.

Then figure out the ongoing costs for each vendor. The largest of these will be for software maintenance, which is usually an annual cost amounting to between 10 percent and 20 percent of the package price.

There are at least two methods of doing a cost analysis of software:

1. A present value analysis based on cash flow streams, without regard to the cost of financing the cash flows.

2. A lease-based method that assumes that the initial costs are written into a five-year lease at whatever interest rate you feel is appropriate for you. This gives you a fixed monthly payment for five years for all front-end costs. Then take the annual maintenance charges, plus any other recurring costs, and divide them by 12 to put them on a monthly basis. This number plus the monthly lease payment gives you a total monthly cost of the software.

Some people prefer the present value method because it is the generally accepted method for analyzing and comparing costs that vary over extended periods of time. But when you use present value, you still have to factor in the method of financing the best choice. The lease-based method, on the other hand, quickly gets you to a monthly cost that most managers can relate to easily, since it will probably be the number that they put into their budget for the next year. This method also allows you to include a vendor's favorable lease terms, when they are available. But both methods will rank the vendors in the same order.

Time to Choose

At last, you're ready to make your selection. When you weigh the strengths and weaknesses of each vendor's offering,

be sure to include a review of how you will address any of your needs that will not be met by that vendor's product. It's unlikely that any package will meet all your requirements, and if you are going to need custom modifications, additional software, or other expedients, they must be considered in making the final decision.

Now comes the day you've been eagerly awaiting, when you choose a vendor and begin detailed negotiations on costs and contractual terms. This is the time to ask for what you want. Of course, be ready to explain completely why you want a change from what is proposed, and maintain an open attitude. Ask the vendor to explain, in detail, any contract clauses you disagree with or do not understand. Remember that the goal of all negotiations is for both sides to get what they want and can reasonably expect from the process.

To recap, here are the basic steps in successfully selecting accounting software:

1. Assign responsibility for the overall selection process and put together a general project plan.

2. Identify your requirements: summarize current problem areas, gather statistics on current and projected accounting activity, make a detailed listing of your requirements, and draw up a preliminary software budget.

3. Do initial screening of vendors to get to a short list of four to six possibilities.

4. Meet with short-listed vendors, see demonstrations of their products, and ask lots of questions.

5. Compare capabilities and costs of the vendors.

6. Make your selection and conclude final negotiations.

In Conclusion...

The end of those negotiations is -- hopefully -- the start of a beautiful friendship (or at least a good working relationship) with the vendor as well as their product. While preparing for the installation of your new software, remember to seek the advice of your vendor in resolving issues and problems. The vendor has done many installations before and is in a good position to provide you with expert advise based on experience.

And once the system is in place, be sure to keep the lines of communication open to your vendor. Let them know of problems you encounter and seek their advice on solutions.

General Ledger Profile for **Vendor Name**

REF #	DESCRIPTION	RESPONSE
000950	GL ACCOUNTING STRUCTURES	
000960	On-line entry.....	_____
000970	Account structure:	
000980	Maximum segments.....	_____
000990	Size of each segment.....	_____
001000	Size of complete account code....	_____
001010	Numeric only or alphanumeric.....	_____
001020	User lengths.....	_____
001030	Sign and amount limits.....	_____
001040	Cross segment edits.....	_____
001050	Customizable screens.....	_____
001060	Co/Division/Department structure:	
001070	Unlimited structures.....	_____
001080	Unlimited levels.....	_____
001090	Simple reorganization.....	_____
001100	Multiple hierarchies.....	_____
001110	Accounting periods/calendars:	
001120	Maximum number/fiscal year.....	_____
001130	Uneven periods.....	_____
001140	GL JOURNAL ENTRIES	
001150	On-line entry:	
001160	Running totals.....	_____
001170	Currency conversion.....	_____
001180	Inter-company balancing.....	_____
001190	Repeating fields.....	_____
001200	High volume interface option.....	_____
001210	Posting controls.....	_____
001220	User source codes.....	_____
001230	Journal type codes.....	_____
001240	Optional batch balancing.....	_____
001250	Suspense accounts.....	_____
001260	Reversing entries.....	_____
001270	Recurring entries.....	_____
001280	Skeleton entries.....	_____
001290	Inter-company entries.....	_____
001300	Statistical entries.....	_____
001310	Foreign currency entries.....	_____
001320	Prior period entries.....	_____
001330	Prior year entries.....	_____
001340	Future period entries.....	_____
001350	Multiple open periods.....	_____
001360	Year end close generates entries...	_____
001370	Independent company close.....	_____
001380	Hi vol error correction (reverse)..	_____

General Ledger Profile for **Vendor Name**

REF #	DESCRIPTION	RESPONSE
001390	Validate presence of std. entries..	
001400	Timing of account balance update...	
001410	GL BUDGETING	
001420	On-line entry.....	
001430	Batch entry option.....	
001440	Number of budgets.....	
001450	Budget creation aids.....	
001460	Budget entry aids.....	
001470	Budget spread aids(smooth,seasonal)	
001480	Budget formulas.....	
001490	Automatic calculations.....	
001500	Lotus 1-2-3 interface.....	
001510	Shorthand budget entry.....	
001520	GL ALLOCATIONS	
001530	On-line entry.....	
001540	Soft allocations (no new entries)..	
001550	Mass allocations.....	
001560	Tiered (Step-down) allocations.....	
001570	Budget allocations.....	
001580	Period allocations.....	
001590	Quarterly allocations.....	
001600	Yearly allocations.....	
001610	Rerun allocations(w/backout of old)	
001620	Formula-based:	
001630	Unlimited factors.....	
001640	Unlimited nesting.....	
001650	Statistical factors.....	
001660	Fixed amount factors.....	
001670	Account balance factors.....	
001680	GL MULTI-COMPANY	
001690	Multiple sets of books.....	
001700	Multiple companies.....	
001710	Inter-company balancing.....	
001720	Consolidation reporting.....	
001730	Reconciliations.....	
001740	Inter-company elimination.....	
001750	Automatic entries:	
001760	Eliminations.....	
001770	Minority interest.....	
001780	Unique or shared:	
001790	Account structure.....	
001800	Organizations.....	
001810	Calendars.....	
001820	Options.....	

General Ledger Profile for **Vendor Name**

REF #	DESCRIPTION	RESPONSE
001830	GL MULTI-CURRENCY	
001840	On-line:	
001850	Journals.....	_____
001860	Exchange rates.....	_____
001870	Automatic conversion.....	_____
001880	Automatic translation.....	_____
001890	Revaluation.....	_____
001900	Revaluation (soft).....	_____
001910	Unlimited currencies.....	_____
001920	Spot rates.....	_____
001930	Period average rates.....	_____
001940	Holding gain and loss.....	_____
001950	GL OTHER FEATURES	
001960	Encumbrance Accounting.....	_____
001970	Fund Accounting.....	_____
001980	Statistical Accounts:	
001990	Budgets.....	_____
002000	Actuals.....	_____
002010	Summary Accounts.....	_____
002020	Project Accounting.....	_____
002030	Unlimited History.....	_____
002040	Simple Archiving.....	_____
002050	Selective Archiving.....	_____
002060	GL FINANCIAL REPORTING	
002070	On-line entry:	
002080	Report requests.....	_____
002090	Report definition.....	_____
002100	Report modification.....	_____
002110	On-line inquiry:	
002120	Ad hoc reporting.....	_____
002130	Account balance.....	_____
002140	Account activity.....	_____
002150	Summary account balances.....	_____
002160	Budgets.....	_____
002170	On-line graphics:	
002180	Pie charts.....	_____
002190	Bar charts.....	_____
002200	Responsibility reporting.....	_____
002210	Multiple indep organization rollups	_____
002220	Multiple indep account rollups.....	_____
002230	Auto elimination of interco account	_____
002240	Standard reports:	
002250	Funds and cash flow.....	_____
002260	Trial balance - detail.....	_____

General Ledger Profile for **Vendor Name**

REF #	DESCRIPTION	RESPONSE
002270	Trial balance - summary.....	
002280	Journal reports.....	
002290	Audit trails.....	
002300	Operating statements.....	
002310	Balance sheets.....	
002320	Chart of accounts.....	
002330	Report sets/groups/families.....	
002340	User defined reports:	
002350	User formats.....	
002360	User rounding.....	
002370	User units.....	
002380	User rows.....	
002390	User columns.....	
002400	User row/column intersections....	
002410	User rounding control.....	
002420	User suppress row of zeros.....	
002430	Copy facility for above.....	
002440	Math operators (+,-,*,/,sum).....	
002450	Boolean logic.....	
002460	Multiple sort sequences.....	
002470	On-line report submission.....	
002480	Report menus.....	
002490	On-line report display.....	
002500	Reporting versatility:	
002510	Forecasts (YTD actual + budget)..	
002520	Multi-year reports.....	
002530	Comparison of budgets.....	
002540	Ratios.....	
002550	Variances.....	
002560	Statistics.....	
002570	Actual versus budget.....	
002580	Across fiscal years.....	
002590	Companies as columns.....	
002600	Multiple reports per pass.....	
002610	Consolidate across charts of accts.	
002620	Financials to Lotus file for PC....	

Appendix B

SOME HP ACCOUNTING SOFTWARE VENDORS

ASK COMPUTER SYSTEMS, INC.
730 Distel Drive
Los Altos CA 94022
(415)-969-4442

BI-TECH SOFTWARE, INC.
1072 Marauder, Suite A
Chico CA 95926
(916)-891-5281

CARDINAL DATA CORP
The Hillsite Building
75 Second Avenue
Needham Heights MA 02194
(617)-449-0066

COMPUTER FINANCIAL SERVICES
13425 N.E. 20th
Bellevue WA 98005
(206)-746-2666

COLLIER-JACKSON, INC.
3707 West Cherry Street
Tampa FL 33607-2596
(813)-872-9990

COGNOS INCORPORATED
3755 Riverside Drive
Ottawa ON K1G 3N3
(613)-738-1440

COMPRO FINANCIAL SYSTEMS, INC.
6025 The Corners Parkway
Suite 200
Norcross GA 30092
(404)-662-8754

DeCARLO, PATERNITE & ASSOC.INC
Rockside Square II
6133 Rockside Road, Suite 400
Independence OH 44131
(216)-524-2121

GTS COMPUTER SYSTEMS, INC.
3529 7th Avenue South
Birmingham AL 35222
(205)-252-9446

HIGH LINE DATA SYSTEMS, INC.
716 Gordon Baker Road
Suite 100
North York ON M2H 3B4
(416)-494-2504

HEWLETT PACKARD
5301 Stevens Creek Blvd.
Mail Stop 52U/92A
Santa Clara CA 95052-8059

INFOCENTRE CORPORATION
7420 Airport Road
Suite 201
Mississauga ON L4T 4E5
(416)-678-6880

MCBA
425 West Broadway
Glendale CA 91204-1269
(818)-242-9600

MITCHELL HUMPHREY & CO.
11720 Borman Drive
St. Louis MI 63146
(800)-237-0028

Appendix B

SOME HP ACCOUNTING SOFTWARE VENDORS

MANAGEMENT TECHNOLOGY INT'L
16 Inveress Place East
Englewood CO 80112
(303)-790-7734

ORACLE CORPORATION
2929 Campus Drive
Suite 200
San Mateo CA 94403

SATCOM
4521 Professional Circle
Virginia Beach VA 23455
(800)-472-8266

CHRIS SCHAEFER & COMPANY
1500 S. Dairy Ashford Road
Suite 400
Houston TX 77077
(713)-558-2273

SMITH, DENNIS & GAYLORD, INC.
3211 Scott Blvd.
Santa Clara CA 95054-3078
(408)-727-1870

SOTAS INTERNATIONAL
192 Merrimack Street
Haverhill MA 01830
(508)-521-1300

TECHNALYSIS CORPORATION
6700 France Ave. South
Minneapolis MN 55435
(612)-925-5900

VOCOS
Dewitt Building
P.O. Box 874
Ithaca NY 14851
(607)-272-8464

Maximising the Return from Integrated Information Solutions

Peter Linkin

Hewlett Packard
Nine Mile Ride
Wokingham
Berks
UK

Introduction

Increasing productivity, revenues, reducing costs and adding value have been attributed to technology investments at some Hewlett Packard installations. However, many organizations are uncertain of what gains can be made from these investments and even after the implementation is complete, what benefits have in fact been obtained. Not having the answers to questions such as "What is the payback to be obtained from this system?" and "What return has been achieved from the installation?" have led to technology projects being cancelled or not being allowed to expand.

Many articles, papers and theories have been put forward as to how these benefits can be identified and how they can be quantified - all offering some good advice and a different methodology. So as a newcomer to this area or even as an existing user of HP computer systems, where should you start in identifying and quantifying benefits from information technology?

Planning or aligning an existing investment with a company's goals and critical success factors has led to the most credible method of cost justification for many Hewlett Packard customers. By doing this the implementation is focussed at achieving a specific goal or set of goals which the organization needs to meet and also helps in quantifying the benefits which are being obtained in business terms.

The benefits to be gained from using this planning methodology are very large. Existing customers often had a feeling that their installation was succesful, but no-one could identify exactly what benefits had been gained. This approach allowed that quantification to be made, giving the existing system and people involved in the implementation a stamp of success. Additionally, new areas of opportunity for technology were often identified. Management have proven to be willing to progress these areas due to the returns which had been identified on existing systems.

A customer example is given to demonstrate how this methodology has been used in practise, and to identify the real benefits which one of Hewlett Packard's customers is obtaining through the implementation of their integrated information system.

Some of the cost justification methodologies which are most prevalent at the moment are described in the following section, outlining the positive and negative aspects to them.

Cost Justification Methods

1. Cost Avoidance / Cost Saving

This methodology can be summed up in the following formula.

Time Saved X Annual Salary = Salary Expense Reduction

There are many variants on this formula, but the basis of the cost justification method is that information technology is going to make people more productive. Some vendors stipulate the areas in which these benefits should be gained and even state the percentage improvements which should be obtained through the implementation. Now everyone can complete their jobs more quickly ie. time is saved. If you add up all of the hours which are going to be saved through these productivity gains in one year and multiply those hours by the annual salary, you obtain the salary expense reduction which the company will achieve.

The end result which organisations should have achieved is therefore either less people doing the same work or the same people doing more work - either cost savings or cost avoidance in salary terms. Varying results have however been achieved.

Some companies have found that after the implementation there are just as many people doing the same jobs as before. In fact in some cases the numbers have increased because there were now trainers, administrators, programming and support staff which had to be added to the salary tab. Unless the spare time which is available for employees due to increased productivity is channelled into other work, the supposed benefits of technology are never realized.

Another pitfall with this methodology can be where employees are asked to undertake more of the same task performed prior to the technology implementation. Doing more of the same does not necessarily save the company money. Unless this incremental work is contributing to the goals of the organisation, money may need to be spent in other areas to achieve the goals. It is therefore essential that employees are rechannelled into work which is of maximum importance to the organisation to ensure that money saved or costs avoided are at the optimum level.

2. Performance Only

Proponents of this methodology feel that the benefits of implementing an Integrated Information System can be measured, but not calculated in dollars. Examples of the benefits which can be gained include:

- % improvements in customer service ratings
- % reduction in error rates
- % improvement in employee morale ratings
- Increased rate of innovation

A company can see whether the implementation is succesful by watching its customer service ratings go up, or by watching its employee morale ratings increase. The implication is that customers perceive the company to provide a higher service and should therefore buy more, or that employees morale increases and so their contribution to the goals of the organisation increase correspondingly.

Unfortunately as your customer service rating is increasing, your competitors could be bringing out a faster, stronger, better product at half the price of yours. Employee morale could be improved as the system allows them to complete their daily tasks in half the time, increasing leisure time. The increased contribution to the organisations goals can be zero, and the impact on the bottom line can be negative instead of positive.

The performance goals need to be aligned with the overall goals of the organisation to ensure that benefits are being made and that these benefits can be directly attributable to a company's bottom line.

3. Value Added Payout

This theory is based on the the premise that any employee's job can be divided up into different categories eg. secretarial and clerical activities, administration, technical analysis, supervisory activities, management etc. Work which an individual carries out to achieve their objectives can span across the low value clerical tasks through to high value management tasks. Information technology should automate the more mundane, clerical aspects of any person's tasks, leaving them free to perform the higher value, higher cost, management activities.

The results have often been very different. Giving managers sophisticated tools can result in more of the clerical work being handled on their own - reducing the dependence on secretarial and support staff. The benefits which they gain in performing these tasks is in having greater control of when and how these activities are done. The tools can often mean that whereas specialised skills were required from secretaries and administrators before, these tasks become simple enough to be completed by a non specialist. ie. the manager can do it himself.

So how do Hewlett Packard advise customers to approach cost justification or identification of opportunity areas which would maximise benefit to the organization?

4. Hewlett Packard approach

We believe that by identifying the following information from management within an organization, you will be able to identify where technology should be focussed for maximum return, as well as identifying in monetary terms the returns the organization will obtain from the investment.

The information which needs to be collected is:

The goals of the organization

Critical Success factors which will enable the goals to be met

Obstacles to achieving the critical success factors

and implications to the organization if the obstacles are not overcome.

In order to be ensure that information technology is being applied in those areas of maximum importance to the organization, the implementation needs to be aligned with the goals of the organization.

It is essential to understand the critical success factors of the business. These are the activities which absolutely MUST be achieved in order for the goals of the organization to be met. Applying technology to meeting the critical success factors will ensure that the implementation is going to be focussed on the highest value payback for the organization.

Understanding the obstacles to meeting those critical success factors will allow you to focus on specific areas where information technology needs to be applied.

Obtaining the management question of "What's the return?" can be achieved by understanding the impact to the organization of not being able to overcome these obstacles. This information can be gained through the management interview process.

There are three major benefit areas that Hewlett Packard believes can be achieved from a information technology implemenation:

A. Hard dollars

This is where there is a 100% correlation between the introduction of technology and the end effect e.g. By automating forms distribution and collection, you may want to redeploy two clerical people to other areas. In this case the only contributor to reducing headcount by two would be the implementation of technology. This category of benefit is

therefore directly auditable ie. payroll amounts for the department should reduce by two staff after the implemenation.

B. Effectiveness

In some instances many different things are changing at the time that technology is implemented. For example, you may be automating your sales force and achieving increases in sales, but there could be other factors which contribute to this change. For instance new products may have been introduced or a new remuneration system could be implemented, all impacting the increase in sales. In this instance it would be up to management to determine how much of the change is attributable to the technology as opposed to other factors. The benefit is however measurable and quantifiable.

C. Added Value

This category of benefit is measurable but not quantifiable or auditable. It relates to benefits which are experienced often by the individual who is using the technology or the end customers of the user. Types of benefits which are included in this category are: improved teamwork, increased responsiveness for customer queries and increased morale.

Depending on the methods used for cost justification in any company, it may be acceptable to include these benefits or there may be a need for just focussing on hard dollar and effectivess benefits.

Customer Example - Thompson Industries

The above approach was taken with a number of customers to identify what returns they had obtained from implementing Hewlett Packard integrated information systems. The example which is explained in detail is for Thompson Industries. One day was spent in management interviews to collect information, this was analyzed by Hewlett Packard, and presented back to the customer for their agreement on the findings. Information collected related to the following:

- Goals of the organization
- Critical Success factors
- Obstacles to meeting the critical success factors
- Implications to not being able to overcome the critical success factors
- Management use of technology and its perceived effect on the organisation

Thompson Industries management provided the following information in the interview process.

Customer Overview

Thompson Industries is headquarted in Phoenix, Arizona and has plants in eleven locations nationwide. Thompson is a

manufacturer and distributor of styrofoam cups, containers and plastic lids. They have the second largest share of this market.

Thompson's customers fall into two categories: institutional, with customers like McDonalds, and consumer, such as Southland Corporation.

Thompson has 1200 employees and annual revenues of $120 million. Its goals are to understand and satisfy its customers, to stay competitive in the industry, and to always deliver defect-free quality products.

Critical Success Factors

Thompson Industries' critical success factors are to:

- Meet quality standards by "doing it right the first time."
- Provide the right information to the right people at the right time
- React to competitive pricing situations
- Provide communication links to plants
- Adhere to a five day turnaround policy to maintain an acceptable and competitive service level
- Maintain good communication with all internal (within Thompson) and external customers

Hewlett Packards Integrated Information system

The company was already using Hewlett Packard for their manufacturing system. They layered an integrated information system on top of the existing manufacturing system, the foundation of which was electronic mail. Feedback from key managers was that the integrated business system was as vital to the organization's business as the manufacturing system. The areas that the integrated business system are being used is listed below:

Solution: DeskManager (Request and approval of pricing deviations)

HP DeskManager is being used in a highly proactive way, particularly for the sales force. Sales representatives who go into a customer location are sometimes required to deviate from their set pricing to counteract a competitor's price. It is now possible for sales reps to immediately request and obain approval for competitive pricing deviations using HPDesk. This allows sales reps to respond to competition in a timely way. Also, there is a record of the deviated pricing and approval which is kept for historical information on the account. This entire process is said to "save business."

Solution: DeskManager (Order Entry)

The on-line electronic messaging capability of HPDesk is used in the order entry process, providing many benefits. It allows

for verification and communication of inbound customer orders to the plants. The process provides the capability to achieve a quick turnaround from the date of order to when the order can be shipped. This is critical to the five-day turnaround policy which is required in the industry.

Solution: DeskManager (Communication Link)

HP DeskManager is also used as the communication link throughout the multiple location organization. This benfits Thompson by being able to provide timely information to the right people in a time-critical business.

Solution: DeskManager (Product Quality Evaluations)

Product Quality Evaluations have benefitted by the use of electronic messaging. Employees can detect a quality problem, and quickly resolve it by notifying the proper individuals. This results in considerable cost savings by immediate troubleshooting.

Solution: DeskManager (Purchase of Packaging Materials)

The purchasing of all plant packaging materials is done with HP DeskManager. What was a paper/telephone process has been converted to an on-line process, which now provides 100% accuracy when placing orders for packaging materials. Before, purchasing errors resulted in buying materials which could never be used, wasting considerable money for the company, as well as delaying shipments, having an impact on revenues and customer satisfaction.

Solution: DeskManager (Improve Customer Service)

There are many areas where customer service has been improved by the use of electronic mail. One area in particular involves the use of portable computers in the field which link the regional sales manager with the plants and headquarters. The field can alert multiple locations to potential customer problems, hold people accountable for sales-critical information, and provide information to customers in a more timely way.

Headquarters can also alert plants to problems allowing quick responses to customers.

Solution: DeskManager (Network Headquarters and Plants)

The strong company communication link provides enhanced communication between departments and the remote plant sites, and eliminates the time-zone problem which is an important benefit as the plants and sales reps are nationwide.

Solution: WR (Integration of WP and E-Mail)

Word processing is another aspect of the Thompson integrated information system. The advantages of the ability to integrate word processing with the vital electronic messaging network benefit by being able to send production plans to remote plants for immediate feedback, speeding up the entire production process. Various key managers are now able to share memos and other important strategic and time-critical information with multiple locations and employees at the same time with no time lag.

Solution: WP (Memo and document production)

The automated memo and document production capabilities contribute to the image of quality which Thompson prides itself on.

Configuration

The total configuration at Thompson consists of one HP3000 Series 70; PCs and terminals at headquarters and elevan plant locations connected via an X.25 network. They are piloting portables in the field for sales force automation productivity benefits. Thompson has ASK manufacturing software installed.

The Benefits Model

The benefits which Thompson Industries has achieved are significant and are outlined in terms of the benefit model. The benefits model takes into consideration the monetary benefits a company can achieve in the dollar, effectiveness and added value areas.

The Dollar Benefits

Thompson Industries achieved significant dollar benefits. In the area of lower operating cost, Thompson's manufacturing department was able to reduce the travel expense associated with visiting plants by $220K/year. Also, there has been less reliance on an inbound WATS line at a conservative savings of $15K/year.

In the area of cost avoidance, a major saving was realized by no longer requiring the IBM System 38 solution which was installed prior to the implentation of Hewlett Packard's integrated information system. By eliminating the IBM equipment, Thompson is able to realize a saving of $190K/year in maintenance and leases. They also managed to net $6K on the sale of their System 38.

In the area of lower personnel cost, the use of electronic messaging to communicate with the plants has also resulted in reducing staff. Although the reduction of these positions is not 100% cause and effect to HP DeskManager, the company attributes a $150K/year personnel reduction in this area to electronic messaging benefits.

Thompson has realized a $50K/year savings in avoiding adding additional administrative support personnel which they either once had, or would have needed with the addition of more management positions as the business grew.

The Effectiveness Benefits

The effectiveness benefits contribute substantially to the value obtained from the system.

Productivity of professionals has increased. Using a factor of salary plus benefits averaging $50/hour, the productivity dollar benefits of 40 professionals who save (conservatively) 2 hours/week equate to $280K/year. Administrative productivity is said to have increased due to reducing the paper-based and telephone-based communication with the plants and now using electronic messaging. As an estimate, each secretary communicates with each plant 2-3 times daily. If there were a reliance on memos, the creation and distribution process would be far more time consuming than the on-line communication of messages which is now the standard method of getting information to the plants. If we estimate that the secretary saves 1 hour/day as a conservative standard, and apply that to the five secretaries, Thompson is realizing $33K/year.

However, these productivity gains are only significant when the emphasis is placed on what it means in a business sense to have the information transmitted to the proper people in a more timely way. One example is that if the plants did not receive information on defect levels, it would affect capacity utilization; if they did not know what level of loss was, Thompson would be selling capacity that was not available. The estimate is that this dollar amount could be translated into a 10% cost of sales. This is when productivity can be ultimately equated to revenues.

The telephone usage at Thompson has been cut by 50%, the primary benefit being the elimination of telephone tag, missed messages and timeliness of information. Also in the timeliness area, remote plants receive information at the same time as corporate employees, resulting in a more coordinated production process. A hard copy audit trail provided by the electronic messaging solution is key to many people who need documentation of communcation and, as in the case of the sales force, need to hold people accountable when dealing directly with customers. The sales force maintains that there is a 25% increase in face-to-face time in front of the customer due to the use of the sales force automation pilot. Customer service levels are improved in numerous ways falling into the effectiveness area, including the automated purchasing process, which saves time in shipping out orders to the customer.

Added Value Benefits

The measurable benefits shown in previous paragraphs demonstrate significant, measurable monetary benefits which the company has achieved. Often companies are looking for paybacks in these areas alone. In addition to these benefits Thompson Industries achieved added value benefits which can convert to more business and a lower operating cost for the enterprise.

The added value benefits which Thompson achieved all relate back to the goals of satisfying the customers, staying competitive and delivering defect-free, quality products. There is a value in reducing the emphasis on working in a crisis mode. The open communication throughout the company includes possible "overlooked" people who could be crucial to the manufacturing process. The benefits of reducing conflict in the organization indirectly impacts the productivity area, which is seen as providing Thompson the ability to maintain their competitive advantage.

Employee morale has been enhanced, which should result in less turnover, and Thompson can protect the investment it has made in its people over time. Importantly, the ability to communicate quickly and with a high level of quality all information relating to the business processes reinforces the emphasis on quality in all aspects of the organization.

Payback Period

After doing a payback analysis based on a sample of the benefits Thompson Industries has achieved, payback was realized in less than one year.

The payback calculation is based on an investment of $75,000 for the integrated information system portion of the HP computer system.

This payback analysis was done using a formula which is very conservative and takes only 50% of the benefits in the first year of system installation. This takes into consideration learning time and time to develop and fully utilize the many applications.

Summary

In summary, Hewlett Packard sees the most credible methodology for ensuring that IT plans are going to achieve maximum gain for the organization by aligning those plans with the goals and critical success factors of an organization.

Going through the interview process with senior management to determine the answers to these questions will enable key opportunity areas to be identified. Information on the impact to the organization in not being able to overcome obstacles to achieving critical success factors will enable you to determine potential areas of cost savings / cost avoidance and added value to the organization.

Building up your cost justification case using the benefits model of

- Hard dollars
- Effectiveness and
- Added value

will enable you to answer questions about where the return on the system is to be obtained and how.

This methodology has now been used succesfully in a number of installed and prospective Hewlett Packard sites. The benefit gained by the proposers of the information system has been sign off of their plans by management. Benefits to higher management has been a high degree of confidence of where benefits will be gained through the implementation. Benefits to end users has been a higher degree of certainty of how the technology would affect them, enabling them to make a smooth transition to their new role.

As management are increasingly asking the question of "Where is the return to be obtained?", I hope that the talk has provided you with some insight as to where the answer to the question can be obtained.

Leasing: A Strategy for Reducing Cost of Ownership

Joanna Wampler
Hewlett-Packard Company
Finance and Remarketing Division
331 East Evelyn Avenue
Mountain View, California 94041

Recently, we heard from a customer who was in a real bind. Two years ago, they needed to expand their computer room. They were interested in acquiring a new technology which was being developed but not yet released. In the interim, they elected to buy an alternate system with the intention of selling it to a broker when the new technology became available. Due to cash constraints, their financial people took out a loan for the equipment.

They are now ready to move to the new technology. Much to their surprise, however, the cost of paying off the loan far exceeds the proceeds they will get from the sale of the existing system. They have the budget to buy the new technology but the acquisition is on hold because they can't afford to dispose of the existing system!

Where did this customer go wrong? They had done a great job evaluating the cost of operating the computer system. They had looked at price performance, training, maintenance and staffing requirements. They had explored their future needs and factored the cost of the new technology into their budget. How could they have possibly anticipated the disposal cost?

In actuality, this customer could have avoided being painted in a corner if they had done a better job of matching the financing with their long-term system needs. They had excelled at evaluating the impact of the improved technology on their overall system costs but failed to recognize that financial factors can have just as big of an impact.

An effective cost of ownership analysis must cover both the technical and financial sides of the solution. A key success factor in this analysis is choosing a financing method which gives the users the amount of flexibility required to meet their long-term needs at the lowest cost.

This paper will describe how selecting the appropriate financial avenue contributes directly to successful cost management. It will give an overview of the most popular financing options available and how they vary in the amount of flexibility they afford. The focus will be on leasing as a financing source and how the lessee can maximize the cost effectiveness of the system through the careful selection of the lease structure and negotiation of the terms.

Financing Options

To successfully integrate the long-term plans into the financing decision, it is very important to understand the different financing options available. For the purpose of this paper, we will focus on the four most popular ways to finance equipment: (1) cash; (2) installment loans; (3) capital leases; and (4) operating leases.

Cash Payments

Companies which pay cash for their systems are generally either cash rich and/or debt averse. Some people do not consider a cash payment as "financing" as there is no interest charge involved. However, there is an opportunity cost associated with a cash payment. By using funds to purchase a system, there are less funds available to invest in other areas of the operation.

Financial theory suggests that long-term assets, such as computer systems, should always be financed through long-term debt. Cash should be reserved for working capital to fund the daily operations of the company and invest in such areas as R&D. However, if a company has more cash than it needs for operations, it may elect to use working capital to buy long-term assets.

When a company pays cash for a system, it is recorded as an asset and depreciated over its economic life. This accounting treatment is the same for both the financial and tax books.

Installment Loans

Installment loans are designed for companies which want to own the equipment without incurring a large cash payment upfront. As it is generally desirable to minimize the monthly cash outflow, a five year term is common on computer equipment. Installment loans are written such that the equipment is completely paid for by the end of the term.

As with a cash payment, the system is recorded as an asset and is depreciated. In addition, the debt is shown on the balance sheet as a liability and the monthly interest charge is expensed. This accounting treatment is the same for both the financial and tax books.

Capital Leases

Capital leases are very similar to installment loans in that the lessee's intent is to own the equipment and they are trying to match the payment with the benefit they receive from the system.

Capital leases are defined by the Financial Accounting Standards Board (FASB) Statement No 13. Basically, FASB 13 states that if the lessee's intention is to own the equipment, they should capitalize it on their balance sheet

just as they would for a cash purchase or installment loan. FASB 13 then goes on to give four criteria which distinguish whether a lessee plans to own or just rent the equipment. If a lease meets any of these four criteria, it is considered a capital lease.

Per FASB 13, a lease is to be capitalized if it has either an automatic transfer of title or an end-of-term purchase option below the equipment's expected fair market value at the date that the option becomes exercisable. FASB also considers a transaction to be a capital lease if the lessee maintains possession of the system for more than 75% of its economic life or if the present value of the non-cancellable lease payments exceeds 90% of the system's fair market value.

A capital lease is treated similarly to the installment contract in the financial statement The asset is capitalized and the corresponding debt is shown as a liability. The value of the debt is calculated using a formula provided by FASB. Each month, the lessee will expense both the interest portion of their lease payment and the depreciation of the system.

One major difference between a capital lease and an installment loan is that a finance lease may have different treatment on the tax books depending upon how it is structured. The IRS has rules addressing the accounting for leases. As FASB and IRS accounting rules differ, it is possible for a lessee to capitalize a lease on their financial books and not capitalize it on their tax books.

The gray area between FASB and IRS classifications is beyond the scope of this paper. Suffice it to say that lessees should always consult with a qualified accountant as to both the financial and tax implications of a lease structure before signing a lease agreement.

Operating Leases

Operating leases are geared toward companies which do not necessarily want to own the equipment. The terms of these leases generally span from twelve to 48 months for computer equipment. A monthly rental fee is assessed and at end of the lease, the equipment can be returned with no additional obligation. In most cases, the rental payments will not have covered the full price of the equipment over the life of the lease. The leasing company must therefore be able to re-sell the system on the used equipment market to make its profit.

Companies which are concerned about obsolesence or have only a limited need for the equipment usually finance through operating leases. However, there are some companies who want to own the equipment and still use operating leases. They do this for accounting reasons. Depending upon the accounting assumptions they make, they may not be required to record either the system as an asset

or the lease obligation as a debt on their financial statements. This "Off Balance Sheet" treatment makes operating leases very attractive to companies who are concerned about the amount of debt they show on their books.

Furthermore, as some companies classify operating lease payments as operating expenses instead of capital expenditures, operating leases also appeal to managers of companies with limited capital budgets.

For the financial books, neither the asset nor the corresponding debt are recorded. Each month, the entire lease payment is expensed. In most cases, an operating lease will receive a similar treatment on the tax books. This is called a true lease - the IRS's equivalent of FASB's operating lease. However, just as with the finance lease, there are some differences between the FASB and IRS classifications. It can be very costly to assume that FASB and the IRS will always treat a given lease structure the same way. Therefore, consult with a qualified accountant about these issues before signing the lease agreement.

If a company truly wants to own the system, an operating lease is not a cost effective method of financing. The cost of exercising the purchase option makes the overall cost of an operating lease extremely high compared to that of an installment loan or a finance lease. This is the price paid by the lessee for the fact that they took no risk in the future value of the system.

Recent Changes in the Leasing Environment

Both capital and operating leases are gaining popularity as sources of computer funding. A Price Waterhouse/Datamation survey estimated that 35% of all systems sales were leased on operating leases in 1987 and that number would grow by over 11% in 1988. The increased demand for computer leasing can be traced to several environmental changes.

1) Alternative Minimum Tax - The Tax Reform Act of 1986 tightened the provisions for the Alternative Minimum Tax (AMT). The end result of this measure was that corporations could be subject to additional tax liabilities if there were significant differences between the income they reported to their stockholders on their financial books and the income they reported to the IRS on their tax books.

In most cases, there will be a variance between the profit on tax and financial reports because companies are allowed to use different accounting methods for the two sets of books. Many companies, for example, use a different depreciation method for tax purposes than they do for financial reporting. If the difference between the two set of books becomes significant, the IRS

requires that the income reported on the tax form be adjusted to more closely reflect the income reported to shareholders. For companies on the brink of becoming subject to the AMT, it has become necessary to avoid creating any more of these differences.

One way to do this is to enter into operating leases. Since the transaction will usually be treated as a rental, it is not recorded as an asset and, therefore, is not depreciated. This off balance sheet treatment averts the depreciation expenses and prevents the potential reporting difference.

2) Computer Technology - A second reason for the growth in leasing has to do with computer technology. The interconnectivity of hardware gives the customer more flexibility in changing hardware vendors. Also, the technology is turning at a much faster rate. Unsure that sufficient upgrade paths are being provided, customers are seeking the ability to walk away from the equipment after a few years. Again, the operating lease is the preferred solution for these customers.

3) Industry Competition - The demand for leasing has also been heightened due to changes within the leasing industry itself. During the early 1980's, the consistent high growth in the leasing market attracted many new players including independent leasing companies, banks, and computer manufacturers. These lessors made huge profits from the tax benefits associated with lease funding.

With the Tax Reform Act of 1986, many of the these tax benefits were repealed. At the same time, the market's growth rate began to slow, reflecting a more mature industry. The result has been a consolidation within the industry and fierce competition for the lessees' business. The Tax Reform Act took away much of the appeal for capital leases for both the lessee and lessor. As lessees began to realize the benefits of operating leases, the leasing industry also turned its attention to developing this business.

Looking Beyond Rates

The effect of this competition has created a tremendous opportunity for lessees as operating lease rates have dropped dramatically. At the same time, however, lessees need to look beyond the rate. They need to integrate the user's needs with the financial decision and seek financing structures which will minimize the cost of meeting them.

In many cases, giving the financial decision maker access to the future plans for the system will be a new way of doing business. It will require meaningful communication between the financial and technical decision makers. This

may involve a substantial time investment depending on how well they currently understand each other's way of evaluating a decision. The following questions can be used as a basis for opening the communication channel.

1) How long will the configuration being bought today meet the user's needs?

   This question will help gauge the useful life of the system to the user. If the user is viewing it as a long term investment, the financing should be geared toward ownership. However, if the system has a limited useful life for him, an operating lease may be a better strategy.

2) When the configuration needs to be changed, will it require the return of all or part of the existing system in exchange for a new product?

   This information is necessary in clarifying the issue of useful life. Sometimes a system can be expanded to meet the user's needs by merely adding some additional equipment. For example, the user may tell you that the system can be used for two years before it will need to be changed. At that time, the system will need another disc drive and some additional terminals. It will then meet his needs for another three years. The useful life of the system is really five years and the financing should result in ownership.

   Compare this to the scenario where after two years he plans to replace the system by a new model which will be introduced next year. Given the fact that the current technology is about to be updated and its market value, therefore, driven down, it may not make sense to commit your firm to owning the system. It may be preferable to merely lease the system for the two years.

3) Is there a long-term commitment to the current computer manufacturer?

   This question is becoming increasingly important as the computer industry moves towards interconnectivity. The fact that you can assemble a network of computers from several different manufacturers means that it will be less expensive to switch from one vendor to another.

   There are definite advantages to partnering with a limited number of vendors. A long-term relationship gives the vendor the opportunity to really understand your business and consult with you on how their technology can work to maximize your profits.

   However, occassionally there are other factors in the relationship which are unsatisfactory. There may be problems with service or support. The user may feel uneasy that the current vendor will have the best

technology available when he needs it. If he has these types of doubts, he may be thinking of switching vendors. In this case, it is important not to commit to ownership of the equipment through the wrong financial arrangement.

4) Where is the proposed system in its life cycle?

This question is extremely relevant if the user is not committed to the current manufacturer. Computer equipment loses market value very rapidly as the technology becomes dated. Therefore, if you are looking at new technology, you may feel comfortable in assuming an ownership position and then selling the system on the used equipment market should the user change vendors. However, if the technology is already mature at the time you obtain it, you may be better off with an operating lease in which the lessor takes the risk on the equipment's future value. The monthly payment will be higher on a short-term operating lease than on an installment contract or capital lease, but the overall cost will be less if the equipment becomes obsolete.

5) Does the manufacturer offer trade-in credits?

Manufacturers offer trade-in credits as an incentive to purchase new technology. While it is impossible for the manufacturer to commit to a set trade-in credit for an upgrade that will take place three years hence, the technical decision maker will probably have a good idea of the manufacturer's commitment to trade-in credits. This is especially true if there has been an on-going relationship with the vendor. For example, the buyer may know that the manufacturer always offers generous credits when a new technology is introduced as a way to establish market share. The existence of a trade-in credit helps offset the risk of ownership. It insures the future value of the equipment as you do not need to go to the used equipment market to sell the system. You, in effect, sell it back to the manufacturer.

Using Net Present Value Analysis

The benefits to be derived from matching the financing to the user's needs can easily be expressed in numbers. To explore this issue, we will be using Net Present Value analysis (NPV). NPV is based upon the premise that a dollar received today has more value than a dollar a year later as today's dollar can be invested to earn interest. Therefore, comparing a dollar received today with a dollar received next year is really comparing apples and oranges. The amount of cash payments and receipts occurring over the course of the year is the annual "cash flow."

NPV analysis takes the cash flows in future years and translates them in terms of today's dollars. By doing

this, all of the cash flows have the same base value and can legitimately be compared. NPV analysis is very prevelant in evaluating financing options. The various options result in different cash flows depending upon the timing of the payments and the tax implications.

For example, assume that your firm has decided to acquire a system costing $100,000. The funds have been budgeted for a cash payments. You calculate the NPV of buying the system and depreciating it over five years as $74,905 (See Appendix B Case 1).

Now assume that in a conversation with the technical buyer, you discover that they only plan to use the system for three years. By that time, the demands on the system will be such that they can justify investing in a new computer architecture which should be available next year. The NPV of buying the system with cash now drops to $59,208 because you will be able to re-sell it on the used equipment market after three years and recoup some of your investment (See Appendix B Case 2).

The fact that a new computer architecture is imminent, however, should serve as a red flag. Perhaps the re-sale value is overestimated as the new technology may obsolete the existing system. There is now a larger degree of risk in taking an ownership position in the equipment since the user's intent is to dispose of it.

To avoid the risk of ownership, you might consider an operating lease. The NPV of a 36 month operating lease when the equipment is returned is $51,837 (See Appendix B Case 3). This cost is significantly lower than cost of a purchase and re-sale and it is does not involve making any risky re-sale estimates.

This is not to imply that an operating lease always leads to cost minimization. As previously mentioned, it is not cost-effective to use an operating lease if the user's intent is to own the system. For example, the 36 month operating lease has an NPV of $71,156 if the purchase option is exercised at the end of the term (See Appendix B Case 4). Compare this to the NPV of a 5 year capital lease ($64,263) or a 5 year installment contract ($62,979) (See Appendix B Cases 5 and 6).

Chart 1 summarizes the cost of use for a $100,000 system. It compares the cost differential between renting equipment on an operating lease and a cash purchase with a re-sale.

Chart 2 compares the cost of ownership for the same system. In this case, it shows the cost differential between exercising the purchase options on leases with various terms and paying cash.

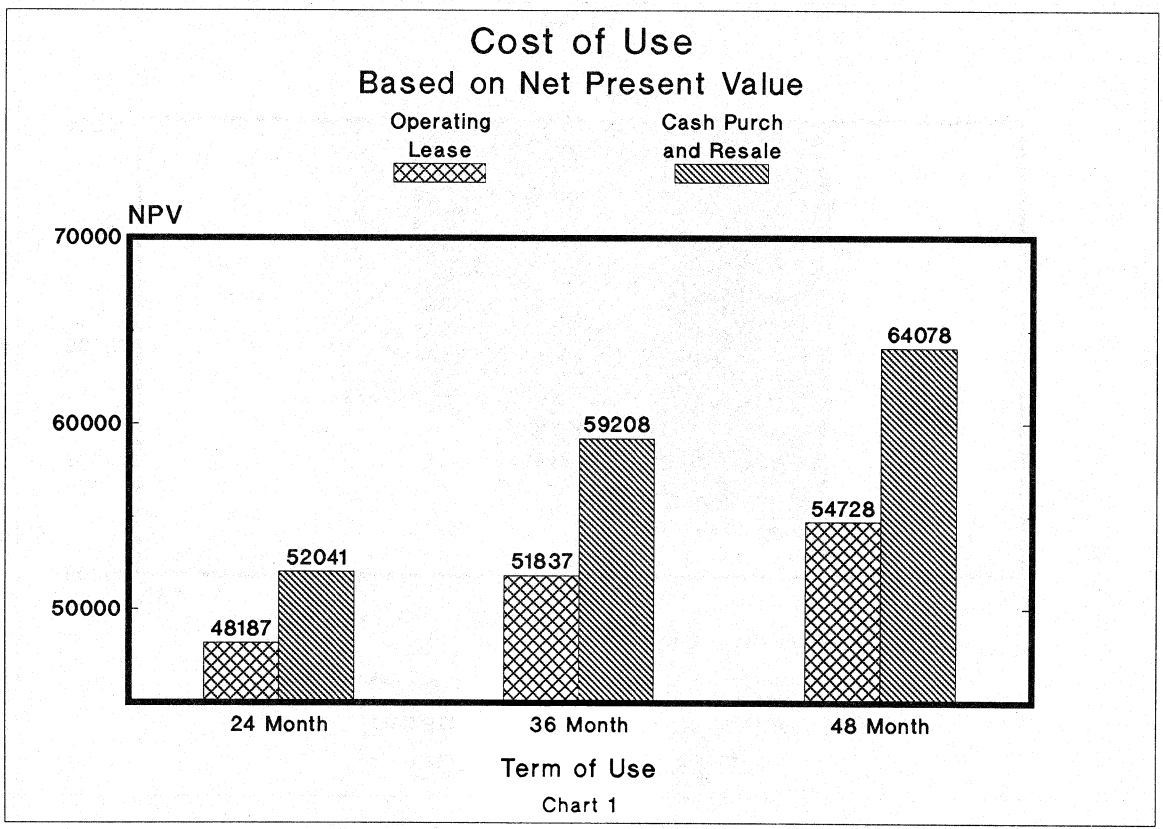

Cost of Use
Based on Net Present Value
Operating Lease
Cash Purch and Resale
NPV
70000
60000
50000
48187
52041
51837
59208
54728
64078
24 Month
36 Month
48 Month
Term of Use

Chart 1

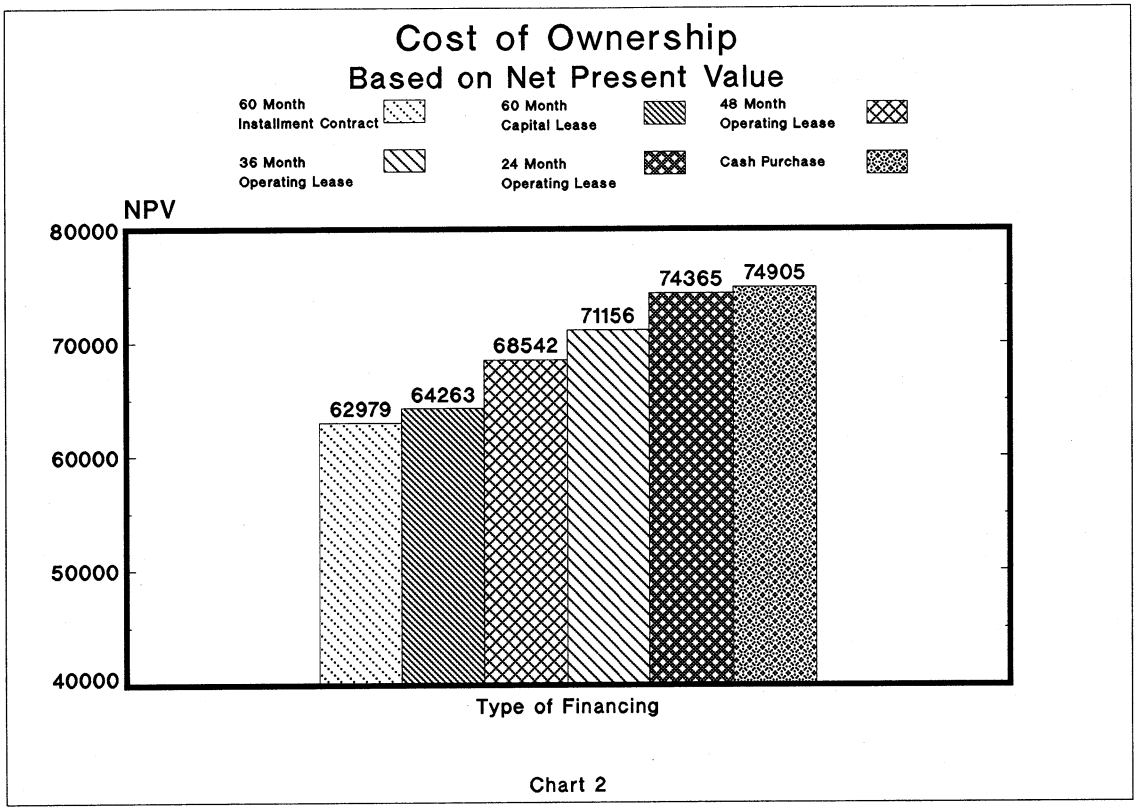

Cost of Ownership
Based on Net Present Value
60 Month Installment Contract
60 Month Capital Lease
48 Month Operating Lease
36 Month Operating Lease
24 Month Operating Lease
Cash Purchase
NPV
80000
70000
60000
50000
40000
62979
64263
68542
71156
74365
74905
Type of Financing

Chart 2

Lease Negotiations

In these examples, leasing has proved less expensive on an NPV basis than purchasing with cash. In many situations, this will be true. However, in some cases, it will not. For example, if your company has a much lower cost of capital than the implicit rate of the lease, it may be less expensive for you to use cash. Also, if your intent is to own the equipment, short-term leases may be more costly than cash.

Leases may also become more expensive than cash purchases if the lease is not properly negotiatied. Just as the financial decision maker needs to ask the appropriate questions of the technical buyer to fit the financing to the system's intended use, he must ask the right questions of the lessor. It is not good enough to merely shop for the lowest monthly payment. The cost of leasing equipment can be significantly reduced if the lessee negotiates financing terms to meet the user's needs. Again, you should enter the lease negotiations with one of two long-term strategies: equipment use or equipment ownership. You should negotiate the terms focused on this strategy. If the user is not totally sure of whether he will want to own the equipment or return it, choose the strategy which he is leaning towards today but negotiate terms which will allow him to go the other way without substantially increasing the costs.

General Questions:

1) Is the monthly payment quoted in advance or arrears?

   The timing of the payments can affect a lease's true cost. Assume that two leasing companies both quote a monthly payment of $2,224 on a five year, $100,000 installment contract. Both leases will start in January 1, but one is in arrears and the other is in advance. The payment for the lease in arrears will be due on January 31. The payment for the lease in advance will be due on January 1. This means that you have access to your money for an additional 30 days with the lease that is in arrears. In terms of Annual Percentage Rate (APR) this translates to rates of 12% for the arrears lease and 12.4% for the advance lease. Therefore, the timing of the monthly payment is a very important factor in comparing lease costs.

2) Are deposits required with the lease?

   Using the above example, assume that the arrears lease requires no deposit while the advance lease requires that the first and last payment be remitted upon signing the lease agreement. What does this do to the APR?

   The APR of the arrears lease stays the same. The APR of

the advance lease, however, now goes all the way to 12.9%. You should also inquire about the deposit refund should you decide to cancel the order after the lease agreement has been signed.

Questions For Equipment Ownership:

1) What is the buy-out option at the end of the lease?

Lessors make their profit from operating leases through recovery of the residual value of the equipment. They do this by selling it to the lessee at the end of the lease term or by selling it on the used equipment market. The lower the initial lease rate, the more residual there is to recover. Therefore, if the monthly payment is significantly lower than other quotes you've received, chances are that the lessor is looking to make up for the low rate by charging a higher buyout price.

2) If the buyout option is priced as "Fair Market Value," how will that figure be derived?

Many lessees feel protected from an unexpectedly high buyout price when they sign a lease agreement with a fair market value purchase option. This purchase option implies that they will be charged the price at which comparable systems are being sold on the wholesale used equipment market. Most lease agreements, however, do not specify who sets the fair market value, how it is determined and what recourse you have if you don't believe that the buyout amount truly reflects the equipment's fair market value. It is important, therefore, to clarify these points before signing any documents. When you and the lessor have reached an agreement on these issues, be sure to incorporate it in writing in the terms of the agreement.

3) What will it cost if you need to buyout the lease before the end of the term?

As seen on Chart 2, the cost of ownership is reduced by entering into longer term lease agreements. If your technical buyer is sure that he wants to own the equipment but is not sure about when he must have title, you may want to sign up for a long term lease with the ability to buy it out early at no penalty.

For example, the technical buyer has selected a system that can meet his long-term needs through the acquisition of additional boards. Therefore, he wants to own it. However, there is a new computer due out next year which may be even better suited to his needs. When the new technology is introduced, he expects that there will be very attractive trade-in credits. Therefore, he still supports taking an ownership position. However, to get the trade-in credits, he will

need title to the leased system. Therefore, there must be a provision for an early buyout.

To be truly safe, the method of determining the early buyout price should be specified in writing in the contract. You should also ask if there will be any additional charges for the early buyout, such as administrative fees, and get these fees in writing.

Questions for Use:

1) What will it cost to be let out of the contract prior to the end of the lease term?

Most leases are non-cancellable and the lessee is responsible for paying all the scheduled payments. However, on occassion lessees have overestimated the length of time for which they will need a system. It is a good idea, therefore, to find out what will happen if you are two years into a three year contract and the user simply doesn't need the system anymore.

For example, assume that the user has misjudged your firm's growth and that the system is too small to meet his needs. He cannot upgrade it and the value on the used equipment market will not offset the cost of buying the lease out. How much will you have to pay the lessor for them to take back the equipment?

As mentioned, you will be responsible for the remaining payments on the lease, in this case 12 payments. However, if you are going to pay them all at once, you should get some credit as the lessor is going to access to the funds prior to the scheduled payment date. The fact that they can now invest that money and earn interest should compel them to charge you less than the full amount of the remaining monthly payments.

2) What are the renewal options at the end of term?

Just as a lessee sometimes overestimates the usefulness of a system, they can also underestimate the amount of time for which the system will be needed. The danger in underestimating the lease term is that it can drive up the cost of use dramatically unless the lease is structured with an affordable renewal option.

Chart 1 shows that the lessor recoups more money from the system for longer-term leases. Conversely, the shorter the lease term, the more risk borne by the lessor. Therefore, to offset this risk, lessors charge higher monthly payments for short term leases. If there is no rate-reducing renewal option, the lessee may end up paying the high, shorter-term rate even if they keep the system for a longer term. Chart 3 shows what happens to the cost of use if the equipment is needed

Cost of Underestimating Usage by 12 Months

No Rate Reduction for Renewal Period

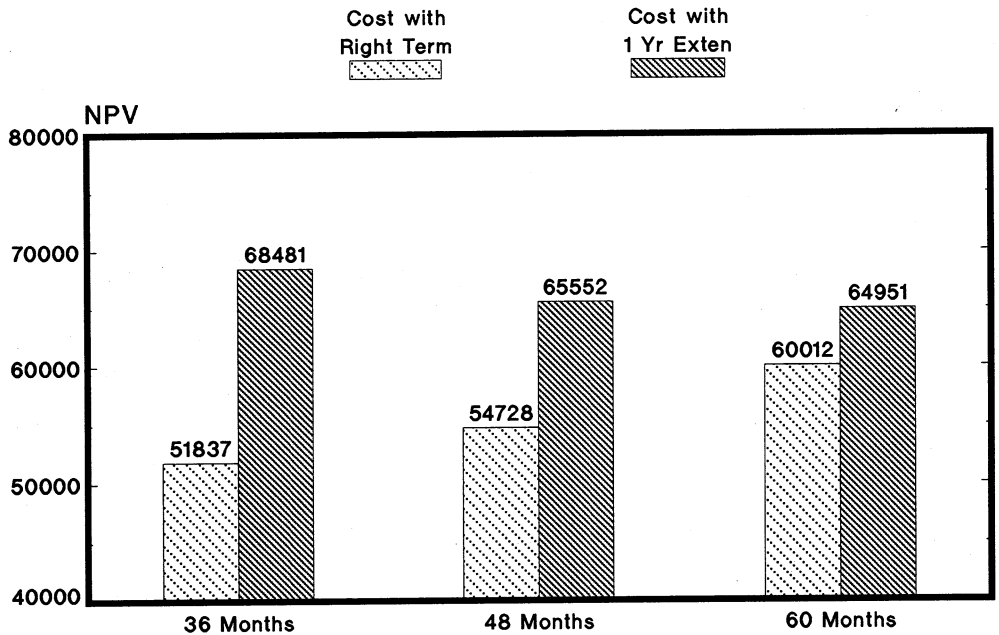


Chart 3

for another 12 months and no rate-reducing renewal option is available. As you can see, if you sign up for a 24 month operating lease and keep the equipment for 36 months, it will cost over $16,000 more than if you had originally signed a 36 month lease. This is almost a 33% cost increase. If a rate reduction had been negotiated into the renewal option, the underestimation of the term requirement would have been much less expensive.

Conclusion

An effective system analysis must cover both the technical and financial sides of acquiring a system. More importantly, the two sides must be viewed as interdependent. Successful integration can be accomplished with a three step process: (1) determine the long-term system plans; (2) identify the financing option which will give the appropriate amount of flexibility for the lowest cost; and (3) effectively negotiate the terms of the financing.

The ability of the financial decision maker to successfully complete this process is enhanced by the current lease market. Competitive forces are compelling lessors to help their customers deal with these issues and find creative ways to meet their needs. While leasing may not always be the preferred solution, its potential effect on the overall system cost should always be considered.

APPENDIX A

Assumptions for Net Present Value Analysis

Customer Data:

System Price	$100,000
Cost of Capital	12%
Tax Rate	34% (Total for all taxing entities)
Tax Dep'n Method	MACRS - 5 Years

Monthly Rate Factors: *

24 Month Operating Lease	3.600%
24 Month Installment Loan	4.731%
36 Month Operating Lease	2.725%
36 Month Installment Loan	3.345%
48 Month Operating Lease	2.275%
48 Month Installment Loan	2.658%
60 Month Finance Lease	2.102%
60 Month Installment Loan	2.250%

End of Term Purchase Options:

Operating Leases	65% Equity Accrual
Finance Lease	10% Purchase Option

Tax Treatment: **

Operating Leases	True Lease (Expense Entire Payment)
Finance Lease	True Lease (Expense Entire Payment)
Installment Contract	Conditional Sale (Expense Interest and Depreciation)

General Assumptions:

1) All cash flows are calculated on an after-tax basis.
2) The cash flows are discounted on a yearly basis. The cash flows are discounted in arrears.
3) The cost of exercising a purchase option is valued net of the present value of the resulting tax benefits. These benefits are gained from the depreciation claimed on the system after the purchase option is exercised.
4) The re-sale value of equipment bought for cash is based upon a decline in the retail value of the equipment of 20% per year. The equipment is then assumed to be sold at 60% of the current retail price.***

*** These rates are for a hypothetical situation. Actual market rates may vary.**

**** Consult a qualified accountant about the tax treatment of a specific lease structure before signing the agreement. Tax treatment will vary depending upon individual circumstances.**

***** The re-sale value assumption for a cash purchase is for demonstration only. This example does not warrant the re-sale value of any Hewlett-Packard computer system.**

APPENDIX B

NET PRESENT VALUE ANALYSES

Case 1 - Pay Cash and Keep System

Year	0	1	2	3	4	5
Investment	(100000)					
Tax Effect Of Dep'n		6800#	10880	6528	3917	5666*
Net Cash Flow	(100000)	6800	10880	6528	3917	5666
PV	(100000)	6071	8673	4647	2489	3215
NPV	(74905)					

\# Year 1 Depreciation * Tax Rate ($20,000 * 34%)

* Includes the $1,958 discounted for one year. Due to a half-year convention, MACRS 5 year depreciation schedule actually spans 6 calendar years. As this analysis is based upon a 5 year cost of ownership, the tax effect of the depreciation in Year 6 has been discounted back one year and added to the tax effect of the depreciation in Year 5.

PV of $1,958 for one year = $1,749

$1,749 + $3,917 = $5,666

Case 2 - Pay Cash and Sell the System After 3 Years

Year	0	1	2	3
Investment	(100000)			
Tax Effect of Dep'n		6800	10880	6528
Salvage Value				30720#
Tax Owed on Gain				(653)*
Net Cash Flow	(100000)	6800	10880	36595
PV	(100000)	6071	8673	26048
NPV	(59208)			

\# Based on $100,000 purchase with a declining retail value of 20% per year. The retail value at the end of three years is $51,200. Assume a broker will pay 60% of the retail price for the equipment (51,200 * 60%)

* There is a capital gain of $1,920 on the sale of the equipment which is taxable.

Sale Price	30720	
Net Book Value	28800	(100000 - 20000 - 32000 - 19200)
Cap Gain	1920	
	*.34	
Tax Owed	653	

Case 3 - 36 Month Operating Lease/System Returned

Year	1	2	3
Annual A/T Payment	(21582)#	(21582)	(21582)
Net Cash Flow	(21582)	(21582)	(21582)
PV	(19270)	(17205)	(15362)
NPV	(51837)		

\# The after-tax of a year's worth of payments:
(2725 * 12) * (1 - .34)

Case 4 - 36 Month Operating Lease/System Purchased

Year	1	2	3
Annual A/T Payment	(21582)	(21582)	(21582)
Purchase Option A/T			(27142)#
Net Cash Flow	(21582)	(21582)	(48724)
PV	(19270)	(17205)	(34681)
NPV	(71156)		

The purchase option is based on a 65% equity acrual:

100000 - (2725 * 36 *.65) = 36,235

The system then is capitalized on the balance sheet and is depreciated over the next 5 years. The present value of the depreciation's tax benefit based on a purchase price of $36,235 is $9,093. The net cost, therefore, is:

36,235 - 9,093 = 27,142

Case 5 - 60 Month Capital Lease/System Purchased

Year	1	2	3	4	5
Ann A/T Paymt	(16648)	(16648)	(16648)	(16648)	(16648)
Purch Opt A/T					(7491)#
Net Cash Flow	(16648)	(16648)	(16648)	(16648)	(24139)
PV	(14864)	(13272)	(11850)	(10580)	(13697)
NPV	(64263)				

The purchase option is based on a 10% purchase option. The system then is capitalized on the balance sheet and is depreciated over the next 5 years. The present value of the depreciation's tax benefit based on a purchase price of $10,000 is $2,509. The net cost, therefore, is:

10,000 - 2,509 = 7,491

Case 6 - 60 Month Installment Loan

Year	1	2	3	4	5
Annual Paymt	(27000)	(27000)	(27000)	(27000)	(27000)
Tax Effect Of Dep'n	6800*	10880	6528	3917	5666*
Tax Sav on Int Exp	3959#	3267	2484	1597	592
Net Cash Flow	(16241)	(12853)	(17988)	(21486)	(20742)
PV	(14501)	(10246)	(12804)	(13655)	(11770)
NPV	(62979)				

* See Footnote in Case 1

\# The interest portion of the annual payment is expensed creating a tax benefit of:

Annual Interest Expense * 34%

Amortizing a $100,000 loan with a monthly payment of $2,250 and an APR of 12.5%, the first year's interest expense is $11,644.

11,644 * .34 = 3,959

TIPS ON MANAGING A 24 HOUR DATA CENTER

Gary Axisa
The NPD Group
900 West Shore Road
Port Washington, New York 11050

Managing a production department is a difficult task in and of itself. Add state-of-the-art computers, printing and data communications equipment and the task becomes even more challenging. Now consider that these resources must be managed 24 hours a day, 7 days a week, and the management task becomes dramatically complex.

The best way to assure that production work is processed efficiently and accurately (short of a "dark" data center, which will be discussed later) is to build a strong departmental organization. This means strong staffs (teams): knowledgeable, dependable, resourceful and willing to do whatever it takes to get the job done.

EXPANDING DATA CENTER COVERAGE

If you currently run a data center with only one shift, and suddenly find the need to expand to evenings and possibly weekends, your first task is to determine the type of work schedule. The two most common schedules are the standard work schedule, consisting of first, swing and grave shifts Monday through Friday, and the three-day-work-week (tdww) schedule. The tdww schedule allows for, 24 hour, seven days a week coverage without overtime. The tdww offers other advantages such as easier sick and vacation time coverage. Additionally, operators usually prefer tdww to working five days a week.

Example TDWW Schedule and Advantages:

A typical TDWW schedule is shown in the table below:

Shift	Time	M	T	W	Th	F	S		S
1	07:00-19:25 (1)	X	X	X					
2	19:00-07:25	X	X	X					
3	07:00-19:25				X	X	X	or	X (2)
4	19:00-07:25				X	X	X	or	X

Table 1, Example Schedule

(1) Twenty-five minute overlap allows adequate time for shift turnover.

(2) One operator works Thursday, Friday, Saturday and the other works Thursday, Friday, Sunday.

- Better coverage - Notice the easier coverage for vacations and overtime. If an operator from Shift 1 is out, you can get an operator from Shift 3 or Shift 4 to cover.

- More cohesive shifts - Tdww operators have a tendency to be closer with one another than operators on standard shifts. This is because people are inclined to let their guard down and be more natural after working twelve hours together.

- Added benefit - Operators perceive the tdww as a tremendous benefit, which can also make recruiting easier.

BUILDING STRONG TEAMS

The easiest way to build strong teams, as obvious as it may sound, is to hire the right person for the job. I think we recognize that some people are better suited for night hours and some for day hours. Some people are better suited for supervisory roles, while others are more inclined towards technical positions. It is important to recognize these characteristics and make the correct matches. If an incorrect match has been made, you must correct it quickly. This is crucial. As managers, we sometimes take a highly technical person and try to mold him into a supervisor. This will fail almost every time. When possible, highly technical people should be given career paths that will allow them to grow, with increased responsibilities, as opposed to promoting them to levels of incompetency in ill-suited administrative roles

Another important consideration is matching personalities, especially in a tdww. Just imagine the problems that can be created at night, when individuals who do not get along must work side-by-side for twelve straight hours.

The most important component of a strong shift is a strong leader. During the off-shifts, this leader must manage and develop staff without the physical presence of top management. It is extremely important that they lead by example; someone who possesses a very strong work ethic; someone you can fully trust. Above all, this leader must truly believe in, and support your management philosophies.

If you have even a slight concern that any of the above characteristics may be lacking in your choice, give your decision some additional serious thought. If a mistake is made, much damage can be done, damage that will take a long time to recognize and repair.

COMMUNICATION PROBLEMS

The biggest challenge is communication - not RS-232 or LANs, but communication with your night and weekend staff.

You must have constant two-way communication with your direct reports, but it is also important to maintain communication with their subordinates as well. A lack of communication can result in:

- Lack of motivation - One of the biggest deficiencies a night shift employee will experience is lack of exposure. They will feel that nobody knows and appreciates the job they are doing.

- Poor morale - Supervisors and operators feel that all the important decisions are made during the day, especially decisions that affect them and the environment they are expected to produce in. Without the opportunity to give input to decisions affecting them, they will feel understandably isolated and resentful.

- Important information falling through the cracks - This will result in production work not running or running incorrectly. If you must implement technical or procedural changes, give your off shifts as much notice as possible.

- High turnover rate - Given the high cost of recruiting and training, this problem is more than a nuisance.

- Decreased quality and productivity - Obviously, lack of motivation and poor morale will affect productivity.

SUCCESSFUL COMMUNICATION STRATEGIES

To insure constant and proper communication with your staff, effective systems must be put in place. If you rely solely on "catch as, catch can" you will experience communication breakdowns. Communications strategies which I have found successful are discussed below in two sections: long term and day-to-day.

Long-term Communication Strategies

- Team approach to problem solving - Once you have your team in place (my team is my direct reports, their team is their shift members) it will clearly behoove you to get them involved in solving problems, especially problems that directly affect them. Once a solution is agreed upon, it is clearly in their best interest to see that it gets carried out and works because they were a part of the solution. Team members should also understand that once a solution is agreed upon, it is not cast in stone. If for some reason the solution is lacking, it is their responsibility to give you immediate feedback and assist in fine tuning. To foster that feedback, you should follow up with them to insure that the problem has been solved.

- Involve the night shift in decisions that will affect them. Night shifts find it extremely frustrating when decisions are "forced down their throats." Because human nature does not like change, you should explain the reasons for the change with staff. Encourage questions at this time. When people understand the reasons for the change, they will accept it easier.

- Appoint communication officers. By appointing one person to be responsible for communicating general information to the shift, you are then only responsible for communicating to that person (usually the shift leader). By appointing one central person, it will make communicating to the entire department much easier for you.

- Hold meetings periodically. Let me state up front that I am NOT an advocate of meetings. But on the tdww schedule it is possible that shifts will not see each other, and for this reason it is important to have periodic meetings to get the entire department together. These meetings should be held about every two months. I use these meetings to address three specific objectives:

  1. As stated above, this gets the entire department together. After the meeting, pizza is served to provide a social setting in addition to the business agenda.

  2. We want to provide the background and information about changes in the department and company. Most computer rooms are in an obscure corner of the building behind locked doors, shielded from what is usually common knowledge to the rest of the company. This meeting presents an ideal opportunity to tell the staff about any relevant issues within the company while keeping them abreast of anticipated changes within the data center.

  3. The meeting is an ideal format for a training session. Usually I invite a guest speaker (someone from Systems Programming or PC or Technical Support) to discuss a technical area such as RISC, LANs, disc caching, data communications, and so forth. This type of meeting has proven to be highly successful and is certainly worth incorporating into your managerial plans.

- Magazine/Newspaper meetings. We all know that it is impossible to read all of the computer mail received in a month. Divide up the media between the shift leaders. Each leader becomes responsible to come to a scheduled meeting with at least two articles to discuss. This keeps everyone informed about what is going on within the industry while spurring interesting discussions.

- Provide feedback on productivity numbers. This puts into perspective the amount of work that is accomplished within the department and gives credit where it is due. In the form of a report, present the number of CPU hours, laser pages printed, lines printed, batch jobs and sessions for the month. Normally, this is the same report which has been circulated to other managers within the company. Given that most operators don't get a sense of accomplishment from running the same jobs day after day, this report will help them see their contributions.

- Provide formal, semi-annual performance reviews. Annual reviews just aren't frequent enough when you don't spend a lot of time with staff.

Day-to-Day Communication Strategies

- Maintain a log book. A hard covered, bound book should be kept at the console area at all times. This book is used lieu of the "gold book" and for entries of daily occurrences such as: active log files on systems, drum count on laser printers, people on shifts and what their daily responsibilities are (we rotate people into different job functions), users that are called and why, all details of any problems or symptoms relating to hardware, software or data communications and just about anything that an operator feels is noteworthy. The log book is especially useful for staff on the tdww. It brings them up to date on the events of the past four days, while they were off.

- Shift turnover sheets. These sheets are used for communication between the shift leaders to communicate any special tasks or high priorities that must be completed. The shift leaders spend most of the 25 minute overlap between shifts going over the shift turnover sheet along with the log book.

- Utilize electronic mail. This has proved to be extremely useful in communicating with staff who work shifts. If you think of something you want to tell someone who works nights - just zip off an e-mail message. I also encourage operators to use it to communicate between themselves and to communicate any ideas or problems to me. We are using HPDesk. It works magnificently.

- Work nights and weekends. I recommend that you work the off shift once in awhile. Probably just a week every six months will do the trick. It is important that the operators know that you appreciate what it is like on their shift. It helps when discussing problems they are encountering. As opposed to saying you can imagine the problem, you can say you fully understand what they mean.

- Meet with your staff personally, daily, or as often as practical. This is probably the one most important thing you can do to facilitate communication. If you work 9 to 5 and have shifts that work midnight to 8, or 7am to 7pm, you could literally go days and weeks without seeing them. It is almost impossible to motivate people and get them to reach their potential if you do not communicate with them, let alone not see them for days on end. This means you may have to come in early or stay late.

DARK DATA CENTER

Given the tremendous amount of attention the "dark data center" concept has received in our professional press, this paper would surely be lacking if I did not address this issue. It can be strongly argued that the best way to manage a 24 hour data center is through complete automation of computer operator functions. I recently completed a formal study and evaluation of how the dark data center concept might benefit our company. As a result, we are generally headed in that direction, but it is a matter of a phased implementation. If you started to plan a dark data center today, you could be as much as two years away from fully implementing it. Because it is probably a more realistic goal to implement a "dim" (partially staffed) data center, the above tips would still be extremely helpful.

My philosophy of the dark data center is to put the responsibility of the operation of the computer in to the hands of the end user. By doing this we are allowing the user to control his own destiny, which will eliminate a level of communication between the user and computer operations. This will make for more productive end users.

To begin you will want to identify and fully understand the functions that the operators of your data center perform. From this basis, you can begin to automate the respective functions. This will free computer operators from their standard tasks such as streaming jobs, printing reports and mounting tapes and packs. Their efforts can be refocused to improve the services you can provide to the users. In the initial stages of implementing the dark data center concept, we have been able to reassign operators or avoided replacement hiring of those lost through attrition.

CONCLUSION

While these tips may not eliminate all the problems associated with the complexities of the day to day managing of your data center, I'm sure you will find them useful. You can create an atmosphere in which operators on any shift feel as they are part Ωof a team. This can result in a cohesive department where everybody is working toward common goals. Additionally, you can develop an organizational environment where individuals can maximize their professional growth and opportunities.

Stress and Job Burnout: An Overview

Robert M. Gignac
Motorola Information Systems
9445 Airport Road
Brampton, Ontario (Canada)
L6S 4J3

Abstract

Does your job make you sick? Do you wake up in the morning already exhausted before you start your day? Do you feel helpless about the way things are, and powerless to change them? Are you trapped and worn down by drudgery? If these feelings are familiar to you, or someone you work with, you may be experiencing what some have called a modern epidemic - job burnout.

This paper will explore the nature of job burnout, suggest some ways to avoid it (if possible), and offer some hope if you have already succumbed. An outline of a five stage scenario (honeymoon, fuel shortage, chronic symptoms, crisis and hitting the wall) will be presented.

Introduction

If a psychologist were to sit down and look at the average DP shop, what would they see? In all probability, they would witness a high technology environment subject to rapid change, filled with achieving, independent thinkers, who are prone to low morale. In short, a very fertile ground for psychological research.

How do we define it?

The terms 'stress' and 'burnout' are often linked together but have entirely different meanings. Stress was originally introduced to the social sciences by Hans Selye in 1936 as 'non-specific response of the body to any demand', and may be more easily defined as 'any situation to which we cannot respond in an efficient manner'. Selye's definition depicted stress as comprising three distinct stages:

Alarm Reaction
Includes the initial shock phase during which resistance is lowered, and a counter-shock phase during which the bodies defenses are mobilized.

Resistance Stage
Bodily signs of alarm reaction disappear as the body makes an effort to adapt. If stressors persist or defenses

prove to be inadequate, adaptation energy will be depleted, leading to....

Exhaustion

Signs of alarm reaction will reappear, but irreversibly because the adaptive mechanisms have collapsed. This stage can end in illness, disease or possibly death.

Burnout is the harder of the two to define, as it is not necessarily a thing, but a state of being. Burnout is a very unpleasant experience, usually preceded by a long period of high stress during which the individual has had no control. Generally it occurs in individuals who pour more into their work than they get back, and at one time was thought to be limited to those in the 'helping' professions (nursing, counselling, doctors, etc). This state is characterized by emotional and psychological fatigue, feeling of helplessness, apathy towards work and life in general, depression, loss of appetite, and disorientation. Routine tasks such as taking a shower can become difficult.

What Causes Stress?

Stress can be caused by a vast number of things, including (but not limited to) the following: sensory overload, sensory underload, marital disharmony, low self esteem, feelings of inadequacy, lost status at work, unfulfilled achievement needs, role ambiguity (at home or work), environmental influences, poor nutrition, poor time management, lack of a social support network, poor physical fitness, loneliness, unrealistic expectations of others, changing social norms, and communication overload among many others not listed here. In fact, stress can be a cause unto itself. As one researcher put it, "Stress, in addition to being itself, and the result of itself, is also the cause of itself".

The problem with stress is the cumulative factor. Any of the above situations alone may not pose much concern, and in fact many of us have encountered these situations and probably never thought about them. However, add together several of the above and you could be on your way to a serious problem. The corollary to this is the fact that a little stress is a good thing. The relationship between stress and performance can be depicted best as an inverted 'U'. Individuals perform at there best when there is a moderate level of arousal. If the arousal level (stress) is low, an individual is likely to be unfocused and unmotivated, conversely the arousal level may be so high that the individual is too anxious to do well. Stress can be the

vehicle by which we mature because it forces us to extend ourselves, and it is this extension that allows us to expand our abilities. The goal of stress management would then be to limit the harmful effects of stress and enhance the positive aspects.

How can we spot someone under stress?

In any individual under stress there are certain telling signs. However, unlike many physical illnesses, the signs may not be obvious (ie: no outward expression of pain). The following is a list of possible signs you may encounter in individuals under stress.

- Declining productivity, both in the quality and quantity of their work
- Working late, or more obsessively than usual, perhaps harder than seems appropriate for the situation
- Absenteeism or chronic lateness
- Making the safe choices, not the best ones
- Constant repetition of the same subject at meetings, especially if the point is not particularly important
- Inappropriate anger, hostility or outbursts of temper
- An efficient worker suddenly becoming careless
- Tendency for team players to suddenly work alone
- Excessive worrying, especially over relatively trivial problems (or, extreme casualness and unconcern in the face of a real problem)
- Confusing or forgetting dates, places, meetings, etc

Any noticeable change from the usual patterns of performance, attitudes, or behaviors can be an indicator of someone under stress.

The Burnout Scenario

Like the human life cycle (or perhaps more familiar, the systems life cycle), job burnout goes through distinct stages. It begins when your stress response mobilizes your body so frequently in the face of chronic stress, that you deplete your energy available for adaptation. This situation can happen after six weeks on a new job or 15 years on an old one.

The process has been separated into five distinct stages, each one being more serious than the last. Let me point out however, that no two people experience job burnout in the same fashion. One might remain in the early stages for many years, then rapidly progress to a later stage. Another may move quickly through all five stages, recover, and remain

free from burnout symptoms the rest of their lives. As well, wide individual variations exist in the symptoms of burnout. It is these variations in progression and symptoms that make it easy to ignore burnout and treat only the visible symptoms rather than the actual cause. Burnout is a complex but elusive condition. No laboratory tests can reveal it. In order to learn about coping with burnout, we should examine the stages and the symptoms that tend to appear at each.

Stage One - The Honeymoon

The vast majority of workers begin their working lives with enthusiasm and a desire to work, enjoying the exhilarating stress of learning new things, meeting new people, and showing their competence. The enthusiasm comes in part from their desire to succeed. Certainly, nobody sets out to become a loser, especially after only a few weeks on the job. As the honeymoon stage continues, we often feel perfectly matched for our jobs. Any stress that we feel only gives meaning to our work and adds to the sense of fulfillment.

Two important things take place in this stage. First, despite our enthusiasm, even enjoyable stress uses up adaptation energy. The psychological bills may not come payable for months or perhaps years, but if stress is not dealt with, eventually you will pay. Second, it is in this phase that we develop our habits for coping with stress. If we are successful, the honeymoon phase can proceed without end, and for many individuals, it does. Should the strategies we develop be ineffective, the burnout process begins in earnest. Perhaps more importantly, we miss the opportunity to equip ourselves to deal with the stress that faces us not only from our jobs, but from life in general.

In summary, the honeymoon stage leaves us quite satisfied, even exhilarated. It causes us to expend valuable energy. Should we be unable to replenish that energy we move forward to the next stage.

Stage Two - Fuel Shortage

To many people, the fuel shortage stage begins with a vague sense of loss. The honeymoon is in fact over, generally with a feeling of contempt towards your job. The challenges of your work and your enthusiasm towards it have waned. People caught in stage two may be referred to as 'tightrope walkers' because they are continually on the verge of losing their balance. Just as unrelieved stress builds up to push

them off the wire, they immediately recover and regain their balance.

At some point late in this stage, many of the most frequent symptoms of job burnout make their appearance: job dissatisfaction, inefficiency at work, fatigue, sleep disturbances, and escape activities.

Job Dissatisfaction

Cyclical 'ups' and 'downs' occur in everyone's jobs, they are a fact of life. Everyone has asked at the end of a particularly tough day 'Why am I in this rat race?'. Job dissatisfaction occurs, when this feeling is prevalent day after day, week after week. In a published survey of banking employees who were asked "If you had enough money to live comfortably, would you continue to work?", over 70% stated they would continue working, but not at their present job. Although job satisfaction is an important barometer of burnout, several traps exist that we can fall into if we are not careful. First, "I need to be 100% satisfied with my work or it isn't worth it". Accept the fact that very few people ever are. In fact, try creating a balance sheet of your job's good points and bad points. If you attain 50/50 or better count yourself lucky.

The second trap, "Everyone hates their job", leads people to perceive their jobs as more stressful than necessary. In fact, an attitude of gloom and dislike for their work can lead people into a slow burnout process. The third trap, "If I enjoy my work, I won't burn out", has been discredited by much evidence to the contrary. Enjoying your work, but working 50-70 hour weeks will eventually take its toll. Unrelieved stress, enjoyable or nonnegotiable can result in burnout.

Inefficiency at Work

This is generally the first noticeable sign if you are trying to spot a worker on their way to burnout. Tasks and decisions take longer than normal. The signs are easy to spot on the faces of bored workers who grind out a workday. The criteria that create this inefficiency are: jadedness, cynicism, lowered creativity, and decision avoidance.

When people become jaded they often work harder, only to end up accomplishing less. Unable to complete their work in eight hours, they stay nine. Those nine hours eventually become ten, eleven or perhaps twelve. If you are suffering from job burnout, you will find an inverse relation between

the amount of time spent in your office and the quality of your work. Inefficiency at work also shows up in a cynical attitude. Every organization has their 'in-house cynic', the person to tell you why something will never work. These people often long for the 'good old days' when things were simpler. What is worse, is they take great delight in shooting their most powerful guns at the creative young employees who with their enthusiasm and ideas, are the ones who could possibly change the situation. One type of inefficiency, perhaps more difficult to measure, shows up in occupations that serve people. It begins when these people are tired of carrying other people's problems around with them. Workers become callous, indifferent to another agonizing story from some poor user who lost their files, and had no current backups (I'm sure you know the story...). This indifference can lead to contempt for the people that you are there to serve, which in turn makes people very unproductive.

Fatigue

To our bodies, fatigue, much like hunger or thirst, is a valuable mechanism for our own self-protection. Fatigue is nature's own way of calling to a halt physical and mental exertion, signalling a need for rest or a change in activity. After a period of vigorous exercise people feel muscle fatigue, a common experience for most of us. But early in the burnout phase, general fatigue is likely to be experienced. It occurs when we have depleted our stores of adaptation energy, and can leave us feeling more tired than if we had run the Boston marathon, when in reality, all we have done was shuffle paper for two hours. The fatigue factor associated with burnout seems to be a common problem. Surveys from as early as 1977 asked workers in all industries about their energy levels after work. Over 60% stated they felt their energy levels were low and they got tired far too quickly. Asked if they would prefer a 10% raise or the ability to work fewer hours, over 35% responded to the shorter work week.

Sleep disturbances

Some burnout victims never suffer from general fatigue and its problems. These people are those that speed along in high gear, feeling tense and perpetually keyed up. Even when they know they should sleep, they can't turn themselves off. Sleep, like fatigue, is a sign our body gives us in order to request 'refueling'. A need for more sleep can signal the onset of burnout, but the need for less sleep is a signal as well. Often we ignore changes in sleep patterns, and when

they become too troublesome, we take medication to help us and ignore the real problem.

Escape Activities

When we are caught in a situation we either don't like or don't understand, one common mechanism is often invoked - escape. If our stress safety values don't work well, we will seek escape through overeating, drinking, cigarettes, shopping sprees, or more dangerously, seeking relief through drugs (legal or otherwise). A Psychology Today survey of 23,000 readers listed the most frequent methods of coping with stress as eating, daydreaming, and shopping. Managers and executives were more likely to drink to handle tension. Foreman and skilled workers were more prone to use drugs, both on and off the job.

It is during this stage that we have the greatest opportunity to break out of the burnout syndrome. If you are here, there is still time to carry out your own personal stress relief program. The key to escaping at this point is to realize what the early warning signals are, and not to pass them off as 'part of modern life'. For those who choose to ignore the signs, we'll carry on to stage three.

Stage Three - Chronic Symptoms

The vague sense of loss encountered at stage two has now turned into 'something is happening to me'. Physical and psychological symptoms are becoming more pronounced, and it will become harder to dismiss them. Whatever symptoms affect an individual, while varying widely, will become more difficult to shake off. Most common to this phase are: chronic exhaustion, physical illness, anger and depression.

When tiredness turns into chronic exhaustion, the sequence of tiredness-rest-recovery has become out of sync. Not only do people feel tired and run down, everything becomes slower, interest in hobbies and friends wane, and irritability becomes the order of the day. Little difference is perceived between the tiredness at the end of the day and the tiredness that remains in the morning when you awake. Attention spans shorten, and things that were once engrossing no longer hold your interest.

Physical illness generally follows as the burnout phase deepens. The connection between work stress and physical illness is not complete, but the facts are hard to ignore. Backaches and headaches are two of the most frequently reported symptoms of burnout. Our gastrointestinal systems

are very prone to stress-related problems. Symptoms can range from difficulty swallowing, recurrent heartburn, diarrhea, colitis or constipation. In time, unchecked work stress can lead to ulcers, perhaps the most widespread occupational disease reported today.

As the third stage progresses, people often find themselves with a perpetual feeling of anger. Almost anything can cause them to erupt, and people begin referring to them as someone with a 'short fuse'. The change is most prevalent in someone who in the past has been easygoing, calm and accepting of ideas or criticisms. As burnout progresses, the general anger often becomes more focused. In place of people in general, computer users, or just plain management, anger will center on one or two individuals.

Others avoid the problems with anger, only to end up suffering from depression. This depression can have numerous causes, and stress is only one of them. Unfortunately, depression is a common response when some aspect of work becomes intolerable for a person. They can't quit, can't fight against it, and it just won't go away. The harder these people try to remove themselves from the situation, or ignore the situation, the worse it gets.

Stage Four - Crisis

For a vast majority of us, we will never progress to stage three type behavior, and we should count this as a blessing. As burnout progresses into stage four - the symptoms only get worse and the time to recover from them lengthens. During this stage, four significant changes occur: physical symptoms become critical, obsession with frustration begins, pessimism permeates thinking, and an escape mentality develops.

What was once regular heartburn becomes a bleeding peptic ulcer. The occasional headaches border on migraines, high blood pressure is common, insomnia often develops. Creative people watch their sources of ideas dry up, and job discontent becomes disillusionment.

Obsession turns people into moody, withdrawn individuals. Specific topics dominate all conversations with this individual. Their jobs, and sources of stress refuse to leave their thoughts, they talk of nothing else, they may even have nightmares about the problems. Those around them feel powerless to help, and for the most part, they are. In time, the obsession with the problems will lead to self-doubt and pessimism. Competent decision makers will hedge their bets

waiting for more information. Second guessing becomes second nature, and the poor decisions that are made only add fuel to the fire.

During this stage, discontent becomes so intense, and those caught in it search for an escape hatch. Surveys of people caught in burnout situations often comment, "I just want out, out of my job, my family, my whole way of life".

A clear progression can be seen in the first four stages of burnout. Satisfaction with work and exhilaration mark the honeymoon stage. Initial stirring of dissatisfaction begin during the fuel shortage stage. If our safety valves don't work, symptoms may begin to be chronic. Frustrations and discontent usually focus on some aspect of their job or career. In the crisis stage, self-doubt and despair blanket perception. Often wishing for a convenient escape hatch, the individual can feel like a trapped animal.

Stage Five - Hitting the Wall

People who are familiar with distance running, especially marathons, will be able to identify with stage five. It is said that a marathon really starts at the twenty mile mark, the last six are the hardest. 'Hitting the wall' is an experience so devastating it can take someone completely out of the race, by paralyzing the body to the degree that you lose all physical control over it. In the contest with stress, people who have hit the wall find themselves unable to work, and perhaps may not be able to for years. For the unfortunate few who reach this stage, burnout has become so entwined with other problems such as alcohol, drugs, heart disease and psychological disorder, it cannot be easily untangled. Recovery from this stage will elude some, others may win the battle, but it will require time and extreme patience and understanding from those close to them.

It is not inevitable that we progress through all the stages of burnout. We must accept the fact that stress is a situation that must be dealt with, and by learning to cope, preventing the process, we can recover should any damage be inflicted. Knowing the stages in the model is only one of the first steps in dealing with burnout.

How can we deal with Stress and Burnout?

A quick look through the psychology section of your local bookstore will bring to your attention a variety of solutions that appears endless. Jogging, diets, meditation, yoga and biofeedback are only a few of the possible alternatives to

alleviate stress. In place of buying every available book on the market (an expensive proposition at that...), I have compiled a list of 30 suggestions for dealing with stress. The list is in no particular order, and not every one will apply to your situation, but as a group I feel they offer some valuable suggestions.

1) Realistically appraise your abilities. People often overestimate their abilities and expect too much from themselves and those around them. Strive for your highest available goal, but don't resist it in vain.

2) Laugh a lot. Research shows that when people laugh, it's often therapeutic. Work on developing your sense of humor and learn to laugh at yourself.

3) Learn to feel comfortable talking about your problems. A close confidant is an invaluable resource, but avoid dumping on the same person all the time. One ground rule is that you must be available for that person in their time of need as well.

4) When stuck in traffic or waiting in line use the time to be by yourself rather than fuming about the frustration. Visualize a more peaceful, tranquil situation or process some ideas for dealing with a problem.

5) Instead of an after work cocktail, try ice water followed by a 10 minute quiet time, relaxing with your eyes closed. Remember the peace there is in silence.

6) When feeling stressed, inhale deeply through your nose, hold, slowly exhale through your mouth. Repeat 5 to 10 times. Oxygen relaxes the body.

7) Accept the things that are out of your control - an overbearing boss or subordinate is unlikely to change, regardless of what you do.

8) Get physical. Brisk walking, jogging, cycling, any type of exercise.

9) Take a daily music break - with your eyes closed sit back and listen to 10 minutes of your favorite music.

10) Stressful events are in the eyes of the beholder and subject to the possibility of misperception. Check your assumptions and perceptions to insure they are correct.

11) Don't procrastinate by doing low priority tasks because they're easy and fun - thereby neglecting a high priority task until it becomes a crisis.

12) Deal with negative thoughts by strength. Positive thinking is better than negative. Be an optimist.

13) Avoid business lunches if you can. Lunch time should be a psychological break, a time to balance out the morning and afternoon.

14) Avoid chronic hurriedness. High stress personalities try to do too many things too fast. Tell yourself that no enterprise ever failed because it was executed too slowly, too well. Cultivate good judgement and decision, not speed.

15) Take note of the things that chronically bother you and work to avoid them if possible. Work on developing an awareness to the stresses in your life.

16) If you've been working 60 hour weeks, cut back. Research has shown that productivity drops among people with high stress, but peaks under moderate stress.

17) Work at building supportive relationships. Remember friendship takes time. Those with supportive relationships suffer less consequences under stress than the socially isolated.

18) Give in when you have little to gain. Save your energy for important things.

19) When you're uptight, tense and untense muscle groups. Start with arms, then face and neck, shoulders, abdomen, and finally the legs.

20) Work on developing the capacity to recognize when you are under stress. If you don't recognize the stress you are under, you'll never deal with it well.

21) If you are aware a stressful event is approaching, visualize yourself doing it well and in a relaxed manner.

22) When you arrive home from work and need to unwind, explain to your spouse and children that you need that half hour to yourself before grappling with family problems. If explained, family resentment will disappear.

23) Studies show that brisk walking (about 4 miles per hour) is as relaxing to the body as chemical tranquilizers.

24) Try to make the workplace as comfortable as possible by adding pictures, plants and ornaments.

25) Take a warm shower or bath to soothe tense muscles and provide a few moments of 'sensory deprivation' - also known as privacy.

26) Learn about the sources of stress. Sometimes, it is hard to accept the fact that stress comes from our own inability to handle life's disappointments.

27) Learn about others. Be aware of the people you deal with from day to day who can cause you much grief if you do not communicate well with them.

28) Realize what damage stress can do. Stress if not controlled can have very damaging effects physically, psychologically, emotionally and mentally.

29) Learn to eat nutritious food. Unfortunately, it is easy to develop a habit of eating those things that satisfy our palate rather than those that satisfy the body.

30) Allow for plenty of rest for mind and body. There is no substitute for rest. Also, those around you may find that rest does wonders for your disposition.

What's happening at Motorola?

Motorola Information Systems in it's quest to provide an excellent workplace for it's employees has implemented several programs. February 1, 1988 saw the opening of the Motorola fitness center. The running of the fitness center is overseen by an outside fitness consultant who is on site two days a week. There are five employees who have become certified fitness instructors, and aerobic classes are held three days a week, during lunch hours and again after work. At present, there are over 170 members of the fitness club, about 25% of the employees at Motorola's Brampton facility.

In addition to the fitness center, several seminars have been held to provide information on stress and stress management, and attendance has exceeded expectation. Upcoming seminars include Yoga and Relaxation, and November 1989 has been unofficially designated as 'de-stress month', with several information seminars planned.

Summary

As this paper was meant only to give a general overview of stress and job burnout, many questions may remain unanswered when you are finished reading. In order to answer these some of these questions, I have provided an outside reading list on the last page for those who desire more information. My hope is that you come away with some new insight into one of today's major problems in the workplace. Stress control can be primarily a matter of self control and everyone owes it to themselves to take control of stress before it takes control of you.

Suggested Additional Reading

Adams, John — Understanding and Managing Stress
San Diego, University Associates,
1980

Albercht, Karl — Stress and the Manager
Englewood Cliffs, N.J., Prentice-Hall
1979

Goldberg, Phillip — Executive Health
New York, McGraw-Hill,
1978

Greenberg, Jerrold — Comprehensive Stress Management
U.S.A., Wm. C. Brown,
1983

Hanson, Peter — The Joy of Stress
Toronto, Hanson Stress Mgmt. Org.
1985

Meichenbaum, Donald — Coping with Stress
Toronto, John Wiley & Sons Ltd.
1983

Seyle, Hans M.D. — The Stress of Life
New York, McGraw-Hill
1976

Simmons, E. R. — Stress Control
Proceedings: HP3000 IUG Anaheim
1984

Wills, James — Coping with Stress: A guide to living
U.S.A., John Wiley & Sons Inc.
1982

Witkin-Lanoil, Georgia — The Female Stress Syndrome
New York, Newmarket Press
1984

Growing A Programmer

Eric S. Messelt
Microwave Applications Group
3030 Industrial Parkway
Santa Maria, Ca. 93455
(805) 928-5711

Growing your own vegetables is the stupidest thing you can do. Or, growing your own vegetables is the smartest thing you can do. The issue of planting a garden is one of those things where "it all depends...". In the same way, committing to "Growing your own" programmers can be the stupidest or the smartest thing you can do. Let's see what it all depends on -

Growing your own vegetables in your back yard is a crazy idea. There is a lot of work involved. You hardly ever save money and sometimes your work can get eaten up by some capricious bugs or bad weather. However. There is no satisfaction like growing your own food. You know exactly what you're eating and the quality is almost always better than "store bought". It takes time, there is frustration, but the end product is worth the wait. I'd even say that not only is the product more enjoyable, but the process is also more enjoyable. Being in the garden, pulling the weeds, fertilizing, watering and then comes the harvest. You have a sense of accomplishment knowing that through your own efforts you have nurtured something wonderful into being. And then you get to eat some really great salad.

"Growing A Programmer" is about taking the seeds of raw talent and nurturing them into a programmer. This is usually called "In-House Training". I don't like that phrase. For me that conjures up images of a VCR, a monitor and a shelf full of Deltak videos on various programming topics. I also think of impersonal programmed learning workbooks. Then I imagine some young worker sitting alone at his table bored to death trying to impress his supervisor by mowing through the material as fast as possible. That kind of "training" doesn't create the kind of programmer I want working on my staff! I want someone who is interested in learning, motivated to do a good job the right way. I want someone who will be interested in coming to work each morning because their job is attractive. And I believe that kind of training can really be done.

My assumptions is this: buying talent off the street may be more cost effective in the short run. But. The experienced people you hire are not all they're cracked up to be. They bring bad habits with them. They may find it easier to spend a shorter time with your organization. They may not be as talented as their resume' and job interviews indicated. They still require orientation to your installation and your specific applications. Indeed, the difference between "buy and make" your programmers is much like that of buying a software package or writing one

yourself. The former takes money and a willingness to accommodate the package. The latter option takes time but the package fits you like a glove. In my experience, most installations have a mix of both "canned" and "home brewed" software. Do you have a mix of Professional and Home Grown programmers? Do you know how to develop your own stable of talent? Do you know what it takes to "Grow your own?"

So this is about how you do it. How can you find the right raw talent? How do you do the training? What are the materials to use? First I'll discuss benefits of Home Grown talent over the alternative. Then we'll look at how to get the right trainee in the right situation. Finally, I summarize some of the training material that I've used and how to successfully use it.

Eating "Garden Fresh"

Growing your own programmer has the same pitfalls and pleasures as growing your own vegetables. But the good outweighs the bad for several reasons. First, when you are growing a programmer you are shaping and nurturing a real live person, not a turnip. The "end product" is more important and more valuable. The process is more interesting since your programmer-in-training is interacting with you. You are not only passing along skills and techniques, but you find yourself interacting with your pupil. A new relationship develops that can even turn into genuine friendship. When was the last time you had a mature dialogue with a tomato?

Understand that In-house training may not be for you. My disposition is already set up for it. I tend to be a teacher. When I learn a new thing, I just can't wait to tell somebody else about it. I love to take complicated things and then explain it in simple terms to someone without my background. I am a natural for this kind of thing. You may not be that kind of person. Yet that doesn't need to stop you from getting some ideas here and perhaps giving them a good faith try.

Let's look at some of the benefits of Growing Your Own.

First, the raw talent comes cheaper. It should be less expensive to hire a non-experienced person than an expert. You probably have access to some salary surveys. What does a programmer with five years experience on an HP3000 go for these days? A lot! You can save up to $20,000 a year (depending on industry and geography plus proportional benefits and perks) in budgeted labor costs. I don't know about you, but I've never walked away from $20,000 in my life! That's a remarkable savings. Isn't it <u>worth considering</u> training your own?

You might object, "Sure I can hire someone for that much less, but how much more will I spend in hidden dollars in my time and

effort getting this person trained?" Well let's say you're making $40,000 a year (I wish that is all it would take, just "saying"). That means that you would need to spend four hours a day, every day of the next year with this person to "break even". I've never spent that kind of time in training! If you need to, you've hired the wrong person.

Another objection, "I can not spend any kind of time training anybody. I have projects where I need programming experience on board NOW. I can't take the time to train. My other staff doesn't have the time - there just is NO TIME!" Good reason. Don't do it then. Hire somebody with what you think is the right experience level. However, even the most qualified isn't going to fit right in right away. Every hire requires some orientation. That is, if you want quality results from him.

At any rate, the inexperienced person can be cheaper. I was able to hire a part time student from a local college. I started him out at just above minimum wage. He started as a part time operator. As I worked with him, he learned programming techniques on the HP3000. I increased his wage in generous yet appropriate amounts - he was still inexpensive for the work I was getting out of him. I gave him simple, then more complex programming assignments. After two years, I left the organization and he took my place! Just by my leaving, the organization saved about $10,000 a year in the difference between my old salary and his new one.

I am not advocating that growing programmers ought to be what every shop ought to do. In fact, I am not advocating anything. I am describing a path for those who wish to start down it or want another perspective on what they are already doing. I do not wish to take jobs away from highly trained and experienced programmers. If you were to rely only on in-house developed talent, your shop would become ingrown in its techniques. It is a refreshing change of perspective to bring someone in who has had different experiences and can lend new blood to your shop.

Let's consider the quality of the end product. I have been constantly frustrated at having to tell an experience programmer how I want things done in my shop. It's not that I'm closed to new ideas. However, it is frustrating when the new man, guided by his own habits, does something differently. That is understandable since he doesn't know how we do it here. There is a constant reorientation for the new person even though he is a competent, qualified and experienced programmer.

That describes the normal situations. I've experienced far worse. I was in a shop where I remember one guy who came in with a wholely different set of programming standards and style. He was experienced and was brought in to be productive ASAP. He was very willing to share his ideas and different ways of doing things

with everyone who would listen. Some of the other programmers even suggested, in an open meeting, that this new guy - who hadn't been on board more than two months - be given authority to redo the shop's standards. It was very embarrassing for management since they did get caught flat-footed without a comprehensive set of standards. The situation was difficult. This man was brought in to be a productive programmer right away, and now everyone wanted him doing "administrative" tasks. At the last moment the man in question had the good grace to resist the popular movement. It turned out that he left after only six months for a more challenging position. Not only did this experienced programmer stir up controversy but then he abandoned us for greener pastures.

I also had the misfortune to attempt to work with a very bright, very motivated, very hard working new programmer who had been hired for me. He just couldn't get into the swing of things. What we were doing in our shop was just too different from what this person was used to. He didn't last two weeks. Of course, then we had to work on the hiring process all over. More wasted time.

When you train your own talent, you have exact control over the quality of the programmer. If you find problems, the person is more malleable towards change since they are in a learning mode anyway. You can encourage them in their strengths. They will begin to adopt your ways of thinking about problems and getting solutions. Think about all the crazy ideas that get stuck in programmer's minds. Here's a chance to put some of your own crazy ideas into this person! The point is: the programmer-in-training will be able to fit right in with your way of doing things.

The next benefit is a more loyal and motivated employee. I would like you to consider the following:

> "I use the rule of 50 percent. Try to find somebody with a record of success and an appetite for the job. If he looks like 50 percent of what you need, give him the job. In six months he'll have grown the other 50 percent and everybody will be satisfied."
>
> - Robert Townsend

Note what's being said here. Your trainee will be learning, growing, challenged. He will be a motivated, interested employee. He will not be bored, and if you manage him well, he shouldn't be swamped. You'll start his training from where he is and take him to a place he's never been to. If he's any kind of human being, he'll feel some sort of gratitude to you and your organization for providing him with that kind of experience. Gratitude translates into loyalty and loyalty translates into longer tenure with the organization.

"How to do it wrong: go outside and get some expensive guy who looks like 110 percent of what you want and a year later, after having raised salaries all around him, you'll still be teaching him the business. The people around him will be frustrated and ineffective."

Your new man, the guy who's going to come in and get your backlog whittled down to nothing, is sure to be a disappointment. In addition, he's going to be bored. Who wants to work for any length of time at something he's over-qualified to do? Soon his eyes will be wandering over the want-ads for the chance to do something different. His discontent will make him difficult to work with or be around. He'll start looking for make-work to keep himself occupied. Even worse for moral, he'll be in attendance eight hours, but only work six. The only thing you've done is train this guy in your industry, which makes him exceptionally valuable to your competitors!

There is the making of yet another one year wonder. Another trip to the newspaper to place the ad. More time of yours interviewing, agonizing over a hire decision, negotiating salary, et cetera, ad nauseam. Just how much of your time did you save to hire the experienced guy and how much more did it cost you to save it? Why do you do this to yourself?

The last benefit I want to mention is that your shop will eventually run better. It will be a more professional enterprise. How is this so? To teach is to learn. As you get yourself together enough to train your neophyte, you may find that your shop needs to get itself together as well. A neophyte needs documentation, standards, consistent interfaces. If your shop doesn't have these, be grateful that the new kid on the block forces you to do things that you should have done long ago. What was the message of the recent books about Excellence? If it's not done excellently, it won't be profitable or fun. If your shop is a half effort of mediocrity and jumbled product, why are you still there? If your not doing this for fun or profit, what are you doing here?

Okay, let's get on with the nuts and bolts.

Picking the Right Seed

This is the most difficult paragraph of this paper. If you can swallow this, you're ready to Grow a Programmer. It is called the Apprenticeship Principle: <u>Hire your trainee with the commitment that they must stay with you for at least two years.</u> At the end of that time, both of you will be able to reassess your mutual commitment. The programmer will be free to seek another opportunity or to stay. During the two year period, you will invest in their lives and teach them the Art of Programming. The first year will be primarily you giving to them, the second year will be them paying you back. You may want to stress that at the end of the

"Apprenticeship" that their salary will make an appropriate jump. You do want to encourage them to stay.

The key is that you are actively planning for employee turnover. You know you'll be losing people - it happens all the time. Why are you always caught short or surprised? Here is a way to insure that you have talent waiting in the wings - ready to step up to new challenges. Not only that, but they are a higher quality of product - you trained them yourself. Expect 50% of your trainees leaving after the two year time, hope for less. This situation forces you do deal actively, positively and constructively with short employee turnovers. Also, if you're going to lose people, wouldn't you rather lose relatively low-paid trainees than very expensive "hired guns?"

Now let's not fool ourselves, this is a dangerous proposition. First, there are laws regarding formal Apprenticeships that you probably don't want any part of. This proposal to the candidate is not an agreement - only an understanding. It is informal and will only work if both parties realize that they are mutually benefited. This is a special relationship that you are forging - you are responsible for making it work. If you aren't up to it for any reason, I would advise against it. This is not for the faint hearted, the half hearted or those intimidated by "the current labor law environment".

It is also a dangerous proposition: this business of expecting people to only be there for two years. Blasphemy! "I don't even want to suggest that any of my employees will be leaving - I don't want to give them any ideas!" They don't need your help, they are all capable of deciding for themselves if or when they will leave. Why keep your head in the sand? Why not bring it out in the open and deal with it like adults? Be a good manager, communicate with your people about the inevitable. Few programmers will stay with your organization until they retire. Talk to them about their eventual departure. Anyway, the point is there is nothing wrong with expecting the inevitable - your experienced programmers will leave someday. It is even more inevitable that your trainees will leave. All I'm suggesting is that you put your cards on the table and ask for a solid commitment for two years.

Let me tell you that it has worked. I know it; I was once an apprentice. I was a recent college graduate with some computer operations experience and was interested in breaking into the computer industry. A programming position came up with the Data Processing department at my alma mater. I interviewed and the proposition was made to me: stay with us for two years, be severely underpaid. Learn from us for the first year, give us back what you learned in the second. At the end of that time, you're free to leave if you'd like - though we'd prefer it if you'd stay. I took them up on it. Most of what I know about doing Data Processing was

learned in that time. Within two years after leaving that organization, I was a Data Processing Manager at my own shop.

Indeed, much of this material was taken from my own "apprenticeship" notes of the time. And now would be a good time to acknowledge my sources. For the outline and my training, I want to thank Robert Catherman and Jan MacDougall. With their patience and experienced, they created a well run shop and a couple of well trained programmers. I moved on and have been Data Processing Manager at three installations. The other programmer who trained with me has been the Director of Computer Services for that organization for many years now. Bob and Jan made those track records possible.

With my first shop, I practiced what I'd been taught. I hired a part time student as an operator and taught him what I knew. I mentioned him above - he was the one who became Data Processing Manager after I left. This technique worked <u>with</u> me and it worked <u>for</u> me. I was trained by it and I used it to train others. This has been such a powerful idea for me that in every supervisory relationship, I have tried to put some part of this into practice. I can not say that this will work for you.

I have not always succeeded. I was once severely misunderstood and, though the relationship was straightened out later, it cost me dearly at the time. Other problems I faced were when hires were made for me and I had no input. On the whole, however, I have been able to make it work.

It should be clear that the success of the programming hire depends on the person being considered. What is the candidate profile for a effective programming trainee? To answer that, you need to give some hard thought as to what kind of programmer you are trying to create. I am not trying to create a "techno-jockey". If I wanted that, I'd hire a Computer Science major or some experienced "Big Gun". No, what I am after are human beings who are able to talk to other human beings and help them do their jobs better. Again, the technical skills can be taught - it's the other intangibles that are worth the gold. Here are some thoughts:

First, look for little experience. Don't get too carried away by this. There seems to be some obvious things a person needs to be productive - but the best candidate is <u>not</u> necessarily a computer science type. What you want is someone who has shown interest and some experience with computers, but is still plastic and malleable. One of my hires was a psychology major who'd taken some computer classes. His computer experience was as a student using Apple personal computers. So he understood basic concepts and had done some programming. He certainly had a teachable attitude and had a success record with his limited exposure.

I also believe in personableness. In that hire decision I just mentioned, I had several candidates who could've succeeded just as well. It was that candidate's record as friendly, reliable, positive, energetic and willing to work that caused me to make him the job offer. I especially would look for a "servant attitude". Perhaps that is a bizarre phrase to you. Let me try to explain. In most companies, Data Processing is not the organization's reason for being. Data Processing is a "service organization". That word, service, is a key attitude barometer. My department exists for only one reason in my organization and that is to serve it. Too many computer folk get their egos in the way and want to put the cart before the horse. We are servants. Our agendas are usually controlled. I would hate to hire someone who wasn't able to adopt that mind-set.

Look for someone with some native intelligence. You want someone who is quick to learn, can concentrate on the task, can retain what you've taught and put theory into practice. I've heard that you want to get someone who has a mind for Mathematics. If so, I'd prefer someone who did well in Geometry rather than Algebra. Algebra is stringing together formulas and solving them. That is not very hard to do as a programmer. The formulas are usually provided and the computer does the math work. However Geometry has all those proofs to do. Remember those? You're given a set of tools in the form of rules and a goal of a thing to prove. Then you had to construct a set of steps using the tools to get to the goal. That is more like the programming I do.

Since about half of the training to be done is reading and exercises, self-motivation is an important characteristic for the trainee. Self motivation also seems to be a significant character element for any programmer. Someone who you want to give an assignment to and then know that he's going off and doing it - without your needing to constantly check on him. Look for real evidences of a self starter.

The next attribute of a good trainee is also the same as for any employee - reliability and commitment. If you opt to use a student or recent graduate, reliability is a very real concern. You want to make sure that the training experience is a consistent and stable one. After all, you want him to learn. You should expect that his position with your organization is his second highest priority - after actual class time. Do make a point that you expect the same level of attendance consistency that you would from the rest of your staff.

There has been a repeated theme here. That is in using a part time college student or a recent graduate. They have a lot of advantages. They are usually mature enough to hold a steady part time job. They are intelligent and their minds are already in gear to learn. I also must admit that most of my experience has been

with younger folk. They do have the understandable problem of wanting to move on to bigger things after they graduate.

However that isn't your only source of talent. Look in your own organization. I keep hearing that Data Processing Managers are more interested in having a programmer who knows the industry than having certain technical skills. That's what this paper is about! Is there someone who is blocked from advancement who has an interest in computers? What a natural candidate! There are pitfalls in this situation, too. There may be resistance from others in the organization to have Freddy the mailroom clerk suddenly becoming Fred the Programmer/Analyst. You may not be able to get that person. Their boss may not want to let them go.

I had a situation like that. There was a guy in Technical Services who had risen to the top of his scale. He had a keen interest in programming and had actually done some for his department. He was rough around the edges, both technically and personally, but there was no reason to suppose that he couldn't be smoothed out. But his boss didn't want to give him up. The almost-candidate was very experienced in his dead-end job and his current boss didn't want to lose the expertise. I was advised to back off that track. We did come to another agreement. Our business was seasonal and we agreed that during the off season, the almost-candidate would report to me for special programming projects. I would train, manage and act as project leader for this pseudo-programmer. It actually worked out rather well. The programmer got his chance to program and we got our backlog chipped down.

Sowing, Nurturing, Reaping

Now we'll discuss just how to put all this together. You've decided to give this a try. You've searched through candidates and found one who has "The Right Stuff". He is now an employee and has reported to you on his first day: What now?

The key is Preparation - YOUR preparation. You must be ready and familiar with the following materials and have your plan established before your "apprentice" shows up. Your plans and materials will center around three things: The Trainee's Workbook, textbooks (only one of which you may not have), and "staff lectures". You must prepare the workbook (and that is not insignificant) and give some thought to the lectures (though you've probably already given them) and buy the textbook (widely available). Now all your ducks are in a row.

I'm going to refer you to our new employee's training worksheets. Obviously I have no space to include all our materials but this should serve to outline the procedure. This worksheet is part of a larger document called "Data Processing Instructions" or DPI. You'll notice references to "DPI 231" or some such. The DPI is our department's Standards Document. We have sections on

Management, Training, Programming and Operations. The DPI number refers to a chapter covering the departmental standards for some topic. Notice that right away you can be integrating your department's way of doing things into your trainee.

Well, let's get on with it. The first part of the training starts like any other new hire: employee orientation. Remember that this trainee is an employee like any other in your organization or in your department. Send them to personnel, introduce them to the folks in the department and walk them around the facility. This is pretty standard.

Then you get more specific. You discuss the Data Processing department's goals and organization. You show the trainee the "Department Library" (manuals, books, other references), review their job description to show them where they fit in and then give and explain their copy of the Data Processing Instructions (which has all their training plans in it). We discuss Management By Objectives though that is optional. You then talk about what the goals are for their professional development and what resources you have to accomplish those goals.

Understand a couple of things here. First, the trainee will be reporting to you, though you will be assigning them to others in the department (notably the Operations Supervisor, more on that later). This will continue until some point in their training when they will begin to perform simple assignments. Then I would turn them over to the Programming Supervisor. Second, up to this point, there is no difference in orientation between your neophyte trainee and what you'd be doing with a newly hired and experienced programmer. The experienced programmer goes on to another orientation that is higher level - centering on the applications and how the systems are held together.

But the trainee begins their introduction to the HP3000 now.

The first assignment is the reading of a memo given to all first time users of the HP3000 system at our organization. It outlines how to log on, what a user and account name is, and some rules to live by. The System Manager assigns a logon to the trainee in an account reserved for the Data Processing department and lets them know that this is where they can play with the system and perform all their assignments later. After the memo is read, they practice logging on to the system, logging off, doing some MPE commands that have been outlined on the memo and then I introduce them to the GAMES account.

Computer games are an invaluable teaching aid for the first time user. It shows them how the computer reacts to their commands, teaches the little protocols that experienced users take for granted - and does all this in a less threatening way. You should make sure that the games available are somewhat idiot proof since you don't

want the program aborting with an intimidating tombstone due to some inadvertent error.

After we've got that taken care of, I make a big deal (as I've done with the orientation section before) of checking off that assignment on the training worksheet and going on to the next. Kind of like putting a little gold star next to their name.

The next assignment is right out of HP's manual, HP3000 Guide for the New User (abbreviated later as "HP3000 GftNU"). The manual has its own set of logon, logoff and MPE exercises. Plain old teaching reinforcement since the trainee has already done this with the System Manager. In the early stages of learning new things, the basics should be repeated in different formats. I try to use this principle throughout: try to arrange new concepts so the full and complete explanations come after some brief introduction during a previous exercise.

The emphasis shifts briefly to a more technical treatment. The trainee is to read through the manual for the HP2392 terminal. The trainee can skim parts that get too deep, but they must skim the whole manual. The idea here is to first give the trainee a taste of the technical nature of their job. Also, this is an example to show that the tools that we are giving the trainee are deeper and more complicated than what they first experienced. The previous exercises used a terminal, but the trainee now learns that the terminal is capable of doing many kinds of things. This is also an introduction to data communication.

Now that they have a grasp on logging on, you can move them along to the larger picture of User/Group/Accounts. This is where you might sit them down and have a "staff lecture". This is no more than you explaining the concepts to the trainee in a one-to-one setting. After you draw your circles and arrows, give them the next chapter of HP3000 GftNU which is learning how to use EDITOR.

We use Chapter Two of the HP3000 GftNU to get the trainee started. Then we have them wade through parts of the EDITOR manual. Since they've already had practical exposure through the easy to master exercise in HP3000 GftNU, diving into the manual is not nearly as traumatic. Encourage them to try out what they're learning in the manual on their own logon. All during this time, you must be available to answer questions, clear up ambiguities, and provide discussion.

They will have plenty of time to get the use of EDITOR straighten out. If they don't get files/users/group/accounts down pat early, they will never succeed. So the next assignment is to continue with Chapter Three of HP3000 GftNU which goes over users, groups, and accounts again. This time we separated the reiteration with the exercises on EDITOR. This gives the first exposure to user/group/account some time to "sink in". By the time you get back to it, the trainee will have lived with the idea for a couple of

days and will be much more aware of the topic the second time around.

In that same spirit, Chapter Four of HP3000 GftNU will repeat the concept of files that was introduced with EDITOR. That's their next assignment.

Now let me stop here for a moment and let you know that your trainee has been busier than you thought. While you've been encouraging your little guy along, he's also been hustled about by your Operations Supervisor. Concurrent to the Programmer Technical training, the trainee is also receiving practical Operations training.

A couple of reasons for this. First, do the Operations stuff concurrently since you can't be immediately available to answer each question that might come up. So what does the trainee do while you're out of the office - pick up the newspaper? No! He marches right over to your Operations folk and picks up with where he last left off with them.

The second reason: every programmer ought to be Operations literate. The content of the Operations training is fairly straight forward: operation of equipment, batch job flow, major applications, etc. But the major thrust of the trainee's time in the computer room is practical. Your trainee should be doing the system backups, be running special jobs, mounting special forms on the printer, cleaning the printer and tape drive, distributing reports, and so forth. You do not want to create a programmer who is uncomfortable in a computer room.

You are, to the great appreciation of your Operations people, imparting this rookie programmer with both the importance and the practice of operations. Let's reason this out. After all, your customers come first. Your customers usually don't care much about how bright or clever your programmers are. Most of the time they want to see their reports in a timely manner - it is assumed that they are correct -and the computer to be available to them when they want it. That is the responsibility of Operations. Most users today don't really think programming is all that difficult anyway. Your customers deal with Operations people on a day to day basis. They see you programmers only when there's trouble - or about to be trouble. Programming is research, Operations is production. Customers care first about production.

It is unfortunate but usually true that D.P. Managers come up through the ranks via programming. We also seem to believe that programming is what runs a D.P. department since that is what we've spent our careers doing. That is simply not true. I hope that kind of arrogance is something that you do not want passed on to your trainee. I want my programmers to have a fine appreciation for the Operators and to realize that we are all a team. Operations

training for the young programmer ought to be considered an integrated part of learning the art of programming.

Despite the crucial nature of Operations, learning the science of programming takes a lot of time since there is much more to learn before you can be useful. We now move away from the new user's guide and training takes on a "directed reading" color.

Instruct your trainee to read through the MPE Commands Manual. He should read everything, but pay particular attention to the commands on the list. They are not in any conceptual order - I leave that to you, if you wish. In any case, there will bound to be many questions so be prepared to spend extra time with the trainee.

Next are the systems utilities. It would be a good idea if you had a short lecture to give a broad overview of what the utilities are used for and what they do. Then turn the trainee lose on the MPE System Utilities Manual for those listed utilities. Again, it is not important that the trainee get every single detail of operation down (I hope you never have to deal with a FCOPY of a BCD file from an IBM system). The ultimate goal is to let the trainee know where such information can be found when needed.

Now we move into more intensive training. The IMAGE DBMS is really not something easily learned straight from the manuals. We have put together some introductory material, using the manuals, and exercises. By the end of this home-made series on IMAGE and QUERY, the trainee actually knows quite a bit about how databases work. They create, access, enter data, modify, and otherwise manipulate their own practice database. They also learn QUERY FIND and REPORT functions very well.

After the strong training in IMAGE/QUERY, we return to more mundane thing like SORT/MERGE.

About now, you ought to have another lecture on your installation's major applications. Your trainee is now ripe to find out how all this information he's learning fits into what goes on at your shop. Discuss any purchased software systems you have - what they do and how they are supported. Put special attention on the in-house systems you've developed. I try to include some history of the in-house systems. Why were they needed? Why did we do them here rather than purchase from a third party? What were some interesting enhancements that we have had to introduce to keep the system useful? You get the idea.

"Well this is all wonderful," you might impatiently be thinking, "But when do we start teaching this kid how to program?!" Of course we'll start right now. Now that the trainee has got some great orientation to the HP3000 system and your organization, the motivation for learning COBOL will be even stronger.

This will involve the textbook I mentioned earlier. We have used McCracken's A Simplified Guide to Structured COBOL Programming. It starts with simple concepts, does a fine job of teaching COBOL syntax, and stresses real structured, top-down programming technique. The trainee uses this textbook with the HP3000 COBOL manual, the department's programming standards (DPI), and other optional materials. But the other big contributor to learning COBOL is a learning plan (which is also in the DPI). This gives reading assignments, exercises, and instruction on how to integrate the other materials like the COBOL manual and the programming standards. This stage will certainly take a couple of weeks - in conjunction with other things that are also going on.

What I've found is that even before the trainee gets to this point, they have already learned enough EDITOR and MPE to be updating jobstreams and other tasks under strict supervision. By the time they get through two thirds of the COBOL learning plan, they are getting supervised programming assignments. They eventually get into actual programming such that they never do finish the programming training. Even so, you'll notice that the trainee's outline doesn't stop with COBOL. There is more on VPLUS, TRANSACT (which is what we use for in-house systems), Personal Computers, SEGMENTER, KSAM and Intrinsics. I don't get too worried that they don't get around to these topics as part of their "formal training". The trainee will eventually need to learn about SEGMENTER for the first time. When the time comes, he'll pick up the manual and get lost, come to someone else in the department, and get his training then. I don't get bothered if the training plan falls apart after the COBOL stuff. Once the trainee gets successfully into actual programming, he is on his way.

Now how do you get them successfully into actual programming? You give them small, short, easy programming assignments. When some simple modifications comes in, your Programming Supervisor takes over. He sits down with the trainee, discusses the application, the part the program plays in the application, what was the program previously doing (in simple terms, please!), and what it should be doing. Then he brings out a copy of the source code and shows how the proposed modification will fix the problem. Note that someone else has already done the trouble shooting. Then lead the trainee by the hand through modifying the source code, compiling, testing, installing and documenting the change. Show the trainee the process, help them through the process, and then let them do more and more of the steps on their own.

All the reading and exercises that went on before where mostly book learning. This is the stage where you really create a programmer. I would start this process even before they pick up the COBOL training. It is also not a big deal if the trainee doesn't understand the program or the application right at first. It was a full year before I understood what my first program

modification was supposed to be doing. I stumbled on it chasing another problem and remembered the modification on sight. It was a profound moment to come across my own code again. Even though my mentor had written every character out for me and my only responsibility was to use the EDITOR correctly, it was still my first piece of programming that someone had paid me to do. I felt like enshrining it in yellow highlighter.

Once you start the trainee on more and more challenging programming tasks, you'll begin to see the results of your training. Be encouraging of good work and nip bad practices in the bud. Remember, your part of the bargain is to make them into the best junior programmer they can be. Your promise to them is to make them marketable. The next challenge for yourself is to be able to afford their new market value!

If you've done your job right, you'll be nearly heartbroken if they decide to leave. But now the metaphor changes from gardening to parenting - you must let them go to make their own way. There will be some who will want to stay for a variety of reasons and you'll find yourself occasionally feeling grateful that such a talent would permit themself to stay. Oh yes, you'll uncover some real gems of talent! Talent, who by your patience and encouragement will have blossomed from dirty clods into hot rocks. Watching that transformation is more satisfying than the mere labor costs you've saved.

So there it is, the best of all worlds: reduced labor costs, improved quality, and rewarding labor. Growing a Programmer is much more fun and challenging than sticking seeds in wet ground and waiting for something to happen. It is a task worthy of a Data Processor!

Appendix

DPI 200

Microwave Applications Group

Data Processing Department

NEW EMPLOYEE INTRODUCTION AND TRAINING

EMPLOYEE ORIENTATION

.. Introduce to Data Processing staff
.. Introduction to President
.. Send to personnel for incoming processing
.. Work station with supplies
.. Organization structure of MAG
.. Organization of Data Processing
.. Job Description Review
.. Tour of building
.. Work standards (hours, dress, decorum)
.. Time card responsibilities
.. Pay day
.. Restrooms, breaks
.. Security responsibilities
.. Employee handbook
.. Vacation and time off
.. Phone network training

INTRODUCTION TO WORK ENVIRONMENT

.. MAG goals
.. Department goals and plans
.. Data Processing library
.. Data Processing Instructions (STNDRD@)
.. MBO
.. Employee development goals (preliminary)
.. Methods of professional development

PROGRAMMER TECHNICAL TRAINING

SUBJECT	REFERENCE
.. New User Orientation	*see memo "New User"
.. Introduction to the HP 3000	*"HP3000 Guide for the New User" - chapter 1.
.. Users/Groups/Accounts	*"HP3000 GftNU"
.. MPE subsystems	*Staff lecture
.. Terminal operation	*TERMINAL REFERENCE MANUAL - 2392
.. EDITOR	*"HP3000 GftNU" chapter 2. *EDIT/3000 manual
.. Files	*"HP3000 GftNU" chapter 4. *MPE FILE SYSTEMS MANUAL

.. MPE OPERATING SYSTEM
--MPE COMMANDS *MPE COMMANDS MANUAL

******** MPE COMMANDS **********	
ABORT	PREPRUN
BASIC	PURGE
BUILD	RECALL
BYE	REDO
COBOL	RELEASE
COBOLGO	RENAME
COBOLPREP	RESET
COMMENT	RESTORE
CONSOLE	RESUME
CONTINUE	RUN
DATA	SAVE
EDITOR	SECURE
EOD	SETCATALOG
EOF	SHOWCATALOG
EOJ	SHOWJOB
FCOPY	SHOWME
FILE	SHOWOUT
HELLO	SHOWTIME
HELP	STORE
JOB	STREAM
LISTF	TELL
PREP	TELLOP

.. SPOOK, LISTDIR5, LISTEQ, FREE5, * SYSTEM UTILITIES MANUAL
.. FCOPY *FCOPY MANUAL
.. IMAGE/QUERY *DPI 201 - SELF-STUDY
*IMAGE MANUAL
*QUERY MANUAL
*Various Articles on Image
.. SORT/MERGE *SORT/MERGE MANUAL
.. Major Applications
#MCBA *Staff lecture
G/L
A/P
A/R
#JOBSHOP/3000
Job Costing
Fixed Assets
#MAG
Payroll/Personnel
Purchasing/Receiving
Job specific software
.. COBOL *"A Simplified Guide To Structured COBOL Programming", MCCRACKEN
*COBOL MANUAL

.. VPLUS

*DPI 110 THRU 131--Staff taught
*USING COBOL - A GUIDE
*"Using VPLUS/V: An Introduction to Forms Design"
*VPLUS/3000 Manual
*"Programming in

.. TRANSACT
TRANSACT"
.. Personal Computer Orientation
#MS-DOS, PAM
#Wordstar/Memomaker
#Lotus 123
#PCF

*HP150, Vectra
*Manuals, tutorials

.. SEGMENTER

*Various articles on SEGMENTER
*SEGMENTER MANAUAL

.. KSAM

*KSAM MANUAL

.. Intrinsics

*INTRINSICS MANUAL

OPERATIONS TECHNICAL TRAINING
(concurrent with programmer training)

SUBJECT	REFERENCE
.. Equipment operation	*Operations Staff
.. Computer Operations experience	*Operations Staff
#The Console	
#Printer and forms	
#Tape drive and backups	
#Notable applications	
.. Console commands	*Console Operators Manual

########## CONSOLE COMMANDS #############

ABORTIO	LOG
ABORTJOB	OUTFENCE
ALTJOB	REPLY
ALTSPOOLFILE	RESUMEJOB
BREAKJOB	RESUMESPOOL
CONSOLE	=SHUTDOWN
DELETESPOOLFILE	STARTSPOOL
DOWN	STOPSPOOL
HEADOFF	STREAMS
HEADON	SUSPENDSPOOL
JOBFENCE	UP
JOBSECURITY	WARN
LIMIT	WELCOME

.. Console UDC's
.. Batch Jobs *DPI 122, 131, 134, 140

ON-THE-JOB TRAINING

SUBJECT	REFERENCE
.. Using HP3000 in programming MPE)	*DPI 132-134 (IMAGE,
.. Data Processing programming	*DPI 140-160
.. Walk through program modification request	*Staff
.. Initial programming assignments	*Staff and Users

The Key To A Successful DP Shop Is PMA

Rhonda C. Skains
Texas Municipal League
211 East 7th, Suite 1020
Austin, Texas 78701
(512)478-6601

INTRODUCTION

Anxiety, controversy, fear and unhappiness - these situations occur in most work environments at one time or another. But if they happen daily, if employees are constantly preoccupied with one or more of these conditions, then the fault may lie with you "The Manager".

The manager usually sets the tone for the work place. If a manager is frustrated and doubtful, it is likely that the employees will have this same pessimistic attitude. Employees entering the manager's office will never know how they will be greeted if the manager exhibits erratic and inconsistent behavior. Many workers have heard of departments in which the employees were "shaking in their boots" with fear of losing their jobs or angering the boss. They worry so much about doing something wrong that they don't have time to do it right.

The principal causes of unhappy workers are:

1. **Failure to give credit for suggestions or work**
2. **Failure to correct grievances**
3. **Failure to encourage**
4. **Criticizing employees in front of others**
5. **Failure to ask employees their opinions**
6. **Failure to inform employees of their progress**
7. **Favoritism**

I can be of no real help to another unless I see that the two of us are in this together, that all of our differences are superficial and meaningless, and that only the countless ways we are alike has any importance at all.

Hugh Prather

If, on the other hand, the manager exhibits rational behavior, an open manner, and is in touch with the employees, it will be communicated to people around the manager. A person who is optimistic, enthusiastic, and carries hope is a person who has a purpose in life, who has self-confidence, self-determination, and self-worth. These people use (PMA) a Positive Mental Attitude, and they reject (NMA) Negative Mental Attitude.

PMA and NMA

Proverb: "That which you fear or expect most will likely come to pass. The body manifests what the mind harbors."

PMA: "I did a good job today and I'll do better tomorrow!"

NMA: "With my luck, I was bound to fail."

When we choose only PMA as the director of our mind we experience definition of purpose. People with PMA expect the best, as a way of life, they prepare themselves physically as well as mentally to achieve a condition of harmony or equilibrium.

Characteristics

PMA	NMA
harmony/peace	**stress**
togetherness	**separation/isolation**
happiness	**depression**
love	**hate**
high self-esteem	**pain**
good health	**illness**
prosperity	**poverty**
success	**failure**
understanding	**false perceptions**
long fulfilled life	**death**

"There is a better way of going through life rather than being pulled through it kicking and screaming." Hugh Prather

Peace and happiness can be accomplished by our willingness to change our goals and by changing our belief systems. This means changing the way we think. Changing the way we think of others. When we desire something from another person or the world and we are not successful we experience NMA. None of us seriously want to experience this pain. But since we hold on to those old beliefs we do not allow ourselves to experience PMA. To experience the world and others differently, we must change the way we think, let past experiences vanish, concentrate on today, and disband any negative thoughts.

Example: Think of the mind as a motion picture reel. This reel contains past experiences. These experiences are superimposed not only on each other but on the lens through which we experience the present. Consequently, we are never really seeing or hearing it as it is; we are just seeing fragments of the present through the tons of distorted old memories that we layer over it. If we are willing, we can wipe away and let go of those negative thoughts.

Self-Appraisal

1. Am I generally optimistic about all aspects of my life?

2. Do I expect the best for myself?

3. When I am discouraged, am I indulging in a form of self-pity?

4. Do I look at problems as potential opportunities?

5. Do I praise or criticize more often?

What can we do to achieve PMA?

PMA our single goal in life

We must first be willing to make PMA our single goal in life. We all juggle many goals and this only creates more conflict in our lives. We can achieve consistency in keeping this single goal in mind by reminding ourselves of the singleness of purpose we would have if we suddenly found ourselves drowning in the ocean. We would, in that situation, put all of our attention into the single goal of staying alive by remaining afloat and breathing for survival.

We must remember that other people do not have to change for us to experience PMA.

Steps to Develop PMA

1. **PMA is our single goal.** Positive Mental Attitude.

2. **Wake up happy.** Optimism and pessimism are learned behavioral attitudes. The best way to develop PMA is to start early in life or early each day. Wake up to music. Sing. Listen to motivational tapes in your car going to work. Read an inspirational message.

3. **Use positive talk morning and evening.** Talk to yourself. Say "This is going to be a great day." "Something good is going to happen to me today." "I did a good job today and I'll do better tomorrow. Replace can't with can. Replace the word try with will. "I will use PMA!"

4. **Look for good in all relationships.** Accentuate the blessings or lessons in even the most trying circumstances. Each and every person has good qualities. Look and you will find them.

5. **Look at problems as opportunities.** Make a list of your most pressing problems, those that block your professional and personal fulfillment. Write a one sentence definition. Next write another definition but this time view this problem as a opportunity or exercise to challenge your creativity. View the solution as if you were advising one of your best friends.

6. **Learn to be relaxed and friendly no matter how much tension you're under.** Instead of participating in group griping, single out someone or something to praise. When tension or anxiety enter the room, that's your signal to lower the tone and pitch of your voice; to breathe slowly and deeply; to sit back and relax your muscles; and to respond calmly to problems with suggested solutions.

7. **Focus on the rewards or benefits opportunities may bring you.** You almost always get what you think of most. Focus all your attention and energy on the achievement. Forget about failure. Make a list of your desires and goals and write down what the benefit will be to you once you have achieved it. Look at this list daily. Don't be afraid to experience new opportunities or things you have often dreamed of doing. Talk to people you know have done this. Find out as much as you can about the subject.

8. **Think positively about your health and others health.** Cure what's curable. Prevent what's preventable. Enjoy life. Tell yourself "I feel energetic." "I'm feeling better now." "I'm reaching my best weight." "I feel young and vital." Try not to pay to much attention to minor health irritations because if you do it places a value or pay-off to being sick. This can develop into a habitual host of allergies, absenteeisms, and exaggerated reactions.

9. **Expect the best from yourself and from others.** Two keys are encouragement and praise. Vocalize daily your optimism and positive expectancy of your associates and family. They will love you for it!

10. **Associate with Positive people.** The best way to retain PMA is to associate with optimistic people. Help your associates who are negative to become more positive by never contributing to their pessimistic attitudes.

How to use PMA in dealing with others

PMA will prove to be a great asset when dealing with others. We cannot go through life without experiencing conflict with others. It is a proven fact that 60 to 90 percent of the failures in the business world are due to human relations. If we can develop the skill to deal with others with confidence, we will automatically improve our own success and happiness. If we learn the principles involved in dealing with people we won't need any gimmicks or politics.

People with low self-esteem.

Whenever a person has low self-esteem it means friction and trouble. Anything can be a threat to a person who has low self-esteem. Braggers, show-offs, and egotists are suffering from it.

The arrogant person, who always attempts to "put you in your place" may be suffering from a low opinion of himself. There are two things you should remember about these people: first, they need to increase their own self-importance and are going to beat you down, and second, they are afraid of you because one good take down would destroy them. They won't take a chance that you might decide to take them down first, so they make sure they beat you to it. Understanding this will help you in dealing with this type. By recognizing this type of person, you can avoid trying to beat them down in any way by not using any sarcastic or cutting remarks. You don't want to argue with them.

The best approach in dealing with these types is to do all you can to help these people like themselves. Feed their ego and they will turn into lambs. Feed them with sincere flattery, genuine compliments and real praise. Adopt in your life the philosophy of paying at least five compliments a day to others and you will soon find your relationships running much smoother.

HELP THE OTHER PERSON LIKE HIMSELF BETTER

Remember the following:

1. We are all egotists. A hungry ego is a mean ego.

2. We are all more interested in ourselves than in anything else in the world.

3. Every person you meet wants to feel important, and amount to something. Help the other person like himself by feeding his ego with sincere comments.

4. Each person has a hunger for approval. Satisfy this hunger in others and they automatically become more friendly, likeable, and cooperative.

5. "Love thy neighbor as thyself." We must like ourselves before we can like others.

6. People act, or fail to act, largely to enhance their own egos.

Making people feel important.

The most important key is to think and believe that each and every person you meet is important and treat them accordingly. If you practice this you will have no problem getting your attitude across to the other person, and it puts your human relations on a sincere basis. If you don't believe in this, the other person will know you are not sincere and your human relations go down the tubes. People play an important role in our lives and what's more interesting than people?

Men and women who have the most influence with other people are those who believe people are important.

Learn to praise others. Praise is a miracle - working power. You can praise a weak body into strength; a fearful heart into peace and trust; shattered nerves into poise and power; and a failing business into prosperity and success. Praise releases energy that sends spirits soaring. You will see immediate upward surge of new energy in those you praise.

Give credit to others. Especially credit for work. In a survey conducted by the National Retail Dry Goods Association, the employees rated "Credit for work" number one in importance where bosses rated it seven. It is very important that you recognize your employees for the work they have done and praise them for a job well done.

If you want to give others life or "put more life into a person", then make it a practice to pay at least five geniune sincere compliments or praises a day. You will find praise will increase employees morale and they will actually turn out more work.

Don't wait for them to do something great, recognize them for small items too. Make a habit of looking for things to "Thank" people for. Don't take them for granted. If you show people how much you appreciate them they will do more for you. Be sincere with your thank you's and make them special. Thank them by name, it adds a personal touch. Look at them when you say thank you. Unexpected thank you's are the best. Do you remember a time when you were thanked for something you didn't necessarily think was significant? This type of thank you is even more powerful because people don't expect it and sometimes don't necessarily think they deserve it.

Notice other people

If you took five people on a trip all five would see different things. Since people tend to notice only things that are important to them, when someone "notices" us, he pays us a high compliment. He boosts our morale and we become more cooperative and may even work harder. The way to get more out of your workers is to notice them and be sincerely interested in their lives. Don't be stingy in feeding the hunger for a feeling of importance.

You have heard the expression, "Negative attention is better than no attention at all". Small children use this often to get the attention they need. You even find it in some adults. If we stop recognizing the bad behavior and look only at the good we will get more cooperation. Criminals are another example of someone trying to get attention. Some commit murder or burglary just to have the world recognize them.

Lack of recognizing tells a person he or she is not important enough to notice closely. Compliments about small things say "You are important and I notice you." Tell him he looks great today! Be on time for appointments with others. Treat your family with as much or more courtesy that you would show a stranger.

When dealing with larger groups of people, make it a point to turn the spotlight on everyone. Look at each of them as you are talking. Let them know that you recognize them as important people. You will be surprized at the small amount of attention that is required to make people feel that you consider them important.

Another rule is to not get trapped into outdoing one another. For example, two children fighting, "My daddy can lick your daddy." or an adult might say "I can tell you one better than that." Let the other person know he impresses you and he will think you are the smartest person he knows. This doesn't take anything away from you by letting him know you are impressed. It only means that you respect him. You don't need to make someone else feel small so you can feel important.

When approached by a need to correct another, ask yourself this question, "Does it make any real difference whether he is right or wrong?" If he says the gun isn't loaded, and you know it is, contradict him, or if he says the bottle contains nail polish and you know it contains nitroglycerine, correct him. But if he says that it is 83 million miles to the sun and you know its not, how important is it to correct him?

You Want to Make a Good Impression on the Other Person. But the Most Effective Way Ever Discovered for Impressing the Other Person Is to Let Him Know That You are Impressed by Him.

Controlling Peoples Attitudes

Adopt the Attitude and Action you Want the Other Person to Express

Whether you believe it or not, you control people's attitudes and actions by your own attitudes and actions. Each of us is constantly influencing and controlling the actions of those with whom we come into contact. The only choice we have is this: shall we use it for good or evil, for our benefit or our disadvantage?

To adopt the attitude and action you want the other person to express, you must approach him in such a manner. If you approach someone in a hostile manner you will receive hostility. People act the part we give them. If you believe a person is going to be difficult, they will. If you believe that person will cooperate, they will be. When dealing with others we simply see our actions and attitudes mirrored back to us. When you smile in the mirror you see a smile returned.

Dealing with anger

Psychology has proven that if you keep your voice soft, you will not become angry. You can control your anger by lowering your tone of voice when you feel anger rising. Remember the biblical injunction, "A soft answer turns away wrath." If you find yourself in an explosive and tense situation that seems likely to get out of hand, deliberately lower the tone of your voice and keep it soft. By lowering your voice it forces the other person to lower his. It will also help him to control his emotion and anger. If a person is already angry this probably won't work. Let him finish and keep your cool.

Express Enthusiasm With Others

If you want others to be enthusiastic about your ideas, you have to show enthusiasm as well. You never sell anything to anyone else until you yourself are sold on it. Enthusiasm convinces others to buy your ideas.

Breeding Confidence

If you want others to have confidence in you, then you have to have confidence in yourself. There are many people of mediocre ability who have gone far just because of their capability to act confidently. Many great leaders would not have been leaders if they had not know the importance of acting confidently.

Signs of confidence or no confidence are:

1. The way you walk. If you walk with your head held high making eye contact with each person you meet it shows everyone how confident you are. However, if a person walks with his head down and droopy shoulders, that tells us, "I'm insecure", "I'm discouraged", or "I'm not sure".

2. Your handshake. A limp dish-rag handshake indicates insecurity where a person with confidence has a firm grip and slight squeeze in his hand. Let people know you're alive when you address them. Let them know who you are.

3. Your tone of voice. The voice is the one thing that gives us away the most. If we are whining, mumbling and express a hopeless attitude we are lacking confidence.

To help others build self-confidence you must let them know that you are trustworthy and then listen to them.

Creating a Good Impression

Everybody in the world is literally waiting for you to tell them what to do. You can control their actions and attitudes to a remarkable extent if you will remember to start the conversation with them on the same keynote that you want it to end on. If you want to be taken serious, act serious. If you want to be taken informal, start out informal. The other person will rise to the occasion and act out the role you set.

Your first meeting with someone new is a lasting impression they will always keep with them. In other words, you are responsible for how you are accepted. People base their opinion of you on the opinion you have of yourself. Act as if you were a nobody, and the world will take you at your own value. Act like somebody and you will be treated accordingly. Do keep in mind, however, that if you show the world what a high opinion you have of yourself to the point of arrogance you will actually be showing the opposite. Anyone who goes to extreme lengths to show his opinion of himself is actually trying to convince himself.

Three Secrets to Attracting People

1. **Acceptance**
   Accept people as they are and allow them to be themselves. Don't expect others to be perfect. And most of all do not bargain for acceptance.

2. **Approval**
   Look for something to sincerely approve of in other people. It can even be something small or insignificant. Once the other person gets a taste of your genuine approval, he will begin to change his behavior so that he will be approved for other things.

3. **Appreciation**
   Let other people know you value them. Treat other people as if they were valuable to you. Don't keep them waiting. Tell them "Thank you". Give them special and individual treatment.

How to Make Others Feel Friendly and Comfortable

Human Relations often become deadlocked because people are afraid to make the first move. We need not wait on other people to make the moves but instead assume he is going to be friendly and act accordingly. If we set the stage and expect that person to like us, he will. If we take the chance to be friendly we will find it accepted 99% of the time. Relax and expect others to respond. The number one asset to making others comfortable and friendly is your smile. Nothing is more magical!

- *Pay someone a compliment and smile, it magnifies the compliment many times.*

- *Ask someone for a favor and smile, he is almost compelled to grant it.*

- *Even when using plain talk, smile, you find anything you say will be right.*

- *Meet someone for the first time and smile, he will think he's known you all his life.*

Using White Magic - "Listening"

To be able to listen to others in a sympathetic and understanding manner is perhaps the most effective mechanism in the world for getting along with people and tying up their friendship for good. Too few people practice the "white magic" of being good listeners.

Justice Holmes

If you want someone to think you are the most intelligent person they have ever met try listening to them. The fact that you listen attentively, proves to him just how intelligent you are. Stop and think for a moment about your friends. Who has the reputation for being intelligent and wise? Do you vote for the fellow who is always shooting off his mouth?

Remember the saying "The Lord gave us two ears and only one mouth. Which must mean he intended for us to listen twice as much as talking."

People will tell you what you want to hear if you will just listen to them long enough. Good communication is a two-way street. It's give and take. If you don't know what the other fellow wants, how he really feels about a situation, what his own peticular needs are, you are out of touch with him. If you can't touch him, you can't move him. If we listen long enough before we start broadcasting we will soon find out what the other persons position is.

You may find times when some encouragement may be needed to get the other person talking. Listening carefully to everything the other person says and paying strict attention to his tone of voice and the inflection of his words.

Use the Seven Techniques to Practice Listening

1. Look at the person who is talking.
2. Appear deeply interested in what he is saying.
3. Lean toward the person who is talking.
4. Ask questions.
5. Don't interrupt; instead, ask him to tell more.
6. Stick to the speaker's subject.
7. Use the speaker's words to get your own point across.

Getting Others to Cooperate

Each day situations arise where we need to persuade another person to accept our viewpoint. It can be with your boss, husband, child, friend, or parents. You hear so many people say, "If only I could get him to see things my way." The most common way we go about getting others to see our way is to argue. Instead of using argument, we need to convince the other person to change his mind. The way to do this is to use low pressure.

Low pressure simply means stating the facts and leaving out any threats or attempts at using force. It means presenting the facts to the other person without emotion. Research by two professors at New York University's speech Department discovered that the one big mistake most of us make in trying to win an argument is we attack the ego of the other person.

If we want to develop our human relations we must learn to work with human nature rather than against it. If you find fault with someone, he has no alternative but to defend his position and ego. If you threaten and use scare tactics a person will simply close his mind toward any of your ideas, regardless of how good they may seem. Survival is the strongest desire of all humans and the human ego is part of this survival.

If we want to sell our ideas we need to remember the key element is to reach the other persons subconscious mind. No one accepts or acts on a idea until the subconscious mind accepts it. An example would be when a person nods in agreement with you but their expression tells you they don't accept the idea. You know then that you have not reached the subconscious. The only way to reach the subconscious is to use suggestion. An example of suggestion is when someone tells you "You can't do that." At that very moment you are bound to try to prove them wrong. By slipping suggestions to the subconscious we can get our ideas noticed. You might tell someone "I know you can help me because of your expertise in the area."

Use the following guidelines to get your ideas across:

- *Let him state his case*

  Use your "white magic" and listen. If you listen you are telling the other person they are important. You may even want to ask questions about his case to show him you are listening.

- *Pause before you answer*

  Show the other person you are thinking about what they said by pausing before you comment. Don't answer with an abrupt answer, if you do they will feel as though you didn't take the time to think about it.

- *Don't expect to win 100 percent of the time*

  Sometimes we have to give a little to get. Acknowledge good points the other person may have. If you cannot agree or negotiate explain your reason why.

- *State your ideas moderately and accurately*

  State clear and concise facts about your ideas.

- *Speak in third person*

Using the third person when you have a difference is very helpful because people are naturally skeptical of you when you are saying things to your own advantage.

- *Let the other person save face*

Many times people must follow through with their ideas simply because they must save face. Allow the other person to save face.

- *Let the other person feel it's their problem too.*

If you want to get full cooperation from someone let them think that it's their problem also. Psychological research has shown that you get more from someone if you allow them to share their ideas. Ask for suggestions and advice. You have then given them a problem to solve and they become interested.

- *Ask for advice*

Ask for comments or suggestions.

Participative Management

"People support what they help create." Robert C. Hood

Participative Management is giving others the opportunity to offer suggestions and advice. Managing does not mean you are to furnish all the ideas and brains. In the "old management days" managers were the brains, and the employees were merely the hands. New management styles encourage employees to participate in solving every day problems. Ask them "What is your opinion?" or "What do you think of this?" A smart manager will use this to magnify his ideas by letting others contribute.

This method can be used at home too. Allow your children and spouse to offer suggestions and ideas. It's a sure way to get them to cooperate.

Criticizing Without Offending Others

The number one error that we make when correcting someone is the attempt to increase our own feeling of self-esteem by lowering the self-esteem of the person. Chronic fault-finding, belittling the other fellow, and nagging are symptoms of low self-esteem. To successfully point out errors and correct others is quite an art form, one that is not perfected by many managers.

Approaches to Criticizing others:

- Always criticize others in complete privacy
- Begin with a kind word or compliment
- Criticize the act and not the person
- Supply an answer
- Ask for cooperation; don't demand it
- One criticism to an offense
- Finish in a friendly manner

A Plan of Action That Will Lead You to Success and Happiness

Your success depends on your willingness to use PMA as the director of your mind. You must strive to apply a Positive Attitude each day in every area of your life.

Remember your single goal is PMA.

Once you have grasped this attitude and apply it with ACTION you will experience happiness and success. The results may seem slow at first. But remember that you are building something very significant - a successful and happy person. If you will practice PMA you will soon discover that you are developing a truly effective system. A system for coping with stressful situations and emotional confrontations, you will get along better with others, you will see your life moving forward to greater expectations, you will get personal satisfaction from setting and achieving goals, and to sum it up, YOU WILL EXPERIENCE SUCCESS AND HAPPINESS.

References

Think and Grow Rich by Napoleon Hill

How To Have Confidence and Power In Dealing With People by Les Giblin

The Psychology Of Winning by Dennis E. Waitley Ph.d

Success Through A Positive Mental Attitude by Napoleon Hill and W. Clement Stone

Love Is Letting Go Of Fear by Gerald G. Jampolsky, M.D.

Operator Training in the Small Shop Environment

Flo Barley
Pekin Memorial Hospital
Court & Fourteenth Streets
Pekin, Illinois 61554-5098

INTRODUCTION

Many times in a small shop environment, formal class training outside of DP isn't possible and inhouse training is the only option available . Whether the new operator has previous experience or not, there are certain basic tasks to be learned regarding the operation and specifics in keeping a smooth running computer system during the training period. The transition to a new operator should be kept invisible to the users and be as painless as possible for everyone else.

With a pre-training checklist you can determine where to start training depending on the new operator's previous experience or knowledge. Then using basic training tools, worksheets and guidelines, the training of a qualified operator can be in your control.

This paper will hopefully set up the beginning guidelines and strategies needed for training and can be modified to suit your particular needs, branching out from hardware to your software requirements, and can be retrofit at the appropriate times in your training schedule.

WHO NEEDS TRAINING AND WHY?

In reviewing the material for my paper and thinking back on past experiences and now on my current responsibilities, I realize that in a small shop, training not only involves the operators, but also the users.

Many small MIS Departments not only have responsibilities for the operations, but also the software, programming, internal documentation & procedures, file & database management, and user training & support. If this is the type of shop you manage or work in, you know the problems that have to be dealt with to accomplish effective training. The MIS Department has to be everything to everyone -- Jack of all trades - and master bluffer of all!! You can tell a user that you don't know the answer, but you still have to find the answer. To them, you're the guru - the only one who can tell them what the answers are. To them, computers are computers and if you know one -- you know them all and can answer all their questions. So we have to learn to find the answers no matter how easy or difficult and knowing where to look or what resources to tap is very important.

Training should be an ongoing process. Each environment is different and must be approached from a different angle. You not only have your HP system with both hardware and software, but you also have various software packages ranging from financial system reporting to hardware monitoring tools. Once you get a basic outline of a training plan that will fit into your organization, keep it updated and include any new hardware or software as it is installed.

No matter what type of background your operator has come from, there will be some type of initial training needed. Even if they are from an HP environment, there will be training involved -- and sometimes, it is actually harder to retrain to your standards than it is to train someone who has never worked on Hewlett Packard equipment before. Regardless of their previous training, the software they will be using may be different and there will be a certain amount of training required for them to both use it and to support the user base.

The first step is to get them acquainted with the various equipment you have throughout the business and to what extent it's used. To help facilitate this process, it would be a handy tool to have a master list of the equipment and what software is used on each. This could help in multiple ways -- ie: you have your list of equipment needed for service calls with serial and model numbers and you would also be able to maintain a master list of the software your users have available and who uses what.

They will need to know what the system hardware consists of and key terms they will need to recognize and understand. The configuration of the hardware is vital to any installation and should play an important part in the training in order for the operator to fully understand and appreciate the complexity of the equipment and how the pieces fit together to create the system you have. There should be an understanding of the accounting structure and why it is necessary.

Explain the terminology within the system and used inside your organization. It is important that they understand what MPE is and what it has to offer. Show them how to use the help subsystem, how to use the MPE Quick Reference Guide and what HP manuals to read. Have them know where to look and find pertinent information regarding the MPE Message System, the configuration guidelines for setting up system tables, and the different subsystem utilities. There are utility programs standard with each system -- explain what they are and their use. What types of Data Base Management tools do you use? These need to be reviewed and the extent of their involvement with each along with any system monitors used. There may be a minimal to heavy explanation of what languages are used and how they will impact the new operator and his work. They will need to know how to monitor the system and manage jobs, sessions and spoolfiles. This would include the use of SPOOK, UDC's, STREAM, file types and equations.

It will be necessary for them to know how to start and shutdown the system and what processes are necessary in your environment. This area would also include what to do when there is a system failure or halt, ie: is there a downtime log or failure/halt log to complete, who to call, what steps are needed before startup such as string dumps, memory dumps, and what type of recovery is needed. They should know the difference between WARMSTART, COOLSTART and COLDLOAD and the importance of each. What is a RELOAD and when and why should it be done? How should it be done? What can be done to save data when a system is down and there is no current backup available? What is a SYSDUMP and why is it done? What types of backup options are there?

All of these are good topics to cover and necessary to maintain efficient operations as far as keeping your system operational, but not necessarily keeping your users happy. There are also the aspects of the hardware that the users have and the micro systems and their software.

What is DOS? To what extent will their knowledge need to be? Are the PC's backed up on a regular basis? Must they deal with hardware problems here and possibly the recovery of files? What tools are available and how do they work? Are they to know how to unpack a PC and set it up and load software before delivery to the user? These are questions you need to think about in training in order to assure yourself that you have not omitted any items from your checklist.

Once helpful tool is to create a workbook for them to use as a reference guide. In this notebook, you should have an outline of the class, any copies of overheads, a pad of paper for notes, and any list of hardware and or software that you have already compiled. As you cover the items on the checklist and copies are printed of material during the training ie: configuration list, add these to the workbook. Make sure you have covered all the bases. Don't use UDC's during the training period -- this forces them to learn the full command -- then have them set up their own UDC file. As they ask questions, have them look them up in their workbooks or the manuals to get them pointed in the right direction of learning where to look for the answers. You may want to set up special accounts and/or groups for the training sessions. This will allow you to keep track of what files are for testing and after each training session is completed, you can stream a job to clear out the accounts and/or groups and recreate them for the next training session. Make up worksheets and tests for them -- give them homework.

It's important that you know your material and look for any answers at the time they are asked. This will build and gain their confidence about your technical ability, knowledge and experience.

When it comes to training users on the micros, don't let them get caught up in the "I'll train myself with this handy-dandy self-paced training guide" syndrome. Take the time to sit with them and learn it -- teach it. If you let them do it on their own, they will do what they can to get it to work, but never fully utilize all of the capabilities of the software. You'll see them using it one day, and the next they'll be doing part of their job the 'old manual' way. When asked why, you'll more than likely get a response of 'Because we've always done it this way'. Some think that learning from the manuals works by osmosis - but what they don't seem to understand is that first, you must take off the shrinkwrap!!

Use your own experiences -- good and bad -- let them know you make mistakes. Tell them about some of the mistakes you've made and they won't feel so bad if & when they do. Let them know how to recover from these mistakes. This will instill confidence in them that they can recover from their mistakes.

Explain to them the impact of their job on the organization and to what degree the users are affected when the system is down or the data inaccessible. Giving them a view of the 'big picture' sometimes opens their eyes and you'll have their full attention during the training sessions.

You should also remember that your staff needs to be cross trained. The inevitable will happen one day and you'll need to have coverage for someone. It is very important that your staff is trained in all the areas necessary to keep your department functional and in operation. Just as you do backups each night for your system, you need a backup for your operators.

This opens the door for having well written procedures and documentation. It's not an easy task to undertake, but getting the users involved in helping to write these procedures can give them some incentive to learn more about the system and what they are doing. It may be easier for you to write the procedures and have the users follow them, but you can turn them into robots this way also. If the procedures are well written, anyone can sit down and use them, which I feel is a good way to do it, but you can bypass the whole idea of having them understand just why they are doing what they are doing. Some explanation is needed in training and you can also expand your procedures to include the explanation. This is also very helpful in helping a user with problems. You can know and understand all of the hardware, but knowing the software well enough to help a user in trouble can be another story. Being somewhat familiar with the logic involved with the system, you should be able to use the procedures for that particular software to walk through the problem. This also lends credibility to keeping them consistent as to content.

It is also good to tap your employees for their prior knowledge. They did not necessarily take this job to gain all of their knowledge from your training. Chances are, you hired them because of their prior knowledge or experience. Don't forget to take advantage of this and perhaps enhance your operations and learn something new yourself.

THE CHECKLIST

Now let's put together a checklist of the ideas that we've just discussed.

I. AREAS TO COVER:

A. Hardware (Mini & Micro)

1. Wiring & Connections
2. Configuration
3. Startup/Shutdown
4. Accounting Structure
5. Tables & Settings
6. Remote & Dialup features
7. Operating Systems

B. Software

1. MPE
2. Commands
3. Help Subsystem
4. Utilities
5. Data Base Management
6. System Monitors
7. Languages
8. Written procedures

II. STRUCTURE

A. What will be taught

1. Keyboard operation
2. PC Assembly
3. Troubleshooting
4. Backup
5. Logon procedures
6. Use of Manuals
7. Installation of software
8. Backup and storage
9. MS-DOS commands

B. When and how long

1. Time frame for each class
2. Repetitive or continuing sessions

SUMMARY

Attract other users to your classes by setting up structured training sessions on a repeating basis. Take the fear out of new users by letting them acquaint themselves with the terminals and PC's by playing games. This helps them gain a comfort level with the equipment and not feel so intimidated by the 'big picture' too soon. Once they have relaxed, they can easily move into more of the sessions to suit their needs. Don't forget to show them the help facility and most of all -- follow up with them within a reasonable amount of time to be sure that they haven't pushed something aside because they didn't have time to read the manual or figure it out on their own.

It's amazing what users can do with a computer if given a chance -- some will progress without a doubt to heights further than you'd ever imagine and others will muddle along and 'adapt' to what they must to stay status quo in their positions. Whatever the case, make sure that your operators are able to handle the users to their fullest capacity and continue to grow with the users to keep the small shop viable. There is much to be said about having a small staff and small shop management. There is versatility in your job, a chance to learn much more than one tedious task and you're given the ability to grow with the system and the users with no fear of ever getting bored. Just remember....

DON'T SETTLE FOR MEDIOCRE -- SETTLE FOR THE BEST YOU CAN DO!!!

Hiring System Professionals -- Lessons Learned the Hard Way!

Robert R. Mattson
9545 Delphi Road S.W.
Olympia, WA 98502
(206) 736-2831
(206) 352-5038

Abstract

The most important factor in the success of systems and system departments is the quality of the people involved. So it would seem, if one hires top quality people...one is well on the way to achieving successful systems. The difficulty is that the hiring of top quality people is not at all easy! Through the years I've used various strategies in this area. The results have been mixed and less than desired. In recent times I have been developing a revised strategy. It is based on evaluating and emphasizing a persons attitudes, aptitudes, approaches, actual skills, accomplishments and appeal. This paper outlines this non-traditional approach which de-emphasizes the value of classical "experience", "education" and "references." The goal is to allow you to avoid a lesson or two I've learned the hard way!

Our World, Profession and Hiring

Introduction

This paper/talk covers certain aspects of the hiring process. This paper is NOT directly about the issues of advertising, search firms, formal steps in the process, etc. These issues have been covered recently; for example see "Putting Your Best 'Feats' Forward", by Diane Amos, Interex Computing Management Symposium Proceeding, Nashville, 1989.

Rather, this paper's focus is mainly on what criteria we should be using in selecting people for a system's position. And further it addresses how much weight one should put on various criteria. The goals are to spur discussion in this critical area and to pass on some of the lessons I've learned the hard way! First let's spend a little time

discussing why we should care about this subject and why we need to spend time focusing directly on it.

The Importance of People

The most important factor in the success and the quality of systems is the quality of the people involved. It is not my purpose here to prove the validity of this statement. I do however want to go on record that I believe strongly in the truth of this statement. My experience agrees with statements found in our professional literature. People are the difference between success and failure. There are ten to one differences in programmer capabilities and productivity. The number of people who are good designers is one in a hundred. There is a five fold difference in the impact on quality brought about by people. So, let's assume for the purposes of this discussion that it is true. This will allow the focus to be on addressing how we can improve our ability to define and find the quality people we need to succeed.

Turnover/Mobility

Even if people are critical to our success, we still wouldn't have to deal with hiring the right ones very often if people didn't switch jobs. But we know that the world of systems is one characterized by personnel turnover and mobility. The increasing growth of the use of computer systems has placed increasing pressure on experienced systems people to move to a new jobs. The last figures I read indicated that the average job span for a system professional is a little over two years.

Changes in Staffing Requirement...Projects, Growth, Etc.

Further, systems is an area where projects are common. Many times these projects require manpower above current staffing to accomplish them. Also in many cases, the increase in the number of installed systems puts a demand to increase staff to support them.

So Hiring Is a Fact of Life

Both of the above result in the need, at various times, to hire people to fill systems jobs. It is the rare department these days that goes for very long without having to deal with the hiring issue. And because people are

critically important, any improvement made in this area will pay the highest possible dividend.

Why Is This a "Neglected" Area?

Yet, given the hiring task's relevance and importance, it appears that we spend very little time improving our skills and results in this area. Why is this? Here are some of the reasons.

First, in talking about how to hire the "right" person, we are dealing with a less than "concrete" area of concern. This type of area seems easier for systems people to avoid than address. And indeed many people would not attribute the importance to this factor that I have given it.

Second, information systems manager's training and skills in this area tend to be minimal. We have had classes and read books on databases, analysis methods and new languages. Yet few if any have done the same for the skill of how to define, find and hire the right person.

Third, there isn't, unfortunately, much sound advice in the literature and from the "experts." Compared to other subjects there is very little coverage devoted to this issue which isn't exceedingly a rehash of the obvious.

Fourth, it appears each vendor in the systems field is sending a hidden message. That message is...."if you use this amazing software, hardware or technique then all your problems of people quality, productivity, etc will go away." We spend billions each year hoping they are right!

Finally, our managers tend to believe that people are a commodity which one simply purchases as needed. This attitude is epitomized by the statement "we'll need to hire a couple programmers!" My experience leads me to believe that good systems people are like rare wine...not cheap beer.

So Let's Understand and Improve This Process

Briefly, I've set the stage for why we need to focus our attention squarely on how we define personnel needs and how we hire the people to fill those needs. And although not the subject of this paper, logically we should be spending

more time on figuring out how we keep the good people we have. Keep in mind that retaining a good person is a much better and cost effective strategy than trying to hire a new one using any technique.

The Hiring Process

The Hiring Lifecycle

There are some pretty common steps that people go through when hiring someone. The steps might look like the following:

1) Decide that you need a person
2) Get approval to hire a position
3) Advertise or otherwise search for candidates
4) Narrow the candidates list down
5) Interview candidates by phone
6) Have a few candidates come for on-site interview
7) Check references
8) Select person to offer job
9) Offer job to person and do negotiating if necessary
10) If they accept then hire..otherwise backup some number of steps and repeat.

Rather than focusing on the steps I would like to focus on what criteria we will use to select the person. Also, to some extent I'd like to focus on how we can "measure" the candidates strength or weakness in an particular area. I want to focus on these two areas because I think they are the hard ones. It is here that we really are doing the selection process...not when we go through the steps above. First, let's look at the more classical approach to hiring one sees most often.

The Classical Approach - Training, Experience and References

When we talk about the criteria for a particular position we usually talk in terms of formal education and experience. Thus very often one will see a requirement that a particular position requires a "four year degree in computer science" and "2 years of programming experience." The problem with this approach is that I've had people with neither of these who could blow the socks off of people with this education plus ten times the experience. In fact, my experience tells me that in the hiring decision we place far

too much weight on the "experience" and "education" criteria compared to some other less obvious criteria.

Part of the reason I believe the education and experience criteria are so popular is that they appear easy to measure. In other words, one either has or doesn't have a degree, etc. Another reason is that personnel systems are set up around this model. And in fact many compensation plans are based around these measures.

A problem with both education and experience is one of definitions. In other words, what does it mean to "have a years experience with COBOL" or have a "degree in computer science." Or even worse "have been a programmer for one year!" I know people who have "experienced" COBOL for ten years who still don't possess the skills in it that I would want from a beginning programmer. Similarly, I know people who have a degree in computer science who are a fraction as well trained as some people with no formal training at all or two year associate degree. Differences are even more evident when "time in a position" is used. I know people who have the title of "programmer" who have more responsibility and do more difficult tasks everyday than people called "system analyst" or whatever advanced title one wishes.

Now, I'm not suggesting that for two otherwise equal individuals that one with zero experience is equal to one with some years of experience. But clearly, years of experience or training are far less an indicator of what an individual can do for you than is commonly thought. And the lesson I've learned the hard way is to base no more than 25% of my hiring decision on these two criteria!

The use of references has its own problems I've found. I was surprised more than once by the differences between what a reference has said about a person and how they appear to me once hired. I'll ignore the issue of whether I am as good a manager as the reference or not. Assuming I am, then clearly either the reference was lying....or as I believe they just have different criteria of what makes a good programmer, analyst, designer or whatever! And though it may be obvious, people seldom give people as references from whom they feel they'll get a bad report.

So, when it comes to references, I do spend some time talking to them. I try to ask open ended questions about what responsibilities they had, strengths and areas for

improvement. But, I discount any positive remarks and magnify any suggested "areas for improvement." And in the final decision the reference data, most times, is not given much weight unless it's negative about some criteria I value.

My less than happy experience with these classic methods has led me to try to figure out a new approach. This approach doesn't ignore education, experience and references but hopefully gives them their proper weight.

The New Approach

Assumptions

My new approach is based on the following strategic premises.

First, do a excellent job of analyzing what we want in the way of a person. Understand the job requirements, my performance expectations and the environment.

Second, it really is what a person can do and how they do it that is important. Conversely, it really is of no importance whether they have "experience", "degrees" or "great references." None of these three does the job...only the person will be around to do or not do the job.

Third, it is worth spending considerable time in the hiring process. I have found it is hard to tell your boss you haven't filled a position or that someone isn't right for the job. But these are NOT hard at all compared with living with someone who doesn't do the job you need and expect or even worse having to firing someone!

Fourth, it is better to do without someone than to have someone that isn't, for whatever reason, able to do the job.

Fifth, there is no perfect person. The real challenge is in deciding characteristics which are essential and those that are not. There are some characteristics which if lacking can cause all the other characteristics to be pretty well useless.

Sixth, hiring someone is a lot like marriage. You know the person will change over time...it is just impossible to predict how much and in what area. So, look carefully and

decide whether you can live with them if they don't change in the area you would like. The lesson I've learned is that it is easy to assume that we can change someone. But, unfortunately, a manager's capability to change an employee in any particular area is far less than we might hope. This makes doubly important that we maximize our understanding of the person we might hire.

What Attributes the Candidate Should Have?

In line with this strategy, I have put a lot more attention into addressing questions such as the following. What is the type of person that I want to fill the position? What skills do I expect the person to have when hired? In six months? What am I like? What type of person do I deal with best? How can I know what a person is really like? All these involve the process of trying to describe the "perfect" but realistic employee and figuring how to determine if I've found one.

The Criteria For A Systems Person

This thought process has lead to a set of attribute areas where I try to do my defining. Once I have a description of the criteria then I try to develop some means of assessing/measuring this criteria in the hiring process.

The following are the criteria areas I have identified as useful to consider: aptitude, attitude, approaches, accomplishments, actual and appeal.

I'll take each in order. I will try to give you a feel for what the criteria are that I value the most. Remember that you and your environment may have additional criteria and/or value one more or less. Then I'll address a couple lessons I've learned about assessing attributes in general.

Aptitude

This criteria deals with whether the person could do the work if trained and motivated. The key here is in determining not just what they can do today but what can they do in six months or with that new database, etc. The lesson that I've learned is that there are a lot of people in the systems business today who are there because it was possible to take and pass a few "programming" classes at some type of institution. Unfortunately too many of these

don't possess the ability to handle even beginning jobs in this field. Sad as it is, our job is to determine whether a particular individual does or does not possess the abilities required.

One measure that has pretty good correlation with ability is good grades in a good educational program. The tough problem here is in evaluating the quality of unknown programs. If you know the program and instructors then discussing the candidate with them is very valuable. The state of most community college and many four-year programs is not rigorous enough to be good test of ability. I have learned, from my experience, that if the person did not do well in their formal programming and system classes that they probably won't do well in "real life." The lesson then is to look for top students and only consider the rest if there are really other overriding attributes.

Here it is also important to focus on "change" and its impact on aptitude. It has been the case that the last ten years has seen tremendous changes in much of what we need to know to do our jobs. In my environment, we have to be able to become "overnight experts" or so it seems many days. As a consequence, I believe that the ability to learn rapidly and easily is essential for long term success. This ability is easily confused with what a person knows today. We need to know how hard the person worked to attain their current skills. Some people can learn to program at a certain level given enough time and tutoring. Unfortunately, they may not be able to learn whatever new "thing" that you require in the time frame you find acceptable.

Under aptitude we need to mention a few other characteristics that are important. The limit on space makes going into depth on these impossible. Nevertheless, they should be given good weight and attention. Three important abilities are analytical problem solving, business judgement and sensible creativity. These three are important abilities required in all the system jobs of programmer and above! These are also rare in the majority of system's people on the job market. Another important ability is good written and verbal communication skills. The lesson is, that if a person cannot articulate on a resume, a cover letter and in an interview...they will almost surely not be able to do it in any important area of their work!

Attitude

The key question here is... what are the critical attitudes that your perfect person should have. Do they have a positive attitude toward the profession/job? What is their attitude toward continuing education and trying to keep up with this changing field? What is their attitude toward quality? Are they motivated toward excellence or will they only do what you tell them? Is their basic attitude one of cheerfulness? What is their attitude toward getting the job done? How about taking responsibility to follow through on agreed commitments? Note that this is only a partial list and your attributes will differ.

A lesson to be learned is that one should not avoid these difficult to assess areas. I've learned the hard way that the person with good ability or experience is much less valuable because they have one or more wrong attitudes. It is worth the time to think through what attitudes you like and what attitudes are problems. Some type of open ended question is helpful in getting data on attitudes. Such questions as...."What's you attitude toward quality?" can get you many times either surprisingly blank or articulate answers. The tough thing is not to ask leading questions or say things that allow the "crafty" interviewee to know what answer you want.

Approaches

This attribute area deals with work methods and problems solving approaches. I address this separately from attitude because I have learned this needs special focus. This area includes questions like the following. Are they self starting or must they be pushed? Do they have multiple strategies for attacking problems? How do they approach something they don't know....do they rely on other people to help them or do they dig in an learn it themselves? Do they learn more than required to do the job? Do they take responsibility for tasks? Can their commitments be relied upon? Can they make good business judgements when deciding on approaches? Can they change their approaches when they aren't productive? Are they analyzers?

As with attitudes, there is no single question or easy way to assess the way a person approaches tasks and problems. Yet, the lesson I've learned is the importance of

the way different people approach things. For example, in my environment, people who are not self motivated to keep commitments cause problems because we are not rigidly organized and I'm not a policeman type manager.

Accomplishments

This criterion deals with the fact that systems is in many ways a task/project oriented business. People who have shown in the past that they can "accomplish" things possess an important attribute for systems. Looking at outside accomplishments as well as systems related is helpful. The open ended question I ask is "Tell me about some of you accomplishments of which you are proud?" I believe that the most successful systems people will in general be able to point to a number of accomplishments.... if they can't then there is a reason for caution.

Actual

This area deals with the criteria of exactly what skills you expect the person to have when hired. In addition, it attempts to assess exactly what skill the person does have today. This is not as hard as some of the other areas but not as simple as it might seem. Also, I believe we don't spend enough time thinking this through.

The challenge here is to do the following:

a) List all the skills we want the person to have
b) Order them in importance
c) Determine what skills the candidate possesses
d) Determine whether we can live with the person not having some skill
e) Determine how long it would take for the person to learn the missing skill

It is not very easy to do! And realistically no candidate will possess all the skills you would like. This makes it doubly important to have solid knowledge of what their potential ability to learn these missing skills are!

The challenge is to assess whether someone really has the major skills you need. It is a difficult task to try to determine this. In this area, more than any, the best method of determining this is to have the person demonstrate their skill. See the section later on "Trials."

Remember, if the person you decide to hire does not have a skill you need then you will be betting they can learn it. At the same time unless you have been able to prove it... don't over estimate an experienced persons skills. For example, I have learned that assuming a person with ten years programming experience could type is a mistake. I have over estimated the skills of "experienced" people enough times now to have learned it is deadly!

Appeal

This criterion simply has to do with whether you like the person. Is this a person you would like to be around? This doesn't mean you should hire people just because you like them. It does mean that this is a factor which should play a reasonably important role in your decision. Depending on the situation, we may spend more time with this person than we do with our spouse or family. So whether they can "do the job" may seem immaterial if they are someone you can't tolerate having around.

I tend to think of this toward the end of my initial evaluation of a person. Although first impressions can be wrong....I think they are right most of the time as to whether we will like the person or not. The key is to believe that it is professional to use this as one of the criteria in our decision process.

This is also a good place to evaluate whether a person will fit with your other people. Some things are obvious, does the person smoke... if your shop is smoke free? I now say "obvious"; but it was a lesson I learned the hard way!

Assessing the Attributes

The preliminary challenge of evaluating candidates is in coming up with the list of attributes/characteristics/skills for which we will search. This is, unfortunately, much easier than actually measuring those in your potential employee! Obviously there is some data that is pretty easy to get like previous positions held, degrees, grades attained,etc. The real problem is that this is only a small part of what we really want to assess. Here are some suggestions for assessing the harder attributes.

First, have multiple people involved in the establishing the assessment attributes. Preferably get your systems associates involved. Then have them involved with the interview and selection process. And at appropriate times get their feedback on the candidate point by point. My experience is that other people can many times see things any one person might miss.

Secondly, use written and verbal questions which address as many possible criteria as you've established. Make them as "open-ended" as possible. Try not to telegraph your expected answers. Ask follow up questions. "What do you mean by.....?" "Tell me more about...." Document the answers so you can review these in black and white later. This helps in the evaluation process to clearly see what they said. It is often easy to forget significant and important answers after the passage of time and intervening interviews, etc.

Thirdly, utilize references to get at many of the same attributes. This will take creativity in question asking. But many times a question like "What are the five words you'd use to describe...?" will tell one a lot about a candidate. Also, you can also ask references questions about what characteristics they value most in an employee.

Fourth, utilize a trial employment. This is a concept that beats every other technique I know. It is worth the expense and any hassle you have to go through. The way it works is to have a person come in and work as a "contractor" for a one to two week period. Pay them for their services plus expenses as appropriate. Then work closely with them on a set of real projects focusing on the most important actual skills needed. During and after the trial assess the skill and attributes you see.

A trial run sounds like a lot of work! Yet think about how much work you've saved yourself if the person turns out to be someone you don't hire. The dollar cost and hassle cost is much less than having to deal with a person who turns out to be much less than you needed.

Fifth, fight hard to see clearly the strengths and weaknesses of the candidate. As much as possible look for actual answers and data to indicate that a person possesses the attribute you desire. Try not to place any positive value on "lack of indication." For example, just because you didn't see anything that says the person "doesn't keep

their commitments doesn't mean you should interpret that to mean they do!

Sixth, base 75% of your decision on the criteria other than "experience", "education" and "references."

Conclusions

People are the most important ingredient in the system's success pie. Hiring the right person is critical. Hiring the wrong person can be disastrous. There is no quick and easy technique or question to apply to assure you hire the "right" person. As with many problems in life... there doesn't appear to be an easy answer. Rather, we improve a little bit at a time by discussing, reading, questioning and trying. Although I offer no easy answer, I do believe we can improve our batting average. This can be done by focusing on the attribute areas of aptitude, attitude, approaches, accomplishments, actual skills and appeal, using the techniques described above. If this sounds like a lot of work... remember you can pay now or pay later. So good luck and I hope that something you have discovered here allows you to avoid one of my lessons learned the hard way!

The Author

Rob Mattson is currently Manager of Information Systems at WIDCO, a large open pit coal mine in the state of Washington. He has been working in the systems field for 14 plus years, the last 11 in the HP3000 environment. His background in addition to many systems related roles includes a degree in Psychology, work as a Certified Public Accountant and as a management consultant/trainer. His current professional areas of interest include general systems theory, project management and exploring the concept of system quality. When not immersed in systems he enjoys his family, sailing, sea kayaking and golf.

Paper Presented Originally at HP3000 International User's Group Meeting - San Francisco, Sept. 1989.

The Renaissance of the Apprentice

Pamela Dickerson
American Data Industries
2465 Campus Drive
Irvine, California 92715
(714) 955-0888

Introduction

In Walt Disney's film, The Sorcerer's Apprentice, the audience sees the disasterous results when an apprentice tries to perform tasks for which he is not properly prepared: water swirls first around his ankles, then his waist, then his neck, until he is finally swept in, quite literally, over his head.

In today's high technology, information-driven world, new employees face much the same challenge: learn to conquer a new job, not be swept away by it.

To that end, apprentice programs are returning. In these greatly specialized training programs, employees are given personal attention in order to help them accomplish predefined goals. Implemented correctly, these programs can help produce valuable employees loyal to the organization and assimilated in the company culture.

This paper examines apprentice programs, including what makes an apprentice, where apprentice programs work best, how to design an apprentice plan, finding the right candidate, managing the program, and benefits/pitfalls.

What is an Apprentice?

Merriam-Webster defines apprentice as a person learning a craft under a skilled worker. In medieval times, a youngster would be apprenticed to a master craftsman in order to learn all aspects of the craft, whether it was a blacksmith or a carpenter. In today's high-tech, service-oriented environment, we have largely moved away from apprentices. We use designates such as 'trainee' or 'junior', or the nebulous, 'associate.' In-house training programs are often particularly lacking, especially in

smaller companies. New employees are expected to learn 'the company way' by osmosis.

An apprentice program in today's highly competitive industry starts with assigning new employees to senior personnel, either on a one-to-one or small group basis. Senior staffers are able to pass on their knowledge while the apprentice gains experience by assuming duties gradually.

The apprentice and his mentor should both understand the tasks and concepts that the apprentice is expected to master, and the estimated time to complete the learning process. By dividing the learning process into different levels, the apprentice is able to start with rudimentary concepts and build to the more advanced. At the end of the designated period, the apprentice is considered to have mastered the skills and concepts required of him and is a full-fledged employee, capable of handling both day-to-day and exceptional tasks.

When (and Where) Apprentice Programs are Appropriate?

Apprentice programs require two primary conditions to succeed: one, the company must have established practices and procedures; two, the job to be apprenticed must require a significant variety of tasks with varying degrees of complexity.

First, start-up companies are poor candidates for apprentice programs because policies are in a constant state of flux. All employees, from the CEO on down the organization chart, are not only learning what those procedures are, they are actually defining them. Instead, apprentice programs should be held in reserve until the procedures are set and understood by the employees, who will by this time have senior status.

The second prerequisite for success, that the job itself require the mastery of many skills, separates apprentice programs from other training programs. For example, a technical writer hired into a medium-sized company not only has to become acquainted with the writing style of the company's manuals, but may have to learn a new word processor and page formatter, the specific business in which his employer is engaged, formats for technical documentation as well as press releases, brochures and training documents, which programmers are responsible for which pieces of

software, how documentation is distributed and how to prepare estimates for different types of documentation. With an increasing number of small-to medium-sized firms etching out market niches for themselves, it is likely that more individuals will be responsible for a larger variety of tasks requiring specialized knowledge.

Designing an Apprentice Program

Identify All the Tasks to be Learned

Start by getting all senior employees associated with the apprentice position together, then brainstorm; write down everything you can think of that the apprentice may have to learn. You'll edit after you have the ideas on paper. Be specific and thorough. In the case of our technical writer, he may have to know how to operate office equipment, including a duplexing photocopier and a sheetfeeding fax machine. MPE is certainly a requirement and a word processor on the HP 3000 -- QUAD or EDITOR or TDP, for example. He'll need to be acquainted with UDCs to increase his productivity and he'll have to understand the different utilities available that have been written in-house. This should be an extensive list.

Distill the Ideas and Assign Priorities

Once you've got the various task areas down on paper, go through and combine similar tasks and eliminate others. Once you've distilled your list to the tasks that a full-fledged employee must have, you're ready to start assigning priorities.

Setting realistic priorities is key to the ultimate success of your program, so get as much feedback as possible. One way to accomplish this is to list all the tasks and let the people involved in the initial definition assign priorities to each. Based on the consensus, you've got your final priorities; figure 1 illustrates a form designed for this purpose. Incorporate any training material you have for specific tasks at this point. For instance, if you use HP's Guided Tour, consult that and determine where it falls in your list of priorities.

FUNCTION	LEVEL				
	1	2	3	4	5
FAX					
Photocopier					
Line Printer					
Laser Printer Operation/Maintenance					
Velo binder					
LaserJet					
Postage Meter/Scale					
Hole Punch					
Phones					
TDP					
MPE					
MPEX					
SP System					
SPOOK					
VooDoo					
Console commands					
Mail System					
Timesheet Program					
ETPs					
3000AD System					
Query Calc					
Equater					
Phantasm					
Menu System					

FIGURE 1

FUNCTION	1	2	3	4	5
Prepare user doc					
Prepare report doc					
Prepare System Audits					
Prepare Contents					
Prepare manual indexes					
Prepare master index					
Prepare system glossary					
Prepare procedural doc					
Prepare training doc					
Prepare contracts					
Prepare price list					
Prepare sales samples					
Prepare overheads					
Prepare press releases					
Prepare newsletters					
Prepare proposals					

FIGURE 1

You've set your priorities. Now lay out, in detail, what the apprentice is to master at each level. In the case of our technical writer, we're going to assume that he will need to learn MPE commands, TDP commands, accounting functions, UDCs and miscellaneous other tasks. Be specific about which commands are required in which categories at which levels.

Now that you've laid out exactly what each level consists of, you're able to assign a timeframe. This is going to help you evaluate the progress of the apprentice and it's going to aid the apprentice in pacing himself. It also offers objective criteria to evaluate one employee against another. Figure 2 illustrates the goals for our technical writer at Level 1. Make sure that all employees who will be working with the apprentice have a copy of these goals for each level. This ensures that they do not talk over the apprentice's head during training, and that they stay on track.

Develop an Evaluation Process

In order to help the employee progress from one level to the next, you'll have to evaluate him at each level. Informal evaluations should be done on a regular basis; formal evaluations should be done when the apprentice feels he's ready to move to the next level. The test should be comprehensive and as real-world as possible. Rather than asking for definitions or explanations, have the apprentice perform specific tasks. In other words, design word problems for the employee.

The point of this evaluation is not to assign a grade; in fact, the formal evaluation should be a pass/no pass situation. In moving the apprentice from Level 1 to Level 2, you're serving notice that he is held responsible for the information contained in Level 1; you don't expect to have to explain rudimentary concepts anymore. Because the evaluation is designed to evaluate the apprentice in the real world, it should be 'open book, open note.'

DOC Apprentice Goals

Level 1

Estimated Time to Complete: 5 - 7 Weeks

FUNCTIONS

TDP
- Use formatting commands to manipulate documents
- Insert running headers/footers
- Use the spellchecker to proof a document
- Make modifications per markup on existing documents
- Use jobs to final/print documents

3000AD
- Work with accounting (A/P, A/R, G/L) software

Equipment
- Make duplex photocopies
- Add toner to the laser printer
- Be able to transfer phone calls
- Send/retrieve faxes
- Retrieve a printout on greenbar
- Understand how to use/place fonts

Administrative
- Request supplies
- Prepare shipments for Universal Reproductions
- Receive shipments for Universal Reproductions
- Ship custom doc to customers
- Be able to post messages in the mail system

CONCEPTS/TERMS TO UNDERSTAND

- AR terms
- Agings
- Alternate character set
- Assets
- Automatic payments
- Backroom processing
- Balance forward
- Balance sheet
- Batch entries
- Budget variance
- Buying period
- Chained payment
- Chart of accounts
- Check reconciliation
- Check register
- Credit account
- Credit memo
- Current month
- Customer statements
- Debit account
- Debit memo
- Division
- Final checks
- Fiscal year
- Font
- GL Distribution
- General ledger
- Gross amount
- Handwritten checks
- Inactive account
- Income statement (P&L)
- Invoice
- Journal
- Lead time
- Liabilities
- Line printer
- MAD
- MPEX
- Month end
- Net amount
- Normal character set
- Paid detail
- Partial payment
- Past Due
- Payment on account
- Posting
- Preliminary checks
- Prior month
- Purchase order
- Recurring checks
- Recurring entries
- Report formats
- Report selection
- Reporting groups
- SPOOK
- Standard list
- Trial balance
- User Defined Command (UDC)
- Year end
- Zero balances

FIGURE 2

MPE COMMANDS

Abortjob	Listf,2	Showtime
Altjob	Redo	Stream
Breakjob	Resumejob	Tell

TDP COMMANDS

@	CENTER	PAGE
^A	CHANGE	PRINT
^F@	COPIES	QUIET
^N	FONTID	REDO
^S	FOOT	RESEQUENCE
^U	FORMAT	SPACE
^W	HEAD	SPELL
^-	IMAGE	STREAM
ADDLINE	INDENT	TOP
ADDSINGLE	LIT	UL
BOTTOM	NEW	UW
	NONLIT	

UDCs

ERASE	ME	SJ
GO	PRNT	TDP
HI	S	UPS
MAIL	SA	XPRINT
	SHOW	

FIGURE 2

If the apprentice doesn't pass the exam the first time, there may be several reasons. One, he may simply not have given himself enough time to learn the concepts. Two, he may need specific work in a certain area, but this is a one-time shortfall and won't be repeated. Three, he may not be the right candidate for the job. Four, the exam may not be a reasonable evaluation of the tasks. Take time to review it. Certainly not passing the exam indicates that extra monitoring is warranted.

Determine the Pay Scale

One of the most powerful incentives behind an apprentice program is an earn-while-you-learn potential. An apprentice at Level 1 is not as valuable to your organization as an apprentice at Level 4. If possible, structure your program so that as the apprentice passes from level to level, his pay increases as well. This enables you to bring the apprentice on at a relatively low salary (since he is presumably not so valuable to the firm) and increase his salary as his worth to the organization increases. This reward system also encourages timely completion of the apprentice process.

Finding the Right Candidate

Once you've established your apprentice goals, developed an evaluation program and determined any salary incentives, you're ready to hire. But what sort of employee will do well with the apprentice approach?

Because of the wide variety of tasks and the number of concepts you're requiring the apprentice to learn, your apprentice should possess higher than average intelligence. This aptitude needs to be accompanied by an eagerness to learn. You want someone whom you can train quickly, and who wants to learn. Interest in a career rather than a job is key. You need someone who's willing to stick out the apprentice program and stay with you after the training. They need to be sold on the company and their role within the organization as well as on a regular paycheck.

Self-motivation is a must, as well. While the apprentice will be monitored, you need a self-starter, someone who is able to push himself (and you) to learn at as efficient a pace as possible. Since not all environments are conducive to the salary incentive, you'll need someone who can be motivated by incentives other than money, too.

For these reasons, recruiting recent college or technical school graduates can be useful. Such recent students are already used to learning prescribed items and have recently developed study habits. The evaluations resemble tests, which the students have also learned to expect. And in order to finish school, students have to be self-motivated. Finally, because they lack the experience to enter into more advanced positions, recent students can be hired at low salary levels.

Managing the Process

Commitment by the organization to the apprentice process is key to the program's success. Senior staff members must not only be given the responsibility for training the apprentice, but also the time. Interdepartmental training should be made available in order that the apprentice have access to as many experts as necessary.

Securing one senior staff member to serve as mentor to the apprentice is important. Select a staff member who understands, at some elementary level, all the goals you've selected for the apprentice. Ideally, this staff member should be in the same position that the apprentice is learning; e.g., senior technical writer, senior programmer/analyst. Monitor the senior staffer to ensure that the apprentice is offered enough guidance to accomplish his goals, but not so much that the learning process is curtailed.

Encourage questions by the apprentice and teach him early on where to go (or to whom) for answers. With an organization committed to the process, the apprentice should have ready access to experts. Monitor the experts as well, though, to make sure that the apprentice is not offered more information than he can handle. For instance, if an apprentice is at Level 2, make sure that the staff members with whom he works do not talk with him using Level 4 terminology.

Regular feedback is important, both from you to the apprentice and from the apprentice to you. First exams ought not to be flunked because the apprentice ought not to take them until he is ready. If the apprentice is doing well, let him know. If he seems to be having trouble, talk with him. Encourage him to let you know if the schedule is too heavy or too light, or if some goals could be better accomplished at a different level.

Benefits

The benefits of implementing an apprentice program can be significant. A well-trained employee, versatile enough to work on different tasks and knowledgeable enough to perform them well, is a valuable asset. You can instill confidence in his ability to perform his job by introducing the employee to the job duties gradually, encouraging him to take on additional responsibilities only when he (and you) are convinced he's ready, and offering a structured, objective approach to the position can instill Such confidence can lead to improved performance in the long run.

Apprentice programs also ensure that new employees are not cast adrift without direction. By assigning a senior staff member to work closely with the apprentice, you help to establish a close working relationship early on in the apprentice's career. The senior staff member can introduce the apprentice to others in the company and serve as an ally in an otherwise unfamiliar environment.

Because the apprentice program uses objective criteria for evaluating employees, employees can be compared to each other. Of perhaps greater importance is the illumination of strengths and weaknesses that such objectivity provides. Responses to the performance evaluation can point out such differences as one writer excelling at press releases while another has a knack for technical documentation.

Using senior employees can also give otherwise nonsupervisory personnel the chance to supervise. While not directly responsible in all cases for the apprentices evaluations, the interaction between the senior staff member and the apprentice can help the senior employee develop management skills.

The primary benefits then, come down to these: structured, objective training process; easy assimilation into the company culture; identification of strengths and weaknesses across employees; and supervisory opportunities for senior employees.

Pitfalls

Of course, there are problems that can crop up in any training program, and the apprentice program is no exception.

Make sure that everyone in the organization understands the goals of the program: grooming a well-rounded and knowledgeable individual in the ins and outs of his position. If everyone isn't fully committed to the project, the apprentice will have a difficult time getting the education and help he needs to be successful.

Vague goals and objectives can be disasterous; that's the reason you should run the risk of being too specific in reference to commands and tasks. Don't assume that the apprentice knows how to perform a task until you've seen him do it. This saves you and your apprentice frustration.

Ineffective or infrequent feedback can cause the apprentice to lose faith in the process. Make sure that you know where the apprentice is in the program and how he is doing. If the employee is unsure about a concept, spend extra time with him. On the other hand, if he is doing extremely well with the program, let him know.

Don't let your program get off to a roaring start and then fade. Once you've set an employee on the apprentice program, be sure that you and he are able to complete it. The effect on morale of starting a program that eventually dissipates can be devastating.

Summary

Well-planned apprentice programs offer an organization efficient training efforts designed to develop highly versatile and motivated individuals. Ideal for companies whose internal procedures are already in place, the apprentice approach involves the entire organization in the successful assimilation of new employees. These programs depend on open communication for success, and require a mandate from the highest levels of the organization.

Successful programs can decrease training time, increase employee morale, and result in a highly efficient and productive environment.

THE MIS PROFESSIONAL AS IN-HOUSE MANAGEMENT CONSULTANT

James Call
The NPD Group
900 West Shore Road
Port Washington, New York 11050

The skills which MIS professionals routinely use in their traditional role can often be effectively applied to a broader range of management problems. These skills include:

- A mindset to look at a problem logically
- Project organization and management skills
- Technical savvy and resourcefulness

Objectives And Benefits Of The In-House Consulting Role

The objectives of taking on projects which have a consulting or a "business analyst" flavor include:

- Being more proactive in projects. By getting involved at an earlier stage and by participating more in the initial problem solving process, there is better prioritizing of tasks and activities.
- Developing personally and professionally.
- Benefiting from seeing the big picture instead of a problem which has been described in terms of limited parameters.
- Being able to tackle projects which might be extremely beneficial to the company as a whole, but which are not viewed as critical by any individual department.

A management consulting approach to projects can lead to some very exciting work. However, this is serious business and we should work hard to assure that the approach does not degenerate into simply a "dilletantey" exercise.

Appropriate Project Areas

Business problems most appropriate to MIS attention in terms of an expanded consultancy role need not necessarily involve programming or traditional systems analysis. A "computer" solution might or might not be recommended. Examples of appropriate project areas would be:

- Disk space utilization, including developing administrative policies for disk management.
- Paper flow efficiency studies and paper flow reduction.
- Streamlining clerical workflow.
- Optimal machine set-up strategies in factories, including issues of optimal lot sizes for production runs.

We should also note that it is never appropriate to take on engagements where the MIS "Consultant" is not professionally prepared to address the issues. For example, it would normally be inappropriate for an MIS specialist to evaluate financial internal controls, which should be done by a certified public accountant.

Being An Effective Internal Consultant

The next section of this paper summarizes suggestions for being an effective internal management consultant. These approaches have been found useful in numerous projects completed successfully over the past 20 years. (Additionally I should credit Mr. Robert Gibbons, a professional management consultant and sometimes mentor who contributed to the list.)

A list like this can be misleadingly simple. It's obviously easier said than done. Nevertheless, here are some things to keep in mind. They are categorized into the major functional areas of a typical project.

TIPS ON BEING AN EFFECTIVE INTERNAL CONSULTANT

Project Beginning

1. Define the purpose and scope of the project in specific and unambiguous terms. Resist the temptation to let the user redefine the project objective and scope in midstream.

2. "Walk through" the departments and areas of the company which are involved in or affected by the study.

3. In conducting fact-finding interviews, explain to the department head and to the person being interviewed what the purpose of the project is.

Fact Finding

1. Be a good listener.

2. Picture in advance what your final report will involve. This will allow you to fill in any holes while fact finding is still in progress. It can help you organize your data logically as it relates to problems and conclusions.

3. Obtain as much background information as possible. Include relevant copies of key documents such as schedules, project and production reports. Try to obtain examples which illustrate the problem.

4. Make detailed notes during the interview, recording short phrases about things which might have a bearing on the project.

5. Interview the person actually doing the work when possible. A supervisor may see only parts of the problem or have a distorted view.

6. Probe without anticipating the answer. Don't ask "Do you use a LISTF to manage these files?" Ask instead "What outside information, if any, do you use to manage these files?"

7. Become aquainted with the physical layout of the area, the environmental features (cluttered? neat?), lighting and noise. Consider effeciency of space usage (square feet as well as cubic) and access to needed equipment and facilities i.e. walking patterns, location of supplies for storage and usage.

Defining the Problem

1. Keep asking yourself whether a given finding is a sympton or a problem.

2. Keep in mind that the job of the business analyst will often be to present available data in a different manner than it has normally been seen. By doing this you can often demonstrate what the problem really is. Ask yourself what data could give insight to the particular problem.

3. Challenge explicit and implicit assumptions. However, do this constructively and tactfully. Never criticize the user.

4. Resist the temptation to assume personal responsibility for the user's problem. Your job is to assist the user in identifying solutions and implementing changes, but you must not take the ultimate responsibility away from the user. To accept such responsibility is even unfair to the user and you will never be able to disengage.

5. Maintain an optimistic and cheerful outlook. Avoid cynicism. Obviously things need improvement. If everything were perfect, the user would not need a consultant.

6. Review preliminary findings (note: findings, not recommendations) with affected persons to give them a chance to critique your understanding of the situation.

Developing Solutions and Recommendations

1. Consider eliminating an entire function or individual steps. That is, sometimes you can eliminate the problem rather than solve it.

2. Stay within the project scope. Fight the tendency to be sidetracked.

3. Let the user be the main participant. Moreover, the user most likely has the key to the solution.

4. Exploit your lack of knowledge of user processes. You can't expect to learn what the user (perhaps with years of experience) already knows. However, your mind is a clean slate. Parts of your lack of knowledge can be turned to advantage because it allows you to objectively challenge each assumption anew. You have a uniquely new perspective.

5. Identify the user's need for missing information.

6. Step back from the problem. Ask yourself what are the key factors or key variables that do or can have a major impact on the problem. For example, look at:

    1. Work load characteristics - transaction volume by time of day, day of week, week of month, seasonals, etc.

    2. What is truly the scarce resource and what affects or determines it - internal and external.

7. Keep an extremely open mind. Anything (moral and legal) is fair game as a problem solution. At the cognition stage there should be absolutely no preediting.

8. Let your imagination go as far as you can carry it. This can lead you to solutions which would never have been considered with a narrower view point.

Identifying Benefits

1. Be alert for side benefits and discoveries. While you must stay within the project scope, in most work you will almost trip over additional areas of opportunity. Expect some of these to be significant, perhaps even far surpassing in benefit the original project. When you discover these, note them for the user to implement or set up as a future project.

2. Use common sense.

3. Apply an 80/20 rule. That is, you can often achieve 80% of the total possible benefits by addressing the "worst" 20% of the problem.

Communicating Results

1. Document your work in a written report with sections as shown on the next page.

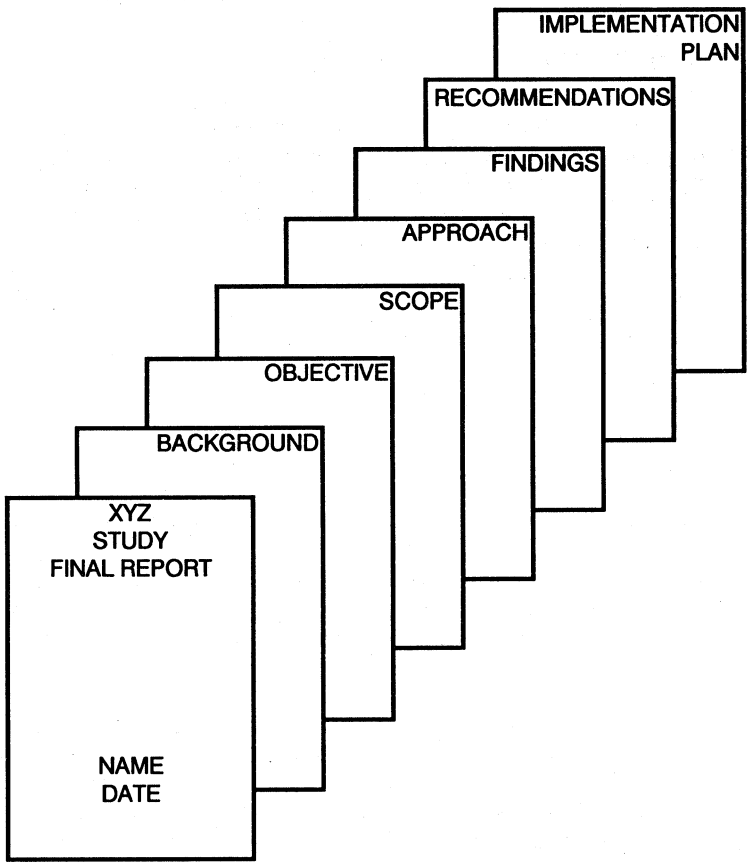
   In addition to the report, you will usually end up with "workpapers", which are the various relevant documents you collected. These workpapers should be kept well organized as you proceed with the project. You will find them invaluable as a reference.

2. Maintain strict confidentiality with respect to user comments. That is, users should understand that you want to utilize their ideas and suggestions in your findings and recommendations, but you must not attribute them when they are personal or politically sensitive. Users must be free to share controversial ideas with you in certain confidence they will go no further, including upper management. (This would not apply, of course, to discovery of unlawful acts, fraud, etc.)

3. It is often more difficult to disengage from a project than it is to get started. Emphasize resolving open issues, documenting your findings, and making decisive actionable recommendations.

4. Try to be a catalyst to bring out and amplify the user's contribution to the solution.

5. A well written report should make sense to a previously uninformed reader, but should be written tightly, without extra words.



AN EFFECTIVE STRUCTURE OF A CONSULTANT'S REPORT

GROWING A PROJECT MANAGER

R. D. Lacy & J. R. McDonnell
Pacific Bell
2600 Camino Ramon, Suite 1E450
San Ramon, California 94583
(415) 823-0992

ABSTRACT

You have been a successful programmer, designer, database administrator or system analyst. Your past successes have now elevated you to the illustrious level of project manager. You naively accept the position thinking, "Wow, I am in charge of my own project"; while management wonders how to structure the assignment so the project succeeds and a project manager is developed.

In this paper, we will share with you the growth and development of a new project manager. We will identify some of the common pitfalls that a new project manager encounters and will suggest ways to overcome them.

GROWING A PROJECT MANAGER

R. D. Lacy & J. R. McDonnell
Pacific Bell
2600 Camino Ramon, Suite 1E450
San Ramon, California 94583
(415) 823-0992

THE SITUATION

Mary had just given John the opportunity assignment of his career. The chance to be the project manager of his own project. John was enthusiastic and could not wait to start. Mary was apprehensive knowing the company could not afford for the project to fail.

John and Mary are data processing professionals who work for a Fortune 100 firm developing business computer systems. Mary has worked in the software development arena for over ten years and has a comprehensive background in the field. She has participated in numerous successful projects and has learned how to solve the dilemmas project managers experience when effecting change in an organization.

John was new to project management. He had been a successful database manager, systems manager, and systems analyst. He had taken the appropriate classes to familiarize himself with the tools of project management but had never managed a project. After accepting this opportunity assignment, Mary asked John when he thought he would have the project completed? John's project management career had begun.

Through John and Mary's experiences, we will discuss the growth and development of a new project manager, identify the common pitfalls that are experienced, and how they can be overcome. We will illustrate how a seasoned project manager can facilitate success without compromising the new project manager's role or allowing the project to fail.

THE PERFECT PLAN

Spirits were running high. John had some great ideas on what tasks should be completed and how the project should be done. After all he had successfully performed many of these individual tasks before and he did not even use a plan. Since the new project plan was his first major deliverable, he wanted it to be the best plan ever!

John and Mary reviewed past project plans and agreed that his plan would contain one concise paragraph articulating:

* The business goal for undertaking the project
* The project objectives that would achieve the business goal
* A strategy for reaching the project objectives
* Who would be affected by the resulting system
* The risk and assumptions involved
* An estimated schedule and cost

In order to produce the project plan, John needed to analyze the project's impact and importance. John was ready to jump in. Six short paragraphs and the project plan would be complete! Systems analysis work had always been easy for John and project planning did not seem that different.

From previous conversations with Mary, John was pretty clear on the project objectives. However, he was having trouble understanding the project business goals. He had always worked in data processing and never paid much attention to business goals. The annual company business goals publication seemed too remote to be applicable to his project. Stuck, he sought Mary's advice.

Before Mary turned the project over, she spent considerable time reviewing the project goals and history with John. She remembered how excited he got when he realized that he might resolve a long standing nasty file maintenance problem and that he might get to program it using one of the new languages. She was not surprised he was having trouble.

"The easiest way to approach a project is to take the view that you are the owner of the company," advised Mary. "Any money spent on the project is money out of your checkbook. So why would you write a check to fund your project? The answer to that question is the project business goal." John walked away muttering.

A few days later Mary stopped by John's office to see how things were coming along. "I'm almost done," John said. "Come by tomorrow at 3 pm and I will show it to you." The next day Mary was in John's office excited about reviewing the plan. John pulled a thick folder out of his desk, turned on the overhead projector, and dimmed the lights. Twenty multi-color slides later, he completed the presentation of the project plan. Her eyes still smarting from the lights going on and a little sleepy from being in the dark for two hours, Mary questioned John about slide one, the project goals and objectives. Satisfied that she and John had a clear understanding of the project goals and objectives, Mary commended John and his team for the hard work they had done on the plan. She approved the project proceeding into the requirements phase. John was triumphant! He and the team had successfully produced their first project deliverable!

Before Mary left John's office she stopped in his doorway and said that she had two little questions. "John, what happened to the six concise paragraphs and what are you going to do when the plan changes?"

John, clutching the 400 task project pert chart, exclaims "Changes? But Mary you just approved the plan!"

John had fallen into a common trap that a new project manager often encounters - the perfect project plan. Because the plan was one of the first deliverables, much care and time was taken to ensure that the right words were chosen, thoughts were clear, each "T" was crossed, and the i's

dotted. Surely, presenting a nicely packaged, completely detailed document to management for approval would demonstrate that the right person was chosen for the job! Mary had only questioned John on slide one of his presentation, the goals and objectives of the project. To ensure the project started out right, it was essential that these were clearly understood, not the 400 task pert chart.

John did not realize that project planning was only complete at the end of the project, never at the beginning when the greatest amount of uncertainty exists. Project planning is like mapping an expedition through an unexplored mountain range. You need to know where you are starting from and where you want to go. The initial route laid out to arrive at the destination will not be the final route. New passes that expedite the journey will be found and unforeseen boulders will block the way.

However, project planning is not completely fluid. The business goals for undertaking the project and the project objectives should not significantly change throughout the life of the project. If they do you have a new project and need to start over. This is analogous to changing the destination of your expedition through the mountains in the middle of the trip. What changes the most are the details on how to achieve the goals and objectives. The road to reach your destination may change but not the destination itself.

AM I DOING IT RIGHT

John was happy that the project plan was approved, but was apprehensive that there might be more to project management than he had anticipated. He knew there were books on the subject and had attended some classes. His new burden of responsibility for the project **and** the project team caused him to be concerned that he might not be completing the project in accordance with the established standards. He asked Mary if there was a project management check list that he should be following. According to Mary, there were tons of books on the subject, but, John did not have time to read books. He only wanted to complete his project on time and do a good job. Why Mary consistently clipped articles, dug out books and put them on his office chair, he did not understand. Didn't she know he did not have time to read this material now in the middle of his project! None of

the material she provided looked like the how to's or rules he was hoping to follow.

John's desire for a structured project management check list was a reflection of his desire to have his thoughts and actions validated. Mary was not helping calm John's fears. Each time John came up with what he thought was a good idea Mary only asked where it fit into his plan. John was seeking confirmation that his ideas were good and that she thought he should proceed; she just kept asking about the plan.

Mary's tendency was to solve problems and provide answers when asked. However, Mary was growing a project manager, and her role was to place the responsibility for making the decisions on his shoulders. As long as the project goals and objectives were not being compromised Mary would not be involved in the project decision making. This forced John, the new project manager, to utilize his team and develop his own decision making skills.

IT'S MY PROJECT

As the project headed into the design phase, the details that John had considered minor came back to haunt him. John started to get questions from individuals who seemed to be minimally impacted. "John, how will the system get this data element?" "How will we update this field, John?" "John, how often do we need tapes and from whom?" "Who will perform the maintenance functions, John?"

These questions were not new. They were questions that were asked early on but not answered because they were related more to design than requirements. Now that the project was in the design phase, John felt he should be in the position to provide workable solutions. He knew he could ask Mary; but this was his project. She would think he couldn't handle it if he kept asking her advice. He could ask Dennis, the chief programmer; but Dennis already assumed that John was an inexperienced project manager who was bent on over-committing the programming staff to an already unreasonable time line. Surely he could not confirm Dennis' thoughts! John was at a loss for alternatives and the project due date was drawing nigh.

Every two days, Mary would stop by John's office and pose her standard question "John, what's new and exciting?" John would share his frustrations with Mary and then sit back and wait for her response. He got no response. At least not the response he was expecting. Instead, Mary followed up with another standard reply "Well, what are you going to do next?" John, frustrated even more by that fact she did not provide "the answer," would respond with "Oh, I'll think of something."

Mary's goal was to help John realize that success hinged on how well he communicated his need for assistance to the rest of the team. John had to learn how to work through the team to accomplish the project goals, meet the client's needs, and resolve open issues - not in a committee of one. John had fallen into a familiar trap - ego. He had experienced success and was confident that he could deliver no matter what the users and programming staff said. HE was in control.

After a week of frustration and the realization that his next due date would be missed if he did not solve the issues at hand, John bit the bullet. He asked Dennis what it would take to do exactly what the users wanted and still meet the due date. To John's surprise, Dennis already had workable solutions and alternatives mapped out. He was just waiting to be asked!

Dennis admitted he needed extra resources to perform certain programming functions in order to meet the due date. It was this concrete information that enabled John to present the team's needs to Mary and obtain adequate resources for the remainder of the project.

John learned a very valuable lesson: The short cut to reaching the desired goal lies in one's ability to lay ego aside and seek the assistance of others on the team - not in the ability to solve all problems yourself. To sum up John's thoughts:

- Let the right people know what obstacles are in the way
- Involve the individuals in planning and problem resolution that are most impacted by the change
- Seek the advice or assistance of experienced staff members
- The appropriate time and place for boosting one's ego is after the project is successfully implemented

FALLING BEHIND

The project proceeded smoothly through the design phase. But in the build phase, the project began to stumble. The first two modules that were built were late. To make matters worse module A passed data to module B in a format that was not readable by module B. John was not sleeping well at night. Mary was beginning to worry.

"John, how often are you having project meetings?" inquired Mary.
"Well, we have project meetings once a month," replied John. "We review the weekly tasks that were scheduled and discuss any project issues. But we still seem to be falling behind and the design seems to be coming apart."

John had fallen into another project management trap. He had mismatched the units of time for the design and build tasks, with the project meetings. The design and build work was set up in weekly units of work, but he was only pulling the project team together once a month to review the project's status. To a large degree the rate of progress of a project is directly proportional to the frequency of the project meetings. If the project meets only monthly, then the project team tends to think of its units of work in terms of months. The project then moves ahead or falls behind in units of months regardless of what was initially planned.

On John's project this was a very serious problem. Fortunately it was a very easy problem to fix. "John, why don't you start having project meetings weekly instead of monthly?" inquired Mary. "I don't really want to take up that much of the team's time in meetings," responded John. "Our monthly meeting takes 2 hours as it is." "Then set a twenty minute time limit on your weekly meetings and just focus on what the goal is for that week," encouraged Mary. Skeptically, John agreed to try it; but right then he would have tried anything !

HE'LL SOLVE ANYTHING

The system was in production and implementation was well underway. The data maintenance staff was beginning to complain about the users and the users were complaining about the maintenance staff. Both compiled lists of problems and gave them to John to resolve. Initially, John did not mind providing a quick answer to the questions or clarifying a process. However, after several weeks of receiving problems daily, he had accumulated a relatively large stack of work. Ten users and five members of the maintenance staff were waiting for him to solve their problems.

One afternoon, Mary walked by John's office and saw him pouring over printouts while trying to pull information from a microfiche reader. Mary was afraid to ask what he was doing, but she led in with her standard question, "What's new and exciting?" John muttered something about trying to figure out a better way to teach the maintenance staff to work out the daily problems. John continued, "Do you see this stack of work they've given me? I don't know when I'll get to the bottom of it. I don't know why they keep coming to me anyway."

John had stumbled into another project management trap. The implementation of John's project resulted in changes to the way the business functions were performed. What John did not understand was that whenever processes change in an organization, it is common for the individuals impacted to continue to seek the advice of the people who

initiated the change or pass the responsibility for resolution of the problems on to someone they could rely on to solve it. In this case, John happened to be the person who fit both molds.

John initially accepted their questions by telling himself the process was new and he was better able to answer the questions immediately. He was helping solve their problems utilizing the system his team built. It was great fun and very satisfying - for awhile. But when he received a complaint because he did not respond soon enough to a request, he knew he was in trouble.

John complained to Mary that the users and the maintenance staff kept coming back to him for help and that he was going to need additional resources to fulfill their needs "John during the months of developing the project you trained the user and maintenance teams to come to you," advised Mary. "During that time, you had all the answers. Before they give you another problem to solve, ask yourself if you are solving the right problem."

Mary's coaching helped John realize that he was the wrong person to solve the users' and the maintenance staff's problems - even though he knew the answers. Now his role was to help the users and the maintenance staff work together to solve their own problems. John's involvement in the problem solving was only prolonging their problems. His project was completed and he needed to realize that.

POST-IMPLEMENTATION BLUES

"Well John, What's new and exciting?" "Not much I'm afraid, everything seems to be fine," replied John.

John was depressed. Hanging on his wall was the project Gantt chart. The long rectangles representing project tasks were all blackened signifying completion. John's phone had not rung in two weeks. John was suffering from post-implementation blues. The system was a success and it would live on without him.

"Hey John let me tell you about this new project."

CONCLUSION

Through the experiences of John and Mary we have tried to share with you the growth and development a new project manager will undergo. We have identified some of the common pitfalls that a new project manager will encounter and have suggested ways to overcome them. None of the pitfalls that beset a new project manager are technical; they are all social in nature. Being responsible for your first project and project team can be burdensome for a new project manager. Proper training and guidance will help ensure the success of the project.

REFERENCES

Block, R., *The Politics of Projects*, Yourdon Press Computing Series, Prentice-Hall, Englewood Cliffs, New Jersey, 1983.

Boddie, J., *Crunch Mode*, Yourdon Press Computing Series, Prentice-Hall, Englewood Cliffs, New Jersey, 1987.

Brooks, F. P., *The Mythical Man-Month*, Addison-Wesley Publishing Company , Reading , Mass. 1979.

DeMarco, T., & Lister, L., *Peopleware: Productive Projects and Teams*, Dorset House Publishing, New York, New York, 1987.

Mattson, R. R., *Software Quality, Let's discuss this ""Can of Worms"*, INTEREX, the International Association of Hewlett-Packard Computer Users, Proceeding of the 1988 Conference of HP Business Computer Users, Orlando, Florida, August 7-12, 1988, Volume 1.

Mattson, R. R., *Secrets of Software Project Management*, INTEREX, the International Association of Hewlett-Packard Computer Users, Proceeding of the 1985 International Meeting - HP3000 Users Group, Washington, D.C., September 8-13, 1985.

Mattson, R. R., *Why System Projects Don't Quite Succeed*, INTEREX, the International Association of Hewlett-Packard Computer Users, Proceeding of the 1988 Conference of HP Business Computer Users, Orlando, Florida, August 7-12, 1988, Volume 1.

Page-Jones, M., *The Best Policy is missing Deadlines - Sometimes*, Computerworld, September 30, 1985.

Peters T. J., & Waterman, R. H., *In Search of Excellence*, Harper & Rowe, New York, New York, 1982.

Thomsett, R. T., *People and Project Management,* Yourdon Press, New York, New York, 1980.

Thayer, R. H., *Tutorial: Software Engineering Project Management,* Computer Society Press of the IEEE, Washington, D. C., 1988.

Weinberg, G. M., & Gause, D., *Are Your Lights On? How to figure out what the problem really is,* Winthrop Publishers, Inc., Cambridge, Massachusetts 1982.

Weinberg, G. M., *The Secrets of Consulting: A guide to giving and getting advice successfully,* Dorset House Publishing, New York, New York, 1985.

Weinberg, G. M., *Becoming a Technical Leader,* Dorset House Publishing, New York, New York, 1986.

SUPPORTING YOUR SUPPORT STAFF

Debby Shermer
PROTOS SOFTWARE COMPANY
300 West Fifth Street, Suite 935
Austin, Texas 78701
512-480-0865

Technical support staffs have been around as long as computers. As the computer has become more advanced so have the users and in turn the technical support needed. This paper will explore four questions:

1. Why is it important to support your support staff; more generally, why is your support staff important?
2. What type of person becomes a support technician?
3. What does the support staff need?
4. How can you give them what they need?

The viewpoint is that of a Manager of Customer Support but the answers can be applied to both customer support and support of the in-house end user.

WHY IS IT IMPORTANT TO SUPPORT YOUR SUPPORT STAFF?

One answer to this question comes from an October 1987 article by Robert E. Kelley entitled "Poorly Served Employees Serve Customers Just as Poorly". The article was in the *Manager's Journal* of the *Wall Street Journal.* Kelley states that the service staff will "treat customers similar to the way they, as employees, are treated by management" and that in many companies "management treats employees as unvalued and unintelligent. The employees in turn convey the identical message to the customer." As a solution to this problem Kelley makes the point that "if managers want to improve service quality they must treat employees the same way they want employees to treat customers." But what of the company where employees are treated well, some might say even like royalty, and customer service is still not good? Does that mean Robert Kelley was wrong? No. What it means is that while good treatment of employees is essential to good customer service, management may be giving the support staff what it thinks the support staff needs, rather than what the staff really needs.

The support staff may be the only part of your company that the customer has contact with; in fact, the support staff may represent your company as much as the

product does. The better the support staff is, the better image the company will have in the eyes of the customer.

WHAT TYPE OF PERSON BECOMES A SUPPORT TECHNICIAN?

This can really be divided into two questions. First, who becomes a support technician, and second, who should be a support technician.

Customer support staffs usually are comprised of young people without a lot of training and education. This is because the salary is usually one of the lowest in the company. The position of customer support technician is not considered to be a very prestigious position. By the time these technicians are trained, they have become disenchanted with helping customers and decide to change positions for a higher salary or more "important" position. They are replaced with other young, undertrained people and the cycle continues. This cycle represents the frequent turnover seen in many support staffs. This frequent turnover can mean either being understaffed or using staff that is undertrained. Either way the customer is "underserved."

A good customer support technician does not become a support technician because he could not do anything else. To provide support he must be a jack of all trades and a master of several. He needs to be able to provide answers at all levels of comprehension and for questions that the customer isn't even sure of. The best qualified support technician will have used one or more of your company's products, be technically sound, like people, and have a background in education or customer service.

If the customer support technician is trained in both customer service and technically in the area he is supporting, the trick is to keep him in the position. A customer support technician is a "burn out" position. Why is it a "burn out" position? Because the needs of the technician are not being met or acknowledged. Supporting your support staff is vital unless you want frequent turnover.

WHAT DOES THE SUPPORT STAFF NEED?

Support for the customer support technician (referred to as technician for the remainder of the paper) should be provided in the areas of technical assistance, communication, empowerment, office space and equipment, background support, training, adequate staff levels, and diversions. Technical assistance is the most critical, followed by communication, and then empowerment. These are followed by office space and equipment, background support, training, adequate staff levels, and diversions, which have equal importance.

Technical Assistance

The technician is under the gun to provide a fast response to the customer. On the occasion when the technician does not have the answer and cannot find it with his

resources, the answer must come from another person in the company and it must come fast. This is where supporting your support staff is the most critical and the hardest to accomplish.

In order to provide that needed support, develop an order of people for which the technician can go to for answers. Do not put as the first person someone who probably will not have any answers. This will only waste everyone's time and slow down the response. The person who can answer the question will probably be also the busiest. This is unavoidable. Remember that he is only busy because the company has customers. Schedule time for questions each day, 15-30 minutes at the end of each day would be ideal, but also allow for interruptions during the day because some problems can't wait until the next morning for a response. The technician will interrupt your day, this is true, but remember that helping a customer is not a minor problem. It is very important. If you didn't have customers, you wouldn't have a company. The technician isn't interrupting because he has nothing else to do, but because he have no place else to go for the answer.

As soon as a customer feels that his needs are not being attended to with the proper speed, or that the technician does not have the proper knowledge or access to the knowledge, his problem will be escalated quickly to a more urgent level. Any time a problem escalates, it costs the company money and the customer's good will.

Communication

For a technician to represent the company, that technician must know what the company is doing, what policies it has, and what has been made public to the customer base.

If there are any changes to the product line, services, literature or even slogans, the technician should be aware of these before the customer is. The technician should not learn about a new product feature from the customer, but from his management. Missing information means missing customer support.

In Linda M. Lash's book *The Complete Guide to Customer Service*, a story of what poor communication can cost a company was related. It seems that in 1981, a major car rental company distributed many new coupons for advertising and customer incentives. The coupons were each a work of art. The conditions and instructions printed on the coupons were a disaster. The customers could not understand them. Worse yet, the service staff didn't understand them. Had they not read the instructions? How could they? There were no instructions available; the service staff was told to "just read the coupons." Needless to say, the coupons caused a great deal of customer (and staff) dissatisfaction. Customers were forced to wait a long time to pick up their cars because of all the coupon confusion. Some customers did not wait, and many who did wait chose another company for their next trip. Clear, concise instructions and communication would have avoided this problem and would have cost much less in the long run.

The customer may be able to see if the company has poor internal communication. This may be true, but the customer should not see it.

Empowerment

Empowerment is an old tool which is starting to be used with a great deal of success in customer support. Empowerment, put simply, means giving the employee that deals directly with the customer the authority to give that customer what he needs. This does not mean giving carte blanche to the technician to give away all the company's money and resources. Empowerment can, and should, have some limitations. Because of the authority, most employees will also feel a strong sense of responsibility for the company's profits because of the trust placed in them.

An example of how empowerment can be useful is the situation in which a customer needs a new tape or a new manual and needs it as soon as possible. If the technician has no empowerment, it could take several days to get the approval to send the tape and another day to get the approval to ship it by Federal Express. The tape that was needed as soon as possible arrives one week later. A technician that has been empowered would have the tape there the next day and the company would actually have saved money.

Office Space and Equipment

The technician's office space does not need to be a palace but it does need to be functional. Noise will harm the technician's ability to concentrate on a problem and communicate with the customer. Also, the noise generated by the technician will interfere with other people's work. If possible provide a separate office for each technician, but do not seclude them from the rest of the technical staff. Remember, the technician is a part of the technical staff and may need to rely on its help occasionally.

For support work done over the telephone, the telephone should be equipped with a good quality headset. This frees up the technician's hand to look up information, use the computer, or just write a note about the call's content. The headset will also relieve a great deal of physical stress on technician's neck and back.

All necessary resource material should be within reach. It slows down response time considerably when the technician must search for the needed information or equipment. If the technician will be dialing into the customer's computer, a good quality modem and a PC with a hard drive are necessary.

Provide an online tracking system for quick solutions. If the technician is able to access information on how this problem was solved last week or last month, the customer will receive a timely answer. The online system should also include information about the customer himself. People feel important if they are remembered. They don't have to know that it was the computer that remembered them.

Background Support

Background support, or behind the scenes support, should be readily available to the support staff for clerical duties and operations. Letters will need to be typed, and facsimiles will need to be sent. Tapes also need to be produced and mailed. The technician has enough to do without having his day broken up with clerical

or operations work. If made to feel like an imposition to the clerical or operations staff or if turnaround is slow, many technicians will find it easier to do the work themselves. Technicians are not noted for their clerical skills so this may not reflect highly on your company. Other technicians may find it easier to just not do it. This also will decrease the level of service to the customer. Give the technician enough authority to assign work to the clerical and operations staff. If your company is large enough, assign a permanent technical assistant to attend exclusively to the needs of the support staff.

Training

Training is necessary in two areas, first, technical, and second, in customer relations. Technical training without customer relations training is like Hewlett without Packard.

The need for technical training was described perfectly by Linda M. Lash in *The Complete Guide to Customer Service* when she wrote:

> Confidence in dealing with customers comes from knowledge. Technical skills training gives customer contact employees the knowledge to deal with customers confidently. Customer contact employees experience intense frustration facing a customer and not having the answer.

One method of training a technician is to "throw them in the water and see if they swim". Fortunately, the medical profession has not chosen to use this method. Surgeons go through extensive training before ever touching a patient. While surgery and customer support are worlds apart, one slip by the technician can leave a very large wound in a customer that would be costly to heal. Training can be done on the job, BUT an experienced technician should be close at hand.

If your company has many products, do not have the new technician support all products. Let the technician get his feet wet one product at a time. This will allow him to become confident in his ability instead of overwhelmed in what he doesn't already know.

Just knowing how to start a call is something that can be taught. Teaching the technician how to respond to a customer who states "your software doesn't work" or "my program is broken" will shorten the response time and provide a more satisfying exchange for the customer and the technician. Training will cut down on the time needed for each call. It will also make the technician more polished and friendlier to the customer. Both of these are benefits to the customer and the company.

Training should be continuous throughout the technician's tenure at the company. New ideas and methods are developed and attitudes can be reconditioned during subsequent training.

Adequate Staff Level

If response time is 30 minutes most of the time and 90 minutes at peak periods,

you are short staffed during those peak periods. It is understood that people must go to lunch but the customer has paid for service, not excuses. The overload put on the staff that is present during the peak period will cause stress and in turn cause their productivity to diminish.

However, no matter how much a technician loves to do support work, a break would be beneficial. This can be as small as someone else covering his calls during lunch so he doesn't come back to a full desk; this may make him not want to come back. It is hard to relax on a vacation day if the technician isn't sure what is happening to his customers or who is taking care of them. (Yes, technicians will become very possessive of their customers.)

Diversions

In order to relieve stress, a technician occasionally needs diversions. Good diversions are small to medium projects that fit into the area of customer support but do not involve direct customer contact. If possible, avoid assigning large projects or those with critical deadlines. Large projects, when worked on for an hour or less a day, will make the technician feel that he will never finish. Projects with critical deadlines will pose the question of which has a higher priority, customer support or the deadline? If you do assign a project with a deadline, be sure to set that priority in advance.

One diversion would be to provide time for the technician and the customer to interact in a situation not involving a problem. This way they could see each other in a very positive light. Regional user group meeting and international conferences, such as this one, are great places for this to take place. Try not to schedule the technician's time too tightly because the customer may want to discuss a problem, an enhancement, or just chat with the technician. This is very hard to do on the phone because of time and it is important for the customer to feel that the technician is his friend as well as his support. It makes calling in with a problem a less stressful situation.

HOW CAN YOU GIVE THEM WHAT THEY NEED?

COMMITMENT. CONSISTENCY IN THAT COMMITMENT. NO EXCUSES.

You have been presented with several of the many ways you can better serve your customer support staff. If you want your customer support staff to be service obsessed, the entire company from the president to the janitor must be service oriented. Attitude is contagious.

This paper has only touched the tip of the iceberg of support. It has only dealt with the needs of the support staff, not those of the company or the customer. Those topics have been discussed by others.

By supporting your support staff in the ways discussed, you are really supporting your company. The support staff is the customer's window to your company. If that window does not provide for the needs of the customers, they may want to find another company. Remember, customers don't interrupt your business, customers are your business. The support staff represents your customers to the company. Help them represent your company to the customer.

DEVELOPING A TRAINING STRATEGY FOR YOUR STAFF

by Kathy McKittrick
McKittrick Associates
5547 S. Yampa Street
Aurora, Colorado 80015

The ongoing training of your Data Processing Staff has become, over the last few years, a necessary factor in keeping your shop in step with the world of computing. If training has not yet become a fairly significant line item on your yearly data processing budget, your staff's lack of training is probably costing your company money.

Ongoing training was not always the issue that it is today. Fifteen or twenty years ago, programmers and programmer analysts typically came to you trained in the programming language that your shop used. (If they weren't conversant in that language, you probably didn't hire them.) Some initial training involving the operating system commands and "job" statements (or JCL for you ex-IBMers) may have been necessary for programmers not familiar with your hardware. You may have even offered some training on the in-house applications that new staff members would be working on, but that was about the extent of it.

Many factors within the Data Processing community have increased the need for initial and ongoing training. Among them are:

An Increase in the Use of Tools and Fourth Gls - Today, most shops use a wide variety of software tools and fourth generation languages. It is impossible for your entire staff to be proficient in all of these areas. As people get promoted out of their current positions, those moving in to take over their old responsibilities will need to be trained.

Programmer shortages - You probably can't always find a person with the exact experience that you need. With all the choices these days of languages, fourth gls, data base management systems and end-user tools, its unlikely that you'll be fortunate enough to find an exact match. Instead, you will have to look for employees who have good job records and some native intelligence, and train them on the software and hardware used in your shop.

Turnover in Staff - You may have people in your employ right now who are conversant in all of your programming languages, tools and hardware. When one of them leaves, however, you need to be prepared to get a new person up to speed quickly.

Increased Demands in System Development - Data Processing shops typically have a significant backlog of user requests. In order to use a new staff member to decrease this backlog, or to use a current staff member in a new area, it is often cost effective to invest in training that person on the software and tools that he or she needs to get the job done. Otherwise, they might waste enormous amounts of time "fighting" unfamiliar syntax. Remember, programmers and analysts are not known for their love of reading documentation.

Increase in the Use of Packaged Applications - Some people believe that packaged applications require user training only. In most situations, this is absolutely not the case. In order for the Data Processing staff to be able to support the users (by answering questions, maintaining data bases, scheduling jobs and trouble shooting problems) they need to be fully trained in how the application works.

Competing with other Companies for Staff - Most DP professionals consider company-financed training a job benefit. Your training program can be one more way that you can attract desirable employees.

It is important to develop an overall plan for getting and keeping your staff fully trained. Without a plan, training can easily become the "step child" of the expense budget.

DEVELOPING A TRAINING PLAN

The first thing that you need to do in developing a training plan is to develop a "training requirements grid" showing the hardware and software used in your shop, the various job descriptions for your staff, and areas where individuals with those job descriptions will require training. An example of a "training requirements grid" is shown in figure 1.

After developing the grid, you can begin to develop a training program for each job description. Some of the people that you currently have on board have probably been trained in some or most of the areas that are important for them. Even if they've never had formal training, many of them will have learned the hard way; through trial and error.

You should determine which of your staff members are lacking training or experience in which areas. This will help you budget and schedule training for these people over the next year. You may want to distribute a questionnaire like the one shown in figure 2 in order to gather the required

FIGURE 1

	Programmers	Program Analysts	System Managers	Operators	User Trainers
Op System	*	*	*	*	
System Commands	*	*	*	*	
Network Systems	*	*	*	*	
Cobol	*	*			
Pascal	*	*			
Quiz	*	*			*
Omnidex	*	*	*		*
HP Desk			*		*
Accounts Receivable System	*	*			*
Order Entry System	*	*			*

information. This form can also be useful as part of your employment application.

The next piece of your plan is to determine in which cases formal training is necessary and where it is available. In cases where it is not available, you may need to develop an in-house program. We'll talk more about this later.

Where formal training is available, you will need to determine what the training method is. In some cases, classroom training is available. Often large vendors (such as Hewlett Packard) and vendors who are either based in your area, or who have a large number of customers in your area, will have classroom training available locally. If it is not available locally, you will need to add the cost of travel to your training budget.

Many companies have self-paced training systems available for their products. In other cases, third party companies offer classroom training and/or self-paced training systems for some of the more widely used products.

Self-paced training, when its well done, can have many advantages. First of all, it is usually less costly. Many times, you can re-use a self-paced training system, (or at least some components of it), each time a new employee comes on board.

The first step is to determine what is available. If the hardware or software is purchased, you should first call the vendor that markets that software. Even if they don't offer training that will suit your needs, they probably know who does.

In choosing training, there are a number of factors to consider:

Cost - Classroom training usually costs between $150 per day and $250 per day, per student. If air travel is required, you can easily add another $1000 to the bill, per week in class. If you have recently purchased a product and need to train five or more staff members, it is usually more economical to ask for an "on-site" class. On-site classes usually cost $1000 to $1500 per day, plus the instructor's travel expenses. The most economical training medium is usually a self-paced training course. These cost anywhere from $500 to $2000, and you can use them many times.

Training Method - Always look for classes that offer "hands-on" exercises or lab assignments. People retain information better when they've used it while performing a task.

FIGURE 2

Programmer Training Profile

Programmer's Name:_____

Supervisor's Name:_____

Time with Company:____ years ____ months

Programming Experience: ____ years ____ months

Software	Formal Training (Y or N)	Years of Experience
Op System	______	______
System Commands	______	______
Network Systems	______	______
Cobol	______	______
Pascal	______	______
Quiz	______	______
Omnidex	______	______
Accounts Receivable System	______	______
Order Entry System	______	______

Without an opportunity for "hands-on", you may as well have your programmers stay home and read the manual.

Proven Track Record - When you purchase software, you ask for references. You should ask for references when you are purchasing training, as well. Ask for references who have not only attended the training class, but who have also had the same instructor that your people will have. All you need to do is think back to your college days to remember what a difference a good instructor can make.

Training Medium - There are a number of training mediums available, depending on the product and company providing the training. I've mentioned several already; classroom training on-site, classroom training off-site, and self-paced training. Moreover, there are several different methods used for self-paced training, such as video tapes, audio tapes, and CBT (computer based training). All of these mediums have advantages and disadvantages.

Classroom training on-site

There are a number of cases where on-site classroom training is advantageous:

- when you have a large number of people to train in a short time frame

- when your own specific requirements can be used as examples in class

- when you want your people to feel free to discuss proprietary information (provided the instructor has signed a non-disclosure agreement, of course)

On-site classroom training also has its drawbacks:

- Class attendants often get called out of class for emergencies, disrupting their own learning process and the class as a whole.

- If you don't have a training facility, or at least an area that can easily be converted to one, don't try this. You'll need to provide hardware, if there's going to be hands-on exercises, and plenty of room to spread out with manuals.

- Instructor availability is sometimes a problem.

Classroom training off-site

Advantages:

- you only have one or two people to train
- classes are offered locally
- You feel that your people will be able to learn from other participants from outside of your company, and bring that knowledge back.

Disadvantages:

- This is usually the most expensive training per student, when travel is required.
- There is often less of an opportunity to involve your own specific requirements.
- Some employees object to being out of town for several days, if this is required.
- Class availability is sometimes a problem.

Self-Paced Training

Advantages:

- The cost of self-paced training is usually the lowest, because an instructor is not required to be there. Moreover, you can use self-paced training over and over again.
- If a student needs to review a topic several times he or she can do so without slowing down the rest of the class or feeling stupid.
- Students can move at their own pace.
- Your staff can review specific areas of interest, whenever the need arises.
- Course availability is never a problem.

Disadvantages:

- There is no student/instructor interaction. (Self-paced training that makes use of audio or video tapes compensates for this, to some degree.)
- Students must discipline themselves to complete the course.

You will need to base your choice of training on all of the factors listed above. Sometimes you'll find that a combination of the mediums works best for you. For example, when you purchase a new product, you may want on-site classroom training for a group of your staff members, in order to get them up to speed in a short period of time. Self-paced training may be of interest to you for future employees or those that are unable to attend or complete the classroom training.

You will find that many products don't require any training at all. Simple command or menu driven tools can often be self-taught in a fairly short period of time. You will need to rely on your staff, other users of the product and the vendor to determine whether or not training is required.

When you've determined what training is required and selected the medium(s) that best suite your company, you can complete your training plan for purchased products. But what about internally developed software?

DEVELOPING AN INTERNAL TRAINING PLAN

Formal training programs for internally developed software are woefully scarce. In many cases, a programmer's introduction to a system that he or she has been assigned to work on is the source code itself. There are many reasons for this.

Training, like documentation, is one of the last items on the system development list. User training is sometimes tackled, but most people don't think of the need for data processing training at system development time. After all, everybody who needs to know about the system has been involved with it for months.

But what happens when new staff members come on board? Perhaps they get a stack of documentation to read, or source code listings. They spend the first few days on the job trying to plow through some of the most boring prose that mankind has ever been subjected to.

It has been proven many times over that most human beings do not learn well through reading alone. Their retention of the information is extremely limited, and any enthusiasm that they may have had for a subject is reduced to a mild interest at best, and dread at worst.

While its true that formal classroom training is often not feasible, due to the cost of developing such training and

the fact that you probably don't hire an entire classroom full of people at one time, there are alternative training programs that you can put into place that will offer your new-hires an interesting, challenging and effective learning process.

You will need to be creative about the resources that you use. Some examples of these are listed below:

User Training - If you have a user training program, have your new-hires attend. Although the information covered may be excruciatingly basic as far a computer knowledge goes, the new-hire will get a feel for the operational aspects of the system and the benefits that the users derive.

"User for a Day or Week" - Send the programmer to live with the users for a day or a week. Again, understanding the use of an application should be the first step, not the last.

Documentation - There's no getting away from it. If your system is documented, this will have to be one of the learning tools that your new-hire uses. But don't just hand it to him or her and ask them to read it. Instead, you can do what one creative manager that did. She asked a new-hire to "test" the documentation. First the programmer had to set up a test data base and account. This required that he learn about system security, and other features of the overall system. Then he had to run each program and ensure that it ran as documented. This included not only checking the screen and report formats, but also checking the data base to ensure that it had been updated as documented. This process made the task of learning the system a "hands-on" process. Since documentation is often allowed to become outdated, this approach can be useful to the data processing department, and made the new employee feel like a contributor from day one.

Written Tutorials - If you can't justify developing formal training programs for your new-hires, you may be able to justify the development of a written tutorial describing system use. A written tutorial is simply a script, showing screen or prompts for on-line applications, and valid data that can be entered. In fact, you may choose to have your new-hire who is testing the documentation develop this tutorial as well.

Face-to-face Tutorials - You should devote at least a half an hour to an hour a day to tutoring your new-hire. If you can't or won't do this, or if someone else on your staff would be better at it because of their system knowledge, assign another person to the task. You should spend 50% of

FIGURE 3

Programmer Training Program

Designed for: Edward Haskel
Employment Date: February 6, 1989

Monday	Tuesday	Wednesday	Thursday	Friday
8:00 Staff Meeting	8:00 Doc Edit	8:00 Accts Payable	8:00 Doc Edit	8:00 attend
9:00 Mgmt. Tutorial	10:00 Acct Recvble	Dept.		user trng.
10:00 Doc Edit	Dept	11:00 Doc Edit		
12:00 lunch	12:00 lunch w. team	12:00 lunch	12:00 lunch	12:00 lunch
1:00 Order Entry	1:00 user meeting	1:00 Doc Edit	1:00 Mgmt Tutorial	1:00 user trng.
Dept.	2:00 Mgmt. Tutorial	4:00 Mgmt. Tutorial	2:00 Doc Edit	4:30 Mgmt Tutorial
	3:00 Doc Edit			

Week Two

Week Three

Week Four

the time describing some aspect of the in-house systems, programming standards, or other areas that will help the new-hire understand and work within your shop. The other 50% should be spent answering questions that have come up during the day. Encourage the new-hire to write these questions down and use the tutorial time for this purpose. This will help to minimize interruptions for you and your staff, and by writing the questions and answers down, the new-hire will be better able to retain the information.

No matter what training methods you decide to use, an important facet of successful training is to schedule the activities. One example of a new-hire schedule is shown in figure 3. Please take notice that the schedule includes real work assignments, from the very first day.

Also, identify the purpose of each aspect of the training. This helps the new hire understand each goal, and illustrates that projects you may give him or her in the first few weeks are not just busy work, but have a purpose.

DEVELOPING COURSES

If you do decide to develop formal classes for internal use, there are several approaches you can take. Many larger companies have training departments that are responsible for developing all user and data processing training. In small and medium size companies, user and data processing training is often the responsibility of the Data Processing Department. If this is the case for you, choose the course developer and instructor(s) carefully.

The Course Developer - Ideally, this should be a person who has some experience with training. If you have someone on staff who has a degree in education, or who has given courses at a previous place of employment, this person could be a good choice. Classroom training should be well organized, so choosing a naturally organized person can be helpful as well.

A person with the most knowledge of the course subject matter is not necessarily the best choice. You need a person who is good at communicating concepts, ideas and techniques. They can research the details of a system, if they need to.

As an alternative, you may choose to have an outside service that specializes in training develop the course for you. If you are short on resources and can afford to do this, it can be one of the best alternatives.

FIGURE 4

Cost Justification
***(product)* Training**

1. Estimates for programmer time saved per program due to (product) training course:

Company A - 10%
Company B - 20%
Company C - 25%
Company D - 25%
Company E - 10%

average = 18%

2. Total estimated hours and cost for development of Sales History system over next six months:

	hours	x hourly rate	= cost
(programmer name)	160	$13.50	= $2,160
(programmer name)	240	$13.50	= $3,240
(programmer name)	480	$17.00	= $8,160
(programmer name)	960	$17.00	= $16,320
Total	1,840	Total	$29,880
			x .18
		savings (at 18%)	$ 5,378
plus overhead savings (30% of hourly rate)			+ 1,613
		Total savings	$ 6,991

3. Cost of on-site course:

Presentation	$2,900
instructor's estimated travel expenses	$1,500
Total	$4,400

NOTE: the savings shown above relate to the development of the Sales History system only. We plan to use *(product)* in the development of the new Customer Service System later in the year, as well. The scope of this project has not been finalized, but it is expected to exceed the scope of the Sales History system.

Have several people review the course outline and materials before the course is given. If the course is for end-users, have a representative of the user department review the course.

The Instructor - The instructor should be a good communicator also. Someone with speaking experience can often be a good choice. They will need to have empathy with their audience, and be able to quickly gain credibility with that audience.

When you are training Data Processing professionals on a system that is used within your shop, a segment of the course should include a presentation by a representative of the users of that system. An overview of the business operations is essential to understanding a system.

JUSTIFYING THE COST OF TRAINING

Some of you would probably love to offer your staff more formalized training, but have had difficulty in the past justifying the cost to the "powers that be". This can be difficult, but it can be done.

Figure 4 shows an example of a cost justification developed by one Data Processing manager. First, the manager asked the training company for five references. Then he called those references and asked them to quantify, on a per program basis, how much time they saved in program development due to the course. These answers were fairly subjective of course, but because the Data Processing Manager was able to find five companies whose answers were fairly consistent, those answers became credible.

The time savings was averaged, and then applied to the development plans that related to the specific product. The hours saved were then applied to the programmers' average hourly rate and the overhead required to employ those programmers.

The MIS director bought into the plan. They often do when they can see a bottom-line benefit.

TRAINING: A KEY TO BETTER PRODUCTIVITY

In summary, be creative about providing training to your staff. Explore alternatives and decide on the best course of action. Because effective training can have a significant impact on meeting deadlines and optimizing the use of your staffs time, it should be thought of as an integral part of product purchases and system development.

THE HP EMPLOYMENT MARKET - WHAT EVERY HIRING MANAGER SHOULD KNOW

Lynn A. Novo, CPC

Network Systems Company
98 South Turnpike Road
Wallingford, CT 06492
203-265-9995

As the number and size of data processing departments with Hewlett Packard computers increases, the supply of experienced programmer/analysts continues to shrink. Since your success as a manager depends on your ability to attract and hire qualified staff members, you can't risk losing a potential employee because your interview process should be improved.

Over the past eight years as a recruiter who has worked exclusively for companies that use HP computers, I have collected some of the best ideas for identifying, interviewing, and hiring in this unique employment market. I will share with you a complete hiring process that will help you feel confident that once you do find someone who can fill your job requirements, you will have done everything possible to get your offer accepted.

We will begin with an overview of the HP employment market and then concentrate on specific guidelines to utilize from the beginning to the end of your hiring process. We will discuss how to define your staffing need, how to identify candidates to interview, and how to conduct the interview itself. We will conclude with how to extend an offer in a way that it will most likely be accepted.

THE HP EMPLOYMENT MARKET

Let's begin our overview of the HP employment market by identifying it. Just who is it that looks for a new job? What are the reasons people give for leaving their current employer?

I am sure you have noticed that in some respects, every person who works in data processing is "in the market". We all have heard those words: "I'm always interested in hearing about new opportunities". But let's separate out the serious job changers from the "always looking" ones and see why they are in the market.

Whenever we talk with people who are seriously considering changing jobs, we ask them to tell us the reason they want to leave their current employer. Remember that we only talk with people who work in HP 3000 environments. All the reasons we have heard over the years fall into the main categories shown below.

REASONS FOR CHANGING JOBS

Want more money, salary too low

Candidate relocating

Company or department moving

No career growth, want more responsibility

Company converting to another vendor system

Want a shorter commute

Position boring, want more diversity and projects

Management problems, unclear direction

Losing job due to termination or lay off

For the purposes of this illustration, we are grouping together the reasons sited by ALL levels and types of HP personnel. Please keep in mind that the bulk of the HP employment market consists of applications personnel such as programmers or programmer/analysts. These are the people who most often change jobs and for whom there is the greatest demand.

Which do you think is the reason we hear most often?

Surprisingly, it's not because of low salary or low raises. In fact, that is one of the reasons we hear LEAST often.

Most people who consider themselves "in the market" say the reason is:

" No career growth, want more responsibility"

Please take note of this reason. Although we're not concentrating on how to keep your current employees, retaining your staff is a challenge just as big as hiring is. If you know that the most common reason for

changing jobs is something that you as a manager can control, such as giving more responsibility, maybe that is the most valuable thing you can take away from this presentation.

As we look at the actual hiring process, it will be helpful to keep in mind the following keys for successful hiring in the HP market.

FLEXIBILITY * TIMELINESS * PREPARATION

We'll see how each of these plays an important role in our success at implementing a hiring process that will result in more "yeses" to our offers.

THE HIRING PROCESS

What are the essential components of the hiring process?

Identifying the need
Sourcing candidates
Interviewing to hire
Extending the offer

IDENTIFYING THE NEED

The successful hiring managers we talk to always begin their hiring process with a well thought out job description. They do not necessarily try to replace someone who leaves with the same set of skills. They take the opportunity to evaluate their current staff and ascertain where there might be in-house skills to fill the gap.

Remember our reasons for changing jobs? This is a good time to make sure that anyone who can be promoted or reassigned to a project gets the opportunity to do so. You'll eliminate a lot of resentment and a possible second resignation if you consider your current staff first, before you hire from the outside.

Take the time to write a job description. We find that one of the biggest complaints we get from candidates after an interview is that they are not sure the manager know exactly what they will be doing on a day to day basis.

If we remember our keys to success in HP hiring, we can see that this first stage in the process requires a great deal of PREPARATION. Your time investment at this stage will be rewarded with successful hires after the interview.

Although I suggest that you have specific requirements for the job, I also recommend FLEXIBILITY. In this highly competitive HP employment market, you must be willing to accept a combination of skills and total experience, not just specific requirements. Managers who fill their vacancies fastest are willing to interview candidates with backgrounds that come close to their needs. They are in the habit of screening in potential candidates, not screening them out because they do not match a rigid set of requirements.

SOURCING CANDIDATES

In this competitive market, hiring managers need to be creative in their approach to identify sources of qualified candidates. It will not suffice to merely place an ad in the newspaper, or wait for your personnel department to forward you resumes. The successful hiring managers of the 1990's will be using a variety of sources and will network with as many people as possible on an ongoing basis to be certain that they are always tuned in to the market.

Let's brainstorm the ways you can identify people to interview. These are the ways hiring managers have told me they have sourced candidates:

Newspaper advertising
Radio or cable TV advertising
Job fairs
Employment agencies/recruiters
College placement office or computer science department
User group meetings/conventions
Personal contacts
Vendor referrals
Employee referrals (give bonus)

Our clients tell us that one of the consistently unreliable ways to identify candidates is through newspaper advertising. I am sure many of you have had successful response to an ad at some point, but our clients tell us they cannot rely on the response. In this unique HP market, there is not a large enough pool of qualified readers to justify the expense of newspaper

advertising in most markets. The person you need to hire probably is not bothering to read the Sunday want ads anyway.

The college campus is an untapped resource in some areas. If you know that a college in your area is using the HP to teach computer science students, you should get to know the department head and placement office. Many of our clients take college students as interns and then hire them to work in full time positions after graduation.

In sourcing candidates to interview, it helps to be willing to try different methods. Timeliness and preparation count when it comes to the all-important networking at user group meetings, with your vendors, or with recruiters. You can't begin to make contacts in the marketplace after your lead programmer resigns. You need to have known what's out there and what sources you can tap before you actually need them.

Flexibility and timeliness are not attributes typically used to describe personnel departments. If your personnel department's idea of recruiting is to put an ad in the paper, screen out resumes based on your ideal job requirements, then forward a resume to you two weeks after it was sent in; chances are the candidate has already started working at the HP shop down the road.

The managers who view themselves as successful in their HP hiring either have extraordinarily efficient and cooperative personnel departments who understand this unique market, or they are handling the hiring process almost entirely without personnel.

INTERVIEWING TO HIRE

Once we have identified our ace candidate for the staff opening, it is time to prepare for the interview. There are some very simple things we can do to make sure that our programmer doesn't get so aggravated with the interview process itself that he or she is turned off to our company. If enough of the following "little things" go wrong, your candidate will not want to work for you regardless of how much you want him or how high your salary offer is.

Have the personnel interview occur AFTER your department's interview, if you must have a personnel interview at all. Your ace doesn't care that his contribution to the medical plan is $14.10 a month. He won't be listening attentively enough to remember after the interview anyway. He wants to find out about you and the technical environment as quickly as possible.

If by chance you cannot persuade your personnel department to agree to this sequence, then at the very least make sure that they see your candidate promptly and that they complete their interview as quickly as possible.

Don't bother with an application form. If an employment application must be completed, either let your programmer take it and return it by mail or have him complete it after your interview. No use having your star programmer getting frustrated before your interview trying to remember if he started that job in February or March of 1980.

Don't keep your candidate waiting. Your potential hire needs to feel that your interview with him is important. When candidates must wait to begin their interview, that is the first thing they mention to us in the follow-up. "They made me wait 20 minutes". That negative first impression is hard to dispel during the course of the interview.

Try to be uninterrupted during the interview. Even if you don't mind if your ace hears your users screaming, he may get the impression that you don't value his time or have enough control of your time to be left undisturbed.

Remember, we're trying to create a setting where all the signals about you and your company are positive. Spending time in preparation for the interview and following these pointers will help your candidate be relaxed enough to interview well and therefore receptive to receiving an offer.

Once you get your programmer into your office (and it should be a private office, even if it isn't your own because yours is too messy) the most successful interviews have been well prepared to include the following four phases:

Rapport screening

Fact finding

Evaluation interviewing

Selling

RAPPORT SCREENING

In rapport screening, you set the stage for a productive interview. You want your candidate to feel comfortable so that he will be open and honest with you. You should ask friendly, conversational questions about his hobbies and interests. These first couple of minutes are when your ace is going to be getting to know you as well. This is the time for establishing shared experiences. Often we hear things like, "We hit it off right from the start. It turns out that the manager likes to ski as much as I do."

It helps to prepare your rapport-building comments. One way to do this is to look at the resume and pick out something you're interested in. Do you share an interest in sports? Did he spend time in Europe? Is he married with kids? Does he live in your hometown? Being prepared right from the start in this way will help you set the tone for a successful interview.

FACT FINDING

After about 10 minutes of rapport screening, you begin to establish if your recruit has the credentials to do the job. You will want to ask about what he does in his current position and probe into specific areas of his background to see how they relate to your needs.

As part of the fact finding session, many managers involve another staff member to ask technical questions. This is quite effective providing the other person doesn't duplicate the questions you have asked.

EVALUATION INTERVIEWING

If there is one secret to successful hiring, this is it. I guarantee that if you ask these three simple questions at this point in the interview, you will be rewarded with some very valuable information. With it you will be better able to determine if your star candidate should be made an offer; you will be able to

use this information to extend an offer with a greater likelihood of acceptance; and you will be better able to retain your ace programmer once he does come to work for you.

The evaluation interview is your time to get inside your candidate's head. It will reveal more about his personality and work ethic than any other type of questions you can ask.

Because of the personal nature of these questions and the fact that you want open and honest answers, it helps to take a break (get a cup of coffee, for example) to make the transition to this phase. I also recommend that you do not take notes during this phase, but rather write notes after the candidate leaves.

For the purposes of this paper, I will present the interpretation of some of the responses along with these questions. During the interview itself, however, I suggest that you keep your focus on the answers and reserve your interpretation until after the interview.

EVALUATION INTERVIEW QUESTION 1

Let's begin the evaluation questions with a topic we have discussed already. Ask:

"Why are you thinking about leaving your current employer?"

We have already seen the range of possible answers to this question. Let's take a closer look at each of the reasons beginning with these tangible reasons when people are forced to find a new job:

Candidate Relocating
Company or department moving
Losing job due to termination or lay off

Most managers feel comfortable with any of these reasons and in fact find that candidates who give these reasons are the most eager to accept offers. However, if your ace has lost or will lose his job, be sure to get the details of the termination. Good HP people are not usually let go; be certain your recruit is the exception.

Continuing with other reasons for leaving:

Want a shorter commute

If your company is significantly closer to your candidate's home or if it will be an easier commute, you have one of the best motivators for changing jobs. In fact, we have found that even people who would not ordinarily listen to other opportunities become receptive to an opening that is closer to home.

Company converting to another system

If your recruit is upset that his company is replacing their HP system with another vendor's system, this should be music to your ears. It demonstrates that your candidate knows and loves the HP and would rather switch employers than deal with new vendor systems. Incidentally, this kind of HP vendor loyalty is very common in the HP marketplace despite the popular notion that everyone would rather work on IBM.

Want more money, salary too low

This reason makes most managers nervous and indeed it should. The candidate who says money is his primary motivator for changing jobs is probably not the one you want to hire. If you hire this type, you will always wonder if you are giving him high enough raises (and our research shows that these people NEVER think they are being paid enough).

The reasons you will hear most often are the intangible ones that remain:

Position boring, want more diversity and projects
No career growth, want more responsibility
Management problems, unclear direction

The savvy hiring manager hears out the candidate's complaints in any of these areas. These refer to the way your candidate perceives his work environment. With these emotional issues, you can see that this information will help you shape your subsequent comments to your candidate in terms that will "feel" positive and that he will view as an improvement over his current situation. I'll show you exactly how when we get to the later phases of the interview process.

EVALUATION INTERVIEW QUESTION 2

"What do you want out of your work?"

This is where you will hear about your ace programmer's plans for his future. Can he verbalize some long term goals? Does he know what he wants? It will be hard for you to keep him happy if he doesn't even know what it is that he wants from his job. If he does tell you some plans that could fit into your department's future, we will see how you can use this information later to "sell" the job to him.

EVALUATION INTERVIEW QUESTION 3

"What effort would you be willing to put forth to achieve what you want?"

Listen for a response that relates to commitment. Your ace should have some idea of what it takes to achieve the success he seeks. Often this response will be conveyed in specific time elements such as, "I would be willing to work 12 hour days to complete a project ahead of schedule so that I could be promoted to a Project Leader position."

EVALUATION INTERVIEW SUMMARY

Those three little questions will give you a great insight into your potential hire's level of commitment, his career goals, and his emotional ties to his workplace. We will use the responses to these questions again later in the hiring process but before we move on, please allow me to make one more suggestion on the evaluation interview.

Once you have hired your ace programmer, I suggest you refer to your notes on this phase of the interview frequently. If you are continuing to provide an environment where your employee is getting what he wants out of work and is able to achieve his goals as given in responses to Questions 2 and 3, you will improve your odds of keeping him longer. An added benefit is that as your staff realizes that you care about their career development and that you are committed to helping them achieve their goals, you will gain a loyalty that is uncommon of in the data processing profession.

Let's move now to finishing up your interview. We have come to the fourth and final phase of the interview:

SELLING

Many hiring managers have told me that this was the hardest phase of the interview to master, but the rewards in terms of successful hires are worth it. What must you sell? You must sell the job, sell your own skills and management style, sell your company.

You begin this phase by telling your ace about your job opening, emphasizing those aspects you garnered will be important to him based on the evaluation interview. You can see why it would be foolish to discuss the details of the job before you had this valuable information.

Here's an example:

You learn from the evaluation interview that your ace wants to leave his current employer because he feels bored and doesn't feel that he's involved in any projects. Even if your opening is for a programmer to maintain your canned financial package, you should include some mention of a project. Think of something on your "To Do" list. Is there some piece of it that an eager new hire could do? Probably, so mention it in terms of being a project:

> *"You will be involved in a project to track user complaints and make recommendations for ways to streamline the problem-solving process."*

Your ace will hear a specific thing about your job that he can distinguish as being better than his current job. He must be able to make this distinction in order to accept your offer.

In this selling phase, you must also be prepared to open up about yourself. Your candidate will not work for you unless he feels comfortable with you. This is the place to discuss your own background including how long you have worked there, why you joined the company, and what you like about the company.

You might be surprised at how important this is. When we surveyed all the candidates we placed in new HP jobs over the past eight years, they sited the personality and skill level of the hiring manager as a major motivation for accepting the job offer. If you're not letting your recruit get to know you in the interview, you're missing a great selling opportunity.

Be enthusiastic about your company. You can't assume that your ace programmer knows anything about your business. You must be able to state at least one clear reason why your ace should go to work for your company. Your enthusiasm will be contagious. He will begin to feel a pride in having the opportunity to work for your organization.

EXTENDING THE OFFER

Let's move ahead now and say that your candidate really proved to be a gem in the end. If you like him and think you want to hire him, you must move toward making the offer immediately.

Remember that we said that successful hiring in the HP market required TIMELINESS. Here's a case of that. This is not a market where you have the luxury of spending three weeks interviewing and another two weeks deciding who to hire. The HP managers who are most successful in their hiring are willing to make an offer to the first candidate who is right for the job. You should know enough about the skills and personality you need before the interview to make a decision quickly after the interview when someone is right.

In determining the appropriate salary to offer, you need to pay as much as necessary to get your ace to accept. In the HP market, most managers tell me they would rather offer slightly higher than the minimum acceptable. That is a wise approach when you consider that you not only want your candidate to work for you, but to come to work <u>happily</u>. It has been our experience that your ace programmer will be more eager to start work quickly and has a better chance of staying with you longer if he doesn't feel that you were frugal in your initial offer.

If you take the responsibility for keeping in tune with the market, you should always have an idea of what the "going rate" for a particular position is in your geographic area. There will be times, however, when you will want to hire a person that has an unusual blend of technical skills that would make him extremely beneficial to your department. In that case, you must not hesitate to pay a higher salary than you had expected. That extra couple of thousand dollars will be recovered quickly in training costs and immediate productivity in your department. Determining salaries in this market requires FLEXIBILITY.

One final point about salaries. If you or your personnel departments are using any national data processing salary surveys for a reference in determining salaries in your HP department, throw them in the garbage. Our specialized market defies the national averages. If you can't get a salary survey based on the HP market itself, you can gauge salary ranges by asking any candidates you interview for their salary history, not just current salary. From that you can see what other companies have paid for different levels of experience over the past years and you can compare it with your staff salaries.

The best offers are made when the call is made to your candidate at home and parts of the selling phase of the interview are reiterated. This is also the place where you want to feed back to your gem that valuable information you gathered in the evaluation interview phase (which he will have forgotten that he told you). Here's an offer that will most likely be accepted:

> *"We were most impressed with the experience you have in maintaining financial applications in COBOL. We feel that we could help you develop those skills further by involving you with our manufacturing systems and special projects like the one I discussed during the interview. Over the next couple of years, we expect to expand our MIS department and hope that you would be someone who could eventually take on additional responsibilities.*
>
> *"Would you be interested? [By asking this now, you save yourself the embarrassment of making an offer to someone who is no longer interested.]*
>
> *"Great! Then it is my pleasure to offer you the job of Programmer/Analyst at a salary of $30,000 plus the benefits we discussed. You will be eligible for a salary increase of up to 10% in 6 months. When can we look forward to you starting?"*

Do you notice how we keep "selling" through the offer? And again, how the information from the evaluation interview gets reworded and used to paint a positive picture of what our ace wants? We present the biggest offer we can and be sure to mention benefits and raises. We're optimistic and enthusiastic about him starting. If you can make your offer like this as the culmination of a well-prepared hiring process, your efforts will be rewarded with an eager new staff member.

We have talked about how your success as a hiring manager in this highly competitive HP employment market requires that you use use flexibility, timeliness, and preparation in your hiring process. Using the guidelines for identifying your staffing need, sourcing HP candidates, interviewing to hire with the evaluation interview, and extending the offer will help make certain that the HP person you want to hire will want to work for you.

IT TAKES PRODUCTIVITY TO MAKE PRODUCTIVITY

Karon Wilkinson
Hewlett-Packard Company
5070 Centennial Blvd.
Colorado Springs, CO 80919

What do a Facilities Manager, Personnel Manager, Repair Center Manager and Computer Operations Manager have in common? They are all service providers. They are often considered overhead areas, with budgets held to a percent of revenue. They are also usually the first areas to be sought out for help when the economy turns south, growth flattens, and expense controls tighten. During this time, everyone looks for ways to increase productivity and do more with less. Everyone wants increased service levels, but of course, without the accompanying increased costs. Sound familiar?

As Data Center Manager for Hewlett-Packard in Colorado Springs, I manage a staff of four computer operators and two technical support people. We provide around-the-clock coverage supporting four HP3000 Series 70 computers and one HP-UX 9000/840 production system for a division of 450 employees. Having a small staff presents the problems of ensuring adequate coverage for peak workload periods, sick leave, and vacations. Our continuous process improvements, however, have allowed us to effectively manage and improve our situation, with our current focus on completely eliminating our third shift operations coverage. Process analysis and improvements over the past two and one half years have allowed us to reduce our operations headcount while providing increased service levels; in other words, we have been able to do more with less. This paper addresses approaches we found effective in achieving our productivity goals.

It all revolves around the buzzword of the 80's, PRODUCTIVITY. Boy, don't we all wish we had more of it. In fact, if you manage your processes effectively, once you are able to get some productivity improvements, you can actually use the savings to gain more. It is just like the old adage, "It takes money to make money". Well, it is also true about productivity, "It takes productivity to make productivity".

So, what is productivity? Webster's new universal unabridged dictionary provides these definitions for <u>productivity</u> and <u>productive</u>:

productivity, *n.*	the quality or status of being productive; productiveness.
productive, *a.* 1.	having the quality or power of producing; bringing as a result; causing to exist (with of); as, productive genius; an age productive of great men.
	:
4.	in economics, of or engaged in the creating of economic value, or the producing of goods or services.

Of the several definitions provided by Webster, I skipped over those pertaining to fertility, though at times I am sure we each, as managers, feel all our problems would be solved if "we could just produce more people". Of course we all know that is not the answer. Therefore, looking at our remaining definitions above, we can see that each lends clarity to our quest for a meaning of productivity. My definition, then, is:

> productivity,/n, 1. the state where individuals possess the power and ability to produce quality business results in the form of goods or services.
>
> 2. the state of doing more with less.

HOW DID WE GET INTO THIS PREDICAMENT?

The root of our business dilemma - in other words the seeking of increased productivity - lies in our reliance on more and more information. We are firmly entrenched in the Information Age. We all need information in order to perform our jobs. This is true of all segments of industry - retail, manufacturing, sales, banking. Industry has spent years producing information--"bit buckets" full of it. Now, how do we manage it before it hampers our productivity? The usual answer is to seek out Information Systems departments for help. Users need to be able to generate, read, analyze, evaluate and update information faster, more accurately, and with less cost than ever before. Who is better at helping than the "keepers" of the data, Information Systems? Meanwhile, Information Systems is processing and printing more and more information. Faster? Better? With less cost? Maybe not. The cost of managing the information explosion is the growing dilemma for Information Systems departments as we exit the 80's and enter the 90's.

WHY NOT JUST STAFF UP?

Figure 1a.

Adding people, as illustrated in Figure 1a, is not the most cost

effective solution. Often times we receive pressure from our internal customers to hire additional operators, programmers, and technical support staff to improve service levels. However, nothing is gained by merely throwing money and people at problems, until the root of the problems and their solutions have been determined. Users forget that adding people to increase service levels generally leads to increased costs. Payroll, taxes, benefits and support overhead go right up with increased staffing.

CUT TO ESSENTIALS BY PRIORITIZING DEMANDS

Increased demands on Information Systems services must be prioritized and serviced as resources allow. Prioritizing is the only way to ensure that the really important demands are met. These are the critical requirements needed to sustain our internal customers through a peak demand period, until productivity or business growth allows for more resources. Often this approach serves to slow down the momentum that leads to escalating demands which can far exceed resource capacity. Existing service commitments can deteriorate if the Information Systems staff is stretched too thin trying to meet each and every demand. When unchecked demands continue to escalate, nothing is really done well, and everyone loses. Information Systems management must start negotiating with our internal customers to determine priorities, and then manage these priorities during periods of change. Shifting priorities have become a fact of life, and that leads us to the next trade off solution.

SOMETIMES IT IS NECESSARY TO GIVE UP SOMETHING TO GET SOMETHING

After prioritization has been completed and assignments made, new demands will often vie for an assigned resource. Shifting priorities inevitably lead to the rise in the urgency of different demands, and so negotiation must continue. Something has to give, and that is when it is necessary to ask our customers, "If we are to shift resources to service this new demand, what are you willing to put on hold?". This is a hard question, but many times resource constraints dictate that the customer cannot have both. Depending on the amount invested to date, sometimes it is wise to continue with the existing project until completion. When the project is put on hold that project and the project that took priority are both impacted. Even if the existing project is relatively new, it may have to be started over from scratch if it is put on hold. Ultimately it is the customer's decision. Without a doubt, shifting priorities negatively affect productivity. It is an indication of being driven by crisis, and a result of ineffective planning. For a while, it might seem that progress is being made, but in the end you realize that no significant progress has been made at all.

In effect, we have experienced this first hand. We realized we could not have our cake and eat it too, so we focused on the highest leverage projects. These were projects whose benefits would get us out of a crisis driven mode and into a productivity gaining mode. Management needed to see the benefits of this approach. We needed to show them the productivity gains already achieved, so that they would allow us to direct our efforts to gain even more.

WE WERE ABLE TO GET THE NECESSARY MANAGEMENT APPROVAL TO INVEST

We had evaluated and purchased an excellent tool that would allow automatic job scheduling on our HP3000 computers. However, we found it increasingly difficult to devote quality time to the implementation and maintenance that this product needed to realize its fullest potential. Other demands and crises kept getting in the way of the project. Finally, we made a presentation to management that convinced them that we needed to prioritize our efforts again and commit resources to this project. This software tool helped us eliminate the potential for human errors introduced while maintaining the production schedules each day. There was repetitive and redundant effort as each day's schedule was literally retyped for each production cycle. Since this was maintained by Operations, it consumed at least a full-time operator just to maintain the manual schedule. When we charted the time savings and showed the reduction in scheduling errors, management readily allocated more resource time to this project. As a result, we are now fully implemented on all of our HP3000s using this automated scheduling tool.

INVEST YOUR PRODUCTIVITY GAINS TO ACHIEVE EVEN MORE

The automated scheduling project was only one example of productivity gains achieved with management support. The time it saved, though, led to free time which could be channeled into other project tasks. We had already formed a technical support group that was responsible for implementing Operations Department productivity tools. This group was staffed with Operations personnel during our down-sizing efforts. As fewer operators were required to handle operation tasks, a path into technical support was provided. Fewer operators were needed as a result of Programmer/Analyst efforts to reduce tape mounts and streamline job processes. Additionally, more efficient application software to manage business needs for our customers was also being implemented. Faster printers, such as the HP2680 laser printer, as well as faster tape drives and disc caching, had sped up processing significantly. A banner program was introduced that aided operators in making distribution of reports to customer output bins. Responsibilities were realigned. Often, tasks fell upon Operations that were not necessarily appropriate. Questioning their appropriateness, along with reassigning these tasks, put the responsibility where it should have been in the first place. Help desks and focus group members distributed throughout the division took advantage of computer savvy people who could be called upon for help within their user department. Today if you look around your own organizations, you will probably notice individuals who are also informally performing assistance to their department members. It usually isn't a full-time effort, and it gives these knowledgeable resources around the organization an opportunity for job enrichment. Obviously, it also results in a decrease in calls made to the Data Center from customers needing help.

WATCH OUT FOR TRAPS!

As I mentioned earlier, technology has helped productivity in our data center. It has also been a double-edged sword. Denser storage devices have made it possible to store more data than ever thought possible.

But, online application performance degraded as databases grew in size during our "information explosion". Disc caching came along and sped up data retrieval significantly for customer online applications. Upgrades to the operating system with expanded tables helped the performance of applications, too. But, all these growing data files had to be backed up and archived which contributed to tape handling and storage problems. Tape drives with file compression features were introduced which reduced the number of tapes needed to back up files. The compression represented anywhere from a 3-to-1 to 5-to-1 reduction in number of tape reels. This single hardware advancement resulted in significant savings for operators. We suddenly went from a 23 reel full dump on our development system to less than 4 reels in half the time. We were truly amazed. We are now investigating software products that will compress disc files. These products should save disc space, require fewer tape volumes for file backups, and make LAN transfer of files a practical solution for a multi-HP3000 environment.

Well, with more data, faster I/O, and a full complement of software applications for the customers, reports became the next headache for the data center. Customers wanted reports, and they wanted lots of them. We had a solution for that, too. Users were encouraged to take advantage of two-up and four-up printing capabilities on the HP2680 laser printer. Compressing data on disc and tape is one thing, but customers just wouldn't accept compressed printout on reports---too hard to read. This was our next challenge, and later in this paper I will address ways we started to solve the report volume problem.

WHAT DO YOU DO WITH THE TIME SAVED FROM PRODUCTIVITY IMPROVEMENTS?
The intent is to free up time spent on tasks that you can eliminate with productivity improvement. Doing the wrong thing faster and easier is not the right solution. Unfortunately, that is what happens sometimes. We apply the improvements to the wrong tasks. These are exactly the tasks we should eliminate, not make easier to perform. Therefore, we took the approach where everything was evaluated from a process standpoint. Look at your processes, even those that appear to be well under control. The information systems environment is changing dynamically every day. What might have been the correct process yesterday, may not be appropriate today. Talk to your customers to find out problem areas and processes that need to be scrutinized. Start with those critical to achieving your most important objectives. Focus on these problem areas first. Pareto the problems. Find their root cause. It is always more simple to address the symptom and overlook the real cause. Produce fishbone diagrams, process flow charts, resource histograms, or whatever method helps you ferret out the source of recovery incidents, resource bottlenecks, scheduling conflicts, project cost overruns, or under and over utilization of equipment or people. Once the true causes are known, the solutions can be brainstormed and evaluated. This is the typical TQC approach. Start with either the easiest to implement or the one with the highest payback depending on the result of your analysis. That's how we were actually able to convince management that resources needed to be directed to the

conversion of our automated scheduling tool. Manual scheduling was sapping our resources both in maintenance and in recovery incidents that occurred as a result of performing this function manually. Throughout the duration of this project, other smaller improvements were made which provided incremental savings along the way. As time was freed from doing either the wrong tasks, or doing them manually, this time was reinvested into additional projects that were a result of our TQC process. Examples of some simple improvements we made which yielded significant savings are:

- We encouraged even greater use of distributed printers in the user departments for local printing, especially for electronic mail messages which were small and easily lost among the larger reports.
- We leased multiple facsimile machines and placed them in departments throughout our division. By training administrative specialists in the user areas, we were able to offload a great deal of our facsimile transmission workload. Users also saw the benefit of faster turnaround since the facsimile machine and a resource resided in their own department. This was an example of a "win-win" situation.
- We had our off-site tape vault moved from another Hewlett-Packard site located a half mile away to a newly acquired adjacent building on our site. With staffing so lean, it had been a real resource drain to cart the backup tapes to and from the off-site vault.
- In conjunction with the moving of the tape vault, a newly purchased tape truck facilitated moving up to 120 tapes to the vault at one time. All of this required less time and avoided an operator being out of the data center for long periods of time while transporting our backup tapes.

SEEK NEW APPROACHES - NEW PERSPECTIVES - NEW EXPECTATIONS - NEW MEASURES

Information Systems departments traditionally strive to maintain processes that are in control, governed by discipline and with documented procedures. Remaining accountable to the division as guardians of the data and the processes that support information flow is still an important responsibility. However, this fundamental value also hampers the department when seeking innovative solutions to improve productivity. For example, we always took all of our systems down at mid-day and performed incremental backups to pick up changed files since the prior day. End-of-day backups were also performed. On the production machine, start-of-day backups were also taken to provide a recovery point following the night's production processing in the event of disc corruption during the day. These backups grew as the size and number of application systems continued to rise. The mid-day backup was cumbersome because customers had to be taken off the system in order to get a thorough backup of changed files. The backups that used to take

less than half an hour had gradually grown to require forty-five minutes. Customers were discontent as their productivity suffered while they could not access the real-time application systems during the backup. Finally, customers questioned the value of the mid-day backups and presented a recommendation to management to discontinue these backups. Because of the active involvement between Information Systems and the user community in the evaluation of the user proposal, management readily accepted discontinuation of all HP3000 system mid-day backups. At that time, each department had to update their procedures for disaster recovery to cover the expanded window for data loss exposure. This was accomplished, and satisfied Information Systems' responsibility for disaster recovery when the backups were eliminated.

Implementing faster mechanisms for backup wasn't the best solution in our case. We had confidence in our decision to discontinue backups because of increased hardware reliability and software integrity. Eliminating the backups, at least for our division, was the better solution. In this single example, a new perspective was taken by the customers, a new approach was taken in the way backups had been handled in the past, and new expectations were set for system accessibility to customers with the elimination of the mid-day backup. Also, the process changes were properly documented. We experienced a definite "win-win" situation.

Measures should follow next to determine productivity gains for customers and for the operators who no longer need to perform the backups. These measures can be used to encourage further improvements as other processes are challenged in the pursuit of productivity. We are now looking at all of our backups, system-wide and application specific, to determine their appropriateness. Backups can become a security blanket, and when excessively used, will negatively affect performance. Just the right amount, appropriately spaced, provides recovery points without jeopardizing production schedules. Recovery is simplified. As in any analysis, you need to weigh the advantages versus disadvantages, and the costs versus savings. Backups sprinkled between short processing cycles don't save much in the event of a recovery. Grouping processes together, with backups more sparingly spaced, buys the most if recovery isn't that significant of an effort. In this case, fewer backups were better. If, on the other hand, a very time intensive set of processes is involved, then backups strategically spaced around these big processes is the best solution. Regardless, a comparative analysis should be done as these types of process improvements are investigated.

WILL I EVER SEE THE PAPERLESS SOCIETY IN MY LIFETIME?

I wish I had the answer. Somehow this concept has not yet been fully embraced by our customers. We are being buried in paper, and from all that I hear, this is becoming a growing headache to many data center managers. It is an intensive manual effort for operators. Distribution errors make for dissatisfied customers. Besides hurting the image of operators, it usually results in rerunning of reports that are redundant

when the original report finally shows up on the desk of the intended customer. That makes the situation doubly bad. We have two laser printers in our data center and they seem to be printing constantly. Report distribution is another problem holding us back from moving towards the unattended data center. Who's going to get all of the night's reports broken down and distributed to the customers by the time they come into work? It still requires an operator to load the boxes of paper on the printer when we go through four boxes in a normal night's processing. In our efforts to solve this distribution problem, here are a few steps we've taken already.

- We considered the Cost of Report Generation. To derive actual cost in dollars and cents, you have to first understand what it really entails:

> **CRG = paper + handling + delivery + receiving + central report distribution + department report distribution + customer handling + mulching (disposal) + printer maintenance + depreciation + supplies (toner. . .)**

  This, in fact is a simplified equation. There are more costs than even those above in the process of raw product to end-customer report. Multiply the cost by the number of cycles of report generation over time and the cost is staggering. It is also often a waste.
- We investigated experiences of other Hewlett-Packard divisions, peers within our organization, articles written on the subject, and any resources we could find. From all this research we started implementing these solutions:

  1. **Networks** - Local Area Networks connected to a division backbone allowed distributed printing to the local printers within each department. It didn't cut down on the amount of printing necessarily, but it improved distribution. Also, customers suddenly began to realize the cost of output. This didn't solve our division's large report output problem, but it did help with the small listings that often got lost among the large reports.
  2. **Distributed Data** - With the widespread introduction of PCs around the division, HP AdvanceLink allowed customers to download data from the large HP3000 databases to their PCs. Once in the user domain, customers could view the data any way they liked. The wealth of PC software tools provided the power they needed to do end-user computing. This was not without some

issues, though, as the customers needed to handle this capability responsibly. Erroneously generated output could potentially hurt customer credibility and might even result in bad management decisions based on invalid information.

3. **Online Review** - More and more online review applications have been introduced on the HP3000, either as part of a business application package or as a separate reporting tool. We have implemented many customer reporting capabilities that have moved some of the burden of reporting to the customers and off Information Systems. We chose to move copies of databases to another HP3000 machine at the end of our nightly processing so as not to degrade real-time processing on our production machine during the day. There is a certain amount of overhead associated with the use of most reporting tools and we wanted to preserve customer productivity for online customers and reporting customers alike.
4. **Exception Viewing/Reporting** - Third-party software products allowed us to defer and read JCL online, as well as reports that did not need to be printed. This served two purposes: less likelihood of lost small listings and easy viewing of listings and reports. This output could be deferred based on whether an abort occurred, or, in the case of a successful execution, by using a specific outclass priority on the output file statement. This feature was particularly popular with accounting and procurement departments. Printed exception reporting has not been extensively used, though accounting does support this concept.
5. **Condensed/summary viewing/reporting** - Suppression of detail data was employed as appropriate to give "bottom line results". This was accomplished in two ways: end-user reporting tools using data extracted from large source databases and by using PCs and HP AdvanceLink to extract subset data from the HP3000 into the customer PC. This data could be reviewed online or printed to local departmental printers. Both mechanisms served to improve the productivity of Information Systems, especially for Operations. HP2680 laser printers provided the advantage of printing two-up and four-up. This resulted in two to four logical pages per physical page of

output. Though a little more difficult for the customers to read, it resulted in a paper savings of two to four times. Archived copies that are seldom read are excellent candidates for four-up printing. This compressed printing hasn't been as popular with customers as we would have liked, but is still an alternative we promote for appropriate applications. We have also promoted the establishing of department reference areas where copies of sharable reports can be maintained. This avoids printing personal copies for everyone in a department. A few general ledger reports, for example, could be referenced by everyone in accounting without printing individual copies.

6. **Other Media** - Microfiche was another alternative to printing. We generate microfiche for transaction logs for materials management and for accounting general ledger reporting. This is better accepted when customers do not need to write on the reports. Another media is PCs. By downloading to PCs, data is stored on floppies for reporting and archiving purposes, or in PC data bases for user data manipulation. We have used HP AdvanceLink to move data from our local HP3000 to the customer PC, where it is massaged before uploading to another HP3000 at our Corporate headquarters. Another media that is now available is optical disk. We receive all of our HP3000 manuals on compact disk which provides easy online viewing and selective printing of our HP3000 manuals.
7. **Intelligent Data Transfer Systems** - There are software products that function as "engines" to pull data from one processor to another without human intervention. This is done by scheduling command files to run at specific times when data is ready to move from one system to another. For example, data can be pulled from the HP3000 after a production process completes. The data then can be ported automatically to a server where it is "pushed" out to other PCs on a LAN. It can happen overnight and be available first thing in the morning when the customers come into work. This is not a "pie in the sky" concept. It can be done today with current technology. So far we have mastered the hard part which is ensuring data compatibility between processors within our network. Image database data and MPE files can be

converted into a format that can be read directly by a variety of PC spreadsheet and database products. The automatic data transfer on the network will be our next challenge. There are many different approaches possible for transferring data bi-directionally between the PC user domain and the HP3000. The best approach should be one that makes the data transfer transparent to the customer. An analysis of the application, customer environment and required objectives should be undertaken to help select one of the many products available in today's market. We've selected an accounting application as our first project to port data on a LAN into spreadsheets and graphs without the need for manual intervention.

8. **End-User Computing** - End-user computing on the PC has advantages and disadvantages. The advantages outweigh the disadvantages. With the widespread introduction of PCs throughout the division, it became apparent that a wealth of computing power was available if we could only use it wisely. We have developed very savvy customers. They know what they want and they have learned how to get it. For focused applications, they are the best ones to provide solutions. The issues arose when the application spanned beyond their domain and affected other systems or decisions based on customer reports. Applications maintained by Information Systems have been centrally controlled and supported with strict guidelines for documentation and revision control. This has not always been the case with end-user applications. What starts as a personal information tool soon becomes something that is heavily relied upon by others. Support is informal, documentation is not up to standard, and backups are not always consistently performed. Those who generate end-user reports need to realize their responsibility to others who might be recipients of their reports.

THE NEW MODEL - INFORMATION SYSTEMS AND USERS COMPLEMENT EACH OTHER

Centrally supported information bases will always play an important role in information management. Information Systems facilitates the transformation of data into information, and assures data integrity by acting as guardian of the data. This information can then be accessed in the mainframe environment or PC environment once the data has been

ported to the user domain. A model that illustrates how well both roles might complement each other is depicted in Figure 2a:

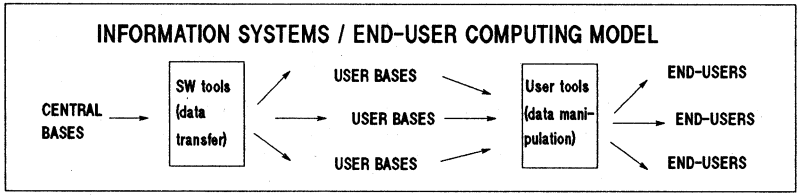


Figure 2a.

The central bases maintained by Information Systems provide for integrity of data and single source data management. Software tools provide for data transfer to other HP3000s and PCs. HP RAPID/3000, and a variety of PC-based tools, provide user reporting in the user domain. The advantages are that data is made available to the customers for end-user reporting and manipulation, and this in turn improves the productivity of both Information Systems and the end user. Information Systems is able to focus on managing the data, while the end-user satisfies individual report requirements.

SUMMARY:

Increased productivity is our main objective, knowing that once gained, it will lead to even greater productivity within our organization. An analysis is paramount in the drive towards productivity improvements. A TQC analysis ensures that our efforts will actually net us the gains we require along with elimination of those tasks we shouldn't be doing in the first place. To achieve maximum leverage, the highest payback projects should be prioritized. The 80/20 rule applies well in this case. If you can invest less to gain more, you will have conquered the "resource catch-22" - not enough people or time to solve problems because you are always trying to solve problems. Fighting fires continually drains resources, giving nothing in return for the effort invested. The problems that contribute to the crisis situations need to be understood, analyzed and then solved once and for all so that they are never repeated. Often during analysis a few basic causes will be found to be at the root of most problems, so that eliminating these few causes will be all that is necessary. Usually the problems are a product of poor processes or processes that are not being followed consistently. Productivity gains cannot be achieved solely by introducing automated tools. If the process is bad, the tools will only allow your organization to "do the wrong thing" faster and easier, not necessarily better.

As your efforts free resource time, you should continue to re-invest this time to generate additional productivity gains. Share your results

with management to further solicit their funding and support for your efforts. Draw on the experiences of others who have risen to be role models for others to follow. Your peers in your company or organization, user groups and a wealth of literature, can help accelerate your efforts. While you are involved in these productivity projects, control the demands on your department through "needs" prioritization. Lack of prioritization and focus can sap any worthwhile effort, and usually perpetuate the "quick fix", "crisis driven" operating mode. Progress suffers as your critical objectives are jeopardized.

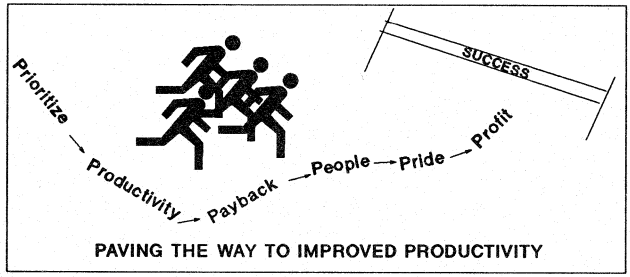


Figure 3a.

This will lead to an organization of people who take pride in the fact that their enhanced productivity leads to a more productive and profitable organization. Figure 3a illustrates the principle that in our cooperative efforts to achieve improved productivity, we all come out winners.

Above all, **start now!** The situation won't get better by just waiting it out until somehow everything suddenly improves.

Getting the Most From HP Customer Education

By Mary M. Humphrey
Manager, ASD Customer Education Engineering

Hewlett-Packard
100 Mayfield Avenue, M/S 36LC
Mountain View, California 94043

Why should you read this? Would you leave half the equipment in the box? What you don't know about HP Customer Education could be equivalent to leaving the most important system component in the box--you!

"Unless you know how to be an effective adult learner, you risk wasting time and money."

It's not enough to know what courses are offered, when, and where. Unless you know how to be an effective adult learner, you risk wasting time and money. This article deals with what you need to know about yourself as an adult learner and how HP Customer Education is designed to help you avoid the frustration of attending classes without receiving training.

You've already demonstrated several principles of adult learning. First, unless the readership of this publication has changed substantially, you're an adult. Next, you're reading something you don't have to read because it might be useful; you're an active learner. I'd better get to the point quickly or you'll stop reading; you want results--now. The next paragraph tells you more about the rest of this article. Read that much more, then decide for yourself if it's worth your effort to finish.

Tll Them What You're Going to Tell Them

Adult learning sounds to most people like a theoretical area for academics to research. I'm going to tell you the basics of adult learning you should know to enable you to recognize your own training needs, to get the most out of the training on which you spend time and money and to avoid being frustrated or mistrained. I'll kepp it simple and practical and I'll provide questions you can ask to know how you're doing. (Simple, practical, self-evaluation--basics of s of adult learning put into use.)

Question One--Do I Need Training?

That's the first useful question. It also follows a key principle: adults need payoffs for learning and they need them early and often. Training requires time and usually money. Adults don't like to spend much of either. If you are a person who hates waiting in line or who gets upset to see an item you have just purchased being sold at a lower price, you need to ask yourself Question One.

Do you need to be able to do your job bbetter, easier or faster to avoid problems or to take advantage of opportunities on the job? If you can specifically state a skill or information area that would help you do this, you meet the first criterion of Question One--you have a need. If it's not something you can get from available documentation, people or services, if YOU are the one who needs to know or to do, then the need is a training need. Keep going.

Does training exist which includes objectives that match your needs? Avoid training which lists product features; you are looking for solutions to job tasks. Training organized around what a product does leaves you to interpret how you will use the product at work. Training organized around job tasks helps you put the learning to use immediately.

By now you should see some principles of adult learning embedded in Question One. Adults learn best when they perceive a personal need to learn. They learn best when the training is directly relevant to what they need to accomplish. Know your needs. When a trainer asks you, "Why are you here? What do you want to get out of this session?" will you be able to answer?

Question Two--Am I Ready For Training?

Instructional designers who create HP training and the instructors who deliver information know that both children and adults learn best when you start with the simple and familiar. This lays the groundwork, enabling you to proceed into more complex and detailed information.

This is what prerequisites and prestudies are all about. Much of HP training is designed to begin at a common point to avoid having to go over material you should already know. Prerequisites and prestudies are created to save you time. Ignore prerequisites and leave prestudy undone and you risk wasting the rest of the training.

Well-designed training also begins with an introduction to presenting the big picture. Be on time to class. Pay attention and ask yourself if you recognize what the instructor is talking about. Do you know what to expect over the course of the training? Course agendas and outlines explain this.

Question Two deals with another principle of adult learning. Instruction should begin with what the learner knows and should be clear on where it will proceed from there. Be an active learner. If you aren't told what is coming up in a training session, ask.

Question Three--Why Am I Learning This?

Adults have a much greater capacity than children of being aware of their own learning. You can use this awareness to make your learning more efficient. Look for the stated objective of each module or section of instruction. What's the main point? In task-oriented training this will be expressed as what you will be able to do after completing that part of the training (in HP training you will see these listed as "Student Performance Objectives" and/or "Module Goals"). You should be able to recognize a relevant job task you could then do better, easier, faster...These are the answers to why you would want to learn something.

You also need to ask yourself how well you need to know the information in order to perform a task. Is it sufficient to be able to look it up later? Do you need to be able to adapt the learning to fit your particular situation? Will you have to create a new procedure or interpret the information for others? Being aware of why you need to know something helps adults decide how well they need to know it. Use your assessment of your needs to guide your learning efforts.

Question Four--Am I Learning?

HP instructors ask many questions to test the understanding of their students. Have you ever looked down or away from the instructor to avoid being called upon? Adult learners are no different from younger students in this way. You may also have been so relieved when you were not called upon that you didn't listen to the question or the answer. This is an opportunity to check your own learning.

It is possible to by shy while being an active learner. While a question is being asked, look down and write out the question the instructor asked. Listen to the answer and write it down. If you don't know th answer or if you don't recognize the correct answer when it is given, you need to "become active." Ask for assistance, a repeat or more information, whatever it takes to ensure you understand the main point. HP instructors are coached on how to answer questions as well as ask them (and they won't look away when you call on them).

The content organization for most HP customer classes is designed to build upon earlier information and skills. To be sure no one "gets lost," several reviews and questions are built into the instruction. If you find you can't answer the questions, don't wait, hoping it will become clear later. Ask now.

HP instructors are trained to keep the pace moving as long as everyone is learning (adult learners become easily bored). They are also experienced in giving examples, demonstrations or explanations when anyone needs extra instruction. The instructor's ability to adapt to your personal need for assistance is the main value of HP instruction. This is a value not found sitting at a desk with a manual of self-paced training.

Question Five--Can I Do It?

Adult learners reach their highest potential and retain the most information when they can practice. Hands-on labs and exercises are costly to develop and require up-to-date facilities and equipment along with expert instructors.

HP courses are designed to contain hands-on practice coupled with instructor feedback. This is the time to be sure you will be able to use your new information or skills on the job. Complete the lab steps, answer the review questions, use any extra time to try it again or ask questions. Many of the labs also have alternate activities available for advanced students, such as customization, troubleshooting or special applications. Ask the instructor.

Use the lab practice to check yourself on how well you will be able to use your training. Active self-evaluation is one capability adults can use to strongly improve their learning.

Question Six--Is That All There Is?

Customer Education at HP is designed as part of an overall support strategy. Do you know what is available to you and how to get the most out of it?

All HP Customer Education courses provide you with a student workbook. It will have student notes and copies of the visual aids used in class. It will also reference other HP documents which support the instruction. Make notes of those references. Do you know which manuals and documents at the classroom site were purchased with your hardware or software and are, therefore, available at your workplace? Do you know the revision number of your documents? Take a few moments before or soon after training class to check your on-site references. It is well worth the time and protects your investment in training and equipment.

Adult learners need an immediate payoff for their learning effort. This sometimes works against the purpose of reference documents and how to use them, which is covered in HP courses. You may not think this is information you need to know now and therefore give it little attention. One way to make it more immediate is to remember that the instructor's presentation of the documentation won't be available later. You need to get it now if you are going to get the full benefit of both your training time and money you spent for the documents in the first place. It's the payoff for earlier expenses.

Do you know how training links to other HP support services? Is the course required for a support contract? Have you worked with your HP account team to determine a training plan for system growth? They have specific tools, such as those in the HP Platinum Book for HP 3000 users, to help you plan your training to achieve overall goals. Ask for training schedules and data sheets on individual courses. There are also curriculum paths available to help you determine how individual courses fit into a larger sequence.

Question Seven--Have I Met My Training Need?

Go back to your response for Question One. This is where you determined what tasks you need to be able to do better, easier, faster, what problems you wanted to avoid and what opportunities you wanted to realize. The final payoff for adult learners is achieving these goals. The realization of these goals will occur as you apply your training to actual work situations.

When will you use the training when you return to work? Look for an opportunity (it may not be the next chance, but should be in the near future). Think ahead about what you have learned and decide what you can actually put into use. This is where knowing WHY you learned something begins to pay off. Are there areas you feel are weak and should be practiced on-the-job? Here you will be using you self-evaluation from class questions and practice labs to guide you. You can make your efforts as an adult learner pay off as much as the learning itself.

If your training has been successful--if your original training need is satisfied--you may find yourself back at Question One with a new need. The HP course content and practice are designed to give you knowledge and skills to make you a more independent and capable user. This often means you will be ready for an advanced level of training. Knowing how your training fits into a curriculum or support plan will pay off in identifying future training needs.

Question Eight--Am I Having Fun Yet?

Adult learners are no different than younger students in that they want to enjoy learning. For adults this means more than coffee, donuts and lots of laughs. Customer satisfaction is the term business uses to imply that you got your money and time's worth, your needs were met and you would recommend it to others. HP's Customer Education program for monitoring customer satisfaction is unique in business training.

If you have attended a customer class, you received a Training Quality Metrics (TQM) form for your feedback. Did you take time to fill it out? Did you write comments as well as fill in the numerical ratings? Do you know what happened to your feedback?

All customers, for all courses, at all sites, are asked to give HP feedback on the course content, labs, instructors and facilities. The instructors read each form (especially the comments) and discuss them with their managers. The forms are also processed and weekly reports are sent to the training facilities and the course developers. Customer satisfaction statistics are reported quarterly to top managers in the company.

What do you get out of this activity? Your immediate payoff is the response of the instructor and education center managers to your comments. Managers often call customers to follow up on specific comments and requests. Your long-term payoff is the impact you have on course revisions, choices in design for future courses and identification of new training needs or opportunities. Individual comments often initiate larger investigations.

In the introduction of this article I mentioned that adult learners like to have control. Customer feedback is an opportunity to extend your control over meeting your immediate and long term training needs. Filling out the forms, with comments, requires you to be active, to deal with your

practical and gets results.

The final point about adult learning is that reviews which end with action are strong methods for ensuring that the information is remembered and used. To get the most out of HP Customer Education, use the key questions in this article during the next training you attend. Being a TURBO student will allow you to learn better, easier and faster.

[This article also appeared in SuperGroup Magazine, January, 1989, pgs. 23-26.]

ELECTRONIC DATA INTERCHANGE - WHAT IS IT?

Larry Oveson
James C. White Company
Box 5495
Greenville, South Carolina 29606
(803) 288-4692

A. Introduction

What would happen if the customers that fund over eighty percent of your business told you to implement Electronic Data Interchange (EDI) by the Summer of 1990 or be 'desourced'? Could your company handle it? If your company is a supplier for Chrysler, Ford or General Motors, you already have been so warned.

The number of companies implementing EDI is progressing at an incredible rate. Some figures I have read indicate that ninety-seven percent of the Fortune 100 and fifty percent of the Fortune 1000 are implementing EDI. Over 5000 total companies in the United States are using EDI and the number is expected to double annually over the next three years. It is not just the number of companies that are implementing EDI but the names of the companies: the automotive giants, DuPont, Texas Instruments, IBM (that's International Business Machine Corporation) and the government's General Services Administration. The fact that there are five sessions this year devoted to EDI compared to only one last year also indicates a growing interest in this topic.

Electronic Data Interchange is the direct, computer-to-computer exchange of standard business documents between trading partners, usually involving no human intervention or paper. There are a few different 'standards' for various industries, but the official standard for implementing EDI is ANSI X12. The standards provide a common reference point for different companies and different computers that enables the exchange of electronic documents.

Companies using EDI are currently enjoying a competitive advantage over their counterparts in many industries. However, in the coming years, companies not using EDI may well find themselves at a competitive disadvantage. In general, industry is faced with two alternatives concerning EDI: either install it willingly or install it because competition and suppliers demand it.

This paper presents a high-level view to EDI focusing on its advantages over paper-based systems and the issues to consider before implementation.

B. Advantages of EDI over Paper-Based Systems

1. The Nature of a Paper-Based Transaction

Transacting business in today's marketplace requires volumes of paper and a good deal of patience. Even a very simple purchase order can produce a significant amount of paper and take up considerable time in processing and handling. Consider the following: a customer sends a vendor a paper purchase order along with copies of the PO for the vendor's internal use. The vendor sends the customer an order acknowledgement. The customer sends the vendor PO change notices. The vendor sends the customer bills of lading and invoices with extra copies for the customer. Finally the customer sends a paper check to the vendor ending the transaction cycle.

Most companies use computers for data processing and document preparation. A paper-based purchase order usually follows the scenario illustrated in figure 1.

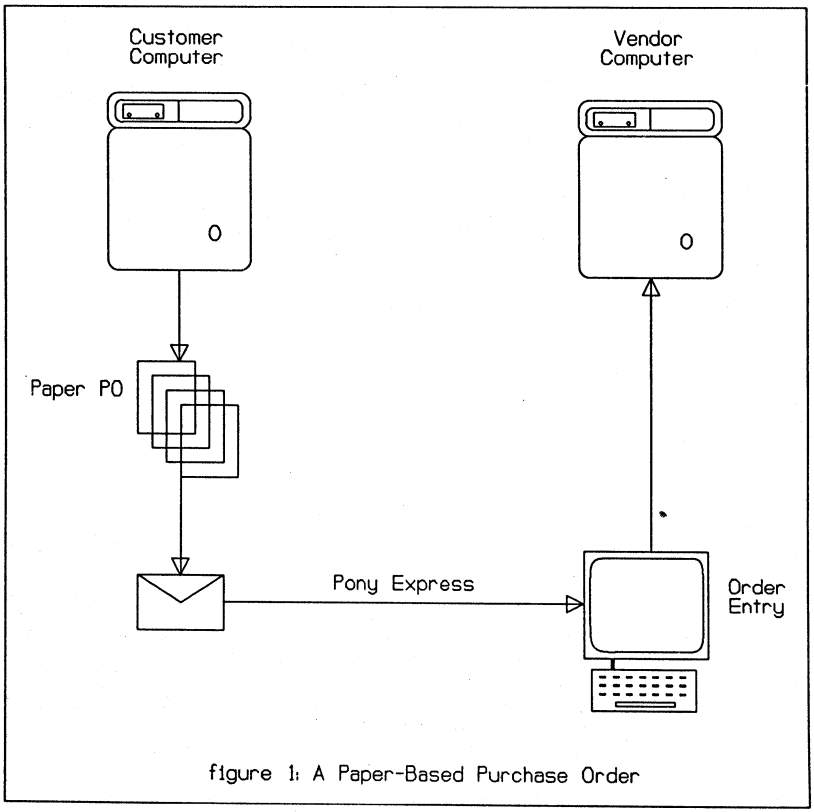


figure 1: A Paper-Based Purchase Order

2. The Nature of an EDI-Based Transaction

An EDI purchase order cycle generally involves the same functions as the paper-based system, yet they are handled in a rather different manner. The customer's computer calls the vendor's computer, either directly or indirectly through a Value-Added Network (VAN), to transmit the purchase order. The vendor's computer automatically records the customer's PO as an order and responds with an order acknowledgement. Changes to the purchase order trigger change notices from the customer's computer to the vendor's computer. When shipments are made, the vendor's computer notifies the customer's computer of the bills of lading and invoices. Then, ideally, the customer's computer notifies its bank to authorize an Electronic Funds Transfer (EFT) to the vendor's bank which then transmits a payment remittance notice to the vendor's computer. This simplified EDI purchase order cycle is illustrated two ways in figure 2.

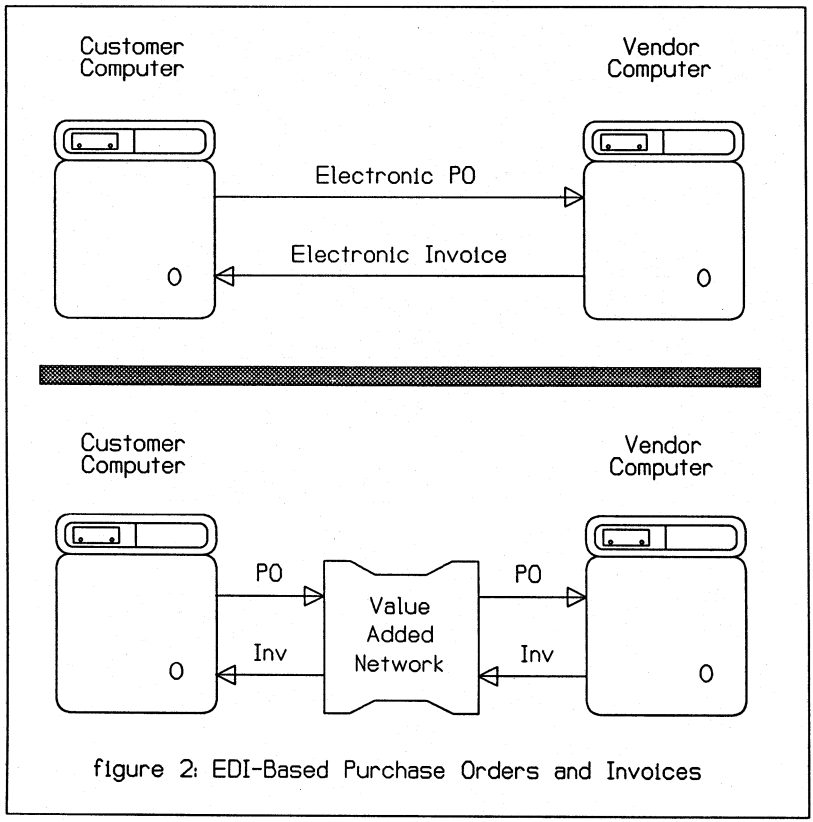


figure 2: EDI-Based Purchase Orders and Invoices

3. Format Comparison of Paper and EDI Invoices

To directly compare the 'look and feel' of an EDI transaction to a paper transaction, consider figures 3 and 4. The paper transaction is fairly standard with a definite 'header' area followed by the line items and a 'summary' area. The EDI transaction incorporates all of the pertinent information from the paper document into electronic form using the X12 standard.

Smith Corporation
900 Easy Street, Big City NJ 15155
(618) 555-6765

INVOICE

No. 1001

BILL TO: Acme Distributing Company
P.O. Box 33327
Anytown NJ 44509

SHIP TO: The Corner Store
601 First Street
Crossroads MI 18106

CUSTOMER ORDER NUMBER	ORDER DATE	INVOICE DATE	TERMS
P98932	6/26/88	7/13/88	2% 10 Days, Net 30

QUANTITY	UNIT	PART NO	DESCRIPTION	UNIT PRICE	TOTAL PRICE
3	Cse	6900	Cellulose Sponges	12.75	38.25
12	Ea	P450	Plastic Pails	.475	5.70
4	Ea	1640X	Yellow Dish Drainer	.94	3.76
1	Dz	1507	6" Plastic Flower Pots	3.40	3.40
			Please direct correspondence to:		
			C.P. Jones		
			(618) 555-8230		

PLEASE PAY THIS AMOUNT $ 51.11

DATE SHIPPED 7/12/88

ORIGINAL

SHIPPED VIA Consolidated Truck

figure 3: A Paper Invoice

EDI Format	Explanation
ISA*00*0000000000*01*passwords*01*123456789* 01*987654321 *880713*2210*U*00200*00008*0*P*~N/L	Interchange Control Header
GS*IN*012345678*987654321*880713*2210*000000001*X*0020000 N/L	Functional Group Header
ST*810*0001 N/L	Transaction Set Header
BIG*880713*1001*880626*P98932 N/L	Date/Order Date/Inv No/PO No
N1*BT*ACME DISTRIBUTING COMPANY N/L N3*P.O. BOX 33327 N/L N4*ANYTOWN*NJ*44509 N/L	Bill To address
N1*ST*THE CORNER STORE N/L N3*001 FIRST STREET N/L N4*CROSSROADS*MI*48106 N/L	Ship To address
N1*SE*SMITH CORPORATION N/L N3*900 EASY STREET N/L N4*BIG CITY*NJ*15155 N/L	Selling Party
PER*IC*C.D. JONES*TE*6185558230 N/L	Correspondence To
ITD*01*3*2**10 N/L	Terms of Sales
IT1**3*CA*12.75**VC*6900 N/L IT1**12*EA*.475**FC*P450 N/L IT1**1*EA*.94**FC*1640Y N/L IT1**1*DZ*3.4**VC*1507 N/L	Line Items
CAD*M****CONSOLIDATED N/L	Freight Carrier
TDS*51111 N/L	Invoice Total
CTT*4*20 N/L	Hash Totals
SE*21*0001 N/L	Transaction Set Trailer
GE*1*0000001 N/L	Functional Group Trailer
IEA*1*000000008 N/L	Interchange Control Trailer

figure 4: An EDI Invoice

4. Summary of Comparison of EDI to Paper-Based Systems

A paper transaction is characterized by:

a) slow communication

b) error-prone rekeying of information

c) high costs of postage, data entry personnel, data error correction and paper

d) increased inventories due to the excessive administrative lead times

An EDI transaction is characterized by speed, accuracy and low operating costs. Some specific benefits of EDI are:

a) dollar savings of reduced labor and materials for outgoing transactions

b) reduced labor for data entry of incoming transactions

c) reduced need for safety stock of inventories due to shorter lead times

d) probable shorter payment cycles

e) improved customer service

f) increased productivity in sales and purchasing

g) improved business efficiency

h) a strategic advantage of doing business faster and more accurately

C. Implementation Issues

1. Top Management Support

In view of the large number of business functions incorporated in EDI, a top level perspective is necessary to coordinate all the departments affected by the rerouting of internal and external information. This executive commitment is also required to negotiate the agreements between trading partners.

A centralized source of EDI expertise and information also will reduce duplication within departments and substantially lower setup costs. Additionally, this will provide a unified entity to which trading partners may communicate.

2. Value-Added Network (VAN) Evaluation

If your company will be communicating directly to one trading partner for all of your EDI transactions, then a VAN is not needed; otherwise, a VAN is probably in your future.

The major advantages provided by a VAN are consistent communication protocols; mailbox services for 24-hour transmissions; files edited for standards; and ready access to a large number of trading partners.

The questions to ask when deciding which VAN to select are:

a) Will I reach and be reached by the trading partners with which I will be doing business?

b) Does the VAN provide constant uptime, providing backup services in emergencies?

c) Are my transmissions secure?

d) How easily can I communicate with a trading partner that is not a customer of this VAN?

e) What are the operational costs for this VAN?

3. Legal Issues

Using a paper-based system, contract terms and conditions are usually part of the written documents sent to trading partners. However, with EDI the contract terms and conditions should be negotiated with each trading partner only once. These terms should spell out the responsibilities and liabilities of each partner. The document should contain clauses stating which partner is responsible for lost transmissions, how acknowledgements are to be handled and if payments will be made through EDI using Electronic Funds Transfer (EFT) or by standard company check.

4. Security

Data encryption and authentication will not be necessary for most sites since paper documents used to travel by U.S. mail. Companies trading with the government or using electronic funds transfer will need to be more concerned about providing a very secure environment.

The other security problems associated with EDI can be broken down into two main areas:

a) Transmitting Documents Directly

If your company will be transmitting documents directly to and from your trading partners, then there exists a greater need for security of data and physical ports.

All incoming files must be editing immediately and placed in a secure location to allow only true EDI documents to be received.

The outgoing files must be edited carefully to avoid accidental corruption of vendor computers. The physical ports and terminals must be secured to restrict access only to authorized users and ensure transmitted files go through normal editing procedures. Also, times of day editing or restrictions ought to be considered for file transmissions.

b) Transmitting Documents through a VAN

If a Value-Added Network (VAN) is used, then a different set of security concerns need to be addressed. The incoming files can be retrieved by the customer when appropriate and under tighter security than allowing transmissions directly at any time of the day. Certain VANs also may edit files for EDI standards before releasing them, eliminating one avenue of threat. All files must still be edited for reasonableness, but possible malignant files should be caught by the VAN.

The area of greatest concern when using a VAN is the integrity of the VAN itself. How secure is the data flowing through the VAN? Do their backups include your files? Are their physical premises secure? If the VAN will edit for EDI standards, they must look at the data to do it.

5. Auditing

When implementing EDI, old paper-based auditing standards must be reevaluated. Gone are the signatures and initial sign-offs. In their place are the authorization fields and date and time stamps. Transactions ought to be date and time stamped at each step of the file conversion process. An electronic audit trail needs to be thought through and implemented. If necessary, internal copies of the transactions being transferred can be produced.

A process can be secure without having a 'paper trail' and even more so. Automatic reconciliation of receipt to invoice to purchase order can ensure correct payment amounts instead of manually comparing paper documents. An 'electronic trail' can be established from vendor to VAN to internal computer. However, an independent system that verifies EDI data may need to be considered.

6. Translator Software

The flow of a transaction through the computer software starts at the application, goes through the 'translator' programs and ends up at the communications software ready to transmit. The receiving end is similar in that it starts at the communications software, goes through the 'translator' and into the application. For most companies, the application is in place and the communications software can be purchased off the shelf. For many companies, the translator software will be the largest single expense implementing EDI.

The translator software needs to be able to retrieve the pertinent information for the transaction out of the application database and format the data into a EDI standard flat file (see figures 3 and 4). The translator also needs to be able to receive an EDI flat file, pull out the necessary information for each transaction set supported and record the transaction in the appropriate application database.

If your application is purchased, the vendor may provide an EDI link. If not the original vendor, possibly a third-party will provide one. If your applications were written or customized in-house, customized translator software may be necessary.

7. Costs Breakdown

The costs involved in implementing EDI can be broken down into setup and operational costs. The operational costs usually consist of VAN contract costs and software maintenance.

The one-time setup costs include:

a) the development or purchase of communications hardware and software

b) the development or purchase of translator software

c) the man-hours reevaluating current business information flows

d) the man-hours spent in various internal and external meetings

e) education and training costs

8. Pilot Setup and Evaluation

The setup and success of a pilot program can make a move to EDI much smoother. The pilot should establish definite goals within a certain time frame with one trading partner. It could start off very small implementing only one transaction in one direction, either receiving or sending. The translator program should process this one transaction correctly before supporting other sets.

A parallel system should remain in effect until the EDI program is in production for a specified time period. The pilot can then either add new trading partners for the one transaction set developed or concentrate on adding all needed transaction sets before adding other trading partners.

D. Summary

Electronic Data Interchange is not just the latest fad in the business computer community, it is here to stay. The companies who ignore the benefits of implementing EDI may find themselves at a competitive disadvantage in the not-too-distant future. We are still left with the two alternatives mentioned earlier: either install EDI willingly or install it because competition or suppliers demand it. The choice is yours.

TITLE: Development Model of a Paperless System

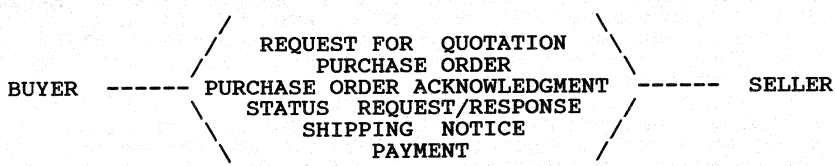
AUTHOR: Dick Pringle

Handouts available at presentation.

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING

PAPER NO. 6202

The typical business relationship between buyers and sellers (which will be referred to more correctly in Electronic Data Interchange (EDI) Terms as business partner) involves a set of transactions that could typically look like:



To begin to realise the benefits of EDI, it is important to consider the current Business Practices for moving the above data between partners. Typically, a request for a quotation may be printed on the buyers machine, vetted by someone and put into an envelope and consigned to the local postal service. This service then delivers the envelope to the seller, who opens the envelope, checks the information and may then key the data into his machine. This can result in another printed piece of paper that is checked by someone, put into an envelope and passed back through the postal system to the buyer. The buyer then checks the contents of the quotation, perhaps keys it into his purchasing system and prints out a purchase order which is validated, put into an envelope and passed to the postal system for conveying to the seller.

70% PLUS OF COMPUTER KEYED INPUT (ON AVERAGE) IS TAKEN FROM A COMPUTER PRODUCED DOCUMENT

What is really wanted is the ability to have your machine "talk" directly to your business partners' machine. Not only does this remove the problems of time associated with traditional methods of delivering data from one company to another (consider the postal system), it also can remove the need for the re-keying of data.

So Electronic Data Interchange can be described as:

THE COMPLETE COMPUTER TO COMPUTER EXCHANGE OF INTERCOMPANY (OR INTRACOMPANY) DOCUMENTS AND INFORMATION

However, a number of factors need to be considered before a business can simply implement the above scenario. Unfortunately,

THE REAL WORLD IS NOT THAT SIMPLE

The first consideration is what kind of communication/ /connectivity protocol can be used. Where your business partner has the same hardware as yourself, it is practical to consider that there will be no problem with the connectivity issue. In the case of Hewlett-Packard Series 3000's, you can use DSN, WAN, NS or even RJE (at a price of course). Yet your partner may not have the same machine as you. They could be using IBM Mainframes, UNISYS, NCR, DEC or ICL so what do you do then?

Assuming that you overcome the connectivity problems, you then have to consider the timings of the connection. The fact that your purchasing system is ready with its Purchase Orders at 4.00am does not mean to say that your supplier can actually receive them at that time, let alone process them. You may now have to plan changing your schedules to accommodate your mutual needs.

This problem of connectivity can now be compounded by the numerous types of participants. It is relatively easy to agree formats with business partners in the same or similar industry to your own. What happens when you find that you start dealing across industries. Historically different industry sectors have been able to agree formats of information for exchanging between them. You now have to think about what an Insurance Broker, a Rail Carrier or indeed what your Supplier would like (or indeed want) to see in the information passing between you and them.

One way of removing most, if not all, of the above obstacles is by using

VALUE ADDED NETWORKS (VANS)

VANS service often allows for the different types of connectivity needed by different businesses, whether they use HP, DEC, IBM etc. As they normally provide a store and forward facility, the question of timings is immediately removed but perhaps more importantly they can provide a

PUSH and PULL

facility. This allows you to decide when you want to connect, whether you are passing data into their network or inquiring on what data is waiting for you. If you then want to receive that data you execute a command set to allow it to be sent into your machine. With direct connect to a business partner machine one of you has to allow read access to the other - an area potentially dangerous (but you need to talk to your auditors about that).

However, having overcome the connectivity, timing and security issues you now have to consider the next stage - the documents themselves. Each document type has its own format. This problem can be overcome by accepting that each document does indeed have its own format and identifying an environment that allows for the different formats that are available, ie. Purchase Order, Invoice, Delivery Advice etc;

The next problem is that no two invoices are identical! Every Company that I have worked with has a different format for its own invoices, this includes separate divisions within a single Company! The information may be similar but you will rarely find the ship-to address in the same location on the different forms. But it is true to say that most of the information is consistent in that an Invoice will contain a Bill-to Address, a Remit-to Address, perhaps an Order-by Address, Registered Office Address, details of items supplied (to include product identifier, quantity, description, unit price, extended price), discounts, terms etc.

Having established the mechanism for transmission of data within an EDI environment, ie. machine to machine, you still have to overcome the problem of format of information. To allow the processing of transported data,

EDI IMPLIES THAT A STANDARD IS USED TO PRECISELY DEFINE ELEMENTS OF THE ELECTRONIC MESSAGE

Again, the real world is not that simple in that there is not just one standard. Some of the currently used standards for the formatting of messages include:

ANSI.X12	**TRADACOMMS**
TDCC - UCS	**ODETTE**
WINS	**EDIFACT**
	SPEC2000M
etc	**etc**

ANSI ASC X12 is the designation for the EDI Standard that is now commonly employed within North America. Conceived in 1978, it has developed into the leading (and most frequently used) standard available the U.S.A.

In the early 80's in Europe, the body that became known as ODETTE (Organisation for Data Exchange by Tele Transmission in Europe) also began work on defining the needs for Business Communication Messages. Although originally set up for the Automotive Industry it is used for a number of industry groups outside that sphere.

In the meantime, in the United Kingdom, the ANA (Article Numbering Association) in conjunction with a number of Industry Groups embarked upon the development of TRADACOMM (Trading Data Communication). This is now widely used in the United Kingdom by many types of Businesses.

The most important development to-date has been the creation of EDIFACT (Electronic Data Interchange for Administration, Commerce and Transportation - or Trade). This single Standard has emerged as being the only true multi-industry / multi-national Standard that is being embraced throughout the World.

WHO USES EDI NOW

The United States Grocery Industry estimated that it has saved over US$150,000,000.00 during 1987 by using EDI. These savings have been realised in both improved communications leading to better availability of Stock (which normally has a short shelf life) and reduction in errors of re-keying data.

ICI have been using EDI in various forms since 1982 and have been able to improve profitability as a result of the savings made in not spending time on errors.

Courtaulds (Clothing Manufacturer) and Marks & Spencer (Major Retail Outlet) at one time used Couriers from Stores to the Manufacturer that cost upwards of US$450,000.00 per year. The Couriers were used to ensure the timely notification of demand for a replenishment product. This ensured that orders could be processed overnight with delivery often the next day. Using EDI they can advise on requirement and often receive replenishment the same day. Apart from improved Customer Service (by the Manufacturer), Marks & Spencer no longer pay for the Courier service!

WHY MOVE INTO EDI

Perhaps the usual reason is savings in money! The examples given earlier go some way to explaining cost benefits but the most significant that we have encountered is one organisation that issued several million purchase orders a year to its suppliers. After analysis, costed each purchase order at about US$50.00 each (most companies can cost about US$30.00 per order). On further analysis it was discovered that 1.3 million purchase orders were actually being sent to seven suppliers. This represented a cost of US$65 million per year on purchase orders to those suppliers. After one year, having implemented EDI with those seven suppliers the exercise of costing the purchase orders was done again and it was found that they now (for the 1.3 million) cost on average US$15.00 - giving a saving of US$45.5 million!

Better Business also comes to mind when considering the benefits of EDI. Levi Strauss USA, clothing manufacturers, were waiting up to twelve weeks for feed-back to its corporate offices on sales. If a product was selling faster than planned, it meant that production was not being geared up to meet the demand in time. By the same token, if a product had "bombed", the knowledge that it was not selling, was not being received until too late and as much as twelve weeks of volume production had been made. By integrating EPOS (Electronic Point of Sale) data direct to EDI messaging to corporate offices, Levi Strauss now know within two weeks how a product is doing!

More efficient Business was the objective of Littlewoods Catalogue Group (a company that sells via agents direct into homes). Littlewoods wanted to improve its service to customers by making sure that a customer did not have to wait more than two weeks from placing an order to delivery. Initially, certain suppliers were targeted and a plan was put in place whereby the suppliers wrapped and delivered product' direct to the end client (as if direct) upon notification of demand by Littlewoods. When dispatched, the supplier advised Littlewoods who then billed the agent and paid the supplier. From the original objective of cutting down lead time for delivery evolved a far greater reliance on direct delivery by suppliers. This reduced the amount of inventory moving into Littlewoods warehouses, cut back on the need for warehousing (with the resultant sale of unused buildings) and did not tie capital up in inventory. Until it was moved from the supplier warehouse - it was not paid for.

STEPS TO BE TAKEN TOWARDS EDI IMPLEMENTATION

At Perwill we consider that there are six steps to go through before you can successfully implement EDI. Other people say there are seven, some as few as four. It does not really matter what the number is, here are our ideas.

1. Fully Understand EDI

   The actual level of understanding really relies on how much of the work that a Company intends to use its internal resources for. If external consultants are not to be used, then this level of understanding must be high. One method of gaining the appropriate knowledge is by joining those groups that develop standards for use Nationally or Internationally. For example ANSI X12 in North America or EDIA in United Kingdom.

2. Agree Standards with Business Partners

   (Some would argue that the first step is to find a Business Partner.) Here it is important to decide on which Standard and (if applicable) its Variants that you intend to use. Once having agreed the Standard, you then need to consider what transactions (documents - ie. Invoices, Purchase Orders?) you are going to use.

3. Modify Existing Systems

This is by far the largest part of the EDI implementation phase. Unless your existing system can accept Batch Input of say, Purchase Orders, you will have to print out Purchase Orders received from Customers and key them in (which defeats one of EDI's objectives). By the same token, if you intend to send Invoices, you must be able to isolate those Invoices for EDI transmission from those that will be printed in the conventional manner.

4. Translate the Data

Having received an EDI Package it must now be translated to a form compatible to your application. By the same token, having isolated the data to be sent to a Business Partner you have to translate it to an EDI Standard Format. This set of translations can be done using a number of software packages available in the market today.

5. Prepare Communications

Having created your EDI Package it has to be delivered to your Business Partner. Also, any packages of Data from your Business Partner have to be delivered to you. Either you plan to have direct connection or you use the services of a VAN.

6. Control & Audit the Whole Process

You must be able to monitor exactly what has been sent to a Business Partner, in the event that the transmission failed (for whatever reason) that you are able to resend the data, you should be able to functionally acknowledge what your partner has sent to you etc;

To reiterate the steps we identify:

1. Fully Understand EDI
2. Agree Standards with Business Partners
3. Modify Existing Systems
4. Translate the Data
5. Prepare Communications
6. Control & Audit the Whole Process

WHATS SO GREAT ABOUT EDI

Having looked at a number of the implications of using EDI, its worth considering one or two more to reinforce the benefits that can be achieved:

MUTUAL BENEFITS

So far the examples provided have only looked at one side of the Business Usage of EDI. Another example, that comes to mind is the situation where one company was sending an average of 40,000 Invoices a month to one Customer. It was taking anything up to 120 days to pay the invoices. The Customer was also having to use a data entry group of eleven people to key the invoices in, upon receipt - with its associated delays and errors. EDI seemed an ideal way of improving processing for both parties.

Each month, the full detail of the 40,000 invoices is passed by EDI to the Customer with four batch totals (about 10,000 Invoices per Batch). Four cheques for payment are raised within three/four working days plus/minus any adjustments from the previous month. The Customer modified its existing Payables System to receive directly into it, the detail invoices and to automatically match against goods receipts. Most of the eleven staff were reallocated to other tasks within the Company, with those remaining concentrating on the mis-matches reported from the Payables System. The earlier payment was justified by the reduced cost of personnel resourcing, the creation of the EDI packages by the Seller, had its cost recovered by the improved cash-flow.

EFFECTIVE COMPETITION

There have been situations where not to use EDI has lost Companies Business. FORD in the USA declared that all suppliers would operate using EDI by the end of 1988. Those that were not prepared to no longer do business with Ford!

One US Corporation that supplied Hospitals throughout North America, although retaining its market value was losing ground in its market share. In 1947 they had about 40% of all Hospital Business but by 1984 they only had 18% (but still same value). They embarked upon a very aggressive strategy that involved the creation of a Network that all Hospitals could tie into with their existing systems (and where the hospitals did not have systems, simple systems were provided). Now, when a Ward Sister has a need for a bandage, she simply keys into a terminal and any understock level will automatically raise an order with the Network Supplier. Allied to excellent quality and comparable prices to their competition, the Network Supplier now has 45% of the market share and it grows each year. They are currently squeezing out their competition!

GOOD BUSINESS

The US Corporation that supplies Hospitals is an excellent example of strategic use of EDI.

Consider that about 80% of administration is spent on tracking the errors often generated by incorrect keying of data. Some organisations pride themselves on only having 2% of errors - but they still use an unrealistic amount of resource in correcting the problems. EDI will not remove errors that originate at source (if someone orders 10,000 instead of 100, EDI is not normally set up to catch that kind of problem), but it will remove the errors of rekeying.

Look at the Littlewoods example. Instead of just improving Customer satisfaction, they were able to release capital by selling no longer used warehouses - something that was not planned for in their early stages of EDI.

CONSIDER THE FOLLOWING

Just as an aid memoir to those readers considering embarking upon EDI it is worth answering the following questions:

- What does your Company do?
- Who are your Trading Partners?
- Which Transactions will you be using?
- Do you have any Special Interests?
- Do you belong to a Trade Association?
- Will you be using in-house, third party or off the shelf translation software?
- Will you be using Direct Connection or a VAN?

As a final note of caution

IT MAY COST MORE THAN YOU THINK

WHAT IS EDI

ELECTRONIC DATA INTERCHANGE

THE COMPLETE COMPUTER TO COMPUTER EXCHANGE OF INTERCOMPANY (OR INTRACOMPANY) DOCUMENTS AND INFORMATION

To begin to realise the benefits of EDI, it is important to consider the current Business Practices for moving the data between partners. Typically, a request for a quotation may be printed on the buyers machine, vetted by someone and put into an envelope and consigned to the local postal service. This service then delivers the envelope to the seller, who opens the envelope, checks the information and may then key the data into his machine. This can result in another printed piece of paper that is checked by someone, put into an envelope and passed back through the postal system to the buyer. The buyer then checks the contents of the quotation, perhaps keys it into his purchasing system and prints out a purchase order which is validated, put into an envelope and passed to the postal system for conveying to the seller.

What is really wanted is the ability to have your machine "talk" directly to your business partners' machine. Not only does this remove the problems of time associated with traditional methods of delivering data from one company to another (consider the postal system), it also can remove the need for the re-keying of data.

However, a number of factors need to be considered before a business can simply implement the above scenario. Unfortunately,

THE REAL WORLD IS NOT THAT SIMPLE

The first consideration is what kind of communication/ /connectivity protocol can be used. Where your business partner has the same hardware as yourself, it is practical to consider that there will be no problem with the connectivity issue. In the case of Hewlett-Packard Series 3000's, you can use DSN, WAN, NS or even RJE (at a price of course). Yet your partner may not have the same machine as you. They could be using IBM Mainframes, UNISYS, NCR, DEC or ICL so what do you do then?

Assuming that you overcome the connectivity problems, you then have to consider the timings of the connection. The fact that your purchasing system is ready with its Purchase Orders at 4.00am does not mean to say that your supplier can actually receive them at that time, let alone process them. You may now have to plan changing your schedules to accommodate your mutual needs.

This problem of connectivity can now be compounded by the numerous types of participants. It is relatively easy to agree formats with business partners in the same or similar industry to your own. What happens when you find that you start dealing across industries. Historically different industry sectors have been able to agree formats of information for exchanging between them. You now have to think about what an Insurance Broker, a Rail Carrier or indeed what your Supplier would like (or indeed want) to see in the information passing between you and them.

WHY STANDARDS

The next problem to consider is the **Numerous Document Types.** However, having established that there are several documents that can be considered for the EDI environment, there is then the issue of the **Various Document Formats.** No two invoices are identical! Every Company that I have worked with has a different format for its own invoices, this includes separate divisions within a single Company! The information may be similar but you will rarely find the ship-to address in the same location on the different forms. But it is true to say that most of the information is consistent in that an Invoice will contain a Bill-to Address, a Remit-to Address, perhaps an Order-by Address, Registered Office Address, details of items supplied (to include product identifier, quantity, description, unit price, extended price), discounts, terms etc.

EDI Will Only Process Documents Within Supported Standards.

EDI IMPLIES THAT A STANDARD IS USED TO PRECISELY DEFINE ELEMENTS OF THE ELECTRONIC MESSAGE

WHAT IS AN ANSI STANDARD

Before examining the Standard, it is worth considering exactly **What Is ANSI?**

It is a **Voluntary Standards Organisation** that **Includes Private and Public Sector** with the objective of creating **Consistent Standards** by the **Consensus Process.**

It may be appropriate to consider the historical development of EDI. In the USA, several standards evolved to meet the needs of the different Industry Sector Users. The United States Grocery Industry adopted UCS, the Warehousing and forwarding groups adopted WINS and the Transportation Group employed TDCC (Transportation Data Coordination Committee).

In the late 70's it was recognised that it was no longer appropriate to have the different groups in the USA and they became consolidated into what has been identified as ANSI ASC X12 (American National Standards Institute Accredited Standards Committee X12 - where X12 is a simple project team designation).

The X12 National Standards can be considered to be comprised of three elements, which are:

Transaction Sets (both creation and maintenance); **Data Dictionary/Segment Directory** and **Transmission Controls.**

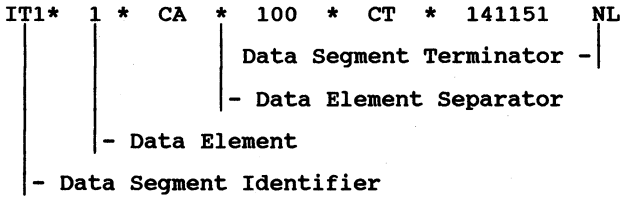
To begin to appreciate these terms it is necessary to look at the EDI environment, where the EDI transmission can consist of several FUNCTIONAL GROUPS which may be likened to sets of Invoices, Purchase Orders etc (Different Documents). Within the FUNCTIONAL GROUP will be the individual TRANSACTION SETS (the Invoices, as distinct from several document types). Yet each TRANSACTION is made up of DATA SEGMENTS, which can be repeated (or LOOP), for example Invoice Address, Invoice Item Line etc; and finally DATA ELEMENTS that are the unique fields within the DATA SEGMENTS. Here a DATA ELEMENT could be a Quantity, Unit of Measure, Description, Unit Price etc; It may be useful to consider DATA ELEMENTS as fields within Records (DATA SEGMENTS) that are held within a file of Invoices (TRANSACTION SETS).

X12 Segment Directory Maintains information about the construction of each logical record (Segment) within ANSI X12. The primary information sets associated with the logical record includes, **Segment Identifier, Segment Title, Purpose, Data Elements (and whether Mandatory/Optional/ Conditional)** and **Transaction Sets Where Used.**

From here it is possible to build a DATA SEGMENT which may look like:

	Quantity	Unit of Measure	Unit Price	Catalogue Number	Description	
IT1*	Quantity Invoiced	Unit of Measure	Unit Price	Product ID Qualifier	Product ID	NL
	1	Case	$1.00	Catalogue Number	141151	
IT1*	1	CA	100	CT	141151	NL

STRUCTURE OF AN X12 DATA SEGMENT



Having described the actual content of an EDI Message we have to now consider how the data gets from one location to another. EDI messages are delivered using an Electronic Envelope.

The **X12 Data Dictionary** contains a further set of information that allows the qualification of the unique elements of data presented in the Segment. This qualification includes **Reference Number** (Unique to each element), **Data Element Name, Type, Length (Minimum/Maximum criteria), Description** and **Code Table.**

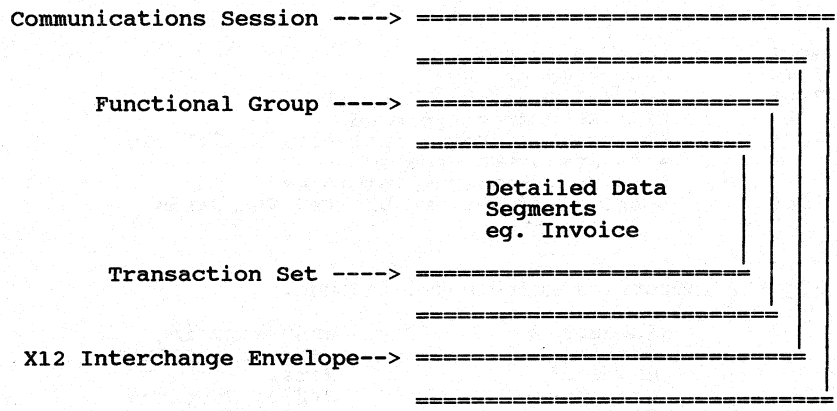
A Sample Invoice of the information Date: July 13th, 1985, Number 1001, Terms of Sale: 2% 10, Days would be shown as:

BIG*850713*1001n/l followed by **IT9*01*03*210n/l**

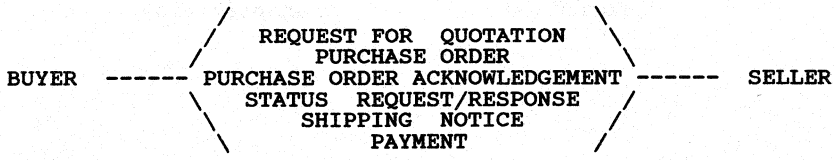
HOW DOES IT GET THERE?

In the Envelope!

The following illustration presents the above information within the constraints of the **X12 INTERCHANGE CONTROL.**



SOME ANSI X12 TRANSACTIONS



WHAT IS ASC X12

Approved by the American National Standards Institute (ANSI), supported by Vendors, Cross Industry utilisation providing Multiple Functions and allowing for Data Communications Flexibility.

ANSI X12 Development took the following Route:

1978	Founded by CRF
1979	Chartered by ANSI
1981	Initial Standards Completed
1983	Five Standards Approved
1984	Alignment of Data Dictionaries with JEDI
	* 5 Standards Updated
	* 13 New Standards Completed
1986	Approval of New and Updated Standards

ASC X12 Industries Represented include:

Automotive	Manufacturing
Banking	Medical Supplies
Chemical	Metal
Computers	Office Supplies
Distribution	Petroleum
Drugs	Publishing
Electrical	Retailing
Electronics	Semiconductor
Grocery	Software
Government	Telecommunications
Information	Transportation

And Many More

ASC X12 Trade Associations Represented include:

Aluminium Association
American Hardware Manufacturers Association
American Iron & Steel Institute
American Paper Institute
American Supply & Machinery Manufacturers Association (ASMMA)
Automotive Industry Action Group (AIAG)
Book Industry Systems Advisory Committee (BISAC)
Graphic Communication Association
Information Industry Association
Motor & Equipment Manufacturing Association (MEDA)
National Association of Electrical Distributors (NAED)
National Association of Purchasing Management-Rail Industry Grou
National Association of Service Merchandising
National Automated Clearing House Association
National Electric Manufacturer Representative Association
National Electric Manufacturer Association
National Industrial Distributors Association (NIDA)
National Office Products Association (NOPA)
National Retail Merchants Association (NRMA)
National Wholesale Druggist Association
Southern Industrial Distributors Association (SIDA)
Wholesale Stationers Association

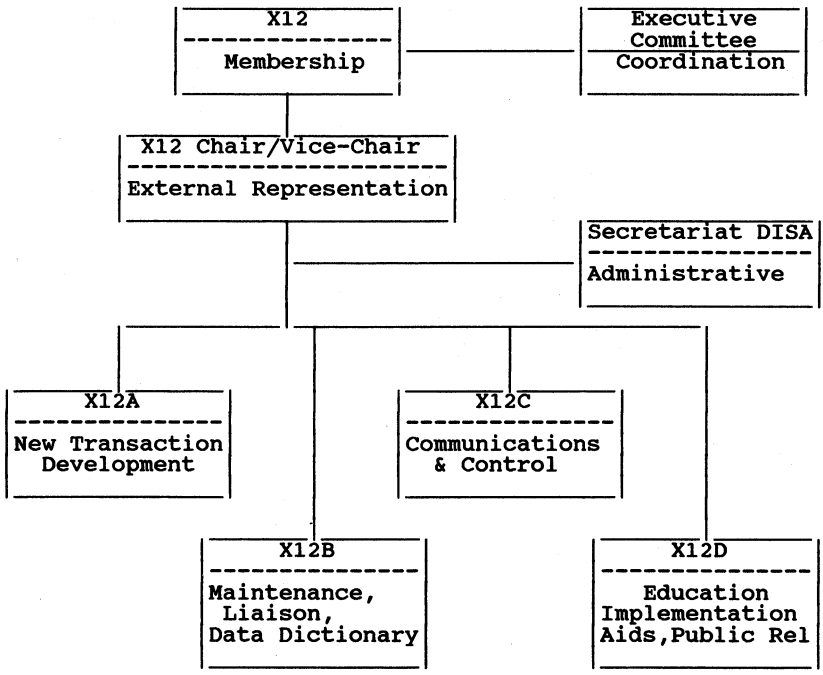
WHY JOIN ASC X12?

By joining the process you will **Increase Understanding, Improve Implementation, Stay Up to Date On Enhancements, Share EDI Ideas With Other Participants** and **Influence Further Improvements.**

WHAT IS THE ASC X12 PROCESS?

There are two facets to the ASC X12 Process. The first is **Organisation** and the second is **Standards Development and Approval.**

ASC X12 Organisation



PROJECT GROUPS

Include:

FINANCE

For Remittance or Banking Functions
- Invoice
- Remittance/Payment Advice
- Financial Transaction Reporting
- Customer Account Analysis

INTERNATIONAL

- Analyse/Resolve Differences in EDI Syntax and Format to Allow Exchange of EDI Data Between U.S. and Europe

MATERIALS MANAGEMENT

For the Control of Materials Movement
- Planning Schedule and Release Capability
- Ship Notice/Manifest
- Shipping Documentation
- Order Status
- Shipping Schedule

PURCHASING

- Price/Sales Catalogue
- Request for Quotation
- Purchase Order
- P.O. - Sales Inventory Calculated
- P.O. - Acknowledgement/Change Request/Change Acknowledgement

PRODUCT DATA

For Miscellaneous Product or Text-Related Information
- Quality Information
- Text Information

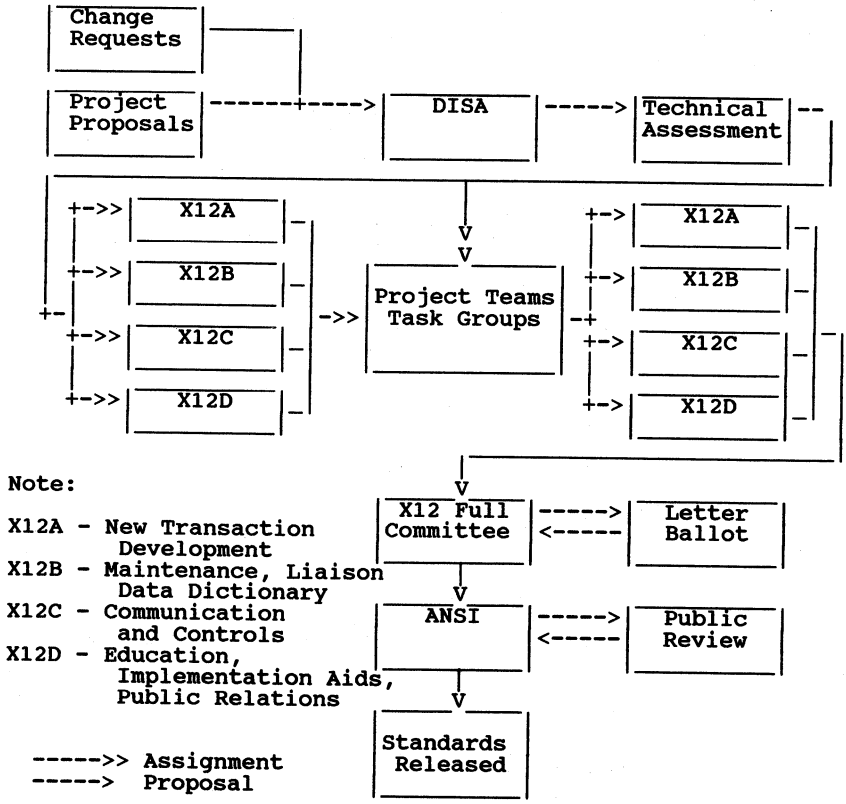
TECHNICAL ASSESSMENT

- Advise on Documentation
- Oversee Data Dictionary Segment Directory
- Oversee Data Element Dictionary

TRANSPORTATION

For Communication Among and Between Carriers, Shippers and Receivers
- Assess feasibility of Generic Transaction Sets
- Work with International Team on Standards

ASC X12 Standards Development



HOW TO GET THE MOST OUT OF AN ANSI MEETING

CONSIDER THE FOLLOWING

- What does your Company do?
- Who are your Trading Partners?
- Which Transactions will you be using?
- Do you have any Special Interests?
- Do you belong to a Trade Association?
- Will you be using in-house, third party or off the shelf translation software?
- Will you be using Direct Connection or a Third Party Network?

EXAMINE AGENDA

- Attend Seminar
- Attend Subcommittee or Task Force Meetings
  - Typical Meeting Structure
  - Membership
- Attend Friday Wrap-Up

INTEGRATED VOICE AND DATA

James S L Cohen
Mecca Leisure Ltd
76 Southwark Street
London SE1 OPP

Telephone (44) 1-928 2323

The function of this paper is to look at some of the options available to you, should you find yourself in the fortunate position of being able to rewire your terminal population.

I do not pretend to cover all the options, but just some of those that are currently available. Furthermore, the options considered as those available in the United Kingdom and there could be some difference else where in the World.

It is not my intention to cover technical aspects of integrated voice and data, unless totally necessary.

So now lets start looking back through the mists of time, before there was a terminal on your desk, to the time before the HP3000, the PC or indeed the microchip .....

Once upon a time, there was just one cable running to each desk, attached to the wire was a telephone handset. The wire was run back through the building (if you were lucky through ducting) to the telephone exchange frame, before going through the telephone exchange to the outside world. Usually the physical link consisted of a cable with two pairs of unshielded wires.

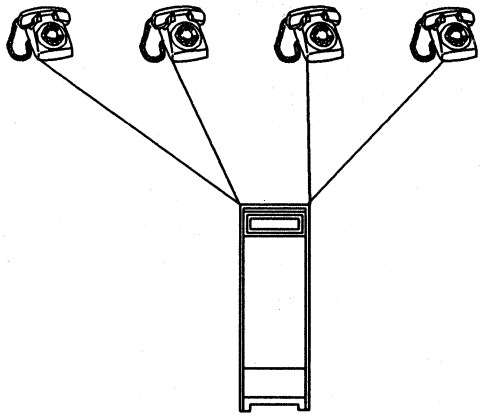


Figure 1

Some time later your company bought its first (HP3000) computer and started placing terminals on some desks. The computer required another cable from the desk this time not to the telephone exchange, but to the computer room. In the early days this cable consisted of many pairs of wires, usually shielded, to provide the data link with the necessary handshaking.

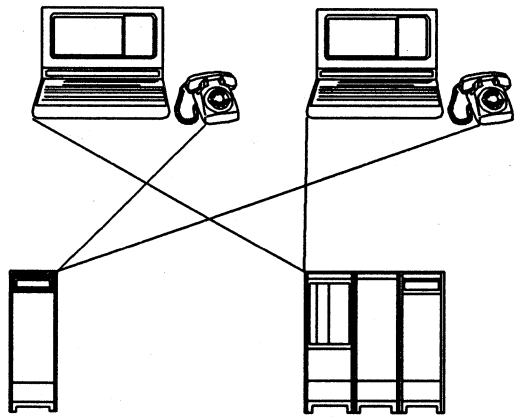


Figure 2

Then later still came the PC, initially no cable was required, but soon it took over in many cases from the terminal. Then later still came networks of PC's, used for shared resources and for multi user applications. These were networked by a number of means, whether coax for networks such as Novell or "telephone cable" for networks such as StarLan.

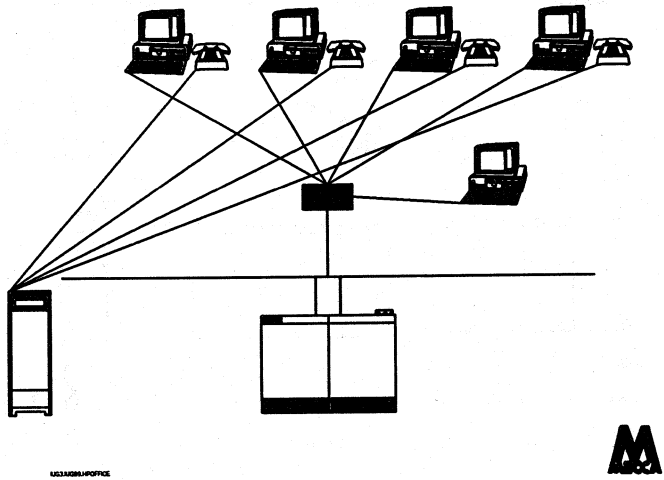


Figure 3

Before long the only certainty was that under floor you had miles of cable, tied in knots, wrapped around your offices. Much of it redundant at any one time.

If you had multiple HP3000's, then to reduce the loading on the systems caused by DS (and later NS), then you might well have considered installing a data switch in front of your processors. Using this users could select which system they required access to.

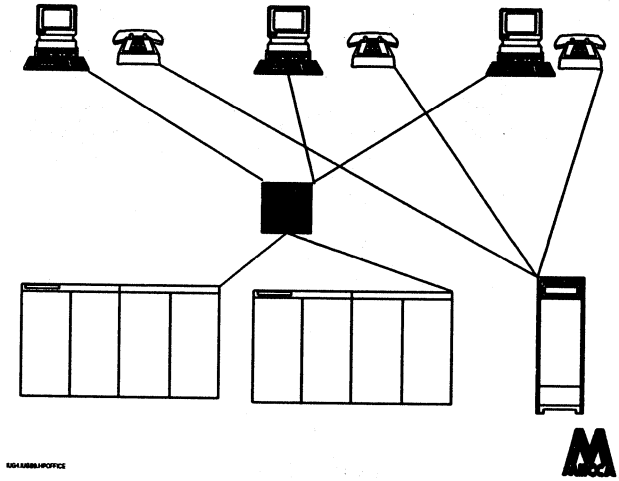


Figure 4

Lets now look at some of the options available to you in 1989, and where integrated voice and data networks can be very cost effective, particularly in a distributed environment.

To reduce the number of options, I have assumed that the site is running Precision Architecture HP3000(s). Solutions for Classic HP3000(s) are available, but are generally more complex.

Option One

Use point to point wiring, from PC's/terminals into Data Terminal Controllers (DTC's). Since the DTC's need not be geographically close to the HP3000's, the asynchronous cable runs can be quite short. The DTC's could, for example, be located on each floor, with horizontal cable runs to the DTC's. The DTC's can then be connected to the HP3000's via either Thick or Thin Lan.

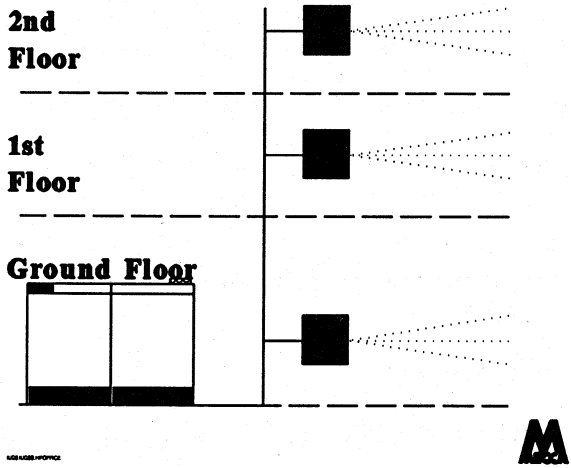


Figure 5

Making use of LAN bridges, the DTC's could be geographically in a remote location, with digital data circuits providing the bridge between one LAN and another.

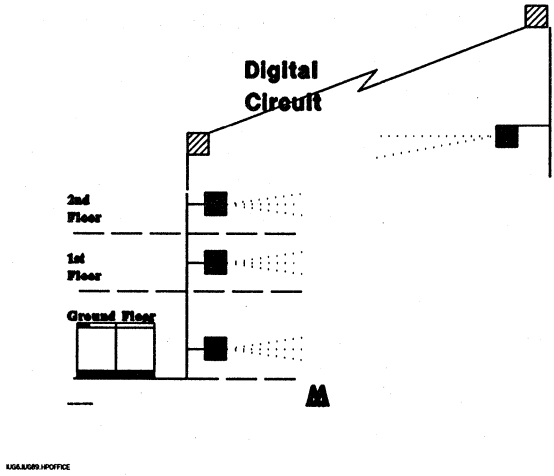


Figure 6

Pro's

- No complex networking
- DTC's switchable between HP3000 (from DTC 2.0 release)
- Provides simple bridging between remote locations

Con's

- High wiring costs, particularly if frequent office moves occur
- Little flexibility
- Relatively long lead times on installation
- No PC networking, for shared resources

Option Two

DTC's are used to provide terminal access, but PC's are to be used as intelligent terminals, these being networked together (either with StarLan or Novell) and then connected to the HP3000. Again networks of PC's on remote sites can make use of LAN bridges to access the main HP3000 computers.

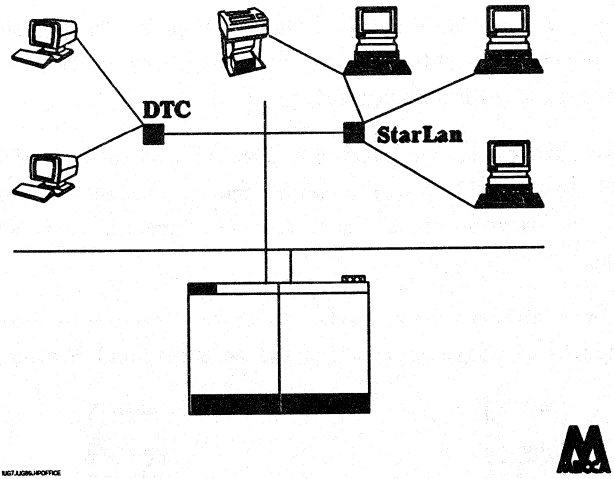


Figure 7

Pro's

- Provides shared PC resources ie shared peripherals (such as printers) and shared PC data
- Allows PC backup via the network onto the HP3000
- Provides disc sharing services on HP3000
- Remote PC networks can bridge to LAN, via LAN bridges and digital data circuits

Con's

- Need for network servers
- Overhead on HP3000 LAN
- PC's and terminals are not easily interchanged, since the cables are not compatible.

Option Three

Make use of Data Over VoicE (DOVE). This system multiplexes data and voice over the same pair of telephone cables, since voice traffic occupies only a small band width. This removes the need to wire for data traffic, but has many of the same restrictions of direct

connections, via a DTC. However, assuming that at the exchange a patch panel is utilised, service can be provided, very quickly to any desk where there is a telephone handset.

Where there are no remote locations DOVE is highly cost effective and very easy to install and maintain, however cannot be extended to terminals on remote sites. Nor does it provide any sort of PC networking.

The costs for DOVE are very similar to that of hardwiring, but with the added benefits of easy installation and no additional wiring.

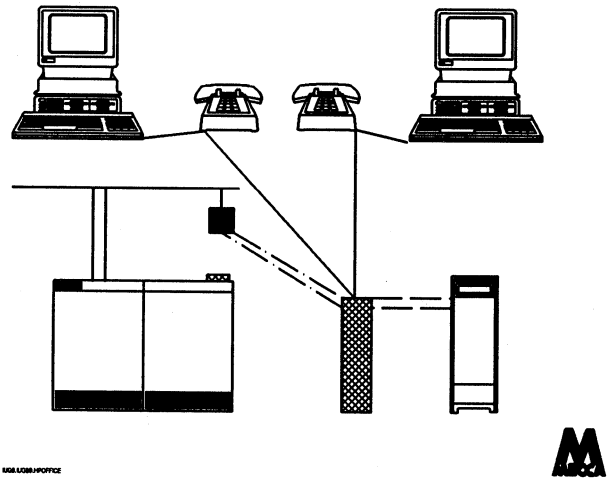


Figure 8

Pro's

- No complex networking
- DTC's switchable between HP3000's (from DTC 2.0 release)
- Fast installation of additional devices
- Easy reconfiguration following from office moves
- No wiring costs, beyond that already incurred for telephone handsets

Con's

- No PC networking, for shared resources
- No remote access

Option Four

The wire running to your telephone has a very wide band width, only a small percentage of which is used for voice traffic, this leaves the remaining band width for other traffic. If you install a suitable handset on each desk and a suitable switch, then you can use your central switch for both voice and data switching. Thus you achieve all the benefits of data switching, without the need to purchase a second data dedicated switch. At the same time wiring costs are kept at the absolute minimum.

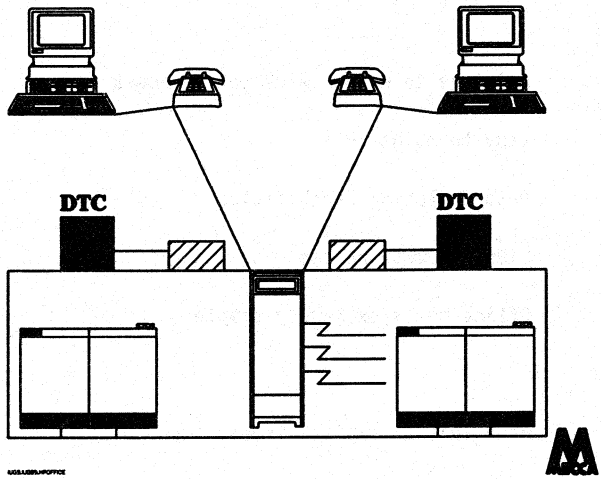


Figure 9

Should you have any remote sites, so long as they have a compatible switch, with the necessary digital network to interconnect the switches, then it is quite feasible to switch from one computer to another without the need of NS or a separate data network.

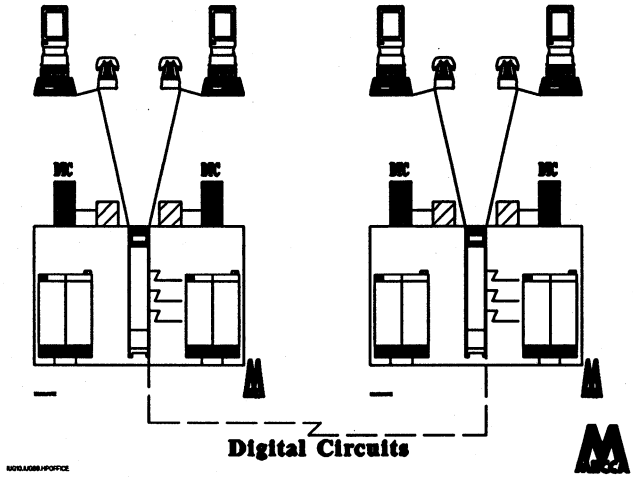


Figure 10

Pro's

- Only one telephone wire to each desk
- Easy to maintain
- Data Switching Capabilities
- Simple remote access
- Office moves extremely simple

Con's

- High initial Capital expenditure
- No PC networking

Now lets look a little further into what makes up an integrated voice/data switch and into the implications of such a purchase, and look into when it might be considered a cost effective alternative to point to point wiring.

The first feature of a voice/data switch is that it must functionally replace your current voice only switchboard. It must have no less functions than those you already enjoy on your current switchboard, such as :

Call Queuing	Call Forwarding
Call Transfer	Abbreviated Dialling
Camp-on when Busy	Call Hold
Call Transfer	Conferencing
Hunt Group	Distribution Group

Many of these features that you have long been using for voice calls are now available to you for data as well. For example.....

> *You are sitting in front of your terminal, and want to access a data base on a computer that is at another location.*

Previously you would have logged onto the computer at your location and then opened a link to the other computer and then logged onto the second computer, with the implied loading on the first. Not now .....

> *You dial a number on your handset, that is the internal number for the remote computer. The voice/data switch establishes a call across a digital circuit that is already carrying voice traffic to the remote site, and so long as there is a port available it will be allocated to you.*

No need to open multiple sessions, no increased load on your local computer, and no duplicated networks. Since previously you would have needed one network to carry voice traffic and a further network to carry data.

Now while it might not be considered cost effective to install an integrated voice and data network, if all your computers are located in a single computer room, once the computers are distributed across a network, the benefits start being realised. If this is likely to be

case, it is not necessary for all the switches to be from the manufacturer. However, for the switches to work they must all reach the same command standard for the interfaces (usually that defined for ISDN). Also the switch should offer at least the following interfaces for connection to the outside world V.24, X.21, and X.25.

It is quite likely that you are already using digital circuits between remote sites, as a cost saving on voice traffic. This would allow handset to handset communications, without the need to use the public network.

The same principle can be applied for the data network. While you may have a circuit between the computers, it too would be dedicated for just data, while the other circuit is dedicated to just voice, using an integrated voice and data exchange at both ends, a single digital circuit can carry both types of transmission. It is as simple as making a telephone call to another internal extension.

From an HP3000 systems management view point, it not only reduces the load on the networked HP3000's, it automatically provides port contention, reducing the costs of your HP3000. In our experience a port to terminal ratio of 1:4 is quite acceptable, which in a 200 terminal installation would save in the region of $60,000, just in DTC savings.

So it would appear that integrated voice and data installations have quite a number of features that are of as much value to an MIS Manager as they are to a voice networking manager. However, there are viable alternatives, for example, if you are predominately PC based, then PC networks may provide you with an alternative - but only if you genuinely require a PC network. You should consider the fact that a PC network costs in the region of $500 per networked PC, which is more than the cost of integrated voice and data. A PC network, if remote to the HP3000's also requires LAN bridging if it to communicate with your central computers, and a further bridge may be required for voice traffic.

If you are responsible for a distributed network where both voice and data communications justify leaseline circuits, then an integrated system not only is cost effective, but easily maintainable.

It does not, however, provide a solution for those users where PC's predominate. In those locations PC networks provide the most flexible networking solution (since there is peripheral sharing etc) as well as access to computer networks. Data and terminal access can be provide by NS, while voice networking can be handled by your private voice exchanges.

CONCLUSION

In many installations, the opportunity of replacing an obsolete voice switch with a fully integrated voice and data switch, will not only reduce your terminal wiring costs, it will increase your productivity (since the load on multiple systems would be reduced). However against these benefits must be weighed the increase cost of the switch, over the cost of traditional point to point networking.

The decision on terminal/PC networking is company specific, but it is hoped that this paper has given you some ideas on some of the options available to you.

BIOGRAPHY - JAMES COHEN

Graduated with degree in Mathematics and Computer Science from Brunel University in 1980. Since then working on HP3000, initially as a programmer/analyst with a consultantcy, and since 1982 with Mecca Leisure being Systems Manager since 1986.

Mecca Leisure is the largest Leisure company in the UK with a turnover of some $2 billion and making $300 million profit. As a group we operate 2, HP3000/950's 2,HP3000/70 and 1, HP3000/925, located on two sites (one in London and one in Southampton).

Responsible for HP3000/950, 925 and 70 located at Head Office in London, with 200 branches connected online via X.25 network.

When not running the computers, am an officer in the Royal Naval Reserve, and am the units Gunnery Officer responsible for the small arms training and ceremonial for a unit of 750 officers and ratings (enlisted men and women). In the week following the conference will be acting as a Liaison Officer on the staff of SACLANT at Norfolk, during a large NATO exercise !!

WORKING EFFECTIVELY WITH YOUR SOFTWARE VENDORS

by Kathy S. McKittrick
McKittrick Associates
5547 South Yampa Street
Aurora, Colorado 80015

Over the course of the last ten to fifteen years, DP shops have become increasingly dependent on the use of third party software. Why? Because the demand for computerized solutions to business problems have increased ten-fold. Fifteen years ago, many companies used there computer systems exclusively for accounting applications. Many small to mid-size companies didn't use in-house computer systems at all. This has changed dramatically. These days it is common for in-house computer systems to handle applications such as word processing, inventory control, sales order processing, payroll, decision support; the list goes on and on. At the same time, it has become apparent that in-house development of all the applications and tools that are required to support such an environment is not only cost prohibitive, but next to impossible.

As the trend to purchase rather than build software continues into the 1990's, a data processing department's success in delivering applications to their end users will be dependent upon their ability to work with their software vendors, to a large degree. While this might seem like an exaggerated statement, when you consider all of the components of a software product, it rings true. Let me give you an example.

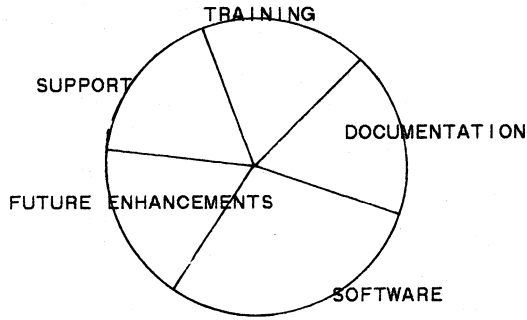
Let's say that you have a software tool that you're currently using. Let's also assume that it works perfectly, you haven't had to call the vendor's support line in months, and you have no complaints. How can this vendor have an impact on your future success?

First of all, you may have a requirement in the future to use this software tool in a different way. You made need to interface the product to other new software that you purchase. This can take you back to square one, in terms of a learning curve, support requirements, and enhancements that the Vendor has or hasn't made to the product.

Secondly, what about turnover in your staff? The "experts" on this software tool may leave your company, and new staff may come on board. Will that new staff need training? How effective is the documentation? Will new people be able to read through it and figure out how the software operates?

Most people have a tendency to think of a software product as having only one component; the software itself. But when you purchase a software product, you are buying something that has five components; The software itself, the documentation that accompanies the software, training on the product's use, product support services and future enhancements. Even if the software itself currently meets your needs, the quality of the other four components may impact you in the future. If you develop and maintain a strong business relationship with your vendors, you can have an impact on the services that your vendors provide.

COMPONENTS OF A SOFTWARE PRODUCT



THIRD PARTY SOFTWARE...A BRIEF HISTORY

In order for you to understand the influence that users of software can have on their vendors, it is helpful to look at a history of the third party software industry.

In the mid-seventies, when I first entered the wonderful world of DP as a programmer, third party software was virtually unheard of. Most application software was developed in-house and if it did come from "outside", it was from a custom software house.

"Tools" software, such as editors, debuggers and sort utilities, were provided by the hardware vendor, or not at all.

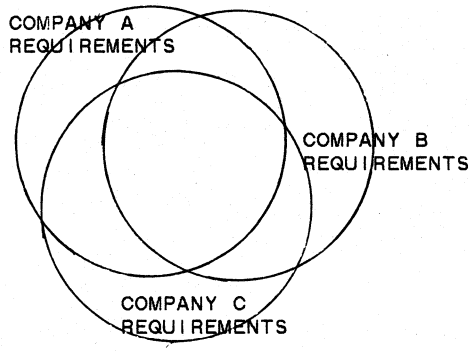
The birth of the third party software industry occurred when software developers realized that the programs that they were developing could be useful to a more than one company. If three companies each develop there own inventory control system, for example, and it costs each company $30,000 to do so, why not develop one inventory control system and share the cost? Each company would save $20,000.

As you can see, the ability to market one system to many customers brought the cost of software down, which was a tremendous benefit to companies with similar application requirements. It was also a tremendous benefit to fledgling software companies. If they developed a product for $30,000, and ten companies decided to purchase for $10,000 each, they made $70,000 on the software. Everybody won.

As is true for any new industry, it took several years to work the bugs out of the overall approach to the manufacturing of software.

First of all, vendors and purchasers alike had to learn that a software product developed for use by many companies could not address 100% of each companies requirements. The most effective approach was to develop an application designed to solve the business requirements that are common to most companies, and leave the rest to custom or in-house development. This is known as the "80 - 20 rule"; 20% of the required functionality of an application software product is used 80% of the time, by most of its users.

PURCHASED SOFTWARE BUILT FOR THE COMMON NEEDS



I had the opportunity to learn this lesson first hand. I worked for a software house in the late seventies whose philosophy was to customize the software package based on 100% of each customer's specific requirements. Sometimes these requirements were at odds with each other, and as a result, the complexity of the software ultimately increased to the point where it was un-maintainable. Moreover, supporting customers became difficult because none of our staff members could possibly become an expert on all of the features available within the software.

A better approach is to build a useful tool that answers most of the business requirements, and allows for easy access to customized routines that take care of the rest.

The development of the custom routines adds to the cost of the total system, but the savings realized on the "core" software product usually more than makes up for this.

Another aspect of the industry that software companies had to work to develop was customer support. Young software companies in the mid to late seventies often provided support on a catch-as-catch-can basis. Support costs were low or non-existent, as was the level of service. This was not the fault of the software industry alone. Often times, naive purchasers of software were reluctant to pay what it costs to finance a good support service. People who made the decisions about software purchases were, in many cases, Vice Presidents of Finance who knew little or nothing about the nature of computers and what it takes to keep the software and hardware running.

Companies that developed "Tools" software had a similar learning curve to get through, but to a lesser degree. This is largely due to the fact that a software tool, such as a sort utility or an on-line editor, often is limited to supporting a smaller number of features that perform discrete tasks. The users of these tools are usually data processing professionals, and as a result are easier to support. They can understand the inner-workings of a software product and as a result, communications between the software developer and user are less time consuming.

Software tools that are on the market today often grew out of a requirement of one particular company. For example, report writers and fourth generation languages were often developed by custom software houses for their internal use. The initial motivation was to cut costs in their own software development cycle. When customers were made aware of these methodologies, they wanted to use them to cut their own development costs.

In other cases, companies that are in business today were founded after a programmer discovered that the software that he or she developed for internal use was marketable to other companies.

Robelle Consulting Ltd., for example, was founded by Bob Green after he had developed a transaction logging facility for his employer. (This was back in the days before a transaction logging facility was available with IMAGE.) His employer proceeded to sell the software to other HP 3000 users, leading Bob to realize that there was a viable market for software utilities. His first product, QEDIT, (an on-line editor for programmers) was introduced in 1977, and has

become one of the most popular editors on the HP market today.

Dave Dummer, of IMACS System Corporation, developed a data dictionary product for use by a particular company and quickly realized that there was a need for such a product in the HP 3000 market at large. He set about to develop Dictionary/3000, which was the foundation for the Rapid family of products. After Hewlett Packard bought the rights to market the Rapid products from Dave, he set about to create a new product that would ultimately facilitate end-user reporting, HP 3000 to PC integration and end-user graphics. This product, called DataExpress, is now used in many HP 3000 shops throughout the world.

Fledgling companies marketing applications and "tools" software began to spring up in the late seventies and early eighties. DP shops soon realized that they could save time and money by investing in these products, but there was some risk involved. These new software vendors were just learning how to market and support their products, and mistakes were made along the way.

By the mid-eighties, most vendors had learned that satisfied customers were essential to their own success. They began to make sure that a prospective customer truly had a need for their product and that once he or she purchased it, they were well supported. They learned the benefits of educating their people about customer requirements. (Customers who find that purchased software meets or exceeds their expectations make good references.)

How did vendors learn about your requirements and how to communicate with you? By paying attention to what you had to say. You are as much of a vital part of their organization as they are of yours. Without satisfied customers, no software company can stay in business for very long. And they know it.

You may be thinking "they've learned, huh? I could tell her a story about something that happened to me last week...", and yes, I know. Things are not perfect. But you can't necessarily place all of the responsibility on your vendor's doorstep. There may be some things that you could be doing that would improve the situation measurably.

One of the ways that customers have come to communicate and work with their software vendors is through user groups. The concerns of a large body of customers can usually bring some influence to bear on a software companies direction.

But your relationship with your software vendors on an individual basis can be just as important, if not more so.

I've developed some basic rules for dealing with software vendors that can help make the relationship more productive. These rules are not only based on my own experiences as a user of software and as a vendor, but also on the experiences of other users who have had success in the use and support of their software products.

PURCHASING SOFTWARE...HOW TO BEGIN THE RELATIONSHIP

The relationship with the software vendor really begins during the evaluation process. This is the time when you first meet (usually via telephone) the sales person. If you have a demonstration version of the software, you also get some exposure to the support staff.

It may surprise you to know that as you are sizing up their company and software, they are sizing you up as well. Not only in terms of whether or not the software suits your needs and whether or not you can or will approve the purchase, but also in terms of what kind of customer you are going to be. Will you be demanding? (Its alright if you are, but if you are, you'd better be fair as well.) Will you be forthright or secretive? Will you trust the vendor, at least until its proven that you shouldn't?

This is your opportunity to lay the groundwork that can effect your success in using a product for years to come.

Rule #1 - Give the vendor as much information as you can

For some mysterious reason, DP shops sometimes think it is a wise practice to keep their potential vendors in the dark. They are reluctant to share information about who can approve the purchase, what competing products are being evaluated, and who is involved in the decision process. This philosophy is self-defeating. Let me explain.

The more information that you share with your sales person up front, the better equipped that sales person is to meet your needs. This is true not only in terms of product features, but also in terms of helping you with the whole buying process. When was the last time that you wanted to purchase a product, but your inability to convince your manager that it had value blocked your purchase? Helping you to handle those issues is a professional sales person's job. Chances are that they are in sales because they happen

to be good at relating the benefits of software to a company's return on investment. They deal with MIS directors and other decision makers every day. Make them work for you. If you need a cost justification, have them help you with it. If you're not sure how to present the product to your boss, ask them for help. This can be part of the service that they provide, if you'll let them.

DURING THE PURCHASING PROCESS

1. ***GIVE AS MUCH INFORMATION AS YOU CAN***
2. ***TELL THEM WHEN THEY DO SOMETHING WRONG***
3. ***TELL THEM WHEN THEY DO SOMETHING RIGHT***
4. ***LAY THE GROUNDWORK FO A LONG TERM RELATIONSHIP***

Don't be shy about telling the sales person about the other products that you're evaluating. It is in your own best interests to tell them who they are competing with. Sales people will work hard to beat the competition and you wind up being the winner; through better service, more information about how the products compare and, in some cases, more aggressive financing options.

Rule #2 - Tell them when they do something wrong

In all business relationships there are miscommunications and misunderstandings. If a software sales person steps out of line, tell them about it. I've heard of companies deciding not to buy a product because of the sales person's behavior. This is a bad basis upon which to make a buying decision. First of all, the sales person may not be representative of the company as a whole. If the software product is the best solution for you, you are biting off your nose to spite your face, as the old saying goes.

There are situations where this is warranted, but only after you have invested a little bit of time to determine:

- was this an honest mistake
- is the sales manager aware of the situation
- is this standard operating procedure for the company, or is this a "bad apple"

In most cases if you discuss the situation openly and calmly, the sales person will work hard to make up for his or her transgression. People can't fix things if they don't know that their broken.

Rule #3 - Tell them when they do something right

One Data Processing manager that I know wrote a letter to a software company's sales manager upon the purchase of their product:

"...and although our primary concern in purchasing [the product] was its functionality and ability to solve our problem, the professionalism and concern for our requirements that was shown by [sales person] definitely tipped the scales in your favor. Its a pleasure to encounter a person who really has the customer's best interests at heart..."

Taking the time to say "job well done" does several things for you. First of all, since not many people do take the time to say "thanks" to a sales person, it is likely that in the future, that sales person will go out of their way to ensure that you are satisfied.

In addition, you will not only be remembered by the sales person, but it is likely that your name and your company's name will become known throughout the company. Sales people aren't shy about sharing their successes.

Rule #4 - Lay the ground work for a long term relationship

Get to know a cross section of people that you will be dealing with after the purchase, as well as your sales person. If possible, take the opportunity to talk to more than one technical support person. Its also a goods idea to talk to the support manager. Again, the better known you are within the company (in a positive light, that is), the better off you will be in the future.

The level of support that you get during the purchasing process should be a significant factor in your decision to buy a product. Even if learning to use the product seems to be an intuitive process, make an effort to pose some questions to the support staff.

Talk to the customer support people, as well. By "customer support" I mean personnel in billing, shipping, documentation and training. You can get a pretty good idea of a company's "culture" if you talk to a cross section of people. And if you're friendly, courteous and genuinely interested in what they do, they'll be glad to hear from you when you need to get a problem resolved.

AFTER THE PURCHASE...THE ONGOING RELATIONSHIP

Many of the rules for successfully dealing with your software vendors are the same after the purchase as they are before. It is in your best interests to tell the staff when they are doing a good job and when they aren't. It is important for you to establish and maintain a good rapport with the support staff. If you communicate with them only when there are problems, they may come to dread your calls.

This is not to say that you should clog their phone lines with idle conversation. But when their product accomplishes something for you that make your life, or a user's life easier, let them know. You'll be a hero in their eyes, and they'll look forward to hearing from you and working with you.

AFTER THE PURCHASE

5. ***LET THE SUPPORT STAFF KNOW WHEN A PROBLEM IS CRITICAL***
6. ***DON'T CRY WOLF***
7. ***BE A REFERENCE ACCOUNT***
8. ***SUBMIT SUGGESTIONS IN WRITING***
9. ***KEEP EACH OTHER INFORMED ABOUT CHANGES IN PERSONNEL***

The nature of the relationship does tend to change after the purchase has occurred, however. Some additional rules that you need to keep in mind are:

Rule #5 - Let the support staff know when a problem is critical

Whether its a product problem or an accounts payable problem, if its costing you money, productivity and/or time, let them know this. Be specific! Software companies sometimes receive hundreds of calls a day from their customers. If you don't make it clear to them that a problem is critical, they might disappoint you by not getting back to you as soon as you'd like. You, then, are likely to become angry and frustrated, often resulting in hard feelings and conflict with your vendor. If you communicate the import of a situation up front, stating clearly how the problem is effecting your company, you can often avoid these confrontations. This doesn't mean that you need to be pushy or hysterical. It simply means that you need to explain the effect that this problem is having clearly and concisely.

This may seem like an obvious approach, but I can assure you that many times the critical nature of a problem is not adequately communicated. I suspect the reason is that when

software users are having a serious problem, they are so painfully aware of the implications themselves, that they believe that the consequences are intuitively obvious. They are not!

Rule #6 - Don't Cry Wolf

A customer whose problems are always critical gets a label, and it usually isn't a pretty one. If all of your problems are "urgent", the result will be that none of them are.

If you've ever had someone work for you who was a constant complainer, you can understand this concept. They complain about the lighting fixtures, the water cooler, the parking facilities and the lunch room. Pretty soon, you tune them out. This is a natural human reaction to a chronic complainer, and whether you're a customer or not, it will happen to you if you don't make distinctions between "annoyances" and "critical problems".

So when you call with a problem, set their expectations for when you need an answer. Get agreement about who is responsible for the next phone call. And if they ask you to do some additional investigation, don't hesitate to ask for the reasons. If you have a clear idea of why you're doing something, you'll feel better about doing it, and do a better job of it.

Rule #7 - Be a reference account

One of the best ways to ensure a high level of support from your vendors is to give references (and even demonstrations) to their prospective customers. You should do this, of course, only if you truly feel that the vendor's product is a good one.

Any sales rep worth his or her salt will fight hard to get good support for any company that has helped to sell software. (If your sales reps don't do this, you should feel free to remind them that they should.)

Although this may seem to be above and beyond what you should have to do, and it may take some time away from your responsibilities, the payoffs are usually worth it. "Word of mouth" is the name of the game in the software business, particularly in the HP user community. Its the age-old concept of one hand washing the other.

Anything you can do to be a "star" customer in your software vendors company will pay off ten-fold. Being a good

reference account is probably the shortest path to achieving this goal.

Rule #8 - Submit your suggestions in written form

If you have suggestions for enhancements to the product, the documentation, the training or anything else, don't just call and express them. Send a letter. Begin the letter by telling them what they're doing right. Then discuss your suggestions. Also, discuss why you think your suggestions are good ideas. Again, what is intuitively obvious to you may not be so to others.

If you take the time to write a letter, your ideas will be taken more seriously. You've put thought into them, they are serious to you, and this is evidenced by your written communication.

Moreover, when you simply call in your suggestions, if its a particularly intense day for the person at the other end of the phone, its possible that the conversation will be forgotten. Its much harder to "forget away" a piece of paper. And if your sales or support person needs to communicate the suggestion to their management, it holds a lot more weight if its communicated on your letter-head.

Rule #9 - Keep each other informed about personnel changes

I am sure that it is disconcerting to you when you phone your software vendor for support and the voice on the other end of the phone is a new one. Perhaps you've been using the product for five years, and only call support under the most unusual circumstances. In the mean time, this brand new support person has been exposed to the product for a grand total of one week.

When this happens to you, you should feel free to call the support manager and let him or her know that while you are willing to work with this new person, you would appreciate knowing that they will be backed up by senior personnel.

Let's face it. Software vendors face personnel turnover just like you do. And they don't like it any better than you do. While it will cost you some time, the best approach is to welcome the new person, give them a background of your experience with the product, and let them know that you have already tried steps one through fifteen in the problem resolution instructions. In most cases they will appreciate the information and do their best to do a good job for you.

Similarly, when you have personnel changes, it is a good idea to let your software vendor's support staff know, in writing. This will save your new employee embarrassment when they phone in as a novice, and it will allow the vendor to update their records. You should also discuss product training for your new staff member. Just because it wasn't available when you purchased the product doesn't mean that it hasn't become available since then.

YOUR SOFTWARE VENDORS ARE AN EXTENSION OF YOUR COMPANY

Its interesting that when you talk to two different people about the same software company, you can get two entirely different reads on how well that software company supports their customers. A large part of the reason for this phenomenon is the personal relationship that has been established between the individual customer and their vendor.

Ron Milam of Monarch Paper Company in Houston Texas understands that his vendors are important to his own success. "I let vendors know where I stand, right from the beginning. I am open with them about my requirements, competitive products that I'm looking at and what I like and dislike about their products and services. I'll take the time to give them a good reference when they have a good product. I treat vendors as I expect to be treated and I usually get good results." He's a favorite with his vendors and the service he gets reflects this.

THE RELATIONSHIP GOES BOTH WAYS

Just like any other person or department in your company, you and your software vendor are interdependent. You share in the responsibility for making the relationship work, and if you do a good job of it, both you and your company will benefit.

Disaster Planning and Recovery
"A True Story"

By
Linda Michel, Information Systems Manager
San Diego Blood Bank
440 Upas Street
San Diego, California 92103
(619) 296-6393 extension 79

Data center disasters and disaster planning have been the topic of much discussion in the past several years. It is difficult to pick up a technical or business publication without reading about computer disasters and their causes. Most companies know that they should have a formal disaster plan in place but, because of the many difficulties involved in trying to establish and test a disaster plan, many do not. It is very hard to plan for something that in, all likelihood, will not happen. Unless you have suffered a loss of data or an unplanned extended period of down time, you find yourself creating a plan that cannot even be thoroughly tested for validity. Only D.P. auditors and pessimists would relish such a task. There are vendors who will provide disaster planning and recovery services, but their services are very expensive and difficult to cost justify. This paper explores the events that occurred when a major disaster struck at The San Diego Blood Bank, without a formal plan, how they survived it, and the steps they took to create a plan and insure that this would not happen again.

The environment for disaster. On Sunday July 12, 1981 between the hours of five and five-thirty a.m. an arson fire was set and gutted the computer room at the San Diego Blood Bank. The San Diego Blood Bank is a non profit community organization and was San Diego's only source of blood supply. The fire was set by an individual with a grievance against the Blood Bank. The arsonist started the fire in the computer room which was located on the third floor of the building. The computer room was destroyed and severe damage was done to the remainder of the third floor. The hardware was scorched beyond repair. The disk packs, containing the blood banks donor history and financial data, were presumed to be a 100 percent loss. The blood bank had been backing up their data on a daily basis and the backup packs were to be removed from the computer room each day. By some stroke of bad luck, the backup packs were not removed that day and both the master and the backup packs were found in the rubble melted down. It is believed that the fire and the breakdown in procedures were unrelated but the two events together made the situation much worse than if one had occurred without the other. The positive side to this situation was that 2,000 units of

blood in inventory was stored on another floor and the fire was contained on the third floor.

In those days there was no data processing department. The computer occupied a room off of the business office. There was no Halon system, no locks to prevent access and no security guards at the facility. The removal of the backup packs was an internal procedure that was assigned to an employee to do. The blood bank was a well respected community service organization. Nobody ever dreamed that this could happen.

The Community of San Diego rallied to aide of the Blood Bank. By 3:00 p.m. that same day the entire operation had been moved to a donated location and was up and running. This was possible because, at that time, testing of blood, making of blood products and the shipping and receiving functions were not automated. The automated financial system and donor history were a another story. The employees pitched in and spent days sifting through smokey charred documents trying to salvage as much as possible in order to continue business. The Comptroller, (currently Chief Financial Officer/ Associate Administrator and my boss) took charge of the data recovery efforts. She worked with the computer vendor and a consultant named David Brown. He specializes in data recovery techniques. He is the same person who recovered the data presumed lost in the MGM Grand Hotel fire in Los Vegas in 1980. His proprietary process of data recovery involves dislodging contaminates lodged between computer heads and disks. He states that the equipment he uses is "not designed for anything which remotely resembles computers". Mr. Brown worked for two days processing the disk surfaces. The packs were so misshapen that they would not fit on the disk drives. He had to modify the disk drives in order to mount the melted disk packs. Unbelievably, they recovered about 75 percent of the financial data and approximately 65 percent of the donor history files.

Operations resumed at the Blood Bank, with the exception of the third floor, in ten days. All departments located on the third floor continued operation at a donated site while the old location was being rebuilt. It was six months before the third floor was ready for occupation. The computer had to be replaced and the computer room rebuild. The fire, smoke and water damage completely destroyed it. To protect their new computer system, a Halon system was installed in the new computer room. Security was tightened and the backup procedures were enhanced to ensure that the disk packs would always be taken off site daily. The Comptroller made a statement at that time and I think it sums up the feeling of loss that occurs at such a time. "I can't tell you what we

felt when we found the computer room and backup packs destroyed. When you think you have everything, that's when disaster hits and it's disheartening. We were lucky to get out of this situation with our data usable. Another company might not be so lucky".

Environment for disaster planning. The Blood Bank continued to grow and it became evident that they would have to automate blood testing and other functions vital to blood bank operations. The industry was changing very rapidly with the introduction of new tests and requirements for Aids screening. It became imperative to have more control and better traceability of blood test results. Many of these functions come under FDA regulations and inspections. The current methods of testing and processing blood had been outgrown.

This is when I came into the picture. I was hired by the San Diego Blood Bank to install a fully automated blood banking system and Hewlett Packard's Financial Application package along with a Hewlett Packard 3000 Series 48. I'm not a D.P. auditor so I must be a pessimist because disaster planning was high on my priority list. Of course, it was high on my bosses priority list too. That helped to encourage me just a little. Having been through the previous disaster, she knew that with all functions being automated, it would be impossible to recover as easily from a disaster of that magnitude.

The first year and a half I spent at the Blood Bank were spent heavily involved with hardware and software implementation. However, I kept the goal of establishing a formal disaster recovery plan in my mind at all times. I was able to do three things immediately. First, I had keyed access locks installed on the computer room doors and limited access to the computer room. Only personnel who absolutely needed access were given the combination. This turned out to be a rather touchy situation and required a lot P.R. work to reassure the excluded people that they were certainly trustworthy but that this was just good procedures. Secondly, I noticed that Blood Bank personnel were becoming too dependant on the computer. Even those who had vehemently resisted the computer could no longer do their work without it. I was fanatical about manual operating procedures and documentation. I told then horror stories about computers that were down for days at a time. I nagged them continually to keep detailed, up to date, manual operating procedures that could be used in the event of computer down time.
To my surprise our new reliable Hewlett Packard 3000 helped me prove my point. It had continual system failures, caused by a faulty GIG cable, which was very difficult to isolate.

One of our high tech 7933xp disk drives, complete with disk caching, intermittently refused to respond to I/O. Finally, a faulty spindle motor caused a complete reload. I didn't plan any of this but it certainly helped to illustrate my point to users. They had to be able to continue work without the computer. As it turned out the manual operating procedures, put together by each department, became an integral part of the formal disaster plan. Thirdly, I implemented a backup routine on the Hewlett Packard 3000 that included a full backup each week and a partial going back to the last full on a daily basis. I continued the daily off site storage strategy and instituted a rigid standard of tape testing and cleaning. I felt a measure of safety knowing that my data was always in two places and that my backup media was in good shape.

Prepare a preliminary plan. In order to begin formalizing our disaster plan, I read as many articles on disaster planning as I could get my hands on. I attended seminars on disaster planning and consulted with my Hewlett Packard S.E. After must evaluation, I decided that it would be desirable to continue operating automated in the event of a disaster. Next I evaluated the blood bank's manual procedures. They served as an my outline of everything that must be accomplished in order to do business. Then I calculated the amount of disk space I would need to operate successfully at a remote site. The Blood Banking system took up most of two 7933 disk drives and the Financial Application package took up just over one 7933 disk drive. With two large systems, it seemed that the easiest disaster planning method would be to hire a professional vendor to assist. I contacted Up Time, Weyerhauser and several other disaster planning vendors. They all had good disaster planning services and disaster recovery methods. The average cost was about $1500.00 a month. I presented the idea to my boss thinking that I was off the hook. Needless to say, given the high cost, I was sent back to the drawing board. Being a small non profit organization, it could not be cost justified. She pointed out that all costs to recover from a disaster were covered by insurance but all costs expended in a disaster planning effort would be out of pocket. It was clear to me that I would have to formulate a plan in house. After I re-evaluated our manual procedures and the disaster planning information I had accumulated, I drafted a preliminary plan which called for a reciprocal agreement with another Hewlett Packard facility. With the size of our applications, it was likely that no other company would be able to accommodate us with disk space. My goal was to impact another facility as little as possible. I looked for a way to run only those functions absolutely necessary during a disaster and perform the other functions in a manual mode.

Identify critical functions. Determining which tasks should be automated during a disaster is a very difficult job. I spent time interviewing our department heads trying to select the tasks that would most need to be performed on the computer in the event of a disaster. Every department felt that it would be impossible for them to operate without the computer and should be one of the choices for automation. It became obvious that there needed to be an individual or group of individuals to decide, with the overall knowledge of the organization in mind, which departmental functions would most need automating during a disaster. In my case, my boss was able to fill that function. I presented my preliminary plan, to create a reciprocal agreement with another company and operate only a portion of our system on their computer, to her. She not only had the authority to make these kinds of decisions, but having been through the previous disaster at the blood bank, she knew exactly what had occurred and what needed to be done at such a time.

The biggest problem for our operation during the previous fire had been in the area of customer billing, accounts receivables, and accounts payable (luckily our payroll is not done in house). It seems that the friendly insurance agent, complete with hard hat, did not appear on the scene immediately with a large insurance check to cover expenses. She recommended that we split the financial system from the blood banking system and automate it during a disaster as our critical resource. This would insure that we could bill our customers, receive revenue, cover payroll and pay our bills. With this decision finally made, I was ready to proceed with the formal plan.

Establish a method. Finding another user that was interested in entering into a reciprocal agreement was not as easy as I thought it would be. Most people I contacted had not addressed this issue and didn't know what they would do in the event of disaster. Several had legal and security concerns. They were all hesitant. I guess it was because disaster planning is an overwhelming task. It occurred to me that the best thing to do would be to finish my plan as if I had a reciprocal site. I felt that if I could present a plan to another site, with a feasible method of operating during a disaster, they might be more interested in entering into an agreement with us. So I proceeded as if I had another company already lined up. I looked at our own operation and thought about how we could accommodate a another site in the event of a disaster. If the Blood bank could act as a reciprocal site any company could. We were short on disk space and short on processing time.

The Blood Bank is a twenty-four hour a day, three hundred and sixty-five day a year operation. Any reciprocal agreement that we entered in could not interrupt our operations too significantly. I needed to find a method of operating on a backup computer that would not interrupt day to day operations and one that would not cost much money up front. I needed a large amount of disk space and I could not offer another company much disk space. Private volumes seemed like a possible solution, but that would limit the companies that could serve as a backup site to only those who had private volumes. At that time I knew very little about private volumes and I had my doubts if that Blood Bank would want to purchase several disk drives, or replace our current drives with removable disk packs, to be available in case of another disaster.

About that time I began to get discouraged. I knew what I wanted to do but still had no idea of how to accomplish it. I turned to Hewlett Packard for some guidance and was able to convince them that they should work with me under the guidelines of our Account Management Support Agreement. After a lot of going back and forth about what services were covered under Account Management Support and whether I should continue with that level of support, Hewlett Packard realized that they could indeed give me some assistance without any additional charges. Hewlett Packard even agreed to make their computer available as a backup computer until a company who was interested in entering into a reciprocal agreement could be found. Amazing how that can happen! Working together, we came up with the idea of renting disk drives when a disaster occurs and configure them into a system as private volumes. This would allow us to have plenty of disk space without impacting a backup computer and would not cost anything up front. It turns out that you can add private volumes on to a system with a cool start. It would be a relatively easy process. Finally I was ready to put the logistics of my plan together. I had all the essential ingredients. 1. Identification of the critical resources to be automated during a disaster. 2. Manual procedures for all non automated tasks. 3. A method for loading a large amount of data on a remote computer without impacting the operation significantly.

Resolve technical issues. With a preliminary plan in hand and a method in mind, I addressed the steps that were needed to configure in private volumes. As part of Hewlett Packard's partnership planning approach to account management support, I worked with our S.E. to set up a procedure to add private volumes to a system, create the accounting structure and load our files. I wanted to have a step by step procedure in place so that at the time of a disaster, I could

follow a predetermined set of instructions. The process is tricky but once you understand how private volumes work, it goes quite smoothly. One thing I learned was that even though our accounts would reside on a private volume, the account names are kept in the system directory and are spanned to the directory on the private volume. To prevent any possibility that a reciprocal site would have an identical account name, we renamed our tools account to SDTOOLS (HP32265). I have listed below the steps necessary to add private volumes to a system, create the accounting structure and load files.

1. Add Private Volume to SYSTEM {fig 1}

{fig 1} SYSDUMP DIALOG REQUIRED TO ADD PRIVATE VOLUME DISC's

<< Must add all 7937 disc drives in this way >>

```
:FILE L;DEV=LP
:FILE T;DEV=TAPE

:SYSDUMP *T,*L
 ANY CHANGES? Y

 SYSTEM ID = HP32033G.03.01.?
 MEMORY SIZE = 3072 (MIN=256, MAX=8192)?
 I/O CONFIGURATION CHANGES? Y
 LIST I/O DEVICES?
 LIST CS DEVICES?
 LIST DEVICE DEFAULTS?
 HIGHEST DRT = 103 (MIN=8, MAX=511)?
 LOGICAL DEVICE #? XX
 DEVICE NAME? HP7937
 DRT #? XX
 UNIT #? 3
 SOFTWARE CHANNEL # = 0?
 TYPE = 3?
 SUB TYPE = 10?
 RECORD WIDTH = 128?
 ACCEPT JOBS/SESSIONS =  N ?
 ACCEPT DATA =  N ?
 INTERACTIVE =  N ?
 DUPLICATIVE =  N ?
 INITIALLY SPOOLED =  N ?
 AUTO REPLY =  N ?
 DRIVER NAME = HIOMDSC2?
 DEVICE CLASSES = DISC?PVDISC
```

<< Cold Load the system from this tape after the physical disc's have been added to the system. Insure there is sufficient system directory space on the system to be used.>

2. Define the Volume Set to the SYSTEM {fig 2}

{fig 2} Commands to define the Volume Set to the SYSTEM

```
:HELLO MANAGER.SYS,PUB

:NEWVSET SDBLOOD;MEMBERS=SDBLOOD:HP7937,SLAVE:HP7937

    Commands necessary to Initialize the 7937 PV's

 << Down the devices, "XX" represents LDEV # in all cases >>
:DOWN XX
:DOWN XX

<< This command will show the status of the private volumes.
   New disc's would be FOREIGN and unallocated in this listing >>

 :DSTAT

 :VINIT
 >FORMAT XX  << Reply with a "Y" when asked if you really want to
               format the drive.  Respond "NO" when asked if there
               known bad tracks.  >>

 >FORMAT XX

 >INIT SDBLOOD,XX,SDBLOOD.PUB.SYS
  10000  << This is in response to the
            directory size question >>
 >INIT SLAVE1,XX,SDBLOOD.PUB.SYS

 >EXIT
```

3. Create Accounting Structure on Private Volumes {fig 3}

{fig 3} Create Accounting Structure on Private Volumes

```
!JOB CREATE,MANAGER.SYS;OUTCLASS=LP,1
!CONTINUE
!TELLOP;  Now creating the Accounting Structure on the SDBLOOD
!TELLOP;  PV's
!TELLOP;  Creating GROUP SDTOOLS within the SYS account
!TELLOP;
!ALTACCT SYS;VS=SDBLOOD.PUB.SYS:SPAN
!NEWGROUP SDTOOLS;CAP=IA,BA,PH,DS,MR,PM; &
!ACCESS=(R,L,X:ANY;W,A,S:AC)
!ALTGROUP SDTOOLS;VS=SDBLOOD.PUB.SYS:SPAN
!COMMENT
!COMMENT    ==========================================
!COMMENT     THIS JOB CREATES THE NEW PRODUCT ACCOUNT.
```

```
!COMMENT    ==========================================
!COMMENT
!CONTINUE
!TELLOP;  Creating the HPFAS Account
!TELLOP;
!NEWACCT HPFAS,MGR;ACCESS=(R,X,L:ANY;W,A:AC);&
!CAP=AM,AL,GL,ND,SF,IA,BA,PH,MR,CV,UV,CS,DS,OP,LG;MAXPRI=BS;&
!VS=SDBLOOD.PUB.SYS:SPAN
!COMMENT
!COMMENT ----------------------------------------
!COMMENT
!COMMENT    THIS JOB CREATES THE NEW ACCOUNT FOR
!COMMENT    THE APPLICATIONS CUSTOMIZER/3000
!COMMENT
!COMMENT ----------------------------------------
!COMMENT
!CONTINUE
!tellop;  Creating the SDTOOLS Account
!tellop;
!NEWACCT SDTOOLS,MGR;&
!CAP=AM,AL,GL,ND,SF,IA,BA,PH,MR,CV,UV,CS,DS,OP,LG; MAXPRI=BS
!ALTACCT SDTOOLS;ACCESS=(R,L,X:ANY;W,A:AC); &
!VS=SDBLOOD.PUB.SYS:SPAN !COMMENT
!STREAM ,>
>JOB HPFAS,MGR.HPFAS,PUB
>tellop;  Creating the Groups under the HPFAS account
>tellop;
>COMMENT
>CONTINUE
>ALTUSER MGR;MAXPRI=BS
>CONTINUE
>NEWUSER USER;CAP=AM,AL,GL,CV,UV,LG,CS,SF,IA,BA,PH,DS,MR,ND
>CONTINUE
>ALTGROUP PUB;ACCESS=(R,X,L:ANY;W,A,S:AL,GU); &
!VS=SDBLOOD.PUB.SYS:SPAN
>COMMENT
>COMMENT  INSTALLATION OF IPC FILES AND GROUPS
>COMMENT
>NEWGROUP GLWORK;ACCESS=(R,X,L:ANY;W,A,S:AC); &
>VS=SDBLOOD.PUB.SYS:SPAN
>COMMENT USE NEWGROUP COMMAND FOR ALL IPC GROUPS
>COMMENT ------------------------------
>COMMENT  BUILD NEW GROUPS FOR REPORT FACILITY:
>COMMENT  THIS JOB CREATES THE NECESSARY GROUP STRUCTURE
>COMMENT  WITHIN THE APPLICATION ACCOUNT
>COMMENT  ---------------------------------
>NEWGROUP REPGLOBS;ACCESS=(R,X,L:ANY;W,A,S:AC); &
!VS=SDBLOOD.PUB.SYS:SPAN
>CONTINUE
>COMMENT    INCLUDE NEWGROUP COMMAND FOR ALL GROUPS IN
>COMMENT       APPLICATION
```

```
>COMMENT
>STREAM ,*
*JOB  SDTOOLS,MGR.SDTOOLS,PUB
*COMMENT
*COMMENT ------------------------------------------------
*COMMENT
*COMMENT     THIS JOB CREATES THE GROUP STRUCTURE FOR
*COMMENT     THE APPLICATION CUSTIOMIZER/3000 ACCOUNT.
*COMMENT
*COMMENT ------------------------------------------------
*COMMENT
*TELLOP;  Creating the Groups under the SDTOOLS Account
*TELLOP;
*CONTINUE
*ALTUSER  MGR; MAXPRI=BS
*CONTINUE
*ALTGROUP PUB;CAP=IA,BA,PH,MR,DS; &
*ACCESS=(R,X,L:ANY;W,A,S:AL,GU);&
*VS=SDBLOOD.PUB.SYS:SPAN
*COMMENT
*COMMENT ---------------------------------------
*COMMENT
*COMMENT     THIS STEP ADDS THE GROUP SYSPGM TO
*COMMENT     THE SDTOOLS ACCOUNT STRUCTURE.
*COMMENT
*COMMENT ---------------------------------------
*COMMENT
*CONTINUE
*NEWGROUP SYSPGM; ACCESS=(R,X,L:ANY; W,A,S:AC); &
*CAP=IA,BA,PH,MR,DS;&
*VS=SDBLOOD.PUB.SYS:SPAN
*COMMENT
*COMMENT     CONTINUE WITH NEWGROUP COMMANDS FOR ALL
*COMMENT     GROUPS IN SDTOOLS ACCOUNT STRUCTURE.
*COMMENT
*tellop;
*TELLOP;  Accounting Structure now created
*TELLOP;  on the SDBLOOD private volume set.
*COMMENT
*EOJ
>EOJ
!EOJ
```

3. Load files to private volumes {fig 4}

{fig 4} JOB to restore files into the PV Accounts

```
!JOB RESTORE,MANAGER.SYS,PUB
!FILE T;DEV=TAPE
!FILE SYSLIST;DEV=LP
!TELLOP; Beginning the restore of files on
```

```
!TELLOP; the SDBLOOD PV's
!RESTORE *T;@.@.HPTOOLS,@.@.HPFAS,&
!@.HPTOOLS.SYS;SHOW;OLDDATE
!EOJ
  FILE = RESTOR.SDBB.SYS
```

4. Allow User Access to Private Volumes {fig 5}

{fig 5} Operator Commands used for Private Volume Management

```
:VMOUNT ON,AUTO  <<  Should be done once user access
                     to the Private Volume files is needed  >>
:DSTAT  <<  Used to check the status of Private Volumes  >>
```

6. Delete Accounts from Private Volumes {fig 6}

{fig 6} Purge Private Volume Accounting Structure

```
!JOB ELIMACCT,MANAGER.SYS,PUB
!PURGEACCT HPFAS;VS=SDBLOOD.PUB.SYS
!PURGEACCT HPFAS
!PURGEACCT SDTOOLS;VS=SDBLOOD.PUB.SYS
!PURGEACCT SDTOOLS
!PURGEGROUP SDTOOLS;VS=SDBLOOD.PUB.SYS
!PURGEGROUP SDTOOLS
!PURGEACCT SYS;VS=SDBLOOD.PUB.SYS
!EOJ
```

7. Eliminate Private Volumes from System {fig 7}

{fig 7} Remove Private Volumes

```
:Hello MANAGER.SYS,PUB
:PURGEVSET SDBLOOD
:FILE T;DEV=TAPE
:FILE L;DEV=LP
:SYSDUMP *T,*L << DELETE THE PV LDEV #S >>
 << COLD LOAD FROM THIS TAPE TO DELETE THE PV'S FROM
    THE SYSTEM >>
```

We create a backup of our Financial Application and Tools Account each morning to be used as our recovery data. These tapes also contain the above procedure and job streams. They are taken off site and rotated daily. At the time of a disaster I would arrange for the rental of the disk drives. The drives would be delivered to our backup location. We would arrange to have Hewlett Packard connect them to the system on a time and materials basis. The companies that I spoke to could deliver the drives within 48 hours subject

to credit approval. The cost is about 350.00 per drive a month. Most of them require a minimum rental period. Since the equipment rental companies cannot guarantee the availability of equipment when you need it, it is best to have several places in mind for renting disk drives.

The Disaster Plan. Putting the written disaster contingency plan together was time consuming, but not difficult, once I had all the components together. The finished disaster plan is a binder that contains the written document and all other information necessary to operate in a disaster. One copy of the plan is kept on site in our safe and one copy is kept off site with our backup tapes. The format of to plan and what each section is comprised of is listed below.

I. Introduction - The introduction contains the goals of the plan, the scope of the plan, and guidelines for periodic review and testing of the plan. This section is intended as a statement of the plan premise and a brief outline of how it will be maintained.

II. Emergency Personnel - This section is a Who's Who of the disaster plan. It specifies the organizational tasks to be done during a disaster and who will be responsible for each. It outlines in detail the steps to be performed by data processing in order to transfer operations to a backup sight and continue operations until normal operations can be resumed. The plan has names and phone numbers of each member. Each member of the recovery team is issued a copy of the plan so that he or she will know what to do and how to contact other members.

III. Hardware and Configurations - All computer room equipment, communications equipment, system software and environmental controls are listed in this section. It is complete with the vendor names, addresses, phone numbers and contacts. This section is intended to specify everything necessary to rebuild the data processing department and to facilitate in ordering replacement equipment and software.

IV. Scheduling - This is a statement regarding computer utilization and the necessity of setting a pre-defined set of priorities for computer work during a disaster. It stipulates that only critical tasks are performed on the backup computer. There will be a shortage of computer time available at the backup site as we committed ourselves to work only during a reciprocal sites off hours. All work must be planned ahead of time to insure that time spent at the reciprocal site

is used wisely. Jobs and tasks to be executed on the computer during a disaster must be reviewed and approved by the Information Systems Manager.

V. Off Site Support - All companies and contacts external to the Blood Bank, necessary to implement the disaster plan, are documented in this section. The name, address, phone number and contact of our data storage company is listed along with a description of our backup strategy. Also the names, addresses, phone numbers of contacts at Hewlett Packard, vendors to be contacted about disk drive rental and our reciprocal site. A copy of the reciprocal agreement or letter is kept here. In addition, it states clearly what steps are to be taken and when to get each person involved. It lists the steps necessary to put the plan in motion and the data processing logistics needed for operation at the backup location.

Vi. Manual Operation Procedures - All the users manual operating procedures are kept with the plan. In the event of a disaster that destroyed the user areas as well as the computer facility, they would be distributed to the appropriate personnel.

VII. Appendices - This section contains a current Sysinfo, copies of all maintenance contracts for both hardware and software, a device list with serial numbers and the location of all peripheral devices, building disaster information and any other information that I thought would be of help. If the entire computer facility were destroyed I might not have access to anything that is kept at the facility. This section is a collection of all information necessary to allow operating at the backup site and working towards the timely replacement of equipment.

Validity testing. The disaster recovery strategy and backup media must be tested on a regular basis. A mock disaster should be staged yearly. The drives should be rented and the procedures for installing and operating on them tested. By testing the plan out in a controlled environment you can become familiar with the steps to be performed. In the event of a real disaster your procedures will be familiar to you. It is also the only way to be sure that the configuration of both your system and your reciprocal site's system are in sync. We tested this plan thoroughly at our local Hewlett Packard office. We went through each step from hooking up extra disk drives to running actual production on our loaded financial application.

Plan maintenance. Nothing is set in concrete and things change very quickly. I never realized how often people change their telephone number and how often we acquire new equipment or software. The fact is that the plan is never finished. The documentation must be reviewed frequently. At first we were changing the plan any time something changed but we just couldn't keep up with it. Now we review the plan quarterly and update it at that time, unless there is a major change. It is also necessary to keep in contact with your reciprocal site to make sure that all their information is still the same. An upgrade or the addition of a device at either site could nullify your agreement.

The disaster recovery experience at the San Diego Blood Bank can serve as a learning tool for all. It forced the Blood Bank to create a disaster recovery plan and I hope it will encourage others to consider disaster planning. As the title of this paper suggests, disaster planning should be addressed before disaster recovery becomes necessary. If there had been a disaster plan at the time of the San Diego Blood Bank fire, no loss of data would had occurred and the recovery process would have been easier. I am sure that there are many approaches to disaster planning, what I have done is just one method. It may not be right for your shop but the key issue is to benefit from our experience. Think about what you would do if one day you went to work and found your data center demolished. It might never happen but I don't recommend that you gamble with your companies resources. Data center disasters can and do happen. They say lightning never strikes twice in the same place, but just in case it does, the San Diego Blood Bank will be ready.

AI–The Three Toed Sloth

robert a. karlin
karlins' korner
7628 van noord ave.
n. hollywood ca.
91605

The three toed sloth, the ai, is a large slow beast that lives in the South American jungles, where he hangs upside down from the trees, feeding on fruits and vegetables. Though slow and normally docile, the ai is a powerful animal, and can be dangerous when aroused.

Artificial Intelligence, that is AI, is also a large slow beast, living far from the ken of normal programmers. Close observation of AI by programmers more used to the mundane could easily lead them to believe that the AI has been programmed upside down, or at least by creatures that live in trees, feeding on fruits and vegetables. And AI, though slow and normally docile, can indeed be very powerful, and also very dangerous if misused.

INTRODUCTION

Since the beginning of the computer age, man has wondered at the idea of a thinking machine. From the Dybbuk of eastern European myth to the clockwork figures of seventeenth and eighteenth century fiction, thinking machines were usually envisioned as human in shape. In 1923, Karel Capek coined the word Robot in his play, *R.U.R.*, and since then, the word has been synonymous with mechanical intelligence. It wasn't until the creation of Eniac and Univac, the first commercial computing machines, in the mid 1950s that the actual mechanism by which intelligence could be imparted to nonliving structures took shape. Since that time, the search for artificial intelligence, has been ceaseless.

In general, AI has not touched the business market place. AI is still too new a field of study to have produced much fruit. However, this is changing. This paper is a look at AI from the businessman's point of view, examining what has gone before, and what is still to come, and how this will affect the business data processing department of the near future.

INTELLIGENCE

Before we enter into a discussion of what constitutes artificial intelligence, we should first see if we can define what it is that we mean by intelligence.

Webster's defines intelligence as 'the ability to learn or understand from experience; the ability to acquire and retain knowledge; the ability to respond quickly and successfully to a new situation'. What part of this definition is applicable to intelligence of the artificial kind?

By far the most striking difference between computer programs and human beings is the ability of humans to alter their behavior pattern based on experience. But it is possible to create programs that learn as well. The simplest example of a program that learns is the game program ANIMAL, a computerized version of twenty questions. To initiate the game, the program says 'THINK OF AN ANIMAL'. After the player responds with a carraige return, the program inquires 'DOES THIS ANIMAL HAVE FOUR FEET?'. If you respond in the affirmative, the program will say 'ARE YOU THINKING OF A CAT?'. If you respond that you are not thinking of a cat, the program will ask you what animal you are thinking of, and when you respond 'elephant', the program will ask for an indicative feature of an elephant. The next time the game is played, if the answer to the first question is affirmative, the program will ask the question saved from the first game to differentiate a cat from an elephant. If the program again guesses wrong, it will store the information acquired this play. In a surprisingly short time, the program can accurately guess thousands of separate animals, usually with less than ten questions. This form of trial by error can be very effective for small problems, but almost worthless for anything larger. To illustrate, imagine a program that plays chess. After each loss, the software stores the last move prior to the mate, and eliminates it from its possible moves. To develop any coherent play in this manner would take years, even with our fastest machines, and the lookup time during play would be prohibitive. Current research in the area of software learning is concentrating on methods for deriving underlying rules of thumb, as opposed to specific courses of action.

It may seem that acquisition and retention of knowledge is the easiest segment of intelligence to emulate in machinery. After all, this is what we believe computers do best, storing data. And yet, just storing data does not really fit what Webster was driving at. Data must be stored in some usable form, indexed appropriately for later

retrieval, and summarized into coherent structures. And this is where we run into trouble. We can store data much more easily than we can produce general purpose rules to identify and classify that data. We have trouble developing rules to allow a program the ability to distinguish between a photo of a beach ball and a photo of the sun. We also have trouble eliminating "noise" data, data that does not belong, from data that just does not fit. A human can easily look into a basket of objects and retrieve a particular size and color block, yet there is no program yet that can perform that task with one hundred percent accuracy.

The ability to respond to new situations based on prior experience may seem to be the most difficult section of our definition to emulate, yet is actually one of the success stories of current AI research. Rule-based 'expert systems' have been developed that can generalize from insufficient data and, responding to new situations with its area of expertise, produce fairly reliable results, but the key to a successful expert system seems to be as much in the artistry of the design analyst as in the developmental technique. We have yet to produce an expert system able to design other expert systems, though this project is certainly being pursued.

EXPERT SYSTEMS

The biggest success story of artificial intelligence research is knowledge-based or 'expert' systems. The first program that could be called knowledge-based was called DENDRAL, and was developed at Stanford University in the mid 1960s in order to help chemists identify compounds by spectrometric analysis. Since that time, expert systems have been designed for applications as diverse as oil prospecting and medicine. In addition to complete applications, expert system 'shells' are now available, allowing customers to tailor the system to their own needs.

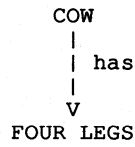
All expert systems contain at least two basic parts. First, obviously is the 'knowledge base' itself and second, some form of knowledge interpreter to input and retrieve data from the knowledge base.

In order to represent a field of knowledge, it must be codified in a way that will make it both accessible and understandable. Some of the different ways of coding are logical coding, procedural representation, semantic nets, production systems, and frames.

Logical coding consists of formal logical statements. For example, if we take the statement 'all cows have four legs', then we could express this in formal logical expression as 'For any object x, if x is a cow, then x has four legs'. The advantage of logical coding is that the rules by which expressions are evaluated is based on centuries of philosophical research, and is known and well understood.

Procedural representation consists of small well defined procedures that process individual portions of the problem. For example, if we were building a natural language parsing program, nouns, verbs, adverbs, etc. each would have their own small procedure that determines what actions should be taken. The major disadvantage to procedural based systems is their complexity, creating problems in understanding the entire interaction of the base procedures and making debugging difficult.

Semantic nets most resemble the database of the business community. Objects, concepts and events are stored as 'nodes', and the interrelation of these nodes are stored as 'links'. A simple net might be:



The major problem with nets is that they are not innately valid, that is, the interpretation of what a link means is entirely in the hands of the software, and, unlike the logical representation, the data itself is no guarantee of its meaning.

Productions systems, also known as rule-based systems, store information as a set of rules, called productions, usually in the form of 'if condition, then action'. An example may be, 'If it starts raining, and you are outside, then open umbrella'. Because of their innate understandability, production systems have been useful for large application systems. DENDRAL, mentioned above, and PROSPECTOR, a geological program, are among the better known of these.

The last representational scheme we will discuss is the 'frame', or object oriented representation. Unlike rule based systems, where each 'unit' is procedural, and nets in which each unit is declarative, each frame includes both a declarative and procedural portion. This allows the frame itself to determine what action it should take to any action against the frame. Object oriented coding is becoming popular outside of the AI research centers, and object oriented PASCAL compilers are even now becoming available.

In addition to our knowledge base, we must have some form of program to analyze our queries and convert them into some form of database access. Each different form of data storage has its own type of 'inference engine' to form the bridge between the user and the data and to represent the methodology of the original expert. Much of the research in this area centers around the ability to control and predict how the system will function under a broad range of circumstances. These techniques are slowly working their way into the business community to produce management reporting and long range data analysis and prediction.

Expert systems have been successful, when they are, due to the narrowness of the scope chosen for each project. Knowledge based systems are massive undertakings, involving skilled 'knowledge engineers', who are responsible for interpreting the methods of each expert chosen for a system

model, and converting this knowledge and methodology into a program and database. As any analyst knows, even the expert may not know what he is doing when he exercises his talent, so separating the substance from the window dressing can be quite an undertaking. Even after the system is complete, improved technology and current information must still be added continuously to keep the system from becoming obsolete. And yet, expert systems can pay for themselves in a single use, capturing an expertise that would be lost forever.

Even though we may seem to have produced the embryo of machine intelligence, we are beginning to realize that humans think differently than the models we have built. Let us look at how researchers have tried to provide machine intelligence with the ability to solve problems.

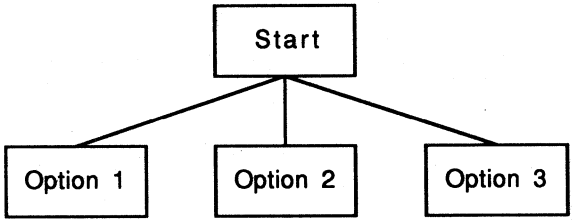
PROBLEM SOLVING

The area of game playing has been one of the most successful areas of research in AI. Most of the leaders of AI were fascinated with games research, due not just to the enjoyment of game playing, but to the limited domain that exist in a game environment, allowing a game simulation to act as a proxy for more complicated real world problems. One of the most popular game simulations has been, of course, chess. The first paper on the subject, 'Programming a Computer for Playing Chess', was published in 1950 by Claude Shannon, and many of the techniques described are used by today's chess playing machines. Chess is a good example of how problem solving techniques have developed.

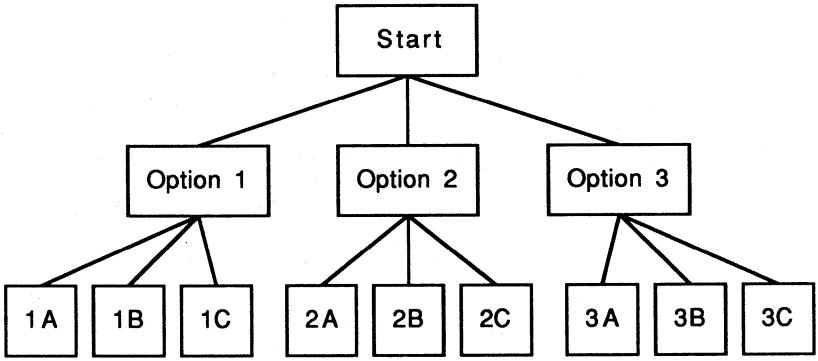
In the beginning, most scholars tended to believe that all that would be needed to develop a good chess program was enough storage and speed to examine all possible moves that could be played for, say, the next ten turns. Most chess experts, by the way, look ahead about six moves. However, if we say that, as an average, we can move at least ten pieces in any one move, and our opponents can move the same, we would need to evaluate 10^{20} moves to look ten moves ahead. Even if we could process a million moves per second, it would take about $3x10^{10}$ hours, or about 3 million years. Obviously, this is not quite acceptable for an afternoon game of chess. Some way must be found to shorten our search. This area of research has been one of the most important areas of AI, and is basic to almost all other AI areas.

In order to examine search techniques, we will represent our problems in the form of a 'search tree'. We start at the top of our tree as so:

From this point, let us say, we have three possible options. We would represent them as so:



Again, let us say that each of these options can have three possible options of their own:



And so forth.

There are a number of ways we can search this tree. The first, and easiest, type of search is called the 'depth-first' search. This, as its name implies, involves searching each branch of the tree to its bottom most leaf, that is, from start to option 1, to 1A, to 1B, to 1C, to option 2, to 2A, etc. Strictly at random, this type of search will, on an average, hit half of the nodes to be searched before finding a 'hit'. The second type of search is known as the 'breadth-first' search. This would involve searching each level, from start to option 1, to option 2, to option 3, to 1A, to 1B, to 1C, to 2A, etc. This search type has the disadvantage of needing a very large intermediate storage area to store the results of searching each level, and, in a random search, will probably hit a greater number of nodes. On the other hand, breadth-first searches are useful in many game situations, since they allow partial evaluation of each major limb of the tree before continuing the search, and this, in turn would allow the program to pick the limb most likely to contain the solution.

Another technique that helps shorten search time is called 'bidirectional reasoning'. In this type of search, not only do you start a breadth-first search from the top of the tree, you can start a breadth-first search from the goal. This is useful for types of manipulations, such as mazes, in which the goal and its precursors are clearly defined, and we are looking for a relationship between the start and the goal.

Another important problem solving tool is called 'means-ends analysis'. This method involves determining, for each node in the tree, what is required to accomplish the execution of the node. These are then analyzed for each complete path from the top of the tree to the required solution, and an optimum strategy for traversing the tree is evolved.

Most problems in AI involve using a combination of techniques to achieve a solution. The dynamic selection of search algorithms based on the type of problem presented is another continuing area of AI research, and is an area that can be of direct benefit to the business community. As the size and complexity of our data base technologies increase, we will need to include many of the problem solving methods of AI just to keep the cost of our inquiries manageable.

NATURAL LANGUAGE SYSTEMS

The ability to understand and use language has always been a yardstick for determining intelligence. Even after the discovery that other species used language, this fact was used as evidence of their intelligence, and closeness to man on the evolutionary scale. Even the intelligence of dogs and cats is not usually considered to be their innate ability to solve complex problems in opening doors and proceeding through mazes, but their ability to understand human speech. Is it any wonder that the use of a natural language, such as English, French, or Swahili as opposed to COBOL, PASCAL, or LISP, is considered prime evidence of machine intelligence?

The problems besetting the analysis of natural language can be easily illustrated in the following examples. First, examine the following sentences:

Time flies like an arrow.
Fruit flies like a banana.

In the first sentence, 'flies' is a verb and 'like' is a preposition. In the second, 'flies' is a noun and 'like' is a verb. This type of uncertainty can give software fits. A second example might be: The ball was hit by the boy with the hat. Did the boy hit the ball using a hat, or is the hat just the means by which we should identify the boy? Change 'hat' to 'bat' and answer the question again.

Natural language research was given an enormous boost by the work of one person, American mathematician Naom Chomsky, who, in 1957, revolutionized linguistics with the publishing of 'Syntactical Structures'. Chomsky was the first person to treat grammar as an area of study that transcends specific languages, an area of study with rules common to all languages and able to be expressed in logical and mathematical terminology. Chomsky's rules of grammar allowed researchers to begin to codify the methods by which humans actually decipher language, and to apply some of these techniques to machine intelligence.

As natural language systems were refined, it became obvious that a rule based language system would become complex to the point of impossibility, not just because of the size of the database, but also because of the inability to debug a system of that magnitude. Object based designs and constrained representation systems are now being applied to the problem in order to simplify the basic systems by reducing the number of rules the system must deal with.

Natural language systems are increasing in importance in the business community. As relational databases become prevalent, natural language interface systems are becoming popular, since this allows a user of no computer sophistication to create complex database queries without learning 'computer shorthand' languages, such as SQL. But even the simplest natural language query processor needs an abundance of data storage and CPU cycles. New ideas from AI research are eagerly awaited.

A second area of natural language systems that is becoming popular in commercial applications is language translation programming. It may seem unimportant to be able to translate *Les Miserables* into Russian using a computer, but the ability to translate technical manuals and documents of all sorts will speed the spread of technologies around the world. It may be possible to read technical journals published in Japan or Germany today, instead of waiting weeks for the translations to be completed. Even computer software interfaces could be translated, expanding the markets for these products, and the hardware they are written for.

We have seen that natural language systems are not simple. On the other hand, the economic incentives for working natural language systems seem to be expanding this area of AI at a rapid rate. Combined with the concurrent research in voice synthesis and speech recognition, we may soon be seeing systems that can be queried in English over the phone to tell us the weather in Bankok, or the balance in our checking accounts.

WHERE ARE WE NOW

We have looked at a bit of AI technology, and we have discussed a few of the applications now under study. We have completely ignored the areas of robotics; visual, aural and tactile perception; self programming and self enhancing systems; etc. The purpose of this paper was to give an overview of the field of AI, with emphasis on those areas that may become applicable to the business community of today.

In general, we have seen that AI is usually very costly, both to design and to execute. Many 'commercial' AI systems were originally developed in traditional AI languages, such as LISP, and PROLOG, and later converted to PASCAL, FORTRAN and even COBOL to achieve the speed necessary to compete in the commercial marketplace. As AI moves from a hit and miss field of study to a fully engineered science, more applications will find their way into the market place. Beware, however, the vendor who is trying to sell you an 'expert' system for $495.90. The most charitable point that may be made would be that an overzealous sales force tacked the appellation to the product without looking the words up. On the other hand, this could be another method of separating the unwary from their money. It will be quite a few years before even a reasonable subset could be found at computer boutique prices.

And yet, certain subsets of expert systems and natural language systems are beginning to appear economical. In not to many years, it will be common to find query languages that seem to be English, because the subset of English implemented will be fairly large. You should expect these programs to be very literal in their interpretation. Asking 'Can you get me the sales figures from October?' may get you an answer of 'YES'. These programs will also tend to get confused often enough for users to comment on the need to desk check the output before treating it as gospel.

We should also see improved database access techniques, with algorithms to optimize inquiries based on prior experience. Our concept of data storage will tend to include concepts such as 'self-defining' databases, whose complete structure, including source, responsibility, editing criteria and report format, will be stored as part of the database itself. We should begin to see object oriented extensions to our current languages. These are extremely useful in rapidly changing businesses, such as life insurance and commodities trading, and would ease the problems of implementing new products or modifying existing

ones.

Though we may not see Karel Capek's robots in our offices next week, AI is here to stay, and will be a large part of the office of tomorrow.

BIBLIOGRAPHY

I have found the following books to be extremely interesting while researching this paper. I would highly recommend them to anyone who is interested in the subject of Artificial Intelligence.

THINKING MACHINES The Search for Artificial Intelligence by Igor Aleksander and Piers Burnett. 1987 (Alfred A. Knopf, Inc). This book is an exceedingly clear book describing the area of AI.

The HANDBOOK of ARTIFICIAL INTELLIGENCE volumes 1, 2, and 3 by Avron Barr and Edward A. Feigenbaum. 1981 (William Kaufmann, Inc). The definitive text on artificial intelligence. No study of the field could be complete without it. Again, this book is quite easy to read, and even the most difficult examples are presented with grace and style.

AI in the 1980s and BEYOND an MIT Survey, edited by W. Eric L. Grimson and Ramesh S. Patil. 1987 (Massachusetts Institute of Technology Press). A provocative look at AI present and future.

GODEL, ESCHER, BACH: an Eternal Golden Braid Douglas R. Hofstadter. 1979 (Basic Books, Inc). One of the unique books of our time, GEB is a marvelous blend of treatise and nonsense, exploring the world of artificial intelligence in a manner akin to Lewis Carrol, creating a world of Zen AI. This book is not a quick read, but it is well worth your while to explore it.

MASTERING AI TOOLS and TECHNIQUES Ernest R. Tello 1988 (Howard W. Sams & Company). Possibly the best overview of Artificial Intellegence written this decade. Tello examines the entire field of Expert Systems, Design Tools and Natural Language systems in terms of real world products and applications. From the PC based systems to DENDRAL and PROSPECTOR, the book goes into enough detail to understand the product, without overwhelming the reader with trivia. A must for anyone interested in the field.

ARTIFICIAL INTELLEGENCE and NATURAL Margaret A. Boden, second edition 1987 (Basic Books, Inc.). An interesting discussion of AI from the humanistic point of view. This book manages to be both technical and philosophical without sacrificing either. Again, this is not a quick read, but many will find it interesting.

EXPERT SYSTEMS IN BUSINESS, a Practical Approach Michael L. Barret and Annabel C Beerel 1988 (Halsted Press, a division

of John Wiley and Sons). As the title fortells, this book describes the practical aspects of implementing an expert system in a business environment, from the initial justification through implementation. The book discusses such topics as choosing an expert, motivating the expert, dealing with management, developement life cycle, etc. This book will not teach you the technology of implementing an expert system, rather, it will show you how to implement an expert system in a business.

STATISTICAL PROCESS CONTROL IN A DATA PROCESSING ENVIRONMENT

LARRY VEALE
HOECHST CELANESE CHEMICAL GROUP
P. O. Box 58190
Houston, Texas 77258-0190
713/474-6200

Insuring quality in manufactured goods began with inspection at the end of the production line. Separate quality control departments were created and quality control inspectors were hired. In the late 1940's a statistician named W. Edwards Deming began teaching the Japanese how to use statistics to sample their processes and to use the techniques he taught to inspect their work during production. This change has enabled industry to reduce costs due to waste and rework. At the same time their products have much less variation than customers expect. In other words, all products meet or exceed customer expectations.

This evolution of the Japanese quality emphasis is shown in Figure 1[1]. This graph shows how the Japanese have grown their quality concept from product inspection (after production) through the control of the process (during production) to the design of the product and process (before production).

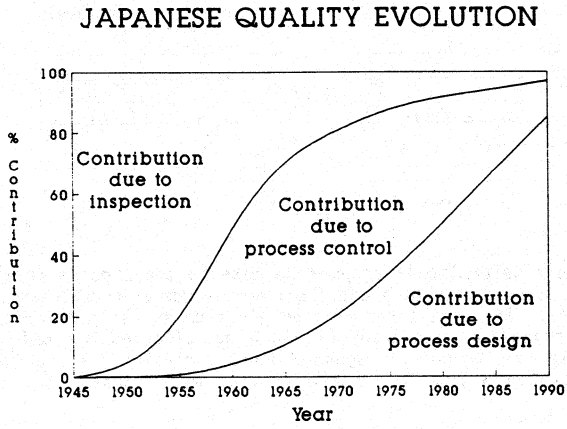


Figure 1

Manufacturing processes apply statistical process control (SPC) techniques quite well today. A process or subprocess is sampled at key points and operators identify causes if it is out of control. Processes are improved by learning about these causes.

As is often the case, people in a different situation tend to dismiss a technique when we fail to see how that application fits in our own area. One question then is, how can we apply the SPC techniques we see applied to a manufacturing process with discreet outputs to a data processing environment whose output is a service?

My purpose in this paper is not to explain the details of SPC, but knowledge of some definitions is required to understand how we use the techniques. A simple definition of SPC is the use of statistics to help maintain or improve processes. One of the SPC tools, the control chart, identifies whether problems are caused by special causes or are part of the random variation of a process. A process is out of control if a sample is outside the control limits, eight consecutive points are above or below the average, or six consecutive points trend up or down. If none of these conditions is violated, then the process is considered stable. Figure 2 illustrates these situations on control charts.

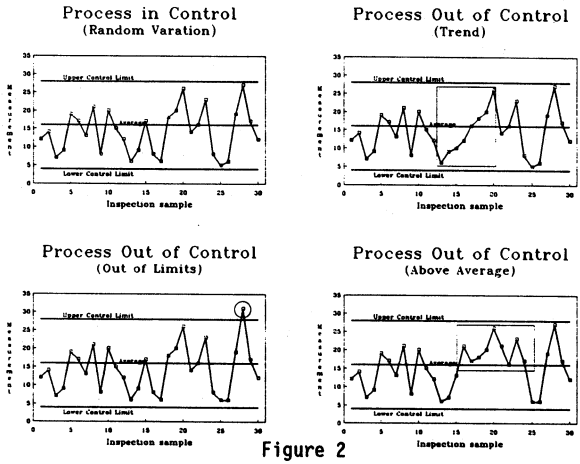


Figure 2

SPC will also help determine if changes we make to the process really improve it. Improvement is evidenced by producing items nearer a target value, with less variation from it. This smaller variation gives us an earlier indication that our process has changed or that a special cause has affected it. Using SPC helps us not waste time chasing normal random process variation. We can concentrate on the significant few and ignore the trivial many causes that affect our processes.

About six years ago, Celanese Corporation (now Hoechst Celanese) began to orient ourselves to a much higher emphasis on quality, to change our corporate culture, and to become a leader in applying SPC techniques to our processes to enable us to stay a leader in commodity chemicals. The use of SPC has been well accepted and integrated into many operations areas of our company. This paper presents my experience with SPC in our data processing environment.

The three case studies I will be talking about will involve a simple application of deciding when to inspect a product, an operations application of control charts to identify and resolve customer problems, and an application of a form of the Shewhart cycle to improve processes.

I. INSPECTION DURING OR AFTER PRODUCTION

The first example was generated after an episode of having a system problem, needing to restore files from a backup tape, and not being able to do so because of inferior quality tapes (as we later found out), even though we verified tapes after they were written. We upgraded our tapes to the best available, and started tracking them with a tape library. We continued to verify our tapes after they were written, but by using higher quality tapes we moved our control point from final production to raw material.

After roughly a year of using these tapes to do reloads and restores with no tape errors, the operators questioned whether we needed to verify the last tapes made at night (which were critical DBSTORES) before they left, or if they could verify them the next morning. This request involved the cost of 30 minutes for an operator every night versus the risk of not having a good set of backup tapes in the event we needed to recover the database and transaction log files. This situation is not an option for our shop. To avoid this situation, all customers (about 40) would have to be logged off, and another set of DBSTORE tapes made, then verified. This would involve an hour of unproductive time for 40 people if this occurred. On the other hand, verifying tapes during the regular workday would reduce the time an operator was here at night, and would make the system available earlier in the evening.

We have accepted the risk of rarely being without a valid backup between the time a backup is made and it would be verified (12 hours). While this is half a day, there are fewer database customers for most of those hours, so the risk of losing transactions is small.

Deming has presented in Chapter 15 of his book, Out of the Crisis,[2] a way to calculate whether the cost of inspection during production or the cost to rework the defective parts upon final inspection is greater. The number of failures for which the cost in terms of time spent to remake a set of backups (rework) equals the time spent every day in verifying the backups at night (inspection) can be calculated.

The calculation for the break-even point is presented below:

$$\text{INSPECT COST: } \frac{0.5 \text{ hr} \times 5 \text{ days/wk} \times 52 \text{ wks/yr}}{\text{REWORK COST: } 40 \text{ customers} \times 1 \text{ hr}} = 3.25$$

This means that for our situation if there are three or fewer verification failures a year, then the least cost approach is to verify during the day.

Another key point is that because we have upgraded our tapes (raw materials), and have improved our process by buying better inputs, we can offer our customers a better output (more uptime) with virtually no cost to us. This is the essence of an effective quality effort - one that improves products or services and reduces cost. Without cost improvements, better quality is only a start.

II. USE OF CONTROL CHARTS FOR EARLY PROBLEM RESOLUTION

The second example involves use of control charts to track problems that the computer operators must spend time on, but are not defined as part of their normal assignments. We had several false starts on this project, but we found the easiest way to approach it was to define what a "problem" was for everyone involved. In our case, it was just what is defined above: a non-routine task that is not expected to occur (Table I). This can involve ports not working, preventing customers from connecting to the HP-3000, application software failures, communications problems between computers, customers not understanding procedures, etc.

Once we defined what we were going to track, we collected data for several weeks to establish a base line for our process. The data was collected on a daily basis, but also tracked on a weekly basis to help smooth out daily variation in the number of incidents. Next, the average of the data was calculated and plotted, and the upper control limits were calculated for the same time interval and plotted. For this system, the average is calculated as the number of incidents divided by the number of days or weeks. The control limits are the average plus or minus 3 times the square root of the average. Thus, if the number of incidents for a six-week period (30 days) is 60, the following calculations apply:

$$\text{daily average} = \frac{60}{30} = 2.0 \qquad \text{weekly average} = \frac{60}{6} = 10.0$$

$$\text{upper control limit (daily)} = 2 + 3\sqrt{2} = 4.2$$

$$\text{upper control limit (weekly)} = 10 + 3\sqrt{10} = 19.4$$

$$\text{lower control limit (weekly)} = 10 - 3\sqrt{10} = 0.6$$

There is no lower control limit in the daily chart since zero errors on any given day is not unusual.

The problems are then tracked on a daily and weekly basis and notes are kept on the daily chart of other events which may affect the number of incidents. The daily plot is shown in Figure 3 and the weekly plot in Figure 4.

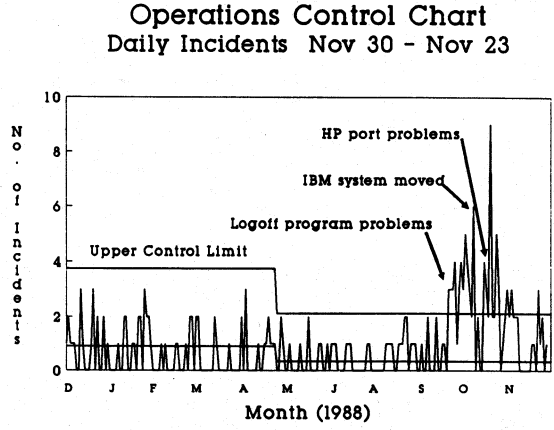


Figure 3

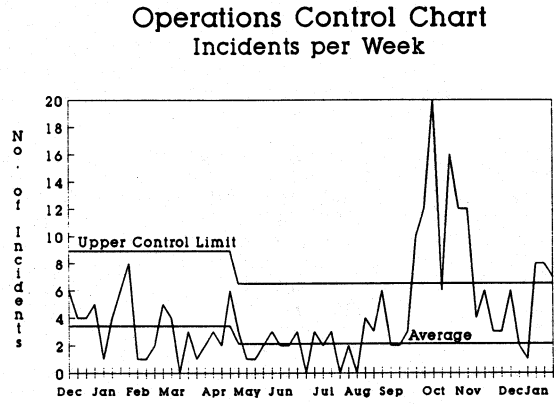


Figure 4

The control chart gives a picture of what is happening in the process, but other SPC tools must be used to get a fuller understanding. A key component is the inclusion of comments on the charts. These note changes to the process or record events which may be forgotten when trying to later analyze the situation. An example: after we had defined our basic process, we noted that many of our problems were associated with the dial-in ports (I'll explain how we knew this in a moment). We developed a procedure to test our dial-in ports weekly and identify whether any problem is on the computer side or the PBX side. With this procedure, we identified a bad card in our PBX that connected to these ports, replaced it, and the number of incidents fell. Our number of problems fell because we had changed our basic process. When we had roughly 20-30 days of data at the new levels, we recalculated the average and upper control limit. This is seen in Figures 3 and 4 as a drop in the average and upper control limit. The weekly upper control limit and average were recalculated at the same time.

On the other hand, there is a time when the number of incidents goes above the upper control limit. This was quickly traced to a special cause. Customers began to let a program that logs off terminals after a specified period of inactivity log their terminals off instead of logging them off themselves at the end of the day. This program was sometimes (but not always) disabling the port on the HP, requiring the operator to reset it. It took several weeks to figure out the solution, but when it was installed, we expected the number of incidents to drop. Surprisingly, the first day after this special cause was resolved another special cause popped up, unrelated to the first and from the control chart alone, it might not be evident that the first problem was really fixed. Another problem also started occurring within days of these problems further complicating the situation, but because we have noted key information as comments on the charts and understand the operations process, we are able to see and address these situations as special causes. We also understand that the process may have also changed due to our increased emphasis on logging these incidents.

Now for how we determined what was causing our problem when the chart went out of control. We had originally divided the problems we noticed into 10 categories. A Pareto chart was used to rank them in number order, the most numerous problem charted first. The original chart is shown in Figure 5. This same type of graph was used to help us understand what was causing our large increase in problems. We plotted two time frames in Pareto form: 31 incidents while we were in control (Figure 6), and 37 incidents after we went out of control (Figure 7). The X axis (causes) was plotted in the same order for each graph to avoid confusion. The number of terminal problems increased to the point that they went from 45% of the total to 83%. This fact is illustrated in the pie chart of the same data (Figures 8 and 9). Checking the detail log sheets for these incidents showed that most of the problems were due to having to reset ports that were disabled the night before by the log-off program, although other causes did contribute to the number of incidents involving terminal hardware.

Pareto Chart of Incidents
by Category

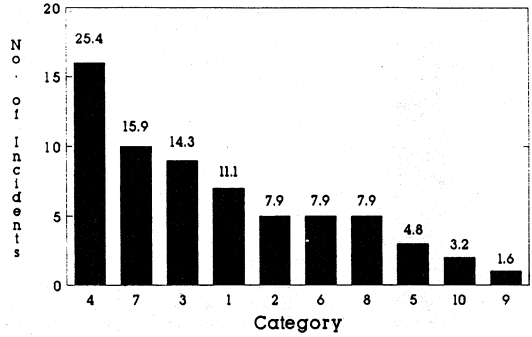


Figure 5

Problem Cause Distribution
In Control (31 Incidents)

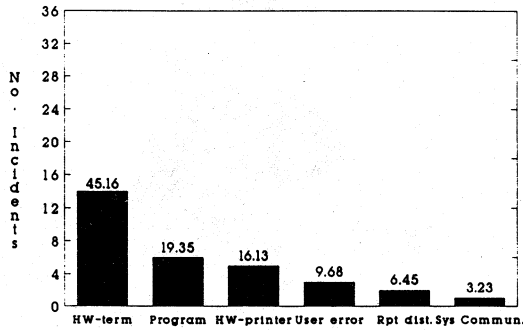


Figure 6

Problem Cause Distribution
Out of Control (37 Incidents)

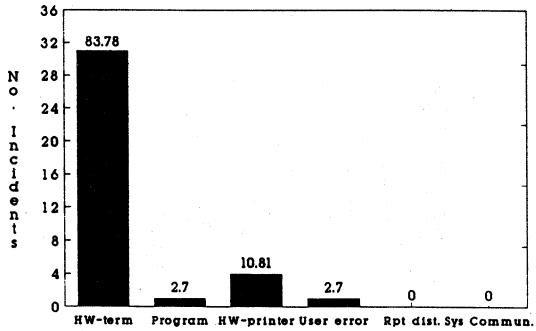


Figure 7

Problem Cause Distribution
31 Incidents

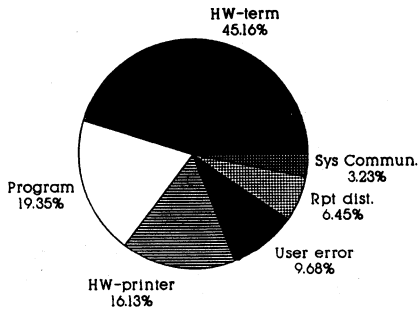


Figure 8

Problem Cause Distribution
Out of Control (37 Incidents)

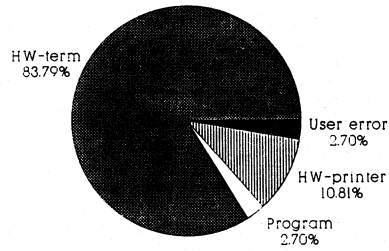


Figure 9

So the control chart helps us know when something has changed for better or worse, and the comments give us clues as to why, but we also need other tools and a good understanding of our process to really grasp whether a change in the process has occurred, or a special cause has impacted our system, or even a combination of them.

III. USING THE SHEWHART CYCLE TO IMPROVE A PROCESS

The Shewhart cycle is not a statistical tool. It is a method to improve a process based upon the knowledge the individual or group has about a process at a given point in time. Like SPC tools, it is used to understand and improve processes. A representation of the cycle is shown in Figure 10. Based on current knowledge, a plan is formulated and predictions are made about what will happen if certain changes are made. The observations and expectations are synthesized into new knowledge and action is taken. This process is repeated until a satisfactory level of performance is reached for the process, or another process is targeted for improvement.

We are using this SPC technique to design a method to address and satisfy programming requests for data processing services. We have four goals in mind for this project (see Table II):

- to better estimate the time it takes to complete a request
- to better understand customer needs when they make a request
- to provide better customer support
- to formalize analyst training on all systems.

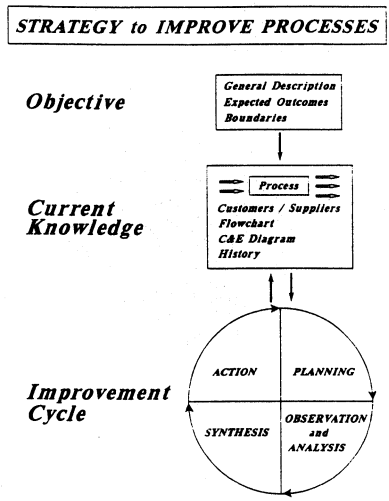


Figure 10

The first goal was addressed in two meetings with the analysts. A current projects board was designed and installed. Projects are broken into components and each day we indicate which parts of which projects we have worked on. This procedure has been in effect for several months, and has given us the expected result of better communication of "maintenance" type projects on which all of us work. It has also created a record that we all can review over a 4-week period for recurring problems, jog our memories to discuss projects in more detail, and communicate who is working on what project.

A point to make in discussing this effort is to demonstrate that the process improvement cycle can work for designing processes as well as improving them. We chose to start from zero base rather than trying to define our process and improve it. We were not happy with what we had, and we all had ideas about what we wanted to achieve with any new process.

We still have a lot of work to do. The middle two objectives are being addressed by assigning each application system to a specific analyst, and by having the analysts work closely with the departments they support on a non-project basis. The intent here is to proactively use our knowledge to help our customers.

We now have a regular weekly review meeting to provide more detailed status reports, discuss problems, planned approaches, and in doing these things, provide some cross-training in the area of skills. With the addition of another analyst, I will be realigning assignments for each of our support areas and applications systems, with the intention of rotating these assignments regularly. This rotation will require all analysts to have a working knowledge of other areas to provide strong support.

IV. CONCLUSION

Statistical control methods are as valuable a tool in the data processing area as they are in a manufacturing area. We may not inspect every fifth widget on an assembly line, but the measurements are very tangible and the results are both graphic and informative.

There are a variety of statistical tools available. The trick is to choose the right one and measure the right variables. But the beauty is that if everything is not correct the first try, a good analysis of the technique combined with knowledge of the process being worked on should indicate how to improve the use of the tools on the next cycle. At the very least, it will improve the process knowledge, which will lead to customer satisfaction ... the ultimate goal. The better the knowledge of the processes we work with, the better we can control them to produce the desired results whether those results are manufacture of automobiles or response time of a computer system or the time it takes to get a question answered.

BIBLIOGRAPHY:

(1) Quality Progress, May, 1986, Sullivan, L. P., "The Seven Stages in Company-Wide Quality Control"

(2) Deming, W. Edwards, Out of the Crisis, pp. 407-411

(3) Provost, Lloyd, Process Control Associates, Personal Communication

OPERATIONS CONTROL CHART PROBLEM DEFINITION

- Not part of the normal duties
- Non-routine situation
- Unexpected
- Operator defined

Table I

CUSTOMER PROJECT REQUEST PROCESS DESIGN OBJECTIVES

- Better project management
  - Good estimates of time and effort required for requests
- Better understanding of customer environment and business
  - More capable of providing solutions
  - Less trial and error prototyping
- Improved customer support
  - By their definition
  - More product line oriented
  - Reduced project backlog
- Provides analyst cross-training

Table II

TRADING UP: Options for your used HP Equipment

Donald G. Cowan
Ross-Dove Company
1118 Chess Drive
Foster City, California 94404
(415) 571-7400

Open the Wall Street Journal and you will read about corporate America's "restructuring," "streamlining operations," "leveraged-buyouts," "layoffs," and the "changing face of corporate America." Corporations are scrutinizing each and every department and division for bottom-line performance, and their employees are under increasing pressure to contribute to the bottom-line to keep the corporation moving ahead. Data processing executives are under the gun to maximize their data processing budgets, and make data processing strategy based on the strictest price/performance guidelines, with bottom-line accountability.

Maximizing the recovery rate on your used data processing equipment is a small but important part of your overall strategy to ensure that the price/performance guidelines of your company are met. It is vital that data processing executives explore all the options available for their used equipment before a system upgrade is made. And the data processing department which uses Hewlett-Packard equipment does have options. These options exist in large part

because used Hewlett-Packard equipment has value in the marketplace.

A BILLION DOLLAR MARKET FOR USED COMPUTER EQUIPMENT

The used or secondary market for computer equipment is estimated to be over one billion dollars annually. The players in this international market range from NYSE-listed companies, which buy,sell,trade and lease millions of dollars worth of equipment, to small, highly specialized brokers. Manufacturers respond differently to the secondary computer market. Many manufacturers encourage the development of a secondary market, realizing seeds are being planted which may yield customers for new equipment and services and may increase market-share and the installed customer base. More than 300 firms are actively buying and selling used IBM equipment, many of which belong to the Computer Dealers and Lessors Association, a national trade organization. More than100 firms buy and sell used DEC equipment, and a rapidly growing number of firms specialize in buying and selling used Hewlett-Packard equipment.

YOUR USED HP EQUIPMENT HAS VALUE

Why are more than 400 firms interested in buying your used IBM, DEC and Hewlett-Packard computer equipment? Value, your used equipment has value. High-technology equipment is prone to a rapid obsolescence rate. You probably know of high-technology

equipment costing hundreds of thousand of dollars that is considered worthless in two or three years. But your used Hewlett-Packard equipment has value long after the initial purchase. Used Hewlett-Packard equipment is in demand because end-users want the reliability, quality, support, access to an extensive library of software, and the security of the Hewlett-Packard name, but do not need the latest generation of computer equipment. Many applications can be satisfactorily performed by systems that are more than one generation old. Even end-users that do buy the latest models may keep an older system on hand for technically undemanding applications. Many end-users would rather purchase a used Hewlett-Packard system than a new system manufactured by someone else.

The secondary computer buy/sell market is a simple economic supply and demand marketplace. It is not unusual for prices in this market to fluctuate from week to week. The manufacturer sponsored trade-in market can be driven by factors and forces not purely economic. And as such, values assigned to your used equipment are oftentimes very different from the purely economic supply and demand market. That is why it is important to compare your alternatives in order to maximize the recovery rate on your used equipment. Maximizing this recovery rate will help ensure your price/performance strategies are met or exceeded.

Because your used Hewlett-Packard has value, options for your used Hewlett-Packard equipment exist: (1) Hewlett-Packard is willing to take old equipment in trade for a new system; (2) secondary market dealers nationwide will make you an offer for your used

equipment over the phone; (3) an array of computer brokers are willing to work for you on a commission basis to sell your used equipment; and (4) auction firms are anxious to offer your equipment at auction, or to enhance an auction by including your equipment in it.

No matter which of these options you exercise, you should keep the following ideas in mind when selling used computer equipment: (1) Keep your equipment properly maintained and eligible for manufacturer service upon transfer. Contact the manufacturer for specific information. Computer equipment not eligible for manufacturers services loses much of its value. (2) Keep your systems installed. A potential buyer may want to inspect your system while it is up and running. You can also sell your system "where is," meaning the buyer is buying it installed and must arrange for de-installation. (3) The used equipment marketplace is a cash market. Do not let your equipment leave your premises without full payment or a high degree of confidence in your buyer. Ask for and check references. 4) Have available a complete inventory of equipment and the date when it will be available. A copy of your service contract will usually do.

Each of the options discussed below have their own advantages and disadvantages. What will work best for you depends upon your particular needs.

SELLING TO A DEALER

Selling your used computer equipment to a secondary market dealer may prove to be an economically desirable alternative to a

manufacturers sponsored trade-in program. A secondary market dealer is in the business of buying and selling used computer equipment. The secondary market dealer will buy your used computer equipment, and incur the cost of de-installation, transportation, storage and marketing until a new buyer is found. The dealer will make his profit on the difference between the buy and sell price.

Selling your used computer equipment to a dealer is easiest and least costly in terms of time, money, possible hassles and involvement but will yield a smaller recovery rate. The dealer will usually give you a written quote good for 30 days, pay you in cash, and allow you to move on to other endeavors. The drawback to selling to a dealer is: the dealer receives the financial upside when the equipment is eventually sold to an end-user.

USING A BROKER

A computer broker is the middleman in the transaction between buyer and seller. The broker will bring together the buyer and seller and facilitate a smooth, orderly transaction, and is paid a commission by the seller upon completion of the sale. Using a broker offers some distinct advantages, but does have some drawbacks which must be carefully weighed. The advantages of using a broker are: (1) the broker is working for you to get the highest amount possible for your used computer equipment; (2) a broker will provide you with knowledge about the current buy/sell market conditions for your equipment; (3) a broker will have extensive end-user contacts and

provide insight on how to best market your equipment. The drawbacks to using a computer broker to sell your used computer equipment are: (1) you do not know when or if your equipment will be sold; and (2) you must incur the costs of keeping your equipment properly maintained until a buyer is found.

THE AUCTION

An auction, as defined by Webster, "is a public sale in which persons bid on property to be sold and property is sold to the highest bidder." In 1979, Sotheby Parke Bernet held the first ever computer-only auction in which it auctioned IBM mainframe computers. Since then, auction firms across the country have been selling computers alongside other equipment and furniture at auction.

The auction method is ideally suited for selling large amounts of equipment fast and effectively. The competitive bidding process ensures the seller of receiving fair market value for his equipment. It is ideal if your equipment is from different manufactures, and the condition ranges from new to scrap. The auction method is particularly effective if you have non-data processing equipment to sell with your computer equipment, such as furniture, test equipment or manufacturing equipment.

An auction firm will either hold an auction for you on your premises, if you have a large amount of equipment, or include your equipment in another auction as a consignor. The auction company will advertise and promote the sale, sell all of your equipment in one

day for cash, supervise the equipment removal, collect the funds, and present you with a net check.

Before hiring an auction firm to sell your used equipment, check to make sure the firm and its auctioneers are licensed by the state and belong to the National Auctioneers Association. Also be sure they are familiar with your equipment and have sold similar equipment in the past, and use a good marketing campaign which will attract a large number of end-users to the sale.

SELLING TO AN END-USER

Selling your used computer equipment to an end-user will yield the highest recovery rate, but involves a commitment of time, manpower and money. It is very important to weigh the costs of selling your equipment yourself against the increased dollar potential. You will not be contributing to your corporation's bottom-line if you spend $20,000 (hard and soft dollars) selling your equipment, and then only receive $10,000 more than a dealer had offered.

In the most basic form, selling to an end-user involves: (1) deciding how much your used equipment is worth; (2) finding potential buyers and informing them of the sale; (3) matching your equipment with the buyer's needs; and (4) coordinating the de-installation and removal with the buyer.

Deciding the correct selling price on used computer equipment is the single most important aspect of a successful sale. What you must decide is the fair market value of your equipment. Fair market value is

the price a willing buyer and seller would agree upon in the open market with no outside pressure or influences. An equipment appraiser who tracks and analyzes computer equipment is the best source of this information. An appraiser will be able to tell you the current "wholesale market price" (what a dealer would pay) and current "retail price" (what an end-user would pay).

Armed with this current price information, you will be ready to find end-users and inform them of the sale. Hewlett-Packard end-users are everywhere. Make inquiries at user's groups, place an ad in trade publications, or rent a list of Hewlett-Packard users. In the ad or mailer, give a complete inventory of equipment, date available, price, contact name, and phone number and state whether it is currently under a maintenance contract.

You will receive many inquiries from potential buyers. Talk with each of them and ascertain: (1) which buyer's needs are most closely met by your equipment; (2) whether funds have been allocated for this acquisition; and (3) who will make the purchasing decision and when will the decision be made. You will want to structure a transaction in which both parties feel they have been treated fairly and have received good value for their money and equipment.

When you have located your buyer, ask for a good-faith deposit to hold the equipment while de-installation and transportation arrangements are being made. Also, consult with your in-house counsel or financial officer to make sure the sale conforms to all UCC regulations and that sales tax is collected, if applicable.

Each method of selling used computer equipment has its distinct advantages and drawbacks. Your unique requirements, and availability of time and resources, will dictate which alternative is best for you. But by quickly exploring these options you will always be assured of maximizing your computer price/performance ratio, and that you are contributing to the corporation's bottom-line by always getting the most computing power for the company's dollar.

A resource guide to assist you in selling used equipment will be available at the conference.

Task Management - The Way to Success in System Projects!

Robert R. Mattson
9545 Delphi Road S.W.
Olympia, WA 98502
(206) 736-2831
(206) 352-5038

Abstract

For most system professionals the successful management of projects is fundamental to their own success. Millions are spent each year on project management software and training. This large sum of money is spent because people know that the risk of failure in projects is large. They know this from their own experience and observations as well as from the "publicized" disasters. Why hasn't the success rate of system projects improved in proportion to the dollars spent? A fundamental reason is that we've focused on project management rather than task management! In other words, one must learn to walk before one runs. We must learn to manage tasks before we can manage projects! This talk outlines what should be done to manage a single system task such as developing a program. Masterful application of these concepts can lead to added success in system projects.

Introduction
or
"Systems Mean Projects, Projects Mean Tasks."

Software systems revolve around projects, big and small. Whether it is putting in a multi-million dollar system or making a modification to one existing program they usually involve "projects" of one sort or another. So, software professionals are involved with projects all the time.

The trouble is, our profession is having a tough time achieving success in these projects. Research indicates that less than half of all projects are finished on time and budget. And a surprisingly large number of projects are cancelled before they are complete or even after completion because they did not deliver as expected. Even for those projects finished within time and budget; I have observed that things seldom went as smooth as one would wish.

To try to help us succeed in these myriad of projects, we look for solutions. We spend millions of dollars each year on new programming languages, new analysis tools, project

management software, etc. Each of these we hope will help us deal better with meeting our project time, quality and cost deadlines.

Yet, can we discerned great improvements in our project track record? The trade press in the last year still has had many examples of missed deadlines, cancelled projects, etc. Why is this still such a problem? There are many reasons. One is that systems are getting more complex and larger. Realistically, it is impossible to know all the reasons. Nevertheless, there is clearly one area that is crucial to all the projects in which we need to improve. This is the area of task management.

Task management is the fundamental building block of project management. I've studied, applied and struggled with the techniques of project management for years. Finally, it dawned on me that almost universally this subject has been focused on too high level of view. The focus has been merely what a project's tasks are and how we manage the "project!" The focus has not been on how each of the individual tasks is defined, planned and managed. Yet, no matter the size of the project, it can and will be reduced to a series of tasks. And it is within these tasks that the project is actually done.

Therefore, this area of how we manage a single task is what I'd like to explore with you.

Why We Should Care About Task Management
or
"How Goes The Task..... So Goes The Project!"

But why should we care about task management? The key here is that you can't manage a project without managing the tasks! The length of time a project takes is equal to the time from start of the first task to the finish of the last task. The cost of a project is the sum of the costs of all the tasks done on the project. The quality of the project is the result of the quality of the work done on each task.

When a project is over budget it is because one or more of the tasks were over budget or because we did tasks for which we never planned.

One of the most common mistakes in project management flows from not understanding this fundamental relationship between project and tasks. A clear example of this is when a project manager sees the first missed task deadline and doesn't take action or assumes the project budget will still

be met. My experience tells me that when this situation occurs then the project will most likely not reach it's cost, time or quality goals. This is because the project manager doesn't understand the key relationship between project and task. Or they don't understand how to manage a task.

The conclusion is clear. If our project's tasks do not meet their time, cost and quality goals then the project cannot! Conversely, if you manage the tasks well then managing the project becomes much easier! So let's address how a software task should be "managed."

How To Manage a Software Project Task
or
"Fundamentals, blocking and tackling..." the coach said.

The old football analogy really has meaning when applied to task management. There are some task management fundamentals that one needs to understand and apply. And, as important, all the project team members have to be skilled in them as well. If these are done right then the task has a good chance of succeeding, and conversely....do them poorly and problems are bound to show-up. What are the fundamentals requirements of excellent task management? The fundamentals are:

1) Written Task Description/Deliverable/Quality Planning
2) Written Task Deadline/Time/Subtask Planning
3) Written Task Cost Planning
4) Doer's Monitoring, Replanning and Status Reports
5) Supervisory Quality, Time and Cost Review and Feedback

Let's address each of these in order.

Written Task Description/Deliverable/Quality Planning
or
"I did what I thought you wanted...give me a break."

The first place to start with any task is to describe what is to be accomplished. This is the specification of the goal, the deliverable. In systems the challenge is to describe a task in adequate detail to insure that all important specification/quality parameters are clearly understood by both the specifier/manager and the doer.

It is <u>easy</u> with software tasks to <u>NOT do even an adequate</u> job of detailing the specification/quality parameters. This unfortunately is done too many times each day! And, if an inadequate specification is done, then all

types of problems can arise. The most serious fundamental problem is that the doer doesn't produce what is desired! In this most common scenario, the time and dollars budgeted are used up ....and suddenly one finds out that deliverable planned for doesn't exist. Then we must scramble to decided how to deal with this reality? How do we get what we need? How much time will it take and at what cost? How will we deal with our "customer"...can we hide it and "make it up somewhere else!"

Yet, identifying that inadequate task specification is a problem is one thing. The question is...how does one go about correctly specifying what is to be done? This is a whole paper in itself. In brief however, the best way I've found to do software task specifications involves defining a comprehensive set of "quality parameters." These are defined for a particular task in terms of what is unacceptable, ok and excellent. These are defined in such a manner as to clearly allow all parties involved to know what is expected in each area. Further each parameter is given a "weight" that represents the relative value if the excellent criteria is achieved. After a task is done, both the doer and reviewer will "score" the resulting product based on the quality parameters. See appendix A for a copy of the Software Quality Form that is used in this technique. For a more in depth discussion of this technique...see the paper Software Quality...Let's Discuss This Can of Worms! (Robert Mattson, Hewlett-Packard Business User Conference Proceedings, Orlando, Florida, August 1989)

A big advantage of this technique is that it is formal and written! I have found, many times, that the manager and doer can have big differences between what they each perceive the deliverable to be. But, when one has to formally write down these expectations the differences and fuzziness become much clearer. Without such a formal written approach the chances for "mistakes" and "misunderstandings" is tremendous.

A sometimes overlooked item, when dealing with a task is to describe the prerequisites that must be available for the task to begin or proceed. I've seen many tasks started; only to discover that some required prerequisite item was missing. This either made finishing the task impossible until the item was made available or required the person to take time to produce the item themselves. Planning a task should require the specification of all expected/required prerequisites.

It must logically follow only when a person knows precisely what is to be produced can they begin to determine

how long it will take them to deliver the product! Yet too often people try to do software task management without first defining what is to be done in a precise manner. Then they wonder why there is such confusion about the success of what has been produced and/or how long it should take. The lesson is that good task management starts with clear task specifications!

Written Task Deadline/Time/Subtask Planning
And Written Task Cost Planning
or
"I'll work hard on it until I'm done....I'm trying"
or
"I guess I just wasn't born a good estimator!"

Assuming we have done a good job of specifying what is to be done, our next "step" of task management is really two very inter-related steps. The first of these involves doing subtask planning, time estimation and deadline setting. The second step closely relate to the first one is preparing a cost estimates. These are very inter-related because of the very close relationship between the time spent on a task and its cost. Generally, the longer something takes to do the more it costs. Somewhat contradictory is the fact that sometimes, if we spend a lot of money, we can shorten the time required to complete a task (ie. buy faster computer, new software tool, etc). The key here is that one cannot estimate the time to do some well defined task with out knowing the resources in people and dollars that one has available.

This issue of people resources mentioned above deserves more discussion. In software systems, as much as any place, there are large differences in the capabilities and productivity of people. This definitely influences the estimates of the time and cost to do any task. Differences between people are so great that without knowing who will do a task..it is literally impossible to accurately estimate how long a task will take. There is not such a thing as a generic programmer/analyst! If one doesn't know the person(s) doing the task then the only other strategy is to use one of three guess strategies...worst case, best case, and some type of average.

Let's assume we know the resources that are available to apply to the task. The next step is to figure out how long it will take, given the resources, and when we can commit to having it done (the deadline). Fundamental to this step is understanding that this estimating must be done primarily by the person assigned the task. These estimates of course

are reviewed by the project manager/supervisor. Regardless of this review, the rule is... the person who must meet the commitment makes the commitment.

Unfortunately, there is probably no harder thing for the average system person to do than estimate the time it will take for he/she to complete a task. Why is this? Partly it is due to the nature of the system tasks. The number of possible variables that can influence the time is large. It is also because very often we haven't defined the actual deliverables well enough. Still further, many people suffer from not understanding how to plan a task for which they are responsible.

I know of no magic way to solve the number of variables that can influence a task. However, a good task specification can help. We've already dealt with how to do a better job of defining the task. So let's address the fact that many system people simply don't understand how to do a task plan. This results in the following observed behaviors.

1) They don't make sure they understand what is to be delivered.
2) They don't know the prerequisites required.
3) They don't make sure they understand any imposed deadlines that the "customer" may have.
4) They don't understand the resource and dollar constraints.
5) They don't subdivide the task into sub-tasks, mini-deliverables and mini-deadlines.
6) They don't place much personal value on meeting any deadline to which they commit.

Further, the person is hampered because, in the past, he/she hasn't kept track of what sub-tasks are done on similar tasks or how long it actually took to do. Further they haven't learned that it is more important to make a longer deadline than to miss a shorter optimistic one.

Here, I'd like to emphasize the fact that too many software professionals aren't aware of what "tasks" they do or even more critically how long it takes them to do these tasks. This is compounded by the limitations in accuracy of human memory. I have personally observed how frequently people cannot accurately assess how long it took them to do something even immediately when they've finished it! Further, if a couple weeks goes by the accuracy may be off by many factors. This means someone can say it took be a couple days when it really took them a week! Look for this phenomenon yourself and I think you'll be surprised at how

common it is. The ultimate cure for this will be when system professionals log the actual tasks and times that are done for their own learning purposes.

One of the inaccuracies associated with knowing how long a task takes is what I call the "lost redo." This is the time spent after the task is "done" to actually finish it. This is usually not included in a person's memory of how long some task took to do...unless it was very large or otherwise significant. And, surprisingly, when an organization has procedures that make people keep track of their time, the lost redo is even more prevalent because it tends to be purposefully hidden! This is because, in those organizations, there are usually serious consequences to a person if they miss an estimate or deadline and so they play games with the recording function to distort what actually happened.

A discussion of this topic is not complete without dealing with deadlines. In excellent task management, a task deadline should be no longer than two weeks. If the total task deadline is longer than two weeks then either the task should be broken into smaller tasks or sub deadlines should be set. My experience shows that task deadlines greater than two weeks tend to be missed more often and have greater over-runs than those of under two weeks. These are the formal deadlines. In practical fact, excellent task management from the doer standpoint really involves "informally" setting many small deadlines associated with the sub-tasks...and meeting those.

Just a word on cost awareness. All hours estimates should be translated into dollar cost. Once we've estimated the time to do a task it is simple multiplication and addition to get the cost of the task. This will aid in giving people a good cost awareness. People who do a good job of task management seem to treat the task as if they were both customer and supplier. They tend to have good cost awareness. They have an attitude that realizes that it isn't just time that is over-run it's money.

We've now dealt with specification, sub-task planning, time and cost estimating and deadlines. Now we'll see what the doer of a task does when working on a task.

Doer's Monitoring, Replanning and Status Reports
or
The "I'm only responsible for doing the task!" syndrome

The next critical step in task management has to do with the what the doer does in self monitoring, replanning and status reporting. Notice the emphasis on the doer. To be successful with any significant project... one must have people who do task self management. The project team must all be capable of and believe in managing themselves, their tasks, and communicating to others their status. This is in contrast to the projects where only the project manager sees himself/herself as the one "responsible" for monitoring, replanning and status reporting!

To manage a task (or project) well requires frequent assessment of where the task is against the plan and specifications. This assessment is used to adjust the plan if required. Also, most importantly, the information from the monitoring and replan needs to be communicated. This communication of the updated plan allows for actions to be initiated as required to accomplish the projects goals.

How often should this monitoring, replanning and communicating be done. The guideline logic that should usually apply is as follows.

Re-evaluate, re-plan and report when:

a) 10%, 50%, 75% and 100% of the "last plan" labor hours or dollars have been expended

or

b) 10%, %50, 75% and 100% of the "last plan" elapsed time has occurred

or

c) Any factor arises which will cause quality of the deliverable to meet less than the excellent criteria.

or

d) Anything has occurred which will cause the delivery date or hours/cost to change by more than 10% of the original.

or

e) The task is completed and deliverable is delivered.

If this schedule is followed then there should be few big surprises! Tasks do not get behind suddenly at 95% complete or just when the task is to be done. A motivated competent task doer should be able to see problems coming much sooner. This schedule requires the doer to stop at key points to assess, replan and report. With this schedule you can quickly spot someone incapable or unwilling to do this. Or you can more easily see a person who is continually providing inaccurate plans and replans. Each person with such a problem needs to be handled differently. The key is that you must have people who can and will do this monitoring, replanning and communicating. Otherwise, successful projects are highly unlikely.

See Appendix B for a form you can adapt to use as the task plan/re-plan document.

I've been focusing mostly on the doer of the task and their responsibilities. This is an over emphasis for effect. The project manager must have critical involvement in all the processes described so far. Further, the next step outlined must be done by the project manager or the method will not work.

Supervisory Quality, Time and Cost Review and Feedback
or
"The manager's squeeky wheel gets attention"

An ongoing Project Manager/Supervisory review is essential to quality task management. The project manager makes or breaks the task management system. He/she does this by reinforcing the correct process by review and providing timely feedback on the plans, status reports and software quality forms. This will keep them coming and make the process a serious part of what the doers take as their responsibility.

Additionally, the project manager is reinforced because they will get early warning of impending problems. This warning should give them enough time find a solution or at least moderate the impact on the project.

This review must be done all through the tasks existence and also most critically at the end of the task. A review should be held when the task is "done." This is where everyone involved should be able to assess the different successes and areas for improvement. On any project of significant complexity and duration, this final review process should do a lot to improve performance over the

project. Without an ongoing review, a project just seems to become hotter, harder and tighter as time goes on!

Keeping Commitments
or
"Don't say you will ....unless you will!"

All of the above techniques don't mean much if the people involved don't keep their commitments. In fact if you have people who are superb at keeping commitments...then some of the fundamentals can be de-emphasized. My experience is that most people are less than perfect at keeping commitments. The techniques outlined helps them to focus on the importance of accomplishing every task on time, cost and quality. And it gives them the means to improve their good intentions.

In spite of everything, people will not always meet commitments. The key is how you handle this. If they perceive that it is not important to you that they missed a commitment then watch out. You've just said it's ok to not keep the next commitment. You must of course balance this with wisdom that say's no one, including yourself, always keeps their commitments. So it's a fine line a project manager must walk. Nevertheless, the bottomline attitude of your project team should be ....never make a commitment that you cannot or will not keep!

Summary Conclusions

As I re-read this paper, it strikes me that I've made the process sound like a simple by the numbers process. It is anything but this...in my experience. All management has to do with people. And no two people are the same! Somehow we must put people first while getting our projects done. At the same time management means stretching people to do things that may not come naturally. Otherwise, it's not management but more like being an bookkeeper looking only backwards.

Task Management is really nothing more than Project Management of a "sub project." Tasks are really just projects with smaller scopes, shorter time-frames, smaller costs and less resources. Project management is done by a partitioning of the project into smaller and smaller pieces and applying the fundamentals of task management to them. And, if one has project members who keep commitments and manage their tasks well then lo and behold the project will be much easier to manage.

Finally, software project tasks are complex. Managing and doing them successfully is not easy. However, with the right approach and attitudes our success in such projects can improve significantly.

The Author

Rob Mattson is currently Manager of Information Systems at WIDCO, a large open pit coal mine in the state of Washington. He has been working in the systems field for 14 plus years, the last 11 in the HP3000 environment. His background in addition to many systems related roles includes a degree in Psychology, work as a Certified Public Accountant and as a management consultant/trainer. His current professional areas of interest include general systems theory, project management and exploring the concept of system quality. When not immersed in systems he enjoys his family, sailing, sea kayaking and golf.

Paper Presented Originally at HP3000 International User's Group Meeting - San Francisco, Sept. 1989.

Software Quality Form

System Name: _____ Est. Time: _____
Program Name: _____ Actual Time: _____
Date: / / Assigned To: Reviewed By:

Quality Parameters	Quality Level					Rating	
	Unacceptable	OK	Value	Excellent	Value	Doer	Rev.
Functional Specifications							
Suitability Job Effectiveness							
Speed Responsiveness							
Resource Impact Overhead							
Robustness Forgivingness							
Adaptability Flexibility							
User Acceptance Satisfaction							
Business Cost Effectiveness							
User Independence Support Required							
User Documentation							
Ease of Learning							
Ease of Use User Efficiency							
Implementation Installation							
User Interface Uniformity							
Development Task Management							
Cost to Develop							
Time to Develop							
Test Plan Testing							
Technical Review Walkthrus							
Defects "Bugs"							
Maintenance Time & Cost							
Maintainability Int. Documentation							
Adherence to Standards							
Integration							
					TOTALS		
					% of Excellent		

Comments:

Appendix A

The Goal is Excellent Systems

TASK MANAGEMENT FORM #1

As of: Last Report: By: Status:

Task ID: Project ID: Customer: Task Resp:

Task Description:

Task Deliverable:

Quality Form Reviewed & Accepted By: Subtask Planning Done (Y or N)?

Prerequisites:

		Commitments			Proposed or Actual Chg From:						
					Original			Revised			Next Update Due:
		Original	Revised	Proposed	Amt	%	$	Amt	%	$	
Labor Hours	P										
	A	Pre:	Cur:	Tot:							
Start Date	P										
	A										
Working Days	P										
	A	Pre:	Cur:	Tot:							
Completion Date	P										
	A										
_____	P										
	A	P:	C:	T:							

Quality will be excellent in all parameters except:

Issues I need to resolve to keep these commitments:

Sign offs--> Plan: Quality: Completion: Task Mgmt Rating (1-10):

R.Mattson 1989

Automating Production Support Tools: Problem/Incident Report Tracking

By Kevin Darling
The Gap, Inc.
Eastern Distribution Center
3434 Mineola Pike
Erlanger, Kentucky 41018
(606) 283-1100

Introduction

Dealing with problems that arise in the everyday operation of systems can be very taxing on personnel. In small shops, the support is usually provided by operators or by the programming staff directly. In some cases, that is all that is needed. But, in my experience, as the business grows and the quantity of software to be supported increases, there comes a point where there is a need to track the problems that occur with the systems. Initially, this is typically handled manually; but, the job of tracking and then reporting the events becomes unmanageable and overwhelming. This usually is followed by the demise of the manual tracking that has been occurring.

Companies experiencing significant growth have begun addressing the problem by developing a staff whose specific function is to support problems that the users report. Now with a staff, problems can be handled in a more controlled fashion. But, this does not address the problem that still exists with a manual Incident Report or IR Tracking system. Enter automated IR systems. These systems can be simple data entry and reporting functions or extravagant systems for data entry, reporting, and cross-reference functionality much like the one used at HP's Response Centers.

Our experience has been very true to this form. We began with a support structure that included development as the front line support staff for the systems. As we grew, we identified key people in our operations and development organizations to become a Production Support staff. Forms were used to keep track of problems that were reported; but, the rich information that these forms contained was rarely being used to track reoccurring problems that needed to be addressed.

With this growing problem, we began to develop a better understanding of the IR tracking process. Formal manual procedures were put into place. But, these procedures were limited in their effectiveness. As a systems developer and then manager of the Computer Operations and Production Support staff, I realized the need for an automated system to better handle the environment in which we worked. Many of you are probably going through the same dilemmas presently or will be in the near future. What will follow here is a discussion of our implementation starting from the manual system definition to the development and use of the automated system.

Defining A Manual System

Setting up a manual system for IR tracking is not something that should be taken lightly. It has to be simple and direct enough to deal with the everyday situations as well as most of the exceptions. From our standpoint, it had to be sound enough to base an automated system on.

First, there has to be a definition of the information that needs to be available for resolution of the problem. Such information includes:

* Date and time reported
* System and type of failure from a predefined list
* Person reporting the problem
* List of people contacted to get problem resolved
* Description of the problem
* Description of the final resolution
* Follow-up information

With this information, the process involved with the documentation through resolution needs to be put into place so all parties follow the same guidelines for resolving the problems. Procedures for such a manual tracking system include:

1. IR's are documented by Computer Operators and/or Production Support staff. The problems can be reported by Users, Computer Operators, or Production Support personnel. Documentation of the problem includes completing an IR form with all the required information available at that time.

2. The IR is then logged into an Incident Report Log Binder. This information is simply the IR number assigned to the form, the date reported, and a one-line description of the problem.

This binder provides a tracking of all IR's and is updated to indicate the status.

3. The first available Production Support staff member takes the most urgent IR from the "Unresolved" bin. By taking the IR, they have assumed responsibility for the resolution of the IR and they note their name and the date and time on the form. Evaluating the "urgency" of an IR is a learned skill and is a major consideration in the implementation of the manual and future automated system. Some questions to consider while determining the "urgency" include:

* Is this IR impeding the processing in the business?
  - On what scale?
  - Does it leave workers without work?
  - Is the impact now or will it be felt later?

* Is there the interest of successfully higher levels of management in particular IRs that imply a level of "urgency"?

4. The person assuming the responsibility for an IR should contact the originator to inform them of the status as well as to gather more information to resolve the problem should that be necessary. This ultimately also provides the originator with some assurance that action is being taken to resolve the problem. Should the person working on the problem find that they are not able to resolve the problem, the escalation procedures should be followed to insure a successful handoff.

5. In many situations and for many reasons, IRs need to be escalated to other staff members or third party support teams for resolution. In our shop, the escalation sequence is from the Production Support staff member to the Production Support Supervisor. From this point the supervisor either resolves the problem or escalates the IR to a System Expert. System experts are Senior Development personnel that have been assigned the support responsibility for a system or part of a system. Hardware or system software escalations are to the Senior System Management personnel. Each System Expert is backed up by another senior staff member. Should support be needed by an third party vendor, the Production Support staff will notify the vendor and follow through as the contact to insure satisfactory resolution of the problem. Copies of the IR and supporting information should be made and kept with the IR Log before distributing to the person resolving the IR.

6. At the time of resolution, the IR should be categorized and the resolution description should be completed.

7. The IR Log should then be updated to indicate that the IR has been resolved, the date and time should be noted, and the time spent should be logged.

8. Final review of the problem should take place. Any problem that seems to be reoccurring should be reported to the appropriate team responsible for the system. Enhancements may be made to provide a tool or to modify the system to correct the problem.

9. The IR should then be filed by filing category.

One of the key factors to the manual system is the possible need for escalation. This implies that some sort of handoff needs to take place. It is critical to make sure the handoff is not one-sided, meaning that the person accepting the IR should be given notification, usually in person or by electronic mail, that the problem is now in their hands. Confirmation that it has been accepted is critical since any "urgent" IRs left unattended only hurt the image of the MIS Department. Any IR that is escalated also should be noted in the IR Log. The escalation carries the responsibility for resolution with it; but, it is critical for the Production Support staff to follow the escalated IRs to resolution and act as the focal point for all communications in most cases between the person resolving the IR and the originator.

Also included in the procedures is the review of the IRs upon completion to "close" the IR. This "closing" of the IR allows one final review by the Production Support staff of the IR and identification of reoccurring problems that should have some further level of resolution that provides better tools or even a correction to the system that has the problem. Any procedures for resolutions that are being documented should be reviewed by the System Experts for correctness.

With the logging of the IRs, the Production Support staff has available a good deal of information concerning open IRs as well as some documentation by system or event that can be reported. The time to develop some of the reports should not be under-estimated and this will be one of the most difficult features to be included in the manual implementation. Steps need to be taken to insure that basic reporting of problems that have occurred during a shift be completed and issued to the key people in support, development, and technical services so monitoring of the system can occur. Further detailed reporting is beneficial; but, it should not be over-killed in such a way that precious time available for support be reduced drastically.

Enter the automated IR Tracking System.

Automating the IR Tracking System

Now that the manual system has been defined and hopefully has had a majority of the "bugs" or exceptional situations defined, work on the automation of the system begins. The automated system brings with it new levels in reporting and tracking. But, other key features that can be included are the use of jobs or programmatic interface to HPDesk or other electronic mail systems and finally the introduction of a security matrix that allows the definition of the capabilities for people using the system.

First, comes the basic definition of the system. The system needs to have the ability to store the data in a form that is easily extracted and reported. Some form of data entry methodology should be used to create and input the data and then report generation programs need to be in place to give management the ability to review and track the progress and problem histories.

Databases

The IR database is the backbone of the system. It contains three master data sets with five detail data sets linked to them in various combinations. The use of the database allows us to concentrate the data into a common location and provide the necessary links between the data to provide more manageable access for data entry, reporting, and cross-referencing. In figure 1, the data base is presented with the links diagrammed.

The specific data sets and their purpose include:

Data Set Name	Description
IRs(M)	This data set is the master set used to store key information about the IR. Included are the IR number, information as to the system and type of failure, priority, and dates and times of creation and resolution.
Keywords(M)	This data set is a master set that contains the keywords that are valid for the cross-referencing of IR's.
Contact(M)	This data set is a master set used to store a list of valid contacts. Also included is the security, capability, and a flag to determine if the contact can receive electronic mail.

IR System Database

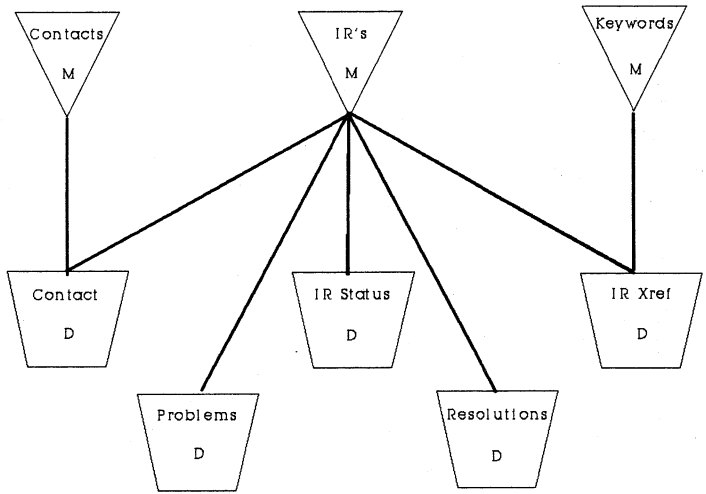


Figure 1

Data Set Name	Description
Problems(D)	This is a detail data set that contains the description of the problem. Each line of text is a separate entry in the data set. Each line of the text is numbered with the number serving as a sort key to insure the text is displayed in the correct order.
Resolutions(D)	This is a detail data set that contains the description of the resolution. Each line of text is a separate entry in the data set. Each line of the text is numbered with the number serving as a sort key to insure the text is displayed in the correct order.
Contacts(D)	This is a detail data set that contains an entry for each contact that is documented to have been involved with the problem and the time that they were notified. The time notified is a sort key and so a reverse chain read gives the most current contact.
IR Status(D)	The is a detail data set the contains the statuses an IR has been through. The various statuses are sorted so that a reverse chain read gets the correct status.
IR Xref(D)	This is a detail data set that contains the IR number and the key-word that is associated to it. There could be more than one entry for an IR since keywords come from the system, type of failure, and key-word entries. Key-words must already exist in the database.

In support of the IR database, there is a utility database that has the definitions for the next IR number, file equations for the reports and data files used, and file build definitions.

IR Tracking System Menu

The critical piece that brings the data to the database structure is the menu software. Here the operators or support staff enter the information concerning a problem so it can be tracked and resolved. The menu consists of the following options:

* Create IR
* Modify IR
* Resolve IR
* Close IR

* Update Contact
* Maintain Keywords
* Reprint IR
* IR Report Request
* Void IR
* Inquire on IR
* IR Index/Cross-reference

Each of these options plays a critical role in providing an automated tracking system. Basic functions are available to create or enter the initial IR information and then options to modify the data should corrections or updated be necessary. In cases where the IR is in a state of disarray, an option is available to void the IR so the it can be reentered.

At times, there is a need to reassign the IR to another person for resolution. To do this, the contact is updated in the entry screen for contacts. As resolution or comments are available, the data can be entered via the resolution entry screen. Upon final review of a resolved IR, if there is no other action required, it can be statused as closed. Due to the nature of the option, it is restricted to a specific user class.

The final options include inquiry of IR detail and IR index and cross-reference facilities. The inquiry function takes an IR number and retrieves all the information available. The information is then displayed to the screen for review. The other option allows indexing or cross-referencing tools for an IR. Here people can use the research tool to review other IR's that are associated with a similar key-word, system, or failure type. This is particularly useful in determining whether there have been any other occurrences of similar problems. With the information in hand, the person resolving the IR can be sure that the procedures that need to be followed are accurate.

Report Generation

Now that we have data in the system and a means of maintaining that data, there needs to be a mechanism to report on the IR's. Some of the reports that need to be available include the document that actually represents the IR. This is simply a document that reports the data that has been entered so the person responsible can understand what the problem is that needs corrective action. The generation of this report is done by a report request processor that is constantly reviewing an input message file for generation requests and processes them when it is available. Also included is a function to reprint an IR in cases when a replacement copy is needed or a copy of the resolution would help solve an IR that is currently open.

Other reports that are useful in the management of the automated system include detailed reports that show all open IR's and optionally include information as to IR's that are closed during the defined time period. These reports specifically can be reported by the contact that currently has the IR and is working on it or by the IR number alone. Reporting IR's by the person who is currently taking responsibility for the resolution can help identify places where there might be bottle-necks and therefore delays in the final resolution. It can also indicate the volumes of IR's that are being handled by each person. The type of data that is maintained in the system also allows for a variety of report possibilities based on the needs of the management. These ad hoc reports can be generated using one of several report generators.

In comparing the manual system to the automated system, one can see that the basic functionality in place in the manual system has been automated and enhanced to allow better use of the computer system resources available. The ability to generate reports easily, control accessing and updating of IR information, and the cross-referencing facilities make the job of tracking the IR's a very easy task. Also, the automated system allows us to research problems for resolutions easily where the manual system required a great deal of manual file searching.

Conclusions

Implementing any form of IR system is not an easy task to undertake. Concern for the manual logistics and then the automated logistics are important. You, the people responsible for supporting the systems or the even the people who manage the support function, need to determine the need for your environment. Not every shop would benefit from an elaborate system; but, there could be some middle-ground that could include some manual and some automated portions.

If there are some concerns as to whether an IR tracking system is needed, ask yourself several questions:

- Are problems to be resolved stacking up?
- Are problems being lost in the paper shuffle causing some delay in getting resolutions?
- Are the functions of support generally out of control?
- Are problems being blatantly thrown away to hide problems?

Answering **yes** to any of these questions may mean that some sort of tracking might help improve situations for getting the job done and could also show the user community there is a resurgence of excellence in the support arena.

Manual or automated tracking systems are only a start to success in the support arena. It also takes some dedication of staff and training for those staff members in support so that they can adequately do the job. Take time to think about your organization. How does support fit in? Who should be really doing the support in your organization? You will find that development of a staff for support and then the solicitation of the staff for ideas and needs for IR tracking can be very beneficial. Including the people involved can make the whole process successful.

Acknowledgements

I would like to thank the senior management and support staff's for working closely in the definition and development of our manual and automated systems. Their input made the process relatively painless and they provided a sound knowledge base for resolving issues that arose during the design and development phases.

THE EVALUATION OF THE EFFECTIVENESS OF INFORMATION SYSTEMS

AUTHOR: PETER W COWIE SENIOR MANAGER MANAGEMENT SERVICES

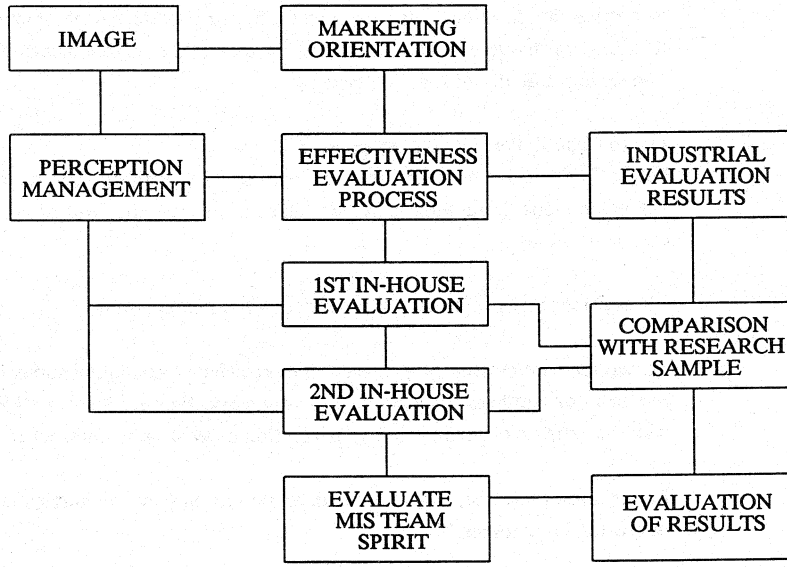
COMPANY: ALUSAF (PTY) LIMITED

BUSINESS: PRIMARY ALUMINIUM PRODUCER

ADDRESS: P.O. BOX 284
EMPANGENI
3880
SOUTH AFRICA

FAX No: 0351-5169
TELEPHONE: +27-351-51111

1. OVERVIEW OF THE PAPER



THIS PAPER SHOULD BE READ IN CONJUNCTION WITH PAPER NO 6707 "THE DEVELOPMENT OF A MARKETING ORIENTATED BUSINESS STRATEGY FOR THE CORPORATE INFORMATION SYSTEM".

Paper Number 6705

1. MARKETING ORIENTATION FOR THE INFORMATIONS DEPARTMENT

PURPOSE - to find out how the customer perceives the quality of the various components of the MIS service.

2. PERCEPTION MANAGEMENT

2.1 Introduction

Only if we measure something can we control it effectively or, if you like, manage it efficiently.

Until now business managers have been able to control all functions of their businesses - except marketing, because these other functions can be measured. The orphan in the mix of business disciplines has always been marketing for the reason that no one has found an effective way of measuring its functions such as advertising, corporate image, public relations, promotions, the sales operation and other related activities. The same is true for information systems.

2.2 The concept of 'Perception Management'

Perception Management has been defined as: 'The efficient control of key perceptions through measurement and the cost-effective direction of communications'.

All decisions that people take are founded in their perceptions.

Perceptions are created only through the receiving of communications in one form or another. The absence of communication has its own effect with the extreme opposite being no perception or awareness whatsoever.

Perceptions, therefore, can be created, or shifted only by communication of one kind or another.

Consequently, if we identify and measure perceptions, we are taking our first step towards creating or shifting them in the certain knowledge that all perception creations and shifts will have to be achieved by communication.

2.3. **The Practice of 'Perception Management'**

Buying decisions are made by people and these decisions are determined by their relative conditions of awareness and their perceptions of products and/or services and the origin of these products and/or services. Once a state of awareness is created perceptions begin to develop.

2.4. **Key Community Groups (KCG's)**

Every organisation has, inside and surrounding it, groups of people whose decisions (and whose perceptions) will determine its future.
The first task is, therefore, to identify those vital groups of people.

2.5 **Introducing Measurement**

Having identified our KCG's within the organisation itself - some in the distribution network and certainly a number in the market place - we commence our measurement process by measuring the current perceptions of our KCG's.

Measurement takes place in two forms and along two dimensions. The dimensions are a rating scale and a set of established criteria.

2.6 **From Perception Measurement to Communication Management**

What is Communication?

Having measured perceptions we have our input of information to the communication process. The other side of the coin is the message or messages we wish to convey to our KCG's.
Communication is the sending out to KCG's of messages designed to specifically and measurably shift perceptions. Outward communication has only one purpose. This is to create awareness or to shift perceptions within one or more KCG's.

Messages and the Five Communication Sets

Perception Management recognises there are five sets of communications from which any, and all, messages derive.

- Corporate awareness and perception.
- Product (including service) awareness and perceptions.
- Response communications.
- Channel communications.
- Customer contact communications.

The key to cost-effectiveness and efficiency in communication is to determine the pertinence of the five sets, their sequence of communication and the emphasis (i.e. allocation of resources) that needs to be applied to each particular set.

It is important to recognise that misjudgement with regard to the five sets is the second most common area of wastage of company communication budgets.

2.7 **Determining the Messages and Developing a Communications Strategy**

The basic issues here are:

- Why are we planning to communicate?
  (Answer: to achieve a measured shift in perceptions).
- With whom are we planning to communicate?
  (Answer: our KCG's, as prioritised).
- What are we to communicate that will shift perceptions?
  (Answer: the messages we identified with our perception measurement projects).
- How will we structure those messages in terms of 'The Five Communication Sets'?
  (Answer: the analysis of our 'Communication Strategy Builder')

Knowing all of these factors leads us to the task of how we will convey the message for the most cost-effective results.

The effect of the communication strategy developed is measurable in only one way. This is by repeats of the original perception measurement projects to measure shifts achieved.

In bringing real measurement to the purpose of our communication endeavours and expenditure we achieve control. Control gives management the ability to manage effectively and to achieve a measurable return on the communications investment.

3. THE EFFECTIVENESS EVALUATION PROCESS

3.1 Introduction

The nature of computer-based information systems in organisations has evolved through several stages over the last four decades. Sprague and Watson name these stages as Basic DP, Integrated DP, Management Information Systems and Decision Support Systems. More recently Arkush observed that the benefits and management objectives of these areas have changed - from clerical efficiency to control costs, to operational efficiency and effectiveness to control process, to management effectiveness and transformation to manage the business better.

This growing importance of IS and its extension throughout the organization has focussed attention on the measurement of IS effectiveness, so much so that this topic has been identified as one of the top ten issues for IS management in the 80's.

Since 1983 a series of studies have been conducted at the Graduate School of Business, Cape Town, South Africa to develop and apply an instrument for measuring IS effectiveness.

3.2 **The Basis of the Evaluation Technique**

A questionnaire was developed by the Cape Town Business School which was an extention of the instruments developed by Ives, Olsen, Baroudi, Alloway and Quillard.

The evaluation is performed by asking each respondent to twice rate (on a scale of 1-7) the full set of questions based on:

1. Importance - How important do you feel that each attribute is in ensuring that the overall systems will be effective.
2. Performance - rate the degree of performance (poor - excellent) for each attribute.

The questionnaire consisted of 37 questions which when logically grouped, recorded perceived performance in 6 major dimensions of MIS activity:

- type 1 information work
- type 2 information work
- MIS staff characteristics
- MIS strategic issues
- User participation
- MIS responsiveness to changing user needs

We modified the questionnaire (See Annexure 1) by increasing the number of questions and by defining the dimensions as follows:

- strategic issues
- technical and application systems
- management reporting
- system efficiencies
- user participation
- responsiveness to changing user needs
- MIS staff characteristics
- decision support systems

3.3 **Reliability and Validity of the Instrument**

Reliability

An analysis of variances for the initial survey were conducted in order to assess the potential influence of **errors of measurement.** For the 37 individual items a reliability coefficient of 0.94 was obtained and for the instrument as a whole, the coefficient was 0.88. These values indicate a high reliability with respect to measurement error.

Validity

The **content validity** of the instrument, i.e. its representativeness or the sampling adequacy of the content, is assumed to be high based on the great care that went into the construction of the questionnaires.

3.4 **Measuring Information Systems Effectiveness in Different Business Sectors**

Altogether 238 firms were selected for the exercise as being the largest in terms of turnover and/or IS activity. Eventually 102 firms participated in the survey and finally 794 usable responses from 83 firms were obtained.

Grouping of Respondent Companies

One of the important objectives of the study was to compare firms performing poorly and well.

GROUPING OF FIRMS BY CLUSTER ANALYSIS
COMPOSITE SUCCESS MEASURES (SCALE 1-7)

SUCCESS GROUP	MANUFACTURING	RETAILING	FINANCIAL SERVICES
1	5.5, 5.2, 5.2 5.1, 5.1, 5.0	5.3, 5.1, 5.0 4.9, 4.9	5.1, 5.0, 4.9 4.8, 4.8 4.9
2	4.9, 4.9, 4.8 4.8, 4.8, 4.7 4.7, 4.7, 4.6 4.6, 4.6, 4.6 4.5	4.6, 4.6, 4.6	4.7, 4.7, 4.7 4.4, 4.3
3	4.8, 4.7, 4.5 4.5, 4.4, 4.4 4.3, 4.2, 4.2 4.2, 4.0	4.6, 4.6, 4.3 4.3, 4.3, 4.3 4.3	4.6, 4.3, 4.3 4.3, 4.3, 4.0
4	4.6, 4.5, 4.5 4.4, 4.3, 4.3 4.3, 4.2, 4.2 4.0, 4.0	4.3, 4.3, 4.3 3.8, 3.8	4.0, 4.0, 4.0 3.8, 3.4

Measure of Performance

The following schedule displays the performance measures for the 83 firms as follows:

- the overall performance for all 794 respondents
- overall performance for all respondents and sector by sector broken down in terms of the 6 dimensions
- overall performance broken down by success groups in each business sector.

MEASURES OF PERFORMANCE
(SCALE 1-7)

SECTOR	RESPONSES	OVERALL RATINGS	FACTORS RATINGS 1 TYPE 1 INFO WORK	2 IS STAFF	3 IS STRATEGY ISSUES	4 USER PARTICIPATION	5 IS RESPONSE TO CHANGE	6 TYPE 2 INFO WORK
COMBINED	794	4.57	4.88	4.76	4.54	4.32	4.17	3.90
MANUFACTURING	361	4.69	5.03	4.89	4.59	4.31	4.31	4.00
RETAILING	157	4.52	4.72	4.80	4.53	4.35	4.18	3.76
FINANCIAL	276	4.44	4.78	4.58	4.47	4.16	3.98	3.84
MANUFACTURING								
GROUP 1	59	5.22	5.60	5.26	5.05	5.06	4.97	4.57
GROUP 2	119	4.82	5.17	5.03	4.63	4.49	4.53	4.20
GROUP 3	79	4.47	4.85	4.68	4.25	4.30	4.15	3.72
GROUP 4	91	4.37	4.67	4.64	4.56	4.06	3.73	3.58
RETAILING								
GROUP 1	36	5.03	5.24	5.30	4.95	4.90	4.73	4.35
GROUP 2	31	4.62	4.85	4.92	4.38	4.54	4.37	4.03
GROUP 3	42	4.45	4.69	4.69	4.58	4.32	3.83	3.82
GROUP 4	47	4.11	4.25	4.42	4.22	3.81	3.95	3.01
FINANCIAL								
GROUP 1	71	4.95	5.35	5.03	4.94	4.67	4.48	4.35
GROUP 2	63	4.60	4.84	4.63	4.63	4.35	4.31	4.25
GROUP 3	72	4.34	4.86	4.40	4.28	4.01	3.80	3.58
GROUP 4	70	3.89	4.08	4.25	4.03	3.63	3.34	3.22

The fact that Type I Work and IS Staff rate most highly is consistent with the fact that the great proportion of IS activity and staff training has always been in the Type I area and still is. For the future, however, there is little doubt that the emphasis will swing to Type 2 Information Work and an associated emphasis on User Participation and IS Responsiveness to Changing Needs, the three factors that rate most poorly.

Regarding the success groups, the factor-to-factor tendencies referred to above are also evident, with Type I Work and IS Staff being rated most highly by all 12 success groups and IS Responsiveness and Type 2 Work lowest. Also the most successful groups are best over all six factors and the least successful, worst.

3.5 **Influences on Information System Effectiveness**

The next phase after measurement, is to understand what influences effectiveness.

Importance Versus Performance

One of the prime objectives of the Alloway and Quillard instrument was to explore the relationship between the importance attached to IS and perceived performance. Analysis of responses to both importance and performance scales in the effectiveness questionnaire showed an inverse relationship and it was suggested that a major reason for ineffective IS was lack of emphasis on those aspects of IS deemed most important.

	CORRELATION COEFFICIENTS		
SUCCESS GROUP	MANUFACTURING	RETAILING	FINANCIAL SERVICES
1	.65	.67	.52
2	.65	.49	.30
3	.55	.44	.31
4	.50	.30	.05

It appears therefore that success in IS is associated with a focus on items regarded as most important. Conversely a possible reason for low success in IS may lie in management trying to 'cover all bases' and achieving limited success as a result.

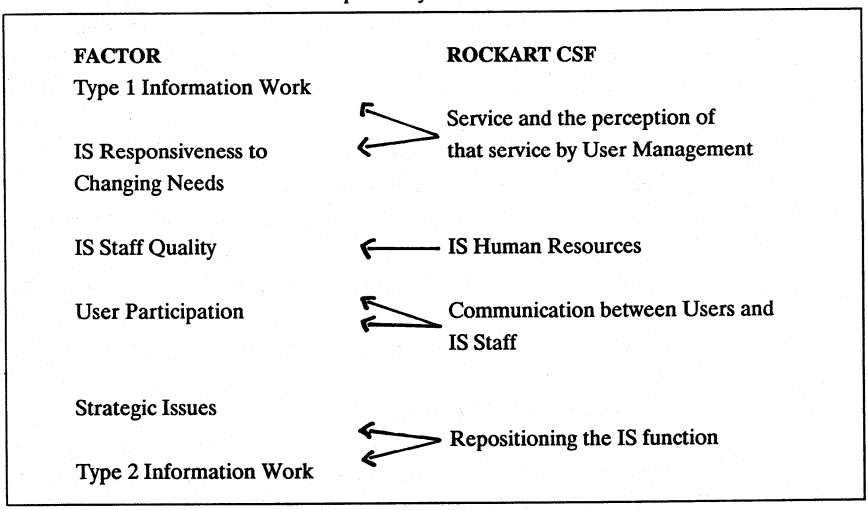
Other Influences on Success

Emanuel's study identified some 177 items considered in the literature to influence success. Through careful interpretation 26 recurring items in three categories were identified - procedural, structural and behavioural. 20 Senior IS practitioners and academics were asked to weight and rank these in importance. 13 Items were isolated in this way as being considered the most important influences on success.

Refer to the annexures for details of these critical success factors.

3.6 **Conclusions**

This instrument can provide the basis for the development of an information's strategy by applying the four so-called Critical Success Factors for IS development by ROCKART.



4. IN-HOUSE EVALUATION OF EFFECTIVENESS

4.1 **Selection of the Respondents**

We divided our sample into 3 main groups:

Management	20
Users	20
Computer Staff	10
Respondents	**50**

The management group included the Managing Director down to Plant Manager level.

The users included the leadergroup of all the applications systems.

The computer staff included 10 out of a total complement of 14.

4.2 **The First Evaluation - June 1987**

The MIS department were not keen for the evaluation to take place for the following reasons:

- we would open ourselves to attack on all flanks
- users do not know enough about MIS to make any useful ratings
- perceptions are not measurable and therefore cannot be trusted

Important Observations

The overall actual and importance ratings of the management and user group were similar but with different internal rankings. (i.e. ranking of specific questions differed)

The MIS actual ratings were 30% higher than the customer group.

Application systems rated the highest with user participation, responsiveness to changing user needs, and Decision Support Systems rated the lowest.

Important MIS Responses to the Evaluation

Recognition of the Customer/Service principle

Acceptance of the perceived messages from a diverse group of 'customers'.

Acceptance of the dynamic role that MIS could play in the corporate dimension.

An enthusiastic action plan was developed to improve the rating to a '5 star' rating.

4.3 **The Second Evaluation - June 1988**

Achieved the '5 star' rating and moved up into the top 15% of the industrial sector. (The significance of the '5 Star' is that our company is rated as a '5 Star' Company by the National Safety Association)

The overall **IMPORTANCE** factor only increased by 4% compared with an overall **ACTUAL** improvement of 12%.

The most significant effectiveness improvements were achieved in the areas of:

Decision Support Models	*
MIS Staff Characteristics	
User Participation	*
Responsiveness to User Needs	*

* The three lowest ratings from the previous year

The MIS department were anxious for the results to become available to see how successful they had been.

Management's perception of the MIS achievements were very positive.

5. BUSINESS SCHOOL EVALUATION

5.1 **Statistical Analysis**

A copy of the in-house detailed results for the two years is attached to this paper for your information.

An overall statistical correlation was completed by the Graduate School as part of their research program and the following observations were made:

The assertion is that optimum effectiveness is achieved when the focus is on those areas considered important and in fact when the achievements are also in these important areas. The perfect degree of 'fit' between importance and actual is therefore 1.0.

The results of the 'degree of fit' analysis are summarised as follows:

MIS RATINGS

	1987	1988	%
Importance : Actual	0.30	0.65	116

Shows an improved achievement in those areas considered important.

MIS AND CUSTOMERS RATINGS

	1987	1988	%
Importance	0.57	0.74	30
Actual	0.39	0.52	33

Shows that MIS and the customers are beginning to agree more on the important "corporate" issues rather than their "own" issues.

5.2 Inter-Company Comparison

Alusaf's order of ranking moved from No 10 to No 2 in the research sample of 12 major companies.

5.3 Summary of the research report

"The two surveys show that there has been a substantial improvement in user perceptions of IS over the intervening period. This has been associated with an increased alignment in the views of the MIS staff regarding importance and performance of various elements of MIS. This can be seen in light of both technical and behavioural shifts over the year. At the time of the survey there was considerable focus on the technical problems of implementation of individual packaged systems. The senior system staff acted relatively in isolation and was struggling to get these systems up and running. There was an emphasis on the DP world and no time for marketing to the users. There was also not that much communication between the systems staff for the same reasons. As a result of the initial survey the focus was shifted towards marketing and through the discussion of the survey results, common ground was identified. There was also a restructuring in the MIS group with the senior systems person being made the systems controller and taking more of a management role than before.

The increased alignment within the MIS and between MIS and the users (as indicated by improvement in the various correlation coefficients) appears to have come about in part through the improved communications throughout Alusaf as a whole, in part through the changed reporting line of the MIS group through the Senior Manager - Management Services to other divisions and in part through a conscious attempt to instill a greater marketing thrust among the MIS people. They have been encouraged to entertain their users and find common ground. Previously DP reported to the General Manager Finance organisationally but informally reported to the Technical Director - in a sense the group had no home and operated as a closed system and removed from the user.

The feature of the MIS department is that the support analysts continued to be responsible individually for each of the four major systems and are expected to do everything in relation to the systems from technical support to marketing and general contact with the users. Alusaf is in the fortunate position that there is no turnover in MIS staff, leading to a stable situation. The MIS staff obtain technical training as well as training in topics such as group management and a need is seen for training in business skills as well. As the MIS group has come closer to the users it can be noted that user participation has in the past been the greatest problem and the MIS staff have had to work very hard to get the human interface right and earn acceptance from the users.

The shift into the PC world has been a major contributor to improved communications between MIS and the users. A very successful subsidised private PC purchase scheme introduced 120 PC's into the homes of the user.

As things stand the Managing Director of Alusaf appears to be very enthusiastic about the MIS effort, and he attributes their success to a very business like approach".

5.4 Evaluation of MIS team spirit

Further discussions took place and it was concluded that there is a crucial factor promoting the success of IS. If it is true that a close association between importance and performance, as perceived by the MIS groups, leads to a high level of user satisfaction then the question becomes "What is the difference between MIS groups showing high and low importance/performance correlations?"

The research was then directed towards the development of a questionnaire to measure cohesiveness of the MIS group.

It was found that the Alusaf MIS group appears to be very cohesive whereas one of the other firms whose results shows very poor perception of MIS, is just the opposite.

6. CONCLUSIONS

- To change perceptions we need to do something positive.
- The evaluation technique provides us with a means of measuring perceptions.
- The introduction of the customer orientation philosophy will have significant impact on attitudes.
- MIS has achieved a high profile status in the organisation.
- A basic success requirement appears to be a cohesive MIS team.
- Success is associated with a focus on items considered important.
- The results of the evaluation can provide the basis for the development of an information strategy.

ACKNOWLEDGEMENTS

1. J Miller - Graduate School of Business, University of Cape Town.
   'Measuring Information Systems Effectiveness'.
2. R M Alloway and J A Quillard 1982. "Top Priorities for the Information Systems Function", CISR Working Paper No 79, Massachusetts Institute of Technology.
3. J E Bailey and S W Pearson 1983. "Developing a Tool for Measuring and Analysing Computer User Satisfaction", **Management Science** Vol 29 No 5.
4. M K Grallert and G H Russell 1985. "An Investigation into the Effectiveness of Computer-Based Information Systems in the Manufacturing Sector in South Africa", MBA Technical Report, Graduate School of Business, University of Cape Town.
5. B Ives, M H Olson and J J Baroudi 1983. "The Measurement of User Information Satisfaction", **Communications of the ACM** Vol 26 No 10.
6. J Miller and B A Doyle 1987. "Measuring the Effectiveness of Computer-Based Information Systems in the Financial Services Sector", **MIS Quarterly,** Vol 11 No 1 March 1987.
7. J F Rockart 1982. "The Changing Role of the Information Systems Executive: A Critical Success Factors Perspective". CISR Working Paper No 85 (Massachusetts Institute of Technology, Cambridge Mass.)

EVALUATION OF THE INFORMATION SYSTEMS EFFECTIVENESS

1. NUMBER OF PARTICIPANTS

	1987	1988
Management	19	17
Users	21	21
Computer Staff	10	9
	50	47

2. EVALUATION OF THE IMPORTANCE RANKING FOR EACH DIMENSION

	Overall	Manage-ment	Users	Computer Staff
Technology & Operations	1	1	1	1
MIS Staff Characteristics	2	3	2	2
Management Reporting	3	2	3	3
User Participation	4	4	4	5
System Efficiency	5	5	6	4
Decision Support Models	6	6	5	6
Responsiveness to Change	7	8	7	8
Strategic Issues	8	7	8	7

3. EVALUATION OF THE RATINGS PER DIMENSION (Scale 1 - 7)

Imp = Importance rating
Act = Actual Rating

	1987		1988		% Change	
	Imp	Act	Imp	Act	Imp	Act
1. Strategic Issues						
Management	5.10	4.20	4.85	4.62	(4.9)	10.0
Users	5.30	4.10	4.94	4.82	(6.7)	17.5
Computer Staff	6.00	5.30	5.66	5.07	(5.7)	(4.3)
OVERALL	5.47	4.53	5.15	4.84	(5.8)	6.8
Actual : Importance %		82 %		94 %		
2. Technology and Operations						
Management	5.10	4.60	5.60	5.32	9.8	15.6
Users	5.80	4.50	5.95	4.80	2.6	6.6
Computer Staff	6.80	5.80	6.58	5.93	(3.2)	2.2
OVERALL	5.90	4.97	6.04	5.35	2.4	7.6
Actual : Importance %		84 %		88 %		

	1987		1988		% Change	
	Imp	Act	Imp	Act	Imp	Act
3. Management Reporting						
Management	5.40	4.43	5.45	4.73	0.9	6.7
Users	5.67	4.31	5.76	4.90	1.6	13.7
Computer Staff	6.36	5.58	6.11	5.59	(4.0)	-
OVERALL	5.81	4.77	5.77	5.07	(0.7)	6.2
Actual : Importance %		82 %		88 %		
4. Systems Efficiency						
Management	5.45	4.25	5.26	4.82	(3.5)	13.4
Users	5.10	4.30	5.49	4.89	7.6	13.7
Computer Staff	6.25	5.50	5.96	5.33	(4.6)	(3.0)
OVERALL	5.60	4.68	5.57	5.01	-	7.0
Actual : Importance %		84 %		90 %		
5. User Participation						
Management	4.70	4.00	5.34	4.84	13.6	21.0
Users	5.30	3.90	5.55	4.77	4.7	22.3
Computer Staff	5.70	5.30	5.79	5.22	1.6	(1.5)
OVERALL	5.23	4.40	5.56	4.94	6.3	12.3
Actual : Importance %		84 %		89 %		
6. Responsiveness to Changing User Needs						
Management	4.70	4.00	4.83	4.76	2.7	19.0
Users	5.30	3.90	5.41	4.96	2.0	27.2
Computer Staff	5.70	5.30	5.61	5.44	(1.6)	2.6
OVERALL	5.23	4.40	5.28	5.05	0.9	14.7
Actual : Importance %		84 %		96 %		
7. MIS Staff Charateristics						
Management	5.30	4.30	5.38	5.19	1.5	20.7
Users	5.30	4.20	5.78	5.21	9.0	24.0
Computer Staff	6.10	5.20	6.32	5.84	3.6	12.3
OVERALL	5.57	4.57	5.82	5.41	4.5	18.3
Actual : Importance %		82 %		93 %		
8. Decision Support Models						
Management	5.00	4.10	5.04	5.30	0.8	29.2
Users	5.00	3.90	5.50	4.86	10.0	24.6
Computer Staff	5.50	4.90	5.76	5.50	4.7	12.2
OVERALL	5.16	4.30	5.43	5.22	5.2	21.4
Actual : Importance %		83 %		96 %		

	1987		1988		% Change	
	Imp	Act	Imp	Act	Imp	Act
9. **Overall Ratings**						
Management	5.00	4.20	5.23	4.94	4.6	17.6
Users	5.20	4.10	5.57	4.89	7.1	19.3
Computer Staff	5.90	5.30	5.98	5.49	1.3	3.6
OVERALL	5.37	4.53	5.59	5.10	4.1	12.6
Actual : Importance %		84 %		91 %		

COMMENTS

- The objective of obtaining a 5-star rating was achieved.
- An effectiveness rating of 5.10 places the Company within the top 15 % category for the manufacturing industry.
- The 'Importance' factor improved by 4 % compared with an overall actual improvement of 12 %.
- The most significant effectiveness improvements were achieved in Decision Support Models, MIS Staff Characteristics, User Participation and Responsiveness to User Needs.

4. **RANKING OF THE MOST IMPORTANT FACTORS**

	QUESTION	Management		Users		Computer Staff		
17	User confidence	1	(1)	7	(8)	1	(2)	*
38	Effective D.R.P.	6		1		3		
18	Accuracy of output	2	(2)	2	(1)	5	(5)	*
11	Currency of output	3	(4)	3	(2)	4	(3)	*
10	Ease of Access to computer	4		4		11		*
26	Data security	-		-	(9)	2	(1)	
25	Quick access to computer	-		3		-	(6)	*
39	Effectiveness of reporting	5		-		8		*
16	Effective training program	-	(9)	5	(6)	-	(10)	
2	Competent System Analysts	7		9		9		*
31	Report content = right level	9		-		7		
13	Low % downtime	10	(8)	-	(4)	-	(4)	*

(previous year's ranking)
* feature in the highest ranked achievements.

COMMENTS

- The perception of the ranking of importance varies between the three groups but generally there is concensus on the top 10 factors.

- The management reporting items are dominant and support the objective of focussing on this are during this year.
- The effectiveness of disaster recovery planning is a new item and it was rated very important by all groups - again supporting the efforts to introduce improvements in this area.
- Nearly 70 % of the important items are reflected in the highest ranked achievements.

5. EVALUATION OF THE HIGHEST RANKED ACHIEVEMENTS

	Question	Management	Users	Computer Staff
11	Currency (up-to-dateness) of output information.	1	11	1
24	Positive attitude of Information Systems personnel towards users.	-	1	8
15	High degree of technical competence of the staff in the Information Systems department.	2	3	-
25	Quick and flexible access to computer data.	-	2	-
10	Ease of access for users to computer facilities via terminals/PC's.	4	-	2
13	A low percentage of hardware and systems downtime.	3	-	-
18	Accuracy and completeness of output information.	-	4	4
47	MIS department to be customer orientated.	-	-	3
17	User confidence in systems.	5	9	-
36	The availability of models to analyse business alternatives e.g. Lotus or EPS spreadsheet models.	-	5	-
14	The improvement of new system development (with respect to time, cost, quality and disruptions)	6	10	-
2	Quality and competence of system analysts developing and supporting major systems.	9	6	5
8	Access to data and models by users without involving the Information Systems Department (e.g. report writer tools)	7	-	-
20	User-orientated systems analysts who KNOW user operations.	-	7	10

COMMENTS

- The range of ranked effectiveness ratings between the groups is much greater than the importance ratings.

6. EVALUATION OF THE LOWEST RANKED ACHIEVEMENTS

	Question	Management	Users	Computer Staff
4	Integration of office communication and information services. e.g. Office Automation/Electronic Mail	1	1	2
27	Information systems providing competitive advantage for the firm e.g. terminals in supplier/customer offices.	2	-	1
12	Short development time required for new systems.	-	2	6
45	More extensive use of 'electronic' transactions and elimination of unnecessary paper flow.	3	8	3
8	Access to data and models by users without involving the Information Systems Department.	-	3	8
43	Hands-on use/understanding of PC applications by management.	4	10	5
16	Effective training programs for users in general Information Systems capabilities.	-	4	-
32	Increasing the effort to develop new systems relative to maintaining existing systems.	-	-	4
41	Encouragement of development of management reporting by users rather than DP using report writer tools.	5	-	-
40	Effective use of user group structures for co-ordinating user development/ training.	10	5	7
3	Communications between Information Systems staff and managerial users.	6	-	-
38	Effective disaster recovery plans and procedures.	-	7	-
5	Prompt processing of requests for changes to existing systems.	7	-	-
30	Systems responsiveness to changing user needs.	8	-	-
23	Flexibility of data and reports available from systems.	9	-	-

Comment

- Office Automation is the next major system to be investigated for possible implementation.
- The question of competitive advantage through Information Systems is becoming increasingly important and requires attention.
- A paper management project has been initiated to improve the paper usage effectiveness.
- Management has been involved in a Lotus Training Course.
- The overall training strategy is being reviewed to meet the user needs.

7. OTHER FACTORS INFLUENCING THE EFFECTIVENESS OF SYSTEMS

Through careful interpretation of 26 recurring items the three undermentioned categories were identified by Emanuel's study

- procedural
- structural
- behavioural

20 Senior IS practitioners were asked to rank these 26 items in importance of influencing success and the result was compared with the way in which the successful companies operate their IS business. After in-depth interviews it was confirmed that these success factors are the critical success factors applied by the excellent companies.

RANK	CATEGORY	DESCRIPTION
1	P	Involvement of operational management in design and implementation of systems
2	P	Relevance of IS projects to needs in organisation
3	B	Positive user attitudes towards IS
4	P	Technical competence of EDP staff
5	S	Nature of outputs (correctness, timeliness etc)
6	B	Support of CEO for IS activities
7	B	Influence of IS leader
8	P	Measurable objectives for IS projects
9	P	Degree of training and knowledge of users in IS
10	P	Adequacy of resources allocated to IS
11	P	Use of Steering Committee
12	P	Existence of a priority scheme for IS projects
13	B	Stability of DP personnel
14	S	Use of data base
15	P	Use of formal planning methodologies
16	P	Minimising length of development cycle
17	B	User's personal stake in systems
18	P	Standardisation of software production
19	S	Use of high level programming languages
20	S	Documentation
21	B	Organisational change caused by MIS
22	P	Regimentation of IS activity
23	B	Managers' decision style
24	S	Volume of output
25	B	Manager/user demographics
26	P	Chargeback method of payment for IS services

B = Behavioural
P = Procedural
S = Structural
(Source : Emanuel [6])

DETAILED EVALUATION OF THE INFORMATION SYSTEM EFFECTIVENESS

Annexure 2

		RATING IMPORTANCE			RATING ACTUAL		
		M	U	C	M	U	C
	STRATEGIC ISSUES						
7	Use of a management committee for overseeing and monitoring all major information systems activity.	4.82	4.79	5.67	5.00	4.75	5.11
19	Preparation of a strategic plan for Information Systems.	5.24	5.30	6.11	4.73	4.91	5.25
27	Information systems providing competitive advantage for the firm i.e. terminals in supplier/customer offices.	3.94	4.30	4.89	3.58	4.65	2.75
29	Setting of systems priorities to reflect overall corporate objectives.	5.06	5.39	5.88	4.64	4.62	5.86
32	Increasing the effort to develop new systems relative to maintaining existing systems.	4.53	4.74	5.00	4.58	4.96	4.67
34	Top management involvement in defining and monitoring information systems policies.	5.35	5.01	5.78	5.14	5.25	5.75
42	MIS strategy to be integrated with corporate strategies e.g. MIS to support by means of decision information.	5.24	4.74	6.00	4.60	4.70	5.71
44	Effective promotion of the benefits of Information Systems e.g. awareness program/attitude changing.	4.59	5.36	6.00	4.71	4.76	5.43
57	**Strategic Issues** **Actual : Importance**	**4.85**	**4.94**	**5.66**	**4.62** **95.2 %**	**4.82** **97.5 %**	**5.07** **89.6 %**
	TECHNOLOGY AND OPERATION						
10	Ease of access for users to computer facilities via terminals/PC's.	5.82	6.15	6.33	6.06	4.75	6.22
13	A low percentage of hardware and systems downtime.	5.65	6.01	6.89	6.08	4.93	6.22
26	Data security and privacy e.g. access controls, system controls and back-up procedures.	5.59	5.80	6.78	4.93	4.80	5.89
33	Application of modern information technology.	5.19	5.29	6.22	5.00	5.10	5.67
38	Effective disaster recovery plans and procedures.	5.76	6.50	6.67	4.55	4.45	5.67
59	**Technology and Operations** **Actual : Importance**	**5.60**	**5.95**	**6.58**	**5.32** **95.0 %**	**4.80** **80.1 %**	**5.93** **90.1 %**

		RATING IMPORTANCE			RATING ACTUAL		
		M	U	C	M	U	C
	MANAGEMENT REPORTING						
1	Availability and timeliness of output information supplied to users.	5.71	6.02	6.33	4.76	5.10	6.00
4	Integration of office communication and information services e.g. Office Automation/Electronic Mail etc.	3.59	4.50	4.67	3.40	3.47	3.78
11	Currency (up-to-dateness) of output information.	5.94	6.20	6.67	6.19	5.20	6.33
18	Accuracy and completeness of output information.	6.06	6.36	6.56	5.31	5.47	6.11
23	Flexibility of data and reports available from systems.	5.35	5.71	5.67	4.40	5.10	5.67
31	Relevance of report contents to intended functions e.g. right level of detail summarised per management group.	5.71	5.75	6.44	4.50	5.10	5.56
39	Effectiveness of reporting e.g. does it meet the key needs of each management function?	5.82	5.75	6.44	4.56	4.87	5.67
61	**Management Reporting**	**5.45**	**5.76**	**6.11**	**4.73**	**4.90**	**5.59**
	Actual : Importance				**86.7 %**	**85.0 %**	**91.4 %**
	SYSTEMS EFFICIENCY						
6	Efficient running of current systems (ease of use, costs, documentation, maintenance).	5.65	5.90	6.33	5.38	5.17	5.78
35	Overall cost-effectiveness of information systems.	5.13	5.41	5.89	5.00	5.04	5.78
45	More extensive use of 'electronic' transactions and elimination of unnecessary paper flow.	5.00	5.16	5.67	4.07	4.46	4.44
63	**Systems Efficiency**	**5.26**	**5.49**	**5.96**	**4.82**	**4.89**	**5.33**
	Actual : Importance				**91.6 %**	**89.0 %**	**89.4%**
	USER PARTICIPATION						
16	Effective training programs for users in general information systems capabilities.	5.65	6.06	6.11	5.80	4.39	5.33
17	User confidence in systems.	6.29	6.00	6.78	6.00	5.24	5.67

		RATING IMPORTANCE			RATING ACTUAL		
		M	U	C	M	U	C
	USER PARTICIPATION (Cont)						
21	The influence the user has over which information services are provided.	5.24	5.30	5.22	4.56	4.97	5.22
22	Users' feeling of participation.	5.29	5.11	5.78	4.50	5.05	5.56
28	Users' understanding of systems.	5.19	5.99	6.11	4.53	4.72	5.11
40	Effective use of user group structures for co-ordinating user development/training.	4.76	5.30	5.44	4.36	4.43	4.89
43	Hands-on use/understanding of PC applications by management.	4.94	5.14	5.11	4.13	4.57	4.75
67	**User Participation**	**5.34**	**5.55**	**5.79**	**4.84**	**4.77**	**5.22**
	Actual : Importance				**90.1 %**	**85.9 %**	**90.1 %**
	RESPONSIVENESS TO CHANGING USER NEEDS						
5	Prompt processing of requests for changes to existing systems.	5.06	5.91	6.00	4.40	5.01	5.67
12	Short development time required for new systems.	4.65	4.53	4.89	5.00	4.24	4.75
25	Quick and flexible access to computer data.	5.29	6.18	5.78	4.93	5.63	5.33
30	Systems responsiveness to changing user needs.	4.29	5.50	5.78	4.40	5.00	5.89
41	Encouragement of development of management reporting by users rather than DP using report writer tools.	4.88	5.25	5.56	4.19	4.67	5.50
69	**Responsiveness to changing User needs**	**4.83**	**5.41**	**5.61**	**4.76**	**4.96**	**5.44**
	Actual : Importance				**98.5 %**	**91.6 %**	**96.9 %**
	MIS STAFFING CHARACTERISTICS						
2	Quality and competence of systems analysts developing and supporting major systems.	5.75	5.91	6.44	5.40	5.32	6.11
3	Communiciations between Information Systems staff and managerial users.	5.24	5.76	6.11	4.38	5.10	5.33
9	Information Systems support for users in preparing proposals for new systems.	5.06	5.39	6.11	5.47	4.46	5.57

THE DEVELOPMENT OF A MARKETING ORIENTATED BUSINESS STRATEGY FOR THE CORPORATE INFORMATION SYSTEM

AUTHOR: PETER W COWIE SENIOR MANAGER MANAGEMENT SERVICES

COMPANY: ALUSAF (PTY) LIMITED

BUSINESS: PRIMARY ALUMINIUM PRODUCER

ADDRESS: P.O. BOX 284
EMPANGENI
3880
SOUTH AFRICA

FAX No: 0351-51697
TELEPHONE: +27-351-51111

THIS PAPER SHOULD BE READ IN CONJUNCTION WITH PAPER NO 6705 "THE EVALUATION OF THE EFFECTIVENESS OF INFORMATION SYSTEMS"

Paper Number 6707

1. PHILOSOPHIES OF MARKETING MANAGEMENT

"Marketing is so basic that it cannot be considered a separate function... It is the whole business seen from the point of view of its final result, i.e. from the point of view of the customer"

Peter Drucker

Marketing

Marketing is human activity directed at satisfying needs and wants through exchange processes.

Product

A **product** is something that is viewed as capable of satisfying a want. Therefore anything capable of rendering a service (i.e. satisfy a need) can be called a product.

Exchange

The fact that man has needs and wants and there are products capable of satisfying them is necessary, but not sufficient to define marketing. Marketing only exists when man decides to satisfy his needs and wants in a certain way that we shall call **exchange.**

Market

A market is an arena for potential exchanges.

Marketing Management

Marketing management is the analysis, planning, implementation and control of programs designed to bring about desired exchanges with target markets for the purposes of achieving the organisation's objectives.

It relies heavily on designing the organisation's offering in terms of the target market's needs and desires and using effective pricing, communication, and distribution to inform, motivate and service the market.

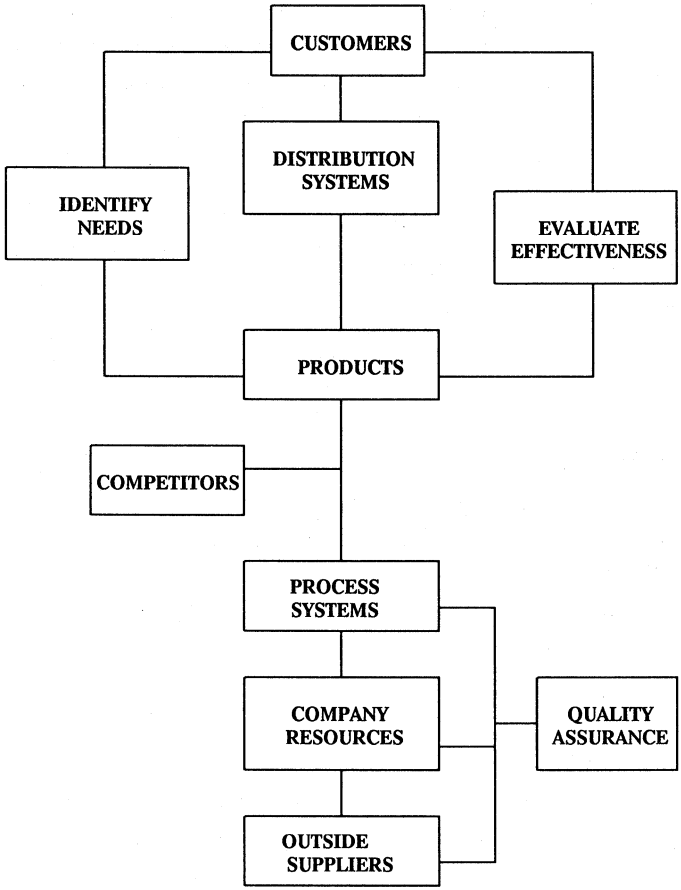
The Production Orientation

A management orientation that assumes that consumers will respond favourably to good products that are reasonably priced and that little marketing effort is required to achieve satisfactory sales and profits.

The Marketing Orientation

A management orientation that assumes that the key task of the organisation is to determine the needs, wants and values of a target market and to adapt the organisation to delivering the desired satisfaction more effectively than its competitors.

2. THE MARKETING SYSTEM AND ITS ENVIRONMENT



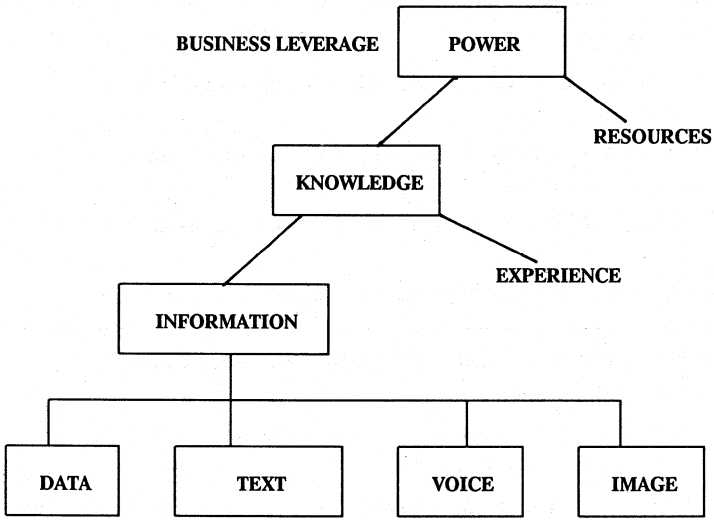
This model displays the many components of an integrated marketing system. It is possible to apply this model to almost any type of business organisation, corporate division or function. The purpose of this paper is to show you how this model can be applied to the Information Systems Division of your Company.

3. DEFINITION OF THE BUSINESS PURPOSE

The purpose should be defined in terms of meeting some need in the business and external environment and should normally not be stated in terms of delivering some product. For example:

- Volkswagen's mission is to provide an economic means of private transportation.
- IBM's mission is to meet the problem solving needs of business.
- Shell Oil's mission is to meet the energy needs of mankind.
- American Telephone and Telegraph's mission is to provide quick and efficient communication capabilities.

The information environment is shown as follows:



The product that we are providing is obviously information but in line with the examples mentioned above we need to think a little further before defining our purpose.

Suggested Business Purpose of the Information Division:

"The information Division's mission is to provide communication capabilities which when combined with business experience, will meet the knowledge needs of the business."

This is a far cry away from saying that we are in the 'hardware availability' and 'programming' business!

What is a business

* A business is a **process** which **converts a resource** into a contribution of economic value in the **marketplace.**
* The whole purpose of a business is to create a customer.

Market realities

Our perception of our customers needs are most probably wrong - only the customer knows.
We sell a product - the customer buys satisfaction.
We tend to focus on our products qualities - not on what the product does for the customer.

Business realities

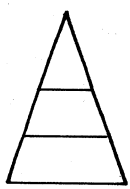
Results and resources do not exist within the business - both exist outside.
Results are not obtained by solving problems - results are obtained by exploiting opportunities.
Results are not obtained by mere competence - results are achieved by leadership.
Concentration is the key to real economic results.

4. AN EVALUATION OF OUR CUSTOMER AND HIS NEEDS

4.1 Who are our Customers?

The customers of the information division include all those who are required to make business decisions and who by definition, are required to gather and evaluate all available information in order to make effective decisions.

The hierarchy of our customer base is shown by means of the classic management hierarchy.



4.2 The Information that Management Needs

Comfort information. This information consists of a few daily figures that tell something about the state of the business, such as sales for a recent period. These figures make chief executives feel comfortable or uncomfortable, but they do little with this information.

Problem information. This is information about a major crisis or about the progress of a significant project, which deserves executives' attention. Executives want this information daily until things seem to be going smoothly.

Information for outside dissemination. This is information about the company that is to be disseminated to outsiders - for instance, company earnings figures that are given to investment analysts. Executives may be very sensitive about seeing this information before it is released.

External intelligence. This is information about the business environment and what competitors are doing.

Internal operations. CEO's typically want a few key figures that indicate how things are going. These most often are financial figures because money is the "universal measure".

4.3 Four Types of Information

	Internal	External
Record Based	Traditional EDP/MIS	Public Databases
Document Based	Word Processing Records Management	Corporate Library

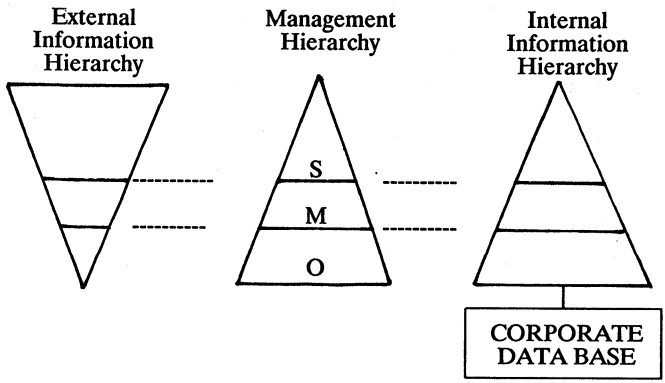
First, there are two types of information generated and managed internally in the organization: (1) information based on data records such as is found in data files, and (2) document based information such as reports, opinions, memos, and estimates.

The first type of internal information pertains primarily to entities, such as individual employees, customers, parts, or accounting codes. Well structured data records are used to hold a set of attributes that describe each entity. The second category of information pertains primarily to **concepts** - ideas, thoughts, and opinions. Less structured documents or messages, with a wide variety of information forms, are used to describe these.

The same two types of information are also generated externally to the organization. There is external record based information, such as government data on economic and financial conditions, stock price quotations. There is also external document based information, such as opinions about economic forecasts or rumours.

Internal record based information has been the focus of attention of information systems because that is the type of information computerbased application systems generate and manage easily. External record based information has become more popular recently in the form of public databases; end users themselves have generally handled the procurement of this data, often using outside time-sharing services. Until recently, practically no attention has been given by information systems management to document-based information, either internal or external, as an information resource. Those areas have been the responsibility of either the administrative manager or the corporate library.

4.4 Sourcing of Information for Each Dimension of Management



It is a fact that at the strategic level the major portion of the information required is sourced from the external environment and at the operations level the detailed internal corporate data base systems are the main source.

It is a sobering thought if you consider that top management - who determine your future and new funding resources - live in a world of uncertainties and desperately need information, yet we as information providers are only normally able to provide them with summarised internally generated data.

5. DEVELOPMENT OF A MARKETING ORIENTATION STRATEGY FOR OUR COMPANY'S INFORMATION DEPARTMENT

5.1 Purpose

A Mission Statement (Annexure I) was developed for the Information Systems based on the Mission Statement of the Company.

5.2 Business Orientation

The focus of the information System was changed to form a dynamic part of the Integrated Business Management System (Annexure II).

5.3 Customer Orientation

A detailed evaluation of our customers was undertaken, based on the considerations set out in paragraph 4, and a hierarchy of customers was established. A program of customer visits was set in motion and the MIS people went into the world of the user to see what he was doing and to learn something of the business environment in which the user worked - this exercise had a considerable impact on user perceptions.

We must remember that MIS people are seldom given training in marketing, business awareness or 'human' communications, and this change is not easy to introduce. Be fair to your MIS staff and send them on some courses other than technical courses.

Extensive use was made of Thomas Peter's video 'A Passion for Customer Excellence' and all our MIS staff as well as our customers were invited to view this video.

We also involved our major MIS suppliers by convincing them that their ultimate success could also only be achieved through the success of the end user. It was quite an experience for certain 'suppliers' to actually meet a real live end user!

5.4 Evaluation of the Effectiveness of the Information Systems

Annual evaluations of the effectiveness of our systems were undertaken using a questionnaire technique and interviews were held with 20 managers and 20 members of the end-user leader group. (Refer to paper 6705 for further details). The evaluation measured the customers level of satisfaction on the following dimensions:

- strategic issues
- technology and operations
- management reporting
- systems efficiency
- decision support models
- user participation
- responsiveness to changing user needs
- MIS staff characteristics

The most significant MIS responses to the evaluation process were
- recognition of the customer/service principle and
- an enthusiastic action plan to improve the ratings

The most significant improvement gains recorded with the second annual evaluation were in the areas of decision support models, MIS staff characteristics, user support and participation - **the areas that were rated the lowest in the previous year.**

5.5 **User Groups**

The creation of user groups for each major system supported by a system support analyst from the information department, went a long way towards developing a group team spirit and in particular it helped to break down the 'them' and 'us' barriers.

We would like to develop this success story a little further by introducing a number of user awards such as the 'Most Improved User', 'Best contribution to management reporting' and 'Information Man of the Year'. It would also be a nice touch if we had the top Hewlett Packard Executive in the country visit our company to present these awards.

5.6 **Promotions**

The importance perception and profile of the information systems also received some attention and a few examples of promotional activities include:

- a presentation to the Board of Directors concerning the strategic planning for the information systems
- invitations were extended to top management to attend certain information conferences as 'guests' of the information services.
- user groups were taken to computer exhibitions and treated to lunch as our guests
- an MIS newsletter was produced which included profiles of the information team members as well as 'advertisements' for the services available to users.
- all key user development milestones are celebrated with a few beers and a barbeque
- A very succesful subsidised private PC purchase scheme was introduced with 120 participants.

5.7 **Organisation**

The organisation structure was considerably revised with the formation of a Management Services Division reporting to the General Manager Finance. (Refer Annexure 3)

As you can see from the organisation chart we were able to bring together under one umbrella, all of the information service related activities of the Company. We were also successful in 'taking over' the library from the technical department on the basis that it formed a logical part of the information system. I might add that the MIS department were very sceptical when we renamed the library as the 'Information Centre', but they have since become more tolerant of the new member of the information family.

One of the key success factors that has supported those changes has been the very good team spirit in the MIS department. It is difficult to evaluate the financial return on an annual MIS bush camp and the informal get-togethers that take place during the year - but one thing is for sure and that is the flexibility to adapt to change is fantastic. Too often we just expect people to adapt to new situations without investing in a 'change program'

5.8 **Quality Assurance**

If you refer to the model of the marketing system you will see that the product is the final output of the process system, company resources and outside suppliers. If we wished to achieve a 100% standard for our product then by definition we must **demand** a 100% standard in each of these three areas.

It is vital to gain acceptance by all parties of this factor before you even attempt to improve the quality of your service.

6. CONCLUSION

It is generally accepted that the rate of technology change in the information industry is faster than in most other industries. However, not only do we need to manage this change but in order to earn our rightful place in the business, we need to develop a new business strategy designed to provide an information service to the whole business as well as to allow the business to make use of information as a competitive weapon.

You should develop this business strategy in three phases:

1. Knowledge - Understand your business
2. Vision - Decide on what it could be
3. Leadership - Set direction on how to reach these goals

WHERE THERE IS NO VISION PEOPLE PERISH

ACKNOWLEDGEMENTS

1. Peter Drucker 'Managing for Results
2. Sprague and McNurlin 'Information Systems Management'
3. Kotler 'Marketing Management'

MISSION STATEMENT : INFORMATION SYSTEM

PURPOSE

We are a service organisation and provide the Shareholders and Management with a quality Management Information Service.

We are committed to relating information systems to business needs in order to provide a competitive advantage to the Company.

We are committed to promoting the need for effective information and Planning and control systems and a general awareness of the business systems to all decision makers within the Company.

We are an integral part of this dynamic organisation and make a significant contribution to the effective management of the business.

VALUES

Marketing Orientation

We believe that the success of our **Customers** is a measure of our own success. We seek to identify their needs and dedicate our efforts to provide them with distinctive professional service and technical support.

Quality

We believe that the quality of our products and services is paramount. We expect a dedication to quality from all our people as well as our suppliers of goods and services. We will concentrate on doing the job right the first time and will not tolerate wastage.

Human Relations

We believe in the development of our people and will encourage training to enable each employee to progress within the Company.

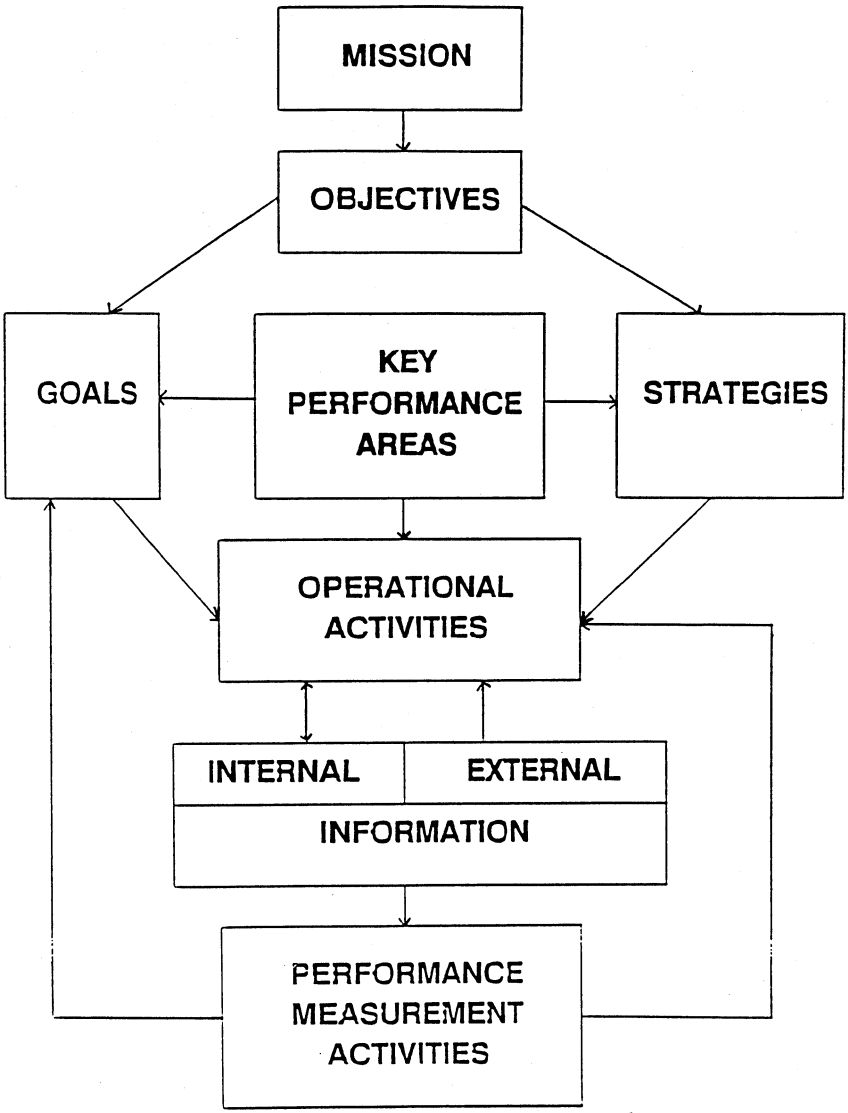
We believe that communication is the cornerstone of good people relations and we encourage sincere face-to-face communication at all levels.

We believe that our own results should be achieved by a team effort from all of our people.

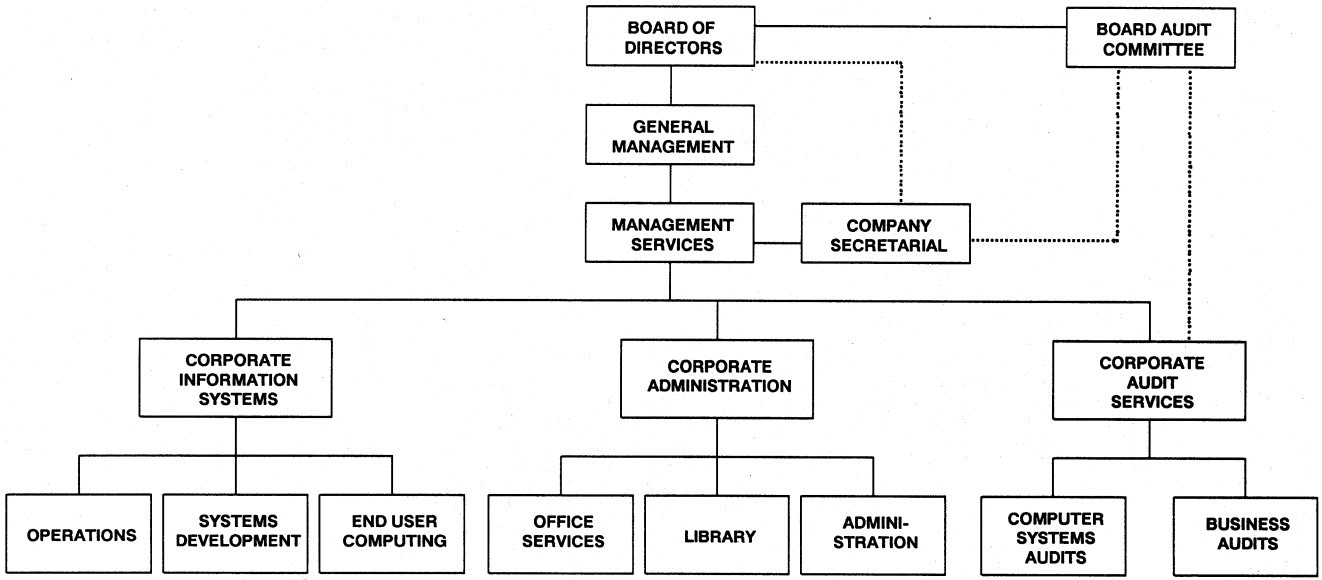
Management Style

We believe in participating in the team building style of management based on clear objectives and strong leadership. We associate ourselves with the sound business and generally accepted practices of this organisation, and will maintain ethical and cordial relations with outside institutions.

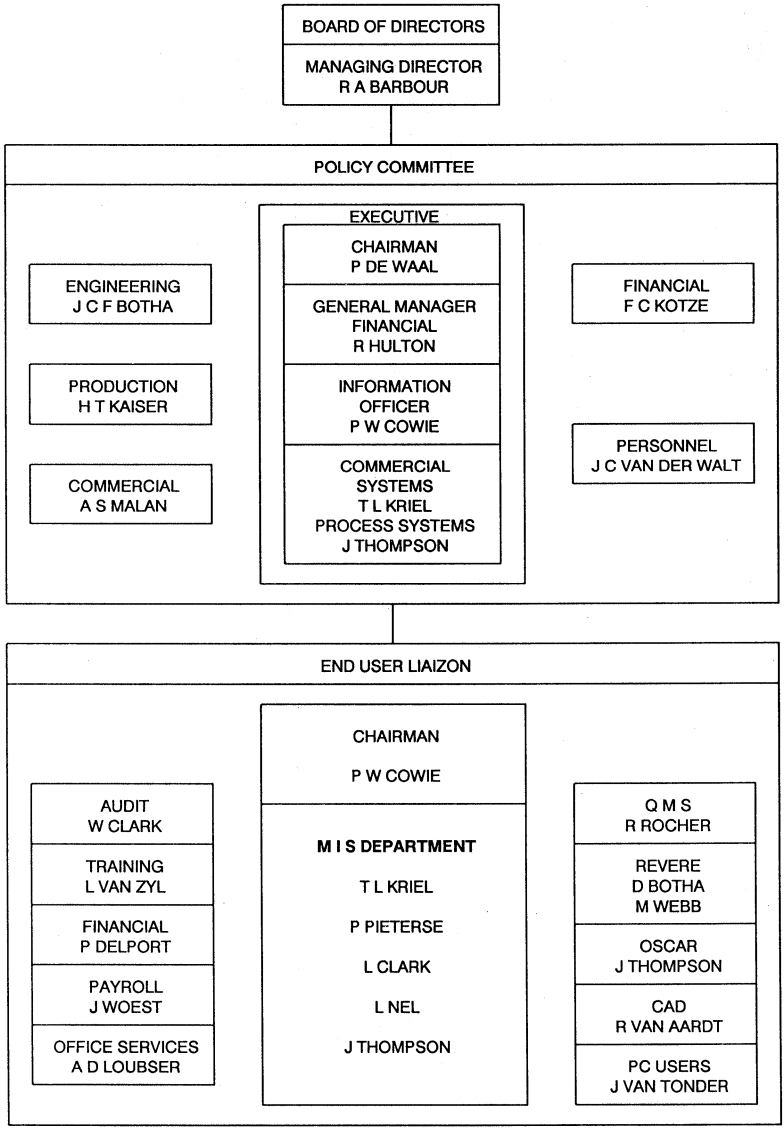
INTEGRATED MANAGEMENT SYSTEMS



6707 - 12


BOARD OF DIRECTORS
BOARD AUDIT COMMITTEE
GENERAL MANAGEMENT
MANAGEMENT SERVICES
COMPANY SECRETARIAL
CORPORATE INFORMATION SYSTEMS
CORPORATE ADMINISTRATION
CORPORATE AUDIT SERVICES
OPERATIONS
SYSTEMS DEVELOPMENT
END USER COMPUTING
OFFICE SERVICES
LIBRARY
ADMINI-STRATION
COMPUTER SYSTEMS AUDITS
BUSINESS AUDITS

CORPORATE INFORMATION SYSTEMS POLICY AND MANAGEMENT STRUCTURE



KEY PERFORMANCE AREAS

MANAGING DIRECTOR

- To inform the Board of Directors of the developments and future strategies of the Corporate Information Resources.
- To provide the Policy Committee with insight into key business strategies and trends in order to merge the business plans and information systems plans.

POLICY COMMITTEE

- To approve the Strategic Information Systems Plan.
- To confirm procedures and policies.
- To approve the capital budget.
- To appoint the Information Officer.
- To evaluate all divisional business strategies and to determine the implications for information systems planning.

EXECUTIVE COMMITTEE

- To make recommendations to the policy committee.
- To identify and evaluate development requirements.
- To evaluate the growth management and capacity planning systems.
- To approve capital expenditure within the broad policy approved by the Policy Committee.

INFORMATION OFFICER

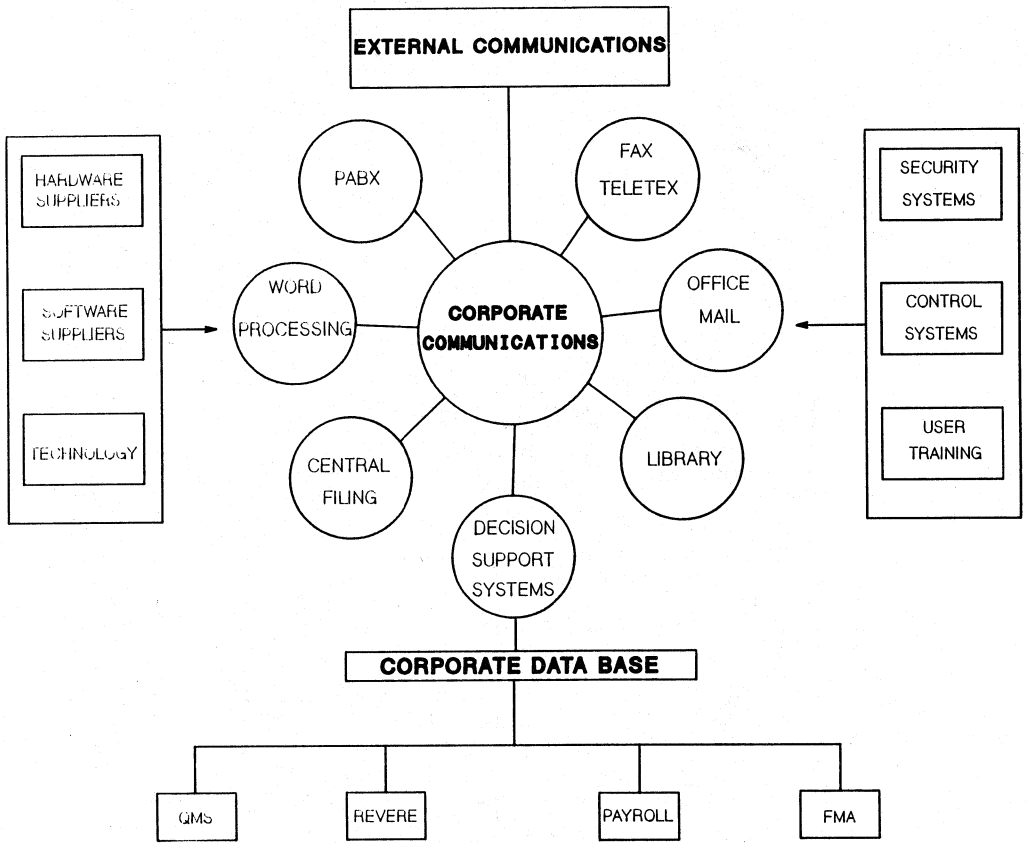
- To formulate the Strategic Information Systems Plan.
- To develop the structure of the information systems framework in which the divisions operate.
- To evaluate the effectiveness of information systems.
- To develop a marketing orientation and business awareness within the information organisation.

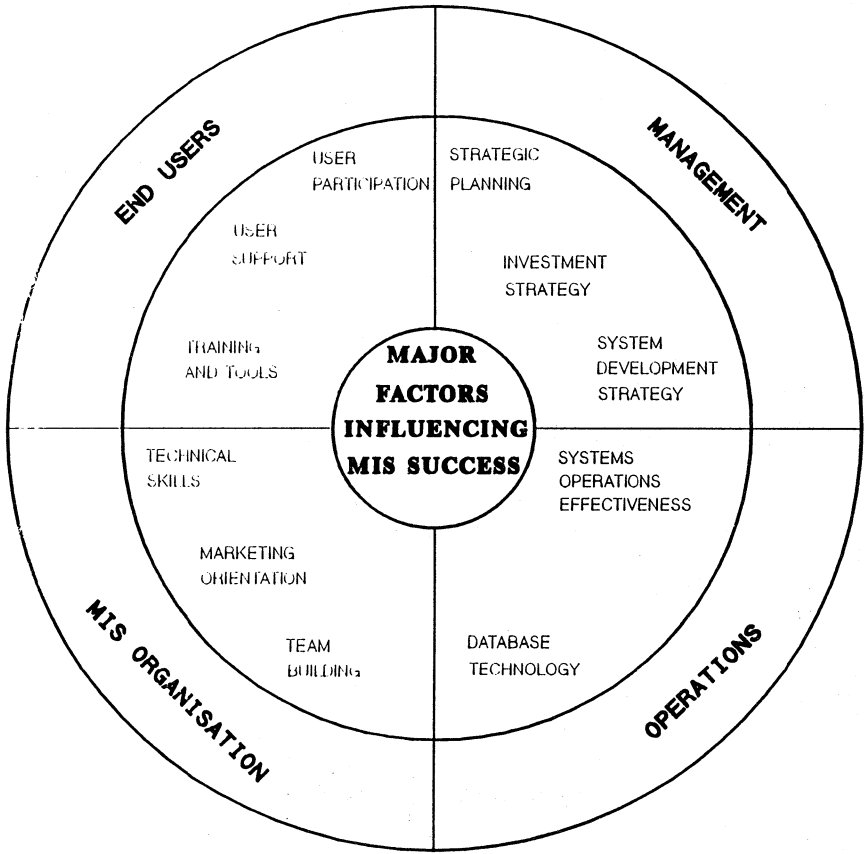
MANAGER - COMMERCIAL INFORMATION SYSTEMS

- To implement the strategic plans and to develop long and short term information systems planning.
- To develop procedures for the management of systems growth and efficiencies of systems.
- To determine policy and procedures for information systems.
- To provide an information systems service to the organisation.
- To co-ordinate the capital expenditure requirements and to prepare capital submissions to the Policy Committee.
- To bridge the gap between top management and the technical people in information services.
- To develop a training and awareness program designed to educate users in the areas of system development disciplines, risk management and effective usage of systems.
- To plan the development of the MIS organisational structure, staffing, development and training of subordinates.
- To develop and implement risk management procedures.

END-USER LIAIZON COMMITTEE

- To provide a forum for the exchange of experience and information and the promotion of a professional service.
- To communicate the Information Systems Plans and developments to the end users.
- To co-ordinate the development priorities and allocation of support service.


EXTERNAL COMMUNICATIONS
HARDWARE SUPPLIERS
SOFTWARE SUPPLIERS
TECHNOLOGY
PABX
FAX
TELETEX
WORD PROCESSING
CORPORATE COMMUNICATIONS
OFFICE MAIL
CENTRAL FILING
LIBRARY
DECISION SUPPORT SYSTEMS
SECURITY SYSTEMS
CONTROL SYSTEMS
USER TRAINING
CORPORATE DATA BASE
QMS
REVERE
PAYROLL
FMA


END USERS
MANAGEMENT
MIS ORGANISATION
OPERATIONS
USER PARTICIPATION
STRATEGIC PLANNING
USER SUPPORT
INVESTMENT STRATEGY
TRAINING AND TOOLS
SYSTEM DEVELOPMENT STRATEGY
MAJOR FACTORS INFLUENCING MIS SUCCESS
TECHNICAL SKILLS
SYSTEMS OPERATIONS EFFECTIVENESS
MARKETING ORIENTATION
TEAM BUILDING
DATABASE TECHNOLOGY

Ethical Issues in Managing Information Systems

Debra B. M. Canfield
Dairylea Cooperative Inc.
831 James Street
Syracuse, New York 13203
315-476-9101

Should we honor software license agreements? Should we tell programmers only half the truth to get or keep them on the job? Should we commit to project deadlines we know we cannot meet?

While these questions might be answered with a simple yes or no, they all have ethical implications which I shall explore in an attempt to establish principles which can help managers make decisions more easily and more wisely.

Should we honor software license agreements?

Certainly most of us would answer this question with a "Yes!" but what do we really mean and why would we answer this way? First, what do we mean by "honor"? A simple definition of honor as used in this context would be "keeping our agreement." It includes a sense of ethical conduct, honesty and integrity.

Second, what exactly is a license agreement? We might have every intention of honoring our agreements, but if we don't understand what they are, it may be hard to do. When we purchase software licenses, we normally have to sign a license agreement with the vendor. Since we would never sign something without reading it, we ought to be able to assume we completely understand and agree with the agreement. Unfortunately, this is often not the case. Either we're so excited about getting the new software (or maybe too busy?) we don't really pay much attention to what the agreement says, or we really do try to read it, but it's just not written in a way which is very understandable.

With much personal computer software it's even easier to overlook the license agreement. It's not necessary to look at an agreement or sign anything before you have the software. When you open it, the label which says you accept the license terms when you break the seal simply looks like a sticker with a lot of little writing designed to keep the media from falling out of the package. The license agreement itself, in perhaps four languages, doesn't look very important or useful, and if it's not printed in a way so that it can be easily put in the binder with the other documentation, it is likely to simply be thrown out with the plastic shrink wrap.

So, exactly what are we agreeing to when we break that seal or sign that agreement? License agreements vary, but after going through many software agreements I have signed or seen over the past several years, I find they usually have the following in common:

1. The software may be used on only one computer. Copies can only be made for backup purposes.
2. The license is non-exclusive (multiple licensees), non-transferable (licensee can't give or sell his rights to somebody else) and limited.

3. Title or ownership in the software remains with licensor. Licensee may not decompile it.
4. Software is copyrighted, confidential, etc., and licensee will maintain its confidential nature.
5. Updates and/or support may be included for a specific amount of time. Continuance of such beyond this period is through a separate agreement.
6. Warranties vary and are generally quite limited.
7. Payment requirements are stated.

A license agreement is a contract between the software supplier (licensor) and user (licensee). The agreements are usually more favorable to the licensor, which is understandable, since he usually writes them. In the preceding list of agreement commonalities, only points 5 and 6 offer anything but limitations to the licensee. Notwithstanding agreement wording such as that the licensor is "not liable for any direct, indirect, special, incidental or consequential damages including profit losses" arising from use of their software and that the "limited warranty is exclusive and in lieu of all other warranties and remedies, express or implied," reputable software suppliers really are attempting to provide a sound product which gives value for your money.

Essentially then, a license agreement is a promise by the software supplier to provide software useful to the licensee and a promise by the licensee to use the software on only one computer and not to divulge its secrets.

It's easy to see how the licensee can break his promise. He can give a copy of the software to all his friends. He can buy just one copy and run it on many of his computers. He can figure out how it works, make a few modifications and start selling it himself, etc.

The software supplier can break his promise by distributing software that doesn't work quite right, by failing to provide agreed upon support, etc. As mentioned earlier, the warranty part of license agreements is what tends to vary the most. Some software suppliers promise the software will work as described or they will fix it. Others only say they will make an attempt to fix it. Even if the agreement doesn't actually say so, and though you might not be able to prove it legally, I believe the expectation is that the licensee is paying for something which will work.

If we assume for a moment that breaking a promise is not a problem in and of itself, what's wrong with making a copy of a software package for a friend? What will the consequences be?

First, your friend will have some free software, and second, the software supplier will have lost the revenue they would have received if your friend had bought the software. This would seem to be a good arrangement for your friend, but not so good for the software supplier. If we take this a step further and assume everyone operates on this principle, we'd have free software for all and tough luck for software companies, right?

Wrong! The natural outcome of such behavior would actually be tough luck for everybody. If software suppliers were no longer paid for their software, the results of their labor, the profit motive would be gone, and software companies would lose their incentive to produce. Fewer people would be drawn to developing software, eliminating competition, which is a spur to make a better

product. Eventually there would be no new software, or if there were, it certainly would be of an inferior grade.

The profit motive and competition are basic tenets of capitalism as are free enterprise (private versus government control) and private ownership of property. Our whole economic system, and in fact the reasonableness of discussing license agreements at all, rests on these principles. While economic systems vary around the world, these principles must be followed to some degree to support the validity of holding copyrights and selling licenses to use software. The seminar to be held outside Moscow in June by several Western companies on copyright and legal issues in East/West trade[1] is evidence of this fact.

By now you may be wondering if whether we should honor software license agreements simply boils down to an economic question, why am I discussing it in a paper entitled "Ethical Issues in Managing Information Systems"? Obviously, it's because I recognize that there are ethical issues involved.

What is ethics?

Ethics is a branch of philosophy. It includes the study of 1) values, what is good and bad; and 2) moral duty, what is right and wrong. Hence the wording of the preliminary questions, "Should we..." Is it good or bad, right or wrong, to do, or not do, a certain thing, or is it ethically neutral?

Where and how do we start thinking about such things? In reality most of our ethical ideas are formed and influenced by our culture and how we were brought up. In the West many of our ethical ideas have their roots in the Judaeo-Christian ideals presented in the Bible. Titus and Keeton point out that while the metaphysical and religious background in the East is quite different from ours, the similarities in the character traits which are approved and disapproved are remarkable.[2]

It is a mistake to equate ethics with the law. Rather than something being wrong because it is against the law, the belief that something is wrong comes first, and the law is made to support it. Hence, the law is actually built upon and is a reflection of our ethical system. When laws do not accurately reflect the values and morality of society, they become difficult to enforce. I believe Prohibition was a good example of this, and the current controversy surrounding abortion is not a controversy about obeying the law but about the ethical issues involved.

Various ethical systems exist, and formal studies of ethics categorize them in various ways. Rather than presenting a formal treatise on all the different thoughts and who thought them, I'll simplify them as coming from three different perspectives. I'll call them 1) Me Theories, 2) We Theories and 3) He Theories.

Me theories start with the interests of the individual. Whether something is good or bad, right or wrong, is determined by how it affects me. Associated thoughts are egoism and individual relativism.

We theories look for the interests of groups, or the majority, of people. Cultural relativism and utilitarianism (the greatest good for the greatest number) characterize these outlooks.

1"Soviet Software Coming!," Datamation, XXXV (April 15, 1989), 13.

2Harold H. Titus and Morris Keeton, Ethics for Today Fifth Edition (New York: D. Van Nostrand Company, 1973), p. 324.

He theories are based on the belief that there is something higher than man, be it God or Nature, which determines the goodness or badness, rightness or wrongness of things. Theism and idealism are examples of He theories.

Me and We theories tend to be subjective and relative. He theories contain universals by which objective decisions can be made.

Having complex minds as we do, and being influenced by many things, our ethical beliefs do not tend to fit into a single pure theory. In fact there are dangers in taking some theories to their pure form. Purtill shows that if we believe the greatest good for the greatest number is the ultimate criterion, we find situations in which eliminating certain people is justified by providing more good for the majority.[3] This extreme conclusion, which most of us consider unthinkable, has actually been followed by some.

How can we apply ethical theories to our software license question?

If I follow a Me theory, I might be tempted to say, "Forget the license, free software for me!" On further examination I would realize the possibility of the consequent demise of the software industry, which would be bad for me, because I would have to write all my own software. However, this conclusion assumes everybody else would act as I would and give away software. If I'm the only one who breaks the license, I'm ahead. Of course, if other people see what I do and follow my example, my gain won't last long, and they might resent me and get back at me in some other way. Then again, making unauthorized copies of software is illegal, and I don't really want to get caught. So... all in all, I guess a logically followed Me theory, taking a long run versus a short run outlook, is going to honor the license agreement.

From the software provider's standpoint, they might at first be tempted not to be concerned about the quality of their software, as long as they were paid for it. In the long run, however, people would not continue to buy defective software, and the software supplier would lose business, so it's also in the licensor's best interest to honor the license agreement under a Me theory.

A We theory, by recognizing what will happen to everyone if license agreements are not honored, will honor them.

In 1984, Future Computing Incorporated sent a survey about business software for personal computers to approximately 70,000 households. Based on 45,000 responses, they believed that for every authorized copy of software there was an unauthorized copy, and that copy-protected software was pirated almost as much as unprotected software. They estimated that piracy cost the business software industry $1.3 billion from 1981 through 1984, and that assuming 25% of pirated copies would have been purchased, another $800 million was lost in 1985.[4]

While some people receive free software from such activities, the incidence is not so severe that major companies are going out of business as a result, but to stay in business their price per copy must be higher to cover their expenses and make a reasonable profit. Ultimately then, the people who honor licenses and pay

---

3Richard L. Purtill, Thinking About Ethics (Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1976), p. 31.

4Future Computing Incorporated, News Release (Dallas: January 17, 1985) I was unable to obtain more current figures from this company.

for software are paying more to support those who don't. Such negative effects on a large group are not justified by a We theory. Similarly, licensing software which doesn't work quite right causes damage to licensees and also is not justifiable.

He theories vary in their content. Being based on the belief that there is a reality higher than man which determines the ethical code, it may be tempting to think of He theories as being based on long lists of dos and don'ts. In actuality, comprehensive He theories begin with broad universals upon which specific values and moral duties are based.

For instance, in considering the tenets of Judaism and Christianity, the Ten Commandments quickly come to mind. "Thou shalt... Thou shalt not..." When Jesus was asked which was the greatest commandment he replied, "'Love the Lord your God with all your heart and with all your soul and with all your mind.' This is the first and greatest commandment. And the second is like it: 'Love your neighbor as yourself.' All the Law and the Prophets hang on these two commandments."[5] Rather than picking out a favorite do or don't, he reached to the heart of the matter and said that proper ethical conduct is based on the reality of God and man and loving relationships between them. All the commandments follow as a logical result of this.

Whatever their basis, He theories in general do not favor stealing, and dishonoring software license agreements is essentially stealing, either through making unauthorized copies, or through taking the licensee's money without providing an adequate product.[6]

Practically speaking, it's probably easier for an MIS manager to make sure software licenses for larger computers are honored than it is for him to make sure software licenses for personal computers are honored. After all, the number of larger computers and the number of people who would want software for them are more limited. With personal computer software in particular, it's important to let other people know both verbally[7] and by example that copying software is not condoned.

When dealing with specific situations, I believe that understanding the basic intent of the agreement, which is to provide a single copy of good software for a given price, and determining to honor the intent, will provide an ethical answer. For instance, I regularly buy personal computers and order software for them. We use several standard packages and have many copies. Sometimes the hardware arrives before the software, and since we're always anxious to get the new computer set up, we go ahead and put a copy of the software we already have on the computer. Even though the copy which is technically for that computer has not yet arrived, I believe this is perfectly ethical. The software licensor is getting their

---

5Matthew 22:37-40 New International Version

6The Free Software Foundation seems to take a different view on this entire issue. According to its originator, Richard Stallman, the "ownership of software is antisocial...because it's based on the obnoxious principle of promising not to share with your neighbor." (Jos Duerinck, "Stallman an Energetic Activist for software free-for-all," Interrupt, VII (April, 1989), 15, from English Computer Magazine, October, 1988). I do not really understand this argument, since if we extended it to other things, why not expect automobile manufacturers to give us free cars, and how does Stallman then justify charging for improvements or modifications?

7Some companies have employees sign written policy statements on software protection policies. An example is found in "Thou Shalt Not Dupe," a brochure published by ADAPSO, which can be obtained by writing to ADAPSO, Suite 300, 1300 North 17th Street, Arlington, VA 22209.

money, and I am paying for all my copies. I don't think the slight timing difference should be a big concern to anyone. The intent of the agreement is honored.

How should we deal with our staff?

Honoring software license agreements is an example of an ethical issue MIS managers face in dealing with those outside our own companies. When dealing with people inside the MIS department, we face other concerns.

Let's begin with the hiring process. Our basic goal in hiring staff is to find the best person for the job and get him to accept it. Other objectives may also be involved, such as getting somebody fast, etc. In an effort to get or keep the best person for the job, an effective approach might seem to be to tell him what he wants to hear and to leave out those things he might not want to hear.

A short run Me theory would support this practice. It would result in the greatest likelihood of the person taking the job, and you would have met your goal of getting the person hired. In the long run, however, when this employee sees the way things really are, is he going to stay with your company? Or is he going to stay but lose his commitment to the company, or maybe even complain to everybody else and contribute to a drop in the morale and productivity of the whole group? A long run Me theory won't support this type of misrepresentation, because you're likely to run into a lot more grief and aggravation as a result of it.

A We theory looking out for both the company's and the employee's interests not only will want the company to get the best person, but also will want the person to get the job which is best for him. The likelihood of a good match is going to be the greatest when the company accurately presents the job and company.

Most He theories frown upon lying and dishonesty. It may be tempting to justify omitting information on the grounds that it is not really lying, but it is certainly not being honest with the other party and is not supported by He theories.

Note that while this example is from the perspective of the ethical behavior of the hiring manager, the same principles apply to the job candidate. I'll leave it to the reader to outline the possible outcomes of exaggerating past experience to get a job and to decide whether you can rationalize such deceit as ethical behavior.

Another example of ethical issues arises in dealing with employees who may or will be leaving the company. In discussing these issues, rather than going through how to look at them according to each of my three general theories, I'm just going to tell you about some actual situations. Judge for yourself where ethical stands were taken and the results.

Last year my company made a decision to sell a major part of our business, and I was faced with a situation of having to let four people (about half my staff) go. While I knew several months before the agreement took place, I also knew part of the agreement was that we would provide computer services for the buying company for over nine months after the agreement became public knowledge.

It would clearly be to the company's advantage to keep the entire staff until the last day we had to provide service for the other company. For employees concerned about their careers whose jobs were about to disappear, the simple clear advantage was to find a good job with a new company as soon as possible.

Two questions were important: when and how to tell everybody and how to handle their actual leaving.

When to tell them was to a large extent controlled by the company. For various reasons, they wanted to keep the deal a secret as long as possible, preferably until it was finalized. One reason was to keep the employees calm. Since there would be plenty of time before my staff needed to make any decisions, I knew this would not hurt them. My preference was that I be able to tell each of my staff myself on a one-to-one basis, and that everybody in the company be told on the same day.

The plan was good, but unfortunately, somebody talked to their staff sooner than they should have, and as such news spreads, not starting with all the facts and getting less reliable and more wild with each telling, soon the company was in turmoil. Not only was a lot of company and personal time wasted in unnecessary worry and discussion, but some of the initial reaction was that those of us who had kept quiet were less than open and honest with our departments. This perception was eliminated once all the facts were known, which was only a few days later, when I had official permission to talk to my people.

In the discussion with each person, I outlined what was happening in the company, whether they were or were not going to still have a job, and the status of all the others in the department. If they were losing their job, I also told them how long their present position would be available and spent time talking to them about future career options. This was a good time to emphasize their strengths and to encourage them to look for a place where they could put them to good use. Since about ten months remained before anybody's job would be gone, I encouraged them not to rush into anything but to take some time to think about what they wanted to do next and to wait for the right opportunity. This would be in the company's best interest as they might stay longer and in their interest, since it's not often one has the option of extended time and his current employer's help in a job search.

Naturally not all of them stayed until the end. The company offered severance pay based on length of service to those who did, but since my staff was all relatively new and had plenty of attractive options, this was not much of an incentive. The first person to go did not leave until more than four months after I had first talked to her.

When another manager heard she was leaving, knowing how important she was to the department he asked me if there wasn't something I could do to get her to stay longer. Certainly I would have liked her to stay until the end; my life would have been simpler. Knowing that she had found a good opportunity, however, and knowing that I had nothing to offer her in the long run, I did not feel I could ask her to stay. Her best interest was in taking the new opportunity, and I believe one of the reasons she and the others stayed as long as they did was because they knew they had been respected and treated well by me and the company.

One good thing that surprised me not long after I had initially talked to my people was that two of them made definite commitments to stay until the end. I had not asked for, or expected, such a commitment, but I was certainly happy for it. I was then able to make definite plans about how all our responsibilities could be covered.

About four months before the end, a programmer who had made a commitment to stay until then began his job search. While it did not seem unreasonable that some openings needed to be filled before he would be available,

and he would thereby not be in consideration for them, he was surprised at the number of people who suggested he break his commitment to our company. Once he had made the commitment to stay, I was certainly counting on him staying, and he was definitely planning to stay, so why did some other people expect him just to walk out as though his commitment meant nothing? Shouldn't they rather be glad to find a person who could be trusted to stand by his word? As a manager I certainly am, because I've always felt, and I think my experience with my staff over the years has proved it true, that I'd rather hire a person with the right attitude and the right aptitude who needs specific training than hire a person who already knows how to do a specific task but whose attitude and aptitude are lacking.

One of the employment agencies with which this programmer interviewed gave him a handout entitled "Beware of the 'Counter Offer'."[8] He thought I would be interested in seeing it, because he knew I was writing this paper, and particularly because of its decidedly negative tone. The handout seems to assume throughout that the company cannot be trusted, and while several points are valid, others seem to grasp for every possible reason why the employee should leave and not accept a counter offer. (However, it doesn't bother to point out a very good reason from the employment agency's standpoint of why a person should leave a current employer for a new job; only then does this type of employment agency get paid!)

Certainly if the things which make a person want a new job cannot be changed as part of a counter offer, the person should not accept it. In the long run, this would benefit neither the person nor the company. But what is the likelihood that a company is going to "buy time" with the counter offer, immediately look for a new person and fire the original person as soon as a replacement is found? Despite the fact that the handout implies this happens frequently, I can think of no reason other than sheer meanness why a person would do such a thing. How could this procedure be logically justified?

You might think that it would be good for the company to keep a critical position filled, but what are the real effects of such a move? A presumably valuable person will still be lost, and what would be the effect on the department and the rest of the company? I can't think of a better way to start a negative downward spiral, and I am quite certain that such a tactic could only be used once. Nobody would ever again trust a counter offer or probably anything else that manager, or maybe even the company, said.

How should we deal with those outside our department?

The preceding issues deal with subordinates. Let's look at an issue where the MIS manager is dealing with equals or superiors. Should we commit to project deadlines we know we cannot meet? For the question to be even reasonable, there must be some perceived benefit to answer in the affirmative. So, let's assume this is a really hot project everybody wants done as soon as possible.

A short run Me theory might want to commit to that deadline. Everybody wants it done, and you will look good by saying you can do it. However, if you

---

8"Beware of the 'Counter Offer'" was given to the programmer by C F A Associates Inc. in Syracuse, New York. A couple months later we were surprised to read an almost identical article in the "PeopleWare" column of the March 1989 issue of SuperGroup Magazine entitled "Beware of the 'Counter Offer'" by Diane Amos. C F A Associates has told me they do not know the original source of the handout but that they have been using it for years. At the time of this writing, we have not been able to determine the original source to whom credit should be given.

really can't, the bad press as a result will more than negate the original favorable outcome.

We theories will evaluate the effects on the whole company, and unless you can think of some good reason why having people think something will be done sooner than it can be, We theories won't justify such a commitment. More likely, We theories will recognize and respond to all the added work and related problems which will be caused by rearranging plans and will consequently favor realistic estimates.[9]

Once again, He theories will make their decision based on the issue of honesty.

The October issue of the HP Chronicle carried a wonderfully funny guest editorial by Virginia Shoemaker entitled "Systems Management: Maintaining the Guru-Like Glow."[10] The story is about what to do when you, as a system manager, accidentally lean against your computer's halt button. While all of the points are hysterical, what I particularly remember is part of the dialogue with the DP manager as follows: "Q: The system went down? A: Yes Q: Do you know what caused the failure? A: No. (Note: you aren't really lying. I mean, who really knows what goes on during a halt?)" As a matter of fact, you are lying, since the definition of lying includes not only saying something which isn't true but saying something which is intended to mislead.

While I accepted and enjoyed the tongue-in-cheek article, I couldn't help but think what my response would be if one of my staff gave me that kind of answer. They would either soon change their ways or not be around long enough to have to answer any more questions. I am willing to accept mistakes. We all make them. I need to know that when I ask one of my staff something, I get a straight answer. That way, I can support them without question when the need arises.

Likewise, other people in the company know the same about me, and I believe that is one of the reasons why I have been given so much responsibility and decision-making authority.

What shall we do?

If acting ethically is so beneficial to people and companies, why do we see so many cases of unethical behavior? I believe the two main reasons are ignorance and failure to think things through.

One reason copying personal computer software is so common is that many people simply do not know that it is stealing. They don't understand the intent of one purchased copy per computer and think they have paid for the copy by buying the disc on which to copy it.

This past year one of the Syracuse area schools had a problem with a student breaking into their HP 3000. When a Hewlett Packard systems engineer

---

9This argument assumes a fairly clean environment. If it's routine in a company to underestimate times, costs, etc., accurate estimates might at first produce some surprising results. Consider all the time and energy wasted by gamesmanship, which could be saved by playing straight.

10Virginia Shoemaker, "Systems Management: Maintaining the Guru-Like Glow," The HP Chronicle, V (October, 1988), 66.

was talking to their system manager about it, he was told that the school teaches the students all about the computer system but nothing about the morality or ethical questions of invading and destroying computers. He was told that it was simply "not in the curriculum."

Failing to think things through properly and failing to look at long run consequences likewise can lead to faulty conclusions. As managers we are paid to look at the big picture and to plan ahead, and if we do so carefully, I believe we will arrive at appropriate ethical conclusions. By thinking through the basic issues ahead of time, we are better able to make sound decisions more quickly.

I believe that in general people want to act ethically. As managers we are responsible not only for our own actions but for those of our staffs. By setting an ethical example, we provide a climate for ethical behavior in others and build and encourage responsibility in the organization.

Bibliography

ADAPSO, "Thou Shalt Not Dupe" Arlington, Virginia: ADAPSO, 1985.

Amos, Diane, "PeopleWare: Beware of the 'Counter Offer'," SuperGroup Magazine, IX (March, 1989), 34-35.

"Beware of the 'Counter Offer'" received from C F A Associates Inc. 5790 Widewaters Parkway Syracuse, New York 13214.

Duerinck, Jos "Stallman an Energetic Activist for software free-for-all," Interrupt VII (April 1989), 15, from English Computer Magazine, October, 1988.

Future Computing Incorporated, News Release, Dallas: Future Computing Inc., January 17, 1985. [obtained from ADAPSO]

Gellerman, Saul W. "Why 'Good' Managers Make Bad Ethical Choices," Harvard Business Review, (July-August, 1986), 85-90.

Hall, Bert M. A Christian Approach to Ethics. Houghton, New York: Houghton College Press, 1972.

"Hiring Programmers Without Headhunters," Managing MIS Personnel, III (January 1989), 1-3.

Purtill, Richard L. Thinking About Ethics. Englewood Cliffs, New Jersey: Prentice-Hall Inc., 1976.

Shoemaker, Virginia "Systems Management: Maintaining the Guru-Like Glow," The HP Chronicle, V (October, 1988), 66.

"Soviet Software Coming!," Datamation, XXXV (April 15, 1989), 13.

Starling, Grover. The Changing Environment of Business. Boston: Kent Publishing Company, 1980.

Titus, Harold H. and Morris Keeton. Ethics for Today. Fifth Edition. New York: D. Van Nostrand Company, 1973.

White, Jerry. Honesty, Morality & Conscience. Colorado Springs, Colorado: NavPress, 1979.

Strategic Planning Through User Input

Lloyd D. Davis and
Elisabeth M. Craig

Center of Excellence for Computer Applications
University of Tennessee at Chattanooga
615 McCallie Avenue
Chattanooga, Tennessee 37403
615-755-4396

Abstract

During the summer of 1988, a task force at The University of Tennessee at Chattanooga (UTC) prepared a strategic plan for academic computing for the next five years. A traditional approach is for administrators and computing personnel to jointly devise such plans. A different approach was used for the UTC study. A six member, campus-wide, task force prepared a survey, received written responses from and interviewed over 70 faculty and administrators to determine their perceived long-range needs. Through the discovery theory and constant comparative method, developed by Glaser and Strauss, the data collected from the interviews with users were transformed into a matrix of needs and priorities. In turn this matrix was used to develop recommendations which were resolved into timetables and schedules for the estimated costs of the strategic plan.

This paper describes in detail this approach to strategic planning and evaluates the strengths and weaknesses of this method. Also, the presentation describes other aspects such as the reactions of those surveyed, and the process of analyzing the large amount of qualitative data. This UTC strategy and its results are compared to those used at other institutions. Although UTC's strategic planning operated within an educational environment, the methodology employed has promising implications for business and industry; this topic also is discussed.

Background Information

From 1975 to 1984, all educational computing resources were under the jurisdiction of the department of Academic Computing Services. In 1974, the UTC department of Academic Computing was established and for interactive computing, the HP2000 system, was purchased. Additional interactive facilities including several HP3000s were added as the demand for resources increased.

In 1984 UTC instituted a Center of Excellence for Computer Applications (CECA). This interdisciplinary center melded faculty and staff from areas such as engineering, computer science, business, education, and Academic Computing. The Center supports undergraduate and graduate instruction (at the master's and post-master's level), as well as faculty and student research.

At the present time, the main controllers of computing at the university are the Center of Excellence for Computer Applications (CECA), the Administrative Computing Center and the Lupton Library. In addition, some academic departments (such as physics, computer science, and engineering) maintain their own computing devices and are smaller resource areas. Thus, the university has decentralized schema for computing.

Based on the Dartmouth tradition, academic computing has always been a "free" resource. All faculty, staff and students are encouraged to make use of the on-campus computing systems without charge. Additionally, UTC faculty have paid out over two-thirds of a million dollars in personal funds to purchase microcomputers, an indication of the importance that these facilities have in the university environment.

Factors such as the decentralization of resources, increased demands for services and the rapidity of technological change made it clear that a strategic plan was needed. The goal of such a plan was to provide a framework for growth, obtain a consensus of opinion and prevent duplication of efforts and material.

Morrell (1986, p. 36) points out that while corporations can use profits to measure results, educational institutions "must attempt to measure the quality and level of service". The data used for the planning process, therefore, was not the more traditional, quantitative information. Instead, UTC obtained qualitative type input, from over 70 faculty and administrators users as the medium to construct a strategic plan. Since analysis of qualitative data differs significantly from quantitative analysis, both the method of collecting data and synthesizing the results had to be treated from a different perspective.

Research Methodology

Seven people were selected for the task force, three from administration and four from the faculty. The first three days were spent in exchanging information and prioritizing topics, and noting issues which should be discussed. A list of 13 major topics was agreed upon. These were: access to computing facilities; communications/networking; computer literacy; consulting/training; faculty rewards; hardware; instruction; policy/organization; public service; security; software and standards. Each topic was discussed and minor issues/problems within the areas were noted. This process permitted the free interchange of ideas and viewpoints.

Using these 13 topics it was relatively easy to isolate those issues which were of major importance to all users and to insure that these were included in a questionnaire. This instrument was distributed to 70 faculty and staff who were considered representative. Users ranged from the novice to the "guru" status.

Each respondent completed a questionnaire composed of 14 questions (see Table 1), thirteen of which were open-ended.

In addition to completing the questionnaire, the subjects were asked to participate in an hour-long interview. The 70 subjects were interviewed in eighteen groups, each consisting of three or four people. Generally all members of the group were from the same or related disciplines. The task force scheduled four or five interviews a day; task force members were present at all interviews. The interviews were designed to be loosely structured and wide-ranging in nature, with the possibility that the interviewees could concentrate attention on a particular topic or experience important to that interviewee. This approach provoked greater spontaneity and candor of discussion than an individual interview since the group members interacted each other. Conducting the interviews in a four-day period was an intensive process, but resulted in a better consensus of opinion among task force.

At the end of each interview a half-hour was devoted to a review of notes taken by a task force member; this activity (using transparencies) was rotated. The review assured that the notes were checked for accuracy while the comments were still fresh in the mind.

After the interviews were completed, the task force was divided into two teams. One team analyzed information obtained from interviewing groups one through nine. The second team paralleled the process for the remaining nine interviewed groups. Designated as belonging to the 19th group were questionnaires submitted by members of the study team .

Classification and Analysis of Data

Once the data were collected from the questionnaires and interviews, the study team performed an analysis to determine the nature of the items discussed and the persistence or frequency of these issues. Since the data were in qualitative form the analysis involved classifying the information into a coherent format and then making the necessary comparisons to produce generalizations.

For the classification of qualitative data, Lazarsfeld (1972, pp. 226-7) suggests that four criteria should be followed: articulation, logical correctness, adaptation to the structure of the situation and adaptation to the respondent's frame of reference. In the articulation process, responses are placed into major categories rather than in long lists. The aspect of logical correctness is needed to ensure that the categories are exhaustive and mutually exclusive. The categories should be based on a comprehensive outline or a structure which is correct for the problem under investigation. Last, the classification must represent clearly the respondents' replies taking into consideration that most of those interviewed will not have discussed all the issues.

The Glaser and Strauss constant comparative method was selected because of its ability to analyze qualitative data. It consists of four stages (1967, pp. 105-113). First, incidents are compared to determine if they are applicable to the category; when necessary a new category is defined. As the data were quantified and integrated, some of the original categories were collapsed and others added. Second, categories are integrated through a constant comparison with previous coding; the link between theoretically related items becomes apparent to the researchers. Third, the researchers examine the categories for underlying uniformities and generalizations. Fourth and last, the researchers use the categories to determine major emphases and complete the process of synthesizing.

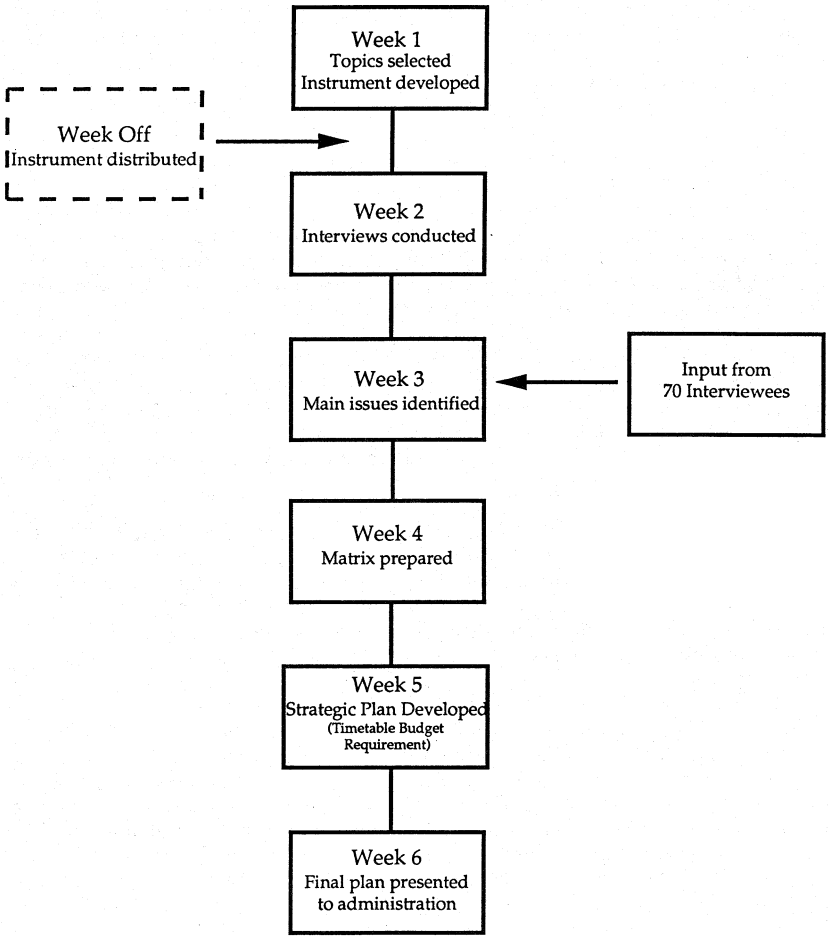
As described by Glaser and Strauss (1967, p. 107), the process of placing the data into categories resulted in two types of coding. In the first type the task force designated the appropriate category based on experiences of the members themselves. In the second type, the appropriate category was abstracted from the language of the respondent because of the uniqueness of the comments.

The method used by Glasner and Strauss stresses the discovery of theory from data. Concepts are worked out in relation to data during the research using systematic choice and a study of several comparison groups (p. 9). After the analysis and synthesis was completed a matrix was derived (Table 2); it listed the major categories and the subcategories within each category. For the sake of brevity only the two top subcategories are listed in the table. The matrix indicates how many individuals in each group addressed the specific issue, thus giving some idea of the impact. It was this matrix which became the basis for the strategic plan. For example, the top two ratings under Consulting/Training were hardware/software support and short courses/seminars which resulted in the request for enlarged user services in the strategic plan.

The matrix made it easier to devise a five year plan which reflected the needs of all users. This plan, which also was in a matrix format, showed the dollar amounts needed for each item as well as a status (e.g. initiate, continue, update etc.) The plan, plus supporting documentation, were presented to the administration and later were circulated to all 70 respondents.

The whole process, from selecting the Task Force to presenting the final plan, was completed over a seven week time span. Figure 1 illustrates the steps in the process and briefly indicates the tasks completed in each week. The Task Force was active for only six of the seven weeks since there was a one-week hiatus for the distribution and completion of the survey instrument by the 70 interviewees. During each of the six weeks the members spent a minimum of 30 hours together. The work load was intensive, but the time constraint insured that the strategic plan would be completed by a specified date and that it would have the quality of immediacy.

Figure 1 - Timetable of Process



Recommendations and Conclusions

Chaffee (1984, pp. 212-13) discusses two models for strategic planning. The adaptive model is one in which the organization is "an entity with its own goals and coherent, goal-directed actions". The opposite of this model is the interpretive one in which the organization "is a network of participants who use their association to pursue individual goals." Chaffee found the latter model was more successful. Although the method used at UTC was not designed to fit either of the Chaffee models, it does correspond more closely to the interpretive approach.

Anyone attempting to use a qualitative rather than a quantitative methodology should be aware that the former is time-consuming and requires extensive coordination and dedication of personnel. For example, each of the seven members of the task force spent a minimum of 150 hours in meetings or interviews; writing and editing material for the final report required an additional 20 hours. However, the method produced a strategic plan which reflected not only the perceived needs of the administration but also those of faculty and staff.

The fact that users were asked for their opinions made them more supportive of the strategic planning process. As Coughlin (1987) notes the concept of collegiality is an important component of strategic planning within the higher education environment. Sibley (1986) echoes this idea by stating that management of change should start with the rebuilding of a sense of community in the institution in order to prepare for later collaborative efforts.

The idea that strategic planning may be an acceptable alternative for educational institutions is not a new one. According to Nielsen (1983) many universities have seen the usefulness of the formal strategic planning process in overcoming an uncertain environment. As early as 1983 Keller, an expert in higher education, advocated the use of strategic planning. A search of literature shows that a number of universities, of varying sizes have realized the usefulness of strategic planning as a tool.

While the user input and content analysis process was used within an academic environment, there is no reason why the methodology would not work equally well in a business or industrial environment. Recent literature (Levinson, 1989) indicates that "campuses are becoming more like each other, and more like corporate bureaucracies." Users, whether in higher education or business, usually have needs which they feel are not being meet. By being part of strategic planning process these users become involved, and more readily ascribe to the recommendations associated with the completed plan.

TABLE 1

SURVEY FOR THE ACADEMIC COMPUTING STRATEGIC TASK FORCE

This survey is intended to develop a picture of the long term needs for computing at UTC. It will be used to help construct a 3 - 5 year strategy for the computer area. Answer the questions both in the light of your needs and the needs of colleagues in your discipline and of your students. Please use additional sheets to answer these questions.

1. Please provide the following demographic information:
   a) Self-rating - Are you a heavy, average, occasional or non-user of computers?
   b) Do you own a personal computer? If so, what type?
   c) How long have you used computers?
   d) What is your academic unit?

2. What long term changes would you make to improve access to mainframe computers and/or micro clusters? What additional types of computer access to the library would be most helpful?

3. What computer resource improvements and additions (hardware, peripherals, software) are critical for: a) Your research? b) Classroom teaching?

4. What types and levels of training and troubleshooting should be provided to faculty and staff by the UTC computer staff?

5. What would you consider to be appropriate university standards for the purchase of software and hardware (particularly PC)? If appropriate, at what level should standardization occur? Department? College/School?University?

6. What should our policies and procedures be on acquiring, maintaining and replacing hardware and software?

7. What should be the campus objectives for electronic telecommunications including electronic mail?

8. How can new technologies such as CD-ROM, Videodisc and Hyper Text be identified and incorporated into the curriculum?

9. How can computer usage be best integrated into the curriculum across the campus?

10. What rewards are required to encourage faculty development of courseware and faculty incorporation of innovative computer use into instruction?

11. What public services involving the computer are on-going but under-publicized at UTC? What are appropriate directions for future computer-based public service efforts?

12. In order to maximize available hours of access, what are appropriate security arrangements for our micro computer labs, clusters, and their software?

13. What single change in your computing environment would most encourage your productivity?

14. What other comments would you like to make on the academic computing system at UTC and/or on a 3 - 5 year computer development strategy?

TABLE 2
MATRIX OF ANALYSIS OF RESPONSES

	Pct.	GROUP																		Pct
ACCESS	Grps	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	Sum	Resp
Micro/terminal in every office	79		1	2	1		1	1		5	2		3	2	2	1	1	1	2	7	32	45
More evening/weekend access	74	3	1	1			3	1	3	3	1		2	1	1	4			1	5	30	42
COMMUNICATIONS/NETWORK'G																						
Campus-wide network	84	3	1	3		4	4	5	2	4	1			1	4	3	1	4	1	7	47	66
Email in each office	90		1		2	2	2	1	1	3	2	1	3	3	4	2	2	1	1	5	36	51
COMPUTER LITERACY																						
Computers across curriculum	39	1					1			2			1			1			1	3	10	14
Separate short course (students)	26			2	1		1							1						3	8	11
CONSULTING/TRAINING																						
Hardware/software support	95	1	2		1	1	3	3	3	4	1	2	2	3	1	4	4	3	2	6	46	65
Short courses/seminars	79	3	2	4	2		2	4	4	4			1		1	1	1	2	2	7	40	56
FACULTY SUPPORT																						
Released time	90	3	2	2	3	1	2	3	4	2	3	1	3	1	2	3	3		2	5	45	63
EDO recognition (softw. publication)	74	1	1	2				1	2		2		3	3	3	3	3	2	1	4	31	44
INSTRUCTION																						
Electronic classroom facilities	74	2			3	2		2	3	4	2			3	1	3	3	1	1	5	35	49
Access to new instructional tech.	47	2	1	1		1	1		1		2						1		1		11	16
HARDWARE/EQUIPMENT																						
More printers/plotters	74			2		1	1		2	2		2	3	1	3	2	1	1	1	4	26	37
State of the art equipment	58	1		1		1		1	1				1	1		2	2		1	2	14	20
LIBRARY																						
End user search (CD-ROM, online DB)	79		3	3	1	2		1		4	3	1	2	1	4	1	1		1	3	31	44
More ports to library	42				1	1	1						3		2		1		2	3	14	20
POLICY AND ORGANIZATION																						
Plan for obsolescence	74	1		2			1	1	1			1	4	2	1	3	1	1	1	7	27	38
Separate computer budget line	68		2	2	1	1			1	2	1		1	1		1	1		3	3	20	28
PUBLIC SERVICE																						
More public access to computers	32				1		1			1			2		2	1					8	11
More media cov. of acad. computing	26	1														1	1		1	2	6	9
SECURITY																						
Staff each micro cluster	95		1	1	3	1	2	2	2	3	1	1	2	1	1	2	1	2	2	5	33	47
Lock down micros	37		2				2				1				1			1	1	3	11	16
STANDARDS																						
Support selected configurations	58		1	1	4	3	4	2	1	1							1		1	3	22	31
Department sets standards	58	1	2	2					2	1	2				2		2	3	2	1	20	28
SOFTWARE																						
Software review mechanism	47				1			1			1			2	2	1	1	2		2	13	18
More micro pkgs. from university	47	2	1	3					1				1	1		1	1		1		12	17

* Note: Only two top-rated items in each category are shown.

Bibliography

Chaffee, E. E. (1984). Successful strategic management in small private colleges. Journal of Higher Education, 55 (2), 212-41.

Coughlin, P. J. (1987). Computing Strategies in small colleges and universities. Business Officer, 21 (2), 16-18.

Filstead, W. J. (1970). Qualitative methodology: firsthand involvement with the social world. Chicago: Markham Publishing Company.

Glaser, B. G & Strauss, A. L. (1967). The discovery of grounded theory. Chicago: Aldine Publishing Company.

Keller, G. (1983). A change in plans: the view from four colleges. Change, 15 (2), 34-43.

Lazarsfeld, P. F. (1972). Qualitative analysis: historical and critical essays. Boston: Allyn and Bacon, Inc.

Levinson, R. M. (1989). The faculty and institutional isomorphism. Academe, 75 (1), 21-25.

Morrell, L. R. (1986). Strategic planning for colleges and universities. College Board Review, 140, 20-23, 36.

Nielsen, R. P. (1983). Socioeconomic trends: strategic management implications for higher education institutions.' College Board Review, 129, 23-25, 31-32.

Sibley, W. M. (1986). Strategic planning and management for change. Canadian Journal of Higher Education, 16 (2), 81-102.

THE FORGOTTEN CONNECTION: THE HP3000 AND LASERJET PRINTERS

By Richard J. Armitage and William C. Tuminaro
Business Systems International, Inc.
20942 Osborne Street
Canoga Park, California
(818) 998-7227

INTRODUCTION

When Hewlett-Packard released the first LaserJet printer in 1985, they set in motion a series of events which have significantly changed the expectation for computer printed output. Quality, speed, quietness, reliability, and flexibility are just a few of the adjectives commonly used to describe the LaserJet family of printers. With over one million printers sold to date, the LaserJet and PCL (the printer's control language) have set a standard for the industry to follow.

That's the good news. Now for the not so good news.

It has become increasingly clear that the Boise Division of Hewlett-Packard, which is responsible for the development and production of the LaserJet family, is primarily interested in the PC marketplace. The resulting lack of support for the HP3000 user has prompted many data processing professionals to avoid or at least postpone using LaserJet printers with their HP3000's. This is a mistake. Actually, desk top LaserJets (the Series II and IID) are well suited for use as department printers with HP3000 based applications. In addition, the availability of higher speed printers (currently up to 26 pages per minute) and the promise of an HPIB interface makes the LaserJet (or similar PCL compatible printers) a practical system printer in many situations. There are, however, a number of technical issues which must be considered and resolved if the LaserJet is to be an effective and reliable output device for the HP3000 user.

In this presentation, we will discuss the practical aspects of using desk top laser printers with the HP3000. We will focus on technical considerations involved in using these printers with the HP3000 for printing standard computer reports, electronic forms, and other specialize output.

LASERJET CAPABILITIES

First, let's review some of the special capabilities of the LaserJet Plus family of printers.

1. Print Quality

The LaserJet prints at 300 x 300 dots per inch resolution which comes very close in quality to typeset documents. It handles both graphic images and text on a single page, and allows a variety of fonts to be used on a page.

2. Fonts

The LaserJet printer offers an almost unlimited variety of fonts, with both fixed and proportional spacing. The available point sizes range from as small as 4 point to over 48 point.

Fonts are available as either resident fonts which are permanently stored in the ROM of the printer, cartridge fonts which plug into cartridge slots in the printer, or soft fonts which are stored on the host computer and can be downloaded into the printer's RAM memory.

3. Programming

The LaserJet provides a control language which can be used to communicate from the host computer application to the printer in order to access the printer features. Hewlett-Packard's Printer Command Language (PCL Level 4) has become a laser printing standard. PCL commands are also referred to as escape sequences. These commands can control cursor movement, font selection, line drawing, definition and placement of graphic images, etc.

4. Memory and Storage

The laser printer has its own memory which, in addition to buffering input and building a page image, is used to hold fonts and macro files which contain PCL commands which direct the printer. The advantage is that once fonts and form macros are downloaded and stored in the printer's memory they are available for use throughout the day. Thus, in most situations, only data has to be transmitted to the printer during a production job.

ADVANTAGES IN SPOOLING A LASERJET OFF THE HP3000

The LaserJet offers a number of definite advantages as a spooled device off the HP3000. You can put a LaserJet Series II into a user department for less than $2,000. The printer is equivalent to a 300 line per minute printer, yet it is quiet, easy to use, and extremely durable. With the LaserJet IID you get the added benefit of additional paper trays and duplex printing.

Some of the advantages to using these printers with the HP3000 are:

1. User reports can be printed on 8.5 x 11 inch (people size) paper in the user department. The users get their reports in a more timely manner, and data processing avoids the hassle of report distribution.

2. Printing in the user department also provides a level of security for sensitive information.

3. With a laser printer you can replace preprinted forms with electronic forms. Again, the forms can be printed on demand in the user department since form mounting and alignment is not necessary. Elimination of preprinted forms is probably the most important benefit of laser printing. The use of traditional preprinted forms is costly and extremely inefficient. The benefits associated with using electronic forms include reduced cost, better control, less work, and of course you never run out of an electronic form.

4. With a laser printer in the user department you can set up your systems to print one or two forms on demand instead of batching all of the output for overnight processing.

5. A spooled laser printer can be used as a shared printer for PC users with any of the currently available PC-to-HP3000 file transfer or print redirection software.

Overall, when you provide a spooled LaserJet for your users, they are happy and data processing operations has less involvement in user printing.

TECHNICAL ISSUES IN CONNECTING LASERJET TO HP3000

There are a number of issues which must be resolved when you are connecting any output device to the HP3000. Because of its unique capabilities, the LaserJet requires additional planning and consideration.

1. Physical Cabling

The physical connection between the HP3000 and the laser printer is the first issue to be determined. Generally, HP only supports an asynchronous or serial interface between the HP3000 and a LaserJet printer. This means using ATP or ADCC controllers. Recently, a number of third party vendors have been offering an HPIB protocol converter, which allows you to connect the LaserJet's parallel (centronics) interface to an HPIB port off the HP3000. Generally, speed of transmission and distance between printer and CPU are the primary factory in determining which method will best serve your requirements.

ADCC and ATP are serial connections. This means that data is sent one bit at a time. It is slow compared with HPIB. ADCC supports transmission speeds up to 9600 baud, and ATP supports a baud rate up to 19,200. With an RS-422 connection the distance from the CPU to the printer may be as much as 4,000 feet. With RS-232C the stated limit is 50 feet.

HPIB is the traditional high speed peripheral device connection for HP. It is a parallel interface. The increased data transmission speed of an HPIB interface is essential for printers with a rated speed of 20 ppm or more. Also, the higher transmission speed significantly reduces the time required to download fonts and electronic forms. The distance from the CPU to the printer, however, is limited to approximately 10 feet.

2. Configuring the Printer

Printer configuration is one of the most confusing technical areas in using laser printers with the HP3000. Approximately 20 percent of the HP3000 Application Notes deal with printers and much of the information on laser printers is confusing and/or contradictory.

TECHNICAL ISSUES (continued)

If you are using a serial connection, you have to select the terminal type which is best suited to your requirements. Generally, TTPCL18, TTPCL19, TTPCL22 and TTPCL26 are recommended for the LaserJet.

In most cases where the printer is directly connected to the HP3000 we use TTPCL18. This does not support status checking, but does use a simple XON/XOFF data stream handshake. The MPE spooler sends data to the printer until an XOFF is received. The only problem arises is if the printer is physically turned off when data is being sent. MPE does not detect this condition and data is lost. However, as long as the printer is on, data is only sent when the printer is ready to receive. XOFF's are transmitted whenever the I/O buffer is full, the printer is OFF-LINE (for example, out of paper, or a paper jam), or the printer is BUSY (such as performing SELF-TEST).

If your printer supports HP status checking you can use one of the other terminal types. These provide for the CPU to send a status check to the printer prior to sending data. At the present time the LaserJet IID is the only member of the LaserJet family which does **not** support status checking. Most of the non-HP PCL printers do not support status checking either.

3. HPIB

The lack of support for an HPIB interface is generally not a problem for users of the 8 or 12 page per minute desk top laser printers, since the asychronous interface transmits data fast enough to allow the printer to operate at its rated speed. However, the asynchronous interface will not transmit data fast enough to support printing at over 20 pages per minute.

HPIB offers not only the advantage of higher data transmission speeds, but also less CPU load since data is transmitted in blocks. A number of third party vendors are now offering protocol converters designed to support connecting the LaserJet to the HP3000's HPIB interface. This is not supported by HP at the present time. In our early testing we discovered that a number of lower level PCL commands generated by the I/O driver were acted on properly by the LaserJet, but were not handled properly by non-HP printers. Adequate support for a high speed interface is essential if the LaserJet is to function as a system printer. We are confident that reliable protocol converters to support HPIB will be available in the near future.

USING TERMINAL TYPE FILES

Hewlett-Packard's terminal type files are a valuable aid in using laser printers connected to the HP3000 through an asynchronous interface. These files allow you to send a sequence of PCL commands to the printer in the same spool file with your print data.

The ability to include printer set-up commands within a single spool file is important because in most spooled printer environments a reset is automatically sent to the printer at the beginning of each spool file. The reset returns the printer to its default environment. Thus, if your application requires you to activate a form macro for overlay, or to set the printer to a specific mode (landscape mode for example), the escape sequences to accomplish these must be sent to the printer before your data. However, if your output data is sent to the laser printer in a separate spool file from the set-up commands, the printer will be reset, and your data will be printed with the default environment.

With terminal type files you can implement laser printed output without modifying your application. Once created, these files can be accessed by using the ENV option of the FILE command.

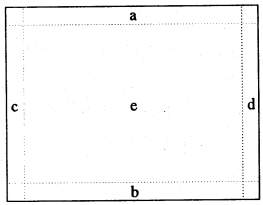
Terminal type files for the classic HP3000 can be created with Workstation Configurator as well as third party packages. At the current time traditional terminal type files are not available on the Spectrum. In addition, the more sophisticated environment files used with an HPIB interface are not directly available to PCL printers.

PRINTING STANDARD COMPUTER REPORTS WITH THE LASERJET

Printing standard computer reports on 8.5 x 11 inch paper with a laser printer is one of the nicest things you can do for management and other users. To do it properly it does, however, require a little work. You can of course print everything in the default environment and avoid the work. But actually it does not take much effort to customize a print environment for each output report. We generally divide the process into three basic steps (illustrated with the attached chart).

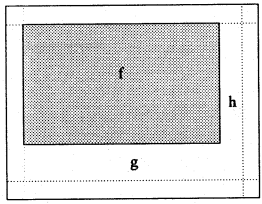
Flex-Set Page Layout Chart

Initial page layout parameters:



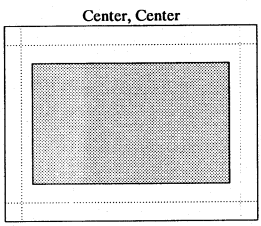
a = minimum top margin
b = minimum bottom margin
c = minimum left margin
d = minimum right margin
e = area where report data may print

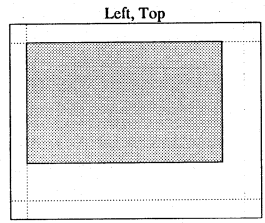
Data size based on selected font:

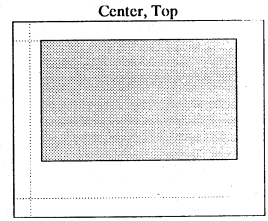


f = report data based on selected font's natural line spacing and pitch
g = additional vertical space not used
h = additional horizontal space not used

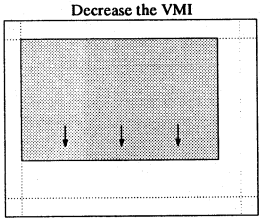
Examples of positioning data within available space:



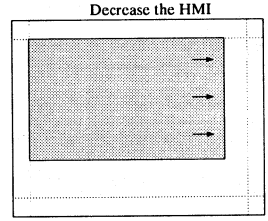




Expanding data to fill available spaces:



Decreasing the number of lines per inch causes the data to expand vertically, conversely, increasing it will compress the data.



Decreasing the number of characters per inch causes the data to expand horizontally, conversely, increasing the characters per inch will compress the data.

PRINTING STANDARD COMPUTER REPORTS (continued)

1. Define the page. This includes determining the maximum amount of data for a page (lines per page and characters per line) and the minimum margin requirements. With this information, plus the paper size and orientation, you can determine the physical size of the print region and the amount of data which must fit into that region.

2. Select a data font. Based on the amount of data to be printed on a page and the physical size of the print region, the minimum pitch and the maximum point size for the data font can be calculated.

3. Data alignment. Based on the font you select for the data, there may be an unused area on the page in addition to the minimum margins initially anticipated. The report data may be repositioned on the page by adjusting the margins, or you may modify the pitch (HMI) and line spacing (VMI) to expand your data.

In addition to the page orientation, paper size, margins, data font, and the line spacing, the LaserJet allows you to select duplex printing, number of copies, and a paper source. Coding all of these escape sequences yourself is not necessary. There are several third party laser printing utilities which will not only build these set-up strings for you, but also put the set-up stings into terminal type files, and download soft fonts.

REPLACING PREPRINTED FORMS WITH ELECTRONIC FORMS

The ability to use electronic forms is one of the greatest assets of the LaserJet. Viewed very basically, an electronic form is nothing more than a file containing instructions which allow a laser printer to print an image of the form. The instructions are stored in the host computer and sent to the laser printer's memory to be stored as a macro.

Using an electronic form with an HP3000 based application is essentially the same as using an impact printer and preprinted forms. First, the form must be available when the data arrives at the printer, and second, the positioning of the data on the form is handled by the computer application.

REPLACING PREPRINTED FORMS WITH ELECTRONIC FORMS (continued)

Actually, the elements needed for successful merging of data from an HP3000 application with an electronic form are:

1) The form to be filled in must be in the printer's memory.

2) Any fonts required by the form or the data must be available.

3) The form must be activated for overlay with the data.

4) The data from the host computer application with proper printer spacing must be available.

When and how the printer is prepared for data merging depends on your requirements. Again the steps that must be handled are: send the soft fonts to the printer, send the form to the printer, and activate the form.

One approach is to have your application handle all three steps. This can be accomplished in one of two ways. First, use a custom environment file which contains all of the soft fonts and the form required by the application, and have it included in the spool file with your data. The other method is to send the fonts and forms to the printer as separate job steps, and then activate the form for overlay either by a terminal type file with the escape sequence or by having your print program send the escape sequence prior to the data. Once a form is activated, it will be overlaid with each page of output data until another form is activated or until the printer is reset.

The advantage of this approach is reliability, that is, all required elements will be available when needed. The main disadvantage comes in if the application runs several times throughout the day or if the same fonts are used by several applications. In that situation, there is excessive overhead involved in sending fonts and form files to the printer each time the application runs.

REPLACING PREPRINTED FORMS WITH ELECTRONIC FORMS (continued)

Another approach for handling printer preparation in situations where forms and fonts are used on a routine basis is to store the soft fonts and the forms required by each of your applications in the printer's memory so that they are available whenever needed. The LaserJet Series II can store up to 32 macro files and the LaserJet IID and the LaserJet 2000 can store over 32,000 files. Under this approach you establish host computer job streams to download the fonts and forms required by each laser printer. The jobs are executed each time the printers are powered up. The fonts and forms stored in the printer's memory as permanent remain resident as long as the printer is powered up.

To work properly, the forms and fonts should be stored in a centralized library on the HP3000 and unique numbers assigned for downloading. This avoids conflicts between users and duplicate storage of the same fonts with different numbers.

HANDLING COMPLEX MULTIPLE PART FORMS

Multiple part forms are perhaps the chief challenge to electronic forms software. The laser printer allows you to print up to 99 copies of any output record, however, the copies are identical. In the traditional multi-part preprinted form set, each part may be slightly different or even completely different. Routing designation and areas of data excluded are examples of some typical differences. In addition, the back side of each copy could be different. While even the most complex multi-part form can be handled if you have the ability to do form switching from within your application, situations where a third party application cannot be modified are far more typical.

The challenge in the multiple part forms environment is to process a single output record to several copies of the same or different forms. Our approach is to create a customized environment file with the multiple part processing requirements. The multiple part form will actually reside in the printer memory as a single form overlay, however, each part may have a unique routing designation printed on it and be routed to a different department printer. In addition, duplex printing may be designated for some or all parts, and the contents of the back page may be different for each part. With this capability, the LaserJet will be able to produce forms which replace even the most complex preprinted form sets, and make forms the efficient tools they were intended to be.

CONCLUSION

The LaserJet printer has the potential to be an invaluable tool capable of assisting you in achieving cost savings and better efficiency within your company. The effort in resolving the technical issues involved in using it with the HP3000 is small compared to the benefits to be achieved. Happy users, quality output, more efficient operations, and the elimination of preprinted forms are all easily achieved when you put the LaserJet to work with your HP3000.

Putting Your Best "FEATS" Forward

by Diane Amos, CPC
Amos & Associates
633-B Chapel Hill Road
Burlington, NC 27215
919-222-0231

Today we're going to discuss all of the issues surrounding making a job change and hiring new employees. The process of changing jobs can be an emotional one for a candidate and should not be taken lightly. Certainly the hiring process is equally important to a company from a monetary as well as a long term investment standpoint. Therefore, I'm going to take you through the whole hiring process from beginning to end from both the company and the candidate's perspective.

I. The first issue to consider is simply **"Is It Time?"...** to look for a job or to hire a new employee. Frequently people jump into this phase too soon or before they've really considered what's important to them.

When thinking about making a job change, I strongly recommend that each prospective candidate sit down and do some "soul searching". Think about the issues involved:

-Why do I want to leave my present company?
-What have I enjoyed about my present company?
-What have I disliked that I don't want to repeat?
-What kind of environment am I looking for?
-What do I need to really make me happy in my next job?
Set your priorities! Is it money, training, advancement, stability/security, challenging work, location or better benefits that you're seeking?

Once you have identified the reasons you're seeking to leave, you will have created the path to follow to get you to the right place, company and position. Visualize what it is you want to do and then you'll be able to communicate that vision to those who can help you find it. More importantly, you won't make the mistake of jumping out of the frying pan into the fire and end up in a job you dislike. Confucious once said, "Choose a job you love and you will never have to work a day in your life".

"Soul searching" should also be done when you, the manager decide you need to hire someone, either through replacement or expansion. You want to sit down and address such issues as:

-What do I want this person to do?
-What kind of person will fit in with our existing group?
-What plans do I have for this new employee?

-Can this job be done by anyone I already have?
-What skills and experience am I looking for?
  1. absolute requirements
  2. "wish" list

Making a check list will help you define what you want. The old saying "if you don't know what you're looking for, then how are you going to know when you've found it"? is certainly appropriate in this situation.

II. The Phone Interview

Moving on, let's now assume that a company and candidate have linked up together, the company has read the candidate's resume and is interested in pursuing him. In today's HP marketplace, it has become commonplace to have a phone interview between interested parties prior to driving or flying in for an on-site interview. Due to the expense of long distance interviewing, this procedure makes sense in that it allows a company to weed out inappropriate candidates with a minimal amount of investment, and it also allows a candidate to explore opportunities in more detail without taking time off from work. It is a valuable hiring tool only if it is handled properly. Unfortunately, many excellent candidates have been "lost" due to a poorly conducted phone interview by the hiring official, and many great opportunities have been "missed" by candidates who didn't know how to handle a phone interview. There are some definite **DO'S** and **DON'TS** to conducting a good phone interview and I'll detail them as follows:

For the company....

-**DO** call when you say you will! The fastest turn off for a candidate is sitting by the phone waiting for a call that never comes.

-**DON'T** leave the interview time open. You might catch the candidate at the worst possible time (in an argument with his spouse or in the midst of a party or with the kids screaming in the background). Certainly the candidate will not be at his best for an interview. Arrange a convenient time for both parties and then follow through at the scheduled time.

-**DON'T** do all the talking. Draw the candidate out on his attitudes and goals as well as his technical qualifications. Lead a "give-and-take" conversation where both parties receive the information they need.

-**FOLLOW** a format such as I'm listing here so that you make the best use of your time. It helps to begin the phone interview with a brief description of your company to break the ice and also to be informative. Then begin your questioning from a general standpoint to get an overall

perspective of this candidate. Be descriptive about the position that you're interviewing him for, what he'll be doing and what challenges you have to offer. Then zero in on how he fits this job with particular technical questions relating to the position.

-**SELL** your company! I've had more candidates turned off by interviewers that were not excited **themselves** about their company or couldn't give good answers to the candidate. Be honest but enthusiastic and positive when discussing your company.

For The Candidate...

Unfortunately, this necessary process of phone interviewing often puts the candidate at a disadvantage. Some people just do not come across well on the phone, but may do perfectly fine on personal interviews. Some helpful hints to get the candidate "past" that phone screen are in order.

-**DO** show enthusiasm and interest in your voice. Watch that voice inflection and be careful not to sound monotone, indifferent or disinterested.

-**ASK** appropriate questions. Have a list next to the phone so that you sound prepared and get the information that you are seeking. Remember, you are interviewing the company as well and have a right to see how this position fits your needs and goals.

-**DON'T** ask about the salary and benefits. This gives the company an impression that your primary interest is dollars rather than opportunity. Salary and benefits should be discussed at the time of an offer. It is tactless, inappropriate and a real turnoff to a company when a candidate's first question is "What does the job pay?"

-**DESCRIBE** how your skills and experience fit the job. Relate stories about similar work that you performed at your present company and how you solved similar problems.

-**PROMOTE** your employability in terms of your attitudes and work ethic as well as your technical skills. Companies want to know what kind of employee you'll be, and they don't always know how to elicit that information over the phone.

-**BE HONEST** yet positive when discussing your skills and experience. You want a job that "fits" you, so bluffing your way through to a hire will only spell disaster later.

If you will use the "phone interview" to your advantage, eliciting all of the information you need while also selling yourself, you can explore many good job possibilities until you find the "right one" for you. Don't underestimate or

minimize the impact of this first step in hiring. It may lead you to that "dream job" you've always wanted.

III. Pre-Interview Preparation

The saying goes..."a little preparation goes a long way". This adage cannot be overstated! In the hiring process, great opportunities are lost time and again because the interviewee did not take the time to properly prepare for the interview. Preparation by the interviewer is also essential in order to attract those cream-of-the-crop candidates. Both the company and the candidate must do their "homework" to assure a successful meeting.

INTERVIEWER

READ and REREAD the candidate's resume so that you are familiar with the candidate, his experience and skills. Highlight areas where he matches and make notes about specific parts of his resume that you wish for him to elaborate on in the interview. A well prepared and organized hiring manager will certainly impress the candidate. At the same time, nothing turns a candidate off more than to be interviewed by a manager that hasn't read his resume or doesn't have it in front of him to refer to.

PREPARE YOUR STAFF! If you have several staff members interview the candidate, discuss how you want the interview process to be handled. Have your staff hold their calls and keep personal interruptions to a minimum. Remember to sell your company! Have your staff make the candidate feel comfortable. Copy the candidate's resume for all who will interview him. So often a group interview resembles a "Chinese fire-drill" with the various interviewers not knowing who the candidate is or why he's there. The same questions by five different people also can be a problem. By the fifth time, the candidate is getting irritated and may not answer well. Plan to have the different staff members focus on different areas of the candidate's background.

REVIEW THE TRAVEL ARRANGEMENTS and make sure there are no loop holes. Having a candidate arrive at a hotel or an airport with no reservation can be disastrous. Lay out an itinerary with names and titles of the interviewers. Send the interview itinerary, your application form and company literature ahead of time. Having the candidate fill out the application before the interview will save you 30 minutes of valuable interview time. The itinerary and literature will help to familiarize the candidate with your company and personnel. Make sure the candidate has the company phone number and possibly the interviewer's home number if he is

arriving the night before. This preventive action has saved the day on that occasion when flights are delayed or there's a problem on the road.

INTERVIEWEE

RESEARCH THE COMPANY! Go to the library and find out all you can about the firm, their products and philosophy of doing business. Run a Dun & Bradstreet on them, or ask your stock broker for an annual report, so that you can get an idea of their financial situation. In addition to the obvious payoff of not "operating in the dark", it will give you a comfortable feeling on the interview of being well-informed. Certainly, this will make you "stand out" in contrast to any competition that arrives less than prepared.

COMPOSE A LIST OF QUESTIONS that you want to ask at the interview and take this with you. It is appropriate to have a notebook with you to jot down information that you receive as well as referring to your list of important questions. This action will impress a company with your seriousness, thoroughness and forethought.

ROLE PLAY anticipated interview questions with your spouse, friend, or anyone who will give you honest and constructive criticism. There may be no right or wrong answer to an interview question, but if you are caught off guard you may stutter or hesitate, and not answer the question as well as if you were prepared.

DRESS TO IMPRESS! Plan what you are going to wear ahead of time making sure that everything matches and is neat and clean. If you do not have an appropriate "interview suit", go out and buy one! Too many offers have been lost SOLELY on presenting a poor professional image. A professional must look the part!

Geographics play a big part in this. Cowboy boots are fine for the Southwest, but are a real turn-off for the Eastern seaboard. Short sleeve shirts without a tie may be OK for Florida and California, but definitely a NO-NO for the rest of the country.

The safest and best policy is to dress conservatively so as not to draw attention to how you look. A "flashy" dresser may offend someone. Bottom line-you want the company to focus on your skills, not be distracted by your appearance. For men, a blue or grey suit, white shirt, conservative tie, and dress shoes are a safe bet. Studies show that brown and green can project a drab image and should be avoided. For women, a dark business suit complimented by a fresh, tailored, light colored blouse (no low neckline!) is appropriate. Pumps or closed heel-and-toe shoes is the best way to put your best foot forward. Jewelry should be

conservative, make-up lightly applied. Do not wear perfume or cologne-this goes for both men and women. It may be offensive to someone without your being aware of it. Remember, THE BETTER YOU LOOK, THE MORE YOU LOOK LIKE YOU ARE WORTH!

PREPARE TO BRING SAMPLES of your work if appropriate. This will provide companies a first hand look at your skills. It will also give you the edge over those candidates that did not do so.

IV. **The "Interview" - Company**

First off, an interview is not a battle of wits, a confrontation or an opportunity to put someone down. I would suggest to you that an interview is an exchange of information by two people, both having needs to fill. It is an opportunity to see if there is a match-up between your need to fill an open position and the candidate's need of a position. It's that simple.

1. With that in mind, the first thing that the company should be aware of in their interview of the candidate is THEIR "first" impression. Lay out the red carpet and DON'T keep the candidate waiting. Schedule your time and make sure that you are prompt.

2. A good way to begin an interview (especially when a candidate is from out of town) is to meet him for breakfast at the hotel to break the ice. This sets a nice tone, and gives you a chance to get to know the candidate's personality in a neutral setting. My clients have often told me that this valuable time really told them what they needed to know about the candidate, and whether he would "fit in" with their organization.

3. Make sure your receptionist or secretary greets the candidate courteously and knows why he is there. There's nothing more demeaning for a candidate than to hear "who are you and why are you here?" But if the receptionist says instead, "make yourself at home, Bill is expecting you. Can I get you a cup of coffee?", the candidate will relax and the stage will be set for a productive interview.

4. Don't start out your valuable interview with Personnel! This could be tragic. I've seen too many interviews where the candidate went in all pumped up from an effective phone interview only to be totally turned off by personnel and walk out not interested in the job. Leave all of those nit-picky items like "how much your insurance will cost you" till the end. You want to excite the candidate in the job opportunity and the staff he'll be working with. So start off the interview with the best first!

5. The purpose of the interview from a company standpoint should be to acquire as much information about a candidate as possible. In this effort, the **means** to get to this **end** is to put the candidate at ease. If he is relaxed, he will open up and tell you what you need to know. If he's not, he'll clam up and you may miss hiring a good employee. Keeping this in mind, your friendly, courteous and non-stressful manner will give you the results you desire.

Now, assuming that you've done all the preparation ahead of time, let's get down to business and interview your candidate.

What do you want to LEARN in this interview? Basically three things:

-Does the candidate have the necessary skills for the job?
-Does the candidate have the proper attitude and motivation?
-Does the candidate seem to "fit in" with the group?

Lead a balanced give-and-take interview and ask the right questions.

SKILLS

If you are not a "technical manager", be sure to have your most technical employee share in the interview process.

1. Ask the candidate about the depth of his skill level in the necessary areas.
2. Have him demonstrate how he used that skill in a given project that he was involved in.
3. Ask to see a sample of his code if he is a programmer.
4. Have him describe the roles that he played in the various projects he worked on.
5. Ask him about his technical accomplishments.
6. Have him describe his work history in his own words, not just relying on his resume.
7. Ask him which functions he enjoys the most in his job and which ones he dislikes.
8. Ask him what skills he can bring to the company that he feels are important.

ATTITUDE AND MOTIVATION

These questions are sometimes the more difficult ones for DP Managers who have not been trained in interviewing techniques. It takes some intuition and "gut feel", but asking the necessary questions can help bring out what you need.

1. Why do you want to change jobs?
2. What causes you to lose your temper?

3. What kind of decisions are most difficult for you?
4. What was your best boss like and why? Worst boss?
5. Have you done the best work of which you are capable?
6. How do you feel about your progress to date?
7. What did you like/dislike the most about your former job?
8. Do you have plans to continue your studies?
9. What do you want out of your work?
10. What would you like to be doing 5 years from now?
11. How long will you stay with this company?
12. Why should I hire you?

FIT IN?

To really know this about a person, it is of course necessary to truly understand your staff and their personalities. If you have a good handle on your existing staff, you'll have a standard and gauge to go by in determining how this candidate will "fit in" with your group. You want to find out whether his goals and aspirations are compatible with yours. Some helpful questions..

1. How do you spend your spare time? Hobbies?
2. Why do you want to relocate here? or how do you feel about relocating here?
3. How do you see yourself fitting in with this company?
4. Why do you want to work here?
5. How hard would you work to get what you want out of your job?
6. What are your long and short term career goals?

You've asked the necessary questions to learn about this candidate. What is it that you want to TELL the candidate?

-Why your company is a great place to work.
-Why this job will offer him what he is seeking.
-What this job is all about: A CLEAR AND THOROUGH JOB DESCRIPTION.

GREAT PLACE TO WORK

If your company has literature, be sure to give as much of this to the candidate as you can. Annual reports, employee newsletters, benefits summaries, relocation packages, etc. will give him the necessary information he needs to make his decision. More importantly, it will impress the candidate with the caliber of your company and stand you apart from the competition.

Have the candidate talk to your most enthusiastic employees and let **them** sell the company for you. If they talk about what a great boss you are, and how much fun it is to work there, their excitement will be contagious.

Be sure to point out the little "extras" that your company

does for it's employees like performance bonuses, college scholarships for employee's children, excellent profit sharing plan, etc. Whatever you think is "out of the ordinary" for a company to do, don't neglect to tell it! By "selling" your company, I do not mean "misrepresent". By all means, be honest! Make the goals of the company clear so that the candidate can assess if they fit his goals as well. Be upfront about any major problems like turnover, possible company mergers, etc. and then discuss how it's being handled. You don't want surprises after the fact because that would only serve to add to your turnover. Lay your cards out on the table but in a positive way.

FIT FOR HIS CAREER GOALS

Describe the exciting projects that the candidate will be working on immediately and what the future holds as well. Let him see his career path if he proves to be an excellent employee. Explain the salary plan and any performance review system you have to determine promotions. State your expectations of job performance.

JOB DESCRIPTION

Show the candidate how his skills will be utilized in this job. What assets he has that will fit your needs. Describe a typical day, a typical work week. Prioritize the job functions: what are the three most important things he'll be doing? Give percentages to each job function to show how they interrelate, ie. 30% design, 30% analysis, 40% programming.

*REMEMBER, even if you do not extend an offer to the candidate, you want him to leave wishing you had. He will spread the word about his impression of you and your company. A positive image is contagious but a negative one spreads like wildfire. Presenting a positive public relations image of your company should be kept in mind at all times, because it can affect your ability to attract top talent in the future. Your job as "interviewer" is to create that positive atmosphere where a balanced give-and-take interview can take place.

V. The "Interview" - Candidate

You're on the way to the interview. What's the first thing to remember to do?

1. Make a good "first" impression! There is never a second chance to give a good first impression. Why work so hard to make a good impression? Shouldn't it be what you know

and what you do that count, not how bad or how good you look? The reality is that we all judge people on the "image" they present, and research has shown that most employers make up their minds about job applicants in the first 30 seconds. So make that first impression the one you intend. Basically you want to do three things: establish eye contact, smile and extend your hand in a firm handshake.

2. Relax! Take a deep breath and go into the interview with the intention of enjoying yourself. Let your personality shine through so that the company will know "who" they're hiring. Don't be afraid to show a bit of your sense of humor (at an appropriate time.) Be sincere and be yourself! (Caution: do not relax so much that you come across as flippant or not serious.)

3. Make a special effort to be friendly and polite to the receptionist or secretary. You better believe that the manager's secretary has some input in the decision making process!

4. Show a high energy level! Companies want to see an alertness, awareness and eagerness. Walk briskly, sit up straight and don't slouch. Watch your body language, especially your eye contact!

5. Demonstrate your enthusiasm! Companies want to see an interest on your part. I've seen too many offers given to the one who "wanted it most" over the more technically qualified.

6. Be positive about everything! Don't raise the "red flag" by sticking your foot in your mouth and expressing a negative about something the company is doing. They'll forget all of the good points you've made and remember that zinger! Keep your mouth shut until you've had some time to think the issue through. After a night's sleep, it may seem insignificant to you and not such a big deal. If it is an important issue, the time to address it and "work it out" is when they make the offer. Once they are committed to hiring you, you have a better chance of working out a concern.

7. Sell yourself! The key is how you "fit" the job and how your skills and experience will benefit the company. Don't try to impress them just for the sake of impressing. This is a turn-off to interviewers. Promote yourself in terms of what you can do <u>for</u> the company.

8. Be honest! Interviewers want prospective employees to be honest with themselves and truthful in presenting their background. They expect to hear about the lessons learned from career disappointments and failures, as well

as from successes. Above all, they seek people with integrity and self-insight. If you get hired based on misrepresentation, disaster might await you.

9. Talk about yourself in terms of what kind of employee you'll be, your motivation, initiative and attitude. I find that technical interviews can often bog down in the technical lingo and the interviewer never really learns about what kind of person you are. Remember, he may not be a skilled interviewer and know how to ask these questions. So it's up to you to bring out this information.

10. Talk about your goals! What life values must your work satisfy? Employers are looking for people who derive satisfaction from both work and a paycheck.

11. Make the company aware that you've done your homework and researched them thoroughly. Interviewers like candidates that are knowledgeable about the organization and are prepared to screen the employer in a two-way conversation.

12. Help the interviewer create a dialogue. Few hiring people, other than personnel people or recruiters, are trained in conducting an interview. Somehow managers are just magically supposed to inherit the art of interviewing along with the title of manager. So do your part. Strive for a balanced exchange of information. The interviewer expects candidates to hold up their side of the conversation by asking questions and giving complete answers. Employers want concise, but full responses, especially to open-ended questions such as "Tell me what you were able to accomplish at ABC". A flat "yes" or "no" is not the way an alert interviewee responds to questions if he want to interact as an equal with the interviewer. Add details, stories and examples to play a co-equal role in the interview.

13. Ask personal questions to get the interviewers talking about themselves and you'll learn much more about the personalities, philosophy and "culture" of the company. Appropriate questions might be: "How long have you been with the company? What is your next step up in this firm? Are the people above you receptive to your ideas? What do you like or dislike the most about this company?" Remember, you are there to interview the company as well. They'll be impressed that you asked, and more importantly you'll get additional information on which to base your decision.

14. Screen the employer technically. Detailed technical questions should be directed to the immediate supervisor or manager. You should ask pointed questions about the

scope of the job, the performance of the department, the philosophy of the organization, how you are expected to interact with others, the turnover in the department and problems or challenges you will be expected to solve. Ask to talk with others (co-workers, other supervisors) to see how they view the employer. You have a responsibility to yourself to size up the employer and organization, as well as make a favorable personal impact.

15. Don't try to solve the company's technical problems as they tour you through the plant. You may give wrong answers because you have not studied the problems. You can say "I don't like to solve problems off the top of my head but prefer time to study the situation. However, when faced with a similar problem at my old company, we did this..." Then tell them a success story!

16. Be **specific** in discussing your experience. Too many candidates generalize their background without telling exactly what they did, what they accomplished or what role they played in a project. Few give details of how successful projects were completed. Remember to relate how this experience "fits" their needs.

17. Handle the salary issue tactfully. I strongly recommend that a candidate not name a figure when asked what they want. If they're working with a recruiter, it's their job to handle the negotiations. More importantly, if you name a figure, it may be less than what they were considering and you'll be stuck with it. Or you may blow it out of the water by overpricing yourself before they've really considered you seriously. Simply state what you are currently making and say "I will give serious consideration to any reasonable offer." My strong opinion is that a company should make an offer based on what the candidate is bringing to them in terms of skills and experience, and not make an arbitrary offer based on a certain percentage over the current salary. With this philosophy in mind, promoting your skills and experience properly at the interview will get you the best offer!

18. End the interview on a positive note. Let the interviewer know that you want the job! Say something like "I'm really excited about what you're doing here, and I'd like to be a part of your team. Based on the interview today, when shall I expect to hear from you?" Offer to get back in touch with them to see if more information would be helpful in reaching the hiring decision.

19. Remember your thank-you letter to the interviewer. It's

not only a polite gesture, it also gives you a chance to point out your interest in the position and highlight why you make a strong candidate.

20. Immediately do a post-interview evaluation of your performance. Check where you shined, where the fit was tight and where it was loose. Critique how you can better answer problem questions. This makes every interview a learning experience and you a better candidate in the future.

I've also included in my handouts some of the pet peeves that I've gathered from interviewers in my 14 years of recruiting.

Interviewers Pet Peeves

-Candidates who talk too much and don't listen enough
-Nervous mannerisms such as smoking or excessive fiddling with hair
-Little or no pre-thought to what the job requires and what the employer does
-Candidates who don't know what skills they have to offer
-Apologetic, tentative answers
-Overbearing, cocky, conceited, know-it-all attitude
-Overemphasis on money, benefits and vacation - interested only in best dollar offer
-Unwilling to start at the bottom-expects too much too soon
-Gum chewers
-Improper dress
-Lack of common sense: telling dirty jokes on the interview, showing impatience, putting foot in mouth

To summarize my interview advice for the candidate: prepare thoroughly, communicate directly, ask penetrating questions, demonstrate how you fit the organization and show self-confidence.

VI. The Offer

The Company:

Timeliness is everything. I've seen too many companies "lose" the candidate they really wanted because they dragged their feet. If the "right" candidate is the first one you interview, and you **know** he's right, make the offer! Good "fits" and excellent candidates don't grow on trees. In the competitive HP marketplace, you can't afford the luxury of interviewing several candidates over a three week time period and then wait two more weeks to make the offer. If you wait around hoping to have "several to choose from", your perfect fit may get away. Evaluate the candidate on his individual merit, and then rule him in or rule him out. Trust your instinct and if you've given a thorough interview, you should

be able to say "yes" or "no" after the interview, not "maybe". Be decisive, make the offer or turn the candidate down.

When it's time to make the offer, be sure to deliver the offer yourself or if you're using a recruiter, benefit from their position as a third party. DO NOT let personnel deliver your valuable offer. They may not be able to answer any questions the candidate may have about the position, responsibilities or his role in your department. And most of all they can't generate the enthusiasm that you want to convey to him. You want him to know that you're excited about him joining your staff, and only **you** can project that. Call him at his home so that you can have a thorough and candid discussion. Be precise when you make the offer indicating to him how he "fits" the job, re-confirming the functions and title, and then describing the full package that your are offering: base salary, bonus(?), benefits and relocation.

In trying to determine the dollar offer to make, don't waste your time or his by making low-ball offer. Make a fair offer and you won't be putting yourself in a position of "bartering". The successful hiring manager will offer the candidate what he's worth based on the skills and experience that he is bringing to the company. They will not try to pinch pennies by hoping to "get the guy cheap", and only end up having the offer turned down or getting a less than enthusiastic new employee. Don't have your new employee be thinking "I could have done better". You want him to think "I'm going to work for a great company"!

What is a **"fair"** offer? There are many factors to consider when looking at the salary ranges in the HP marketplace.

You cannot compare the HP market to any other hardware vendor, such as IBM or DEC. HP is unique to itself. So don't lump an HP programmer analyst in with all the other programmer analysts in the country to determine a fair salary.

"Geographics" play a major role in determining salaries. Where the cost of living is low, the salaries are much lower than where the cost of living is high. A company's task is to recognize these factors, make the best offer they can afford without disrupting their whole internal salary structure, and help the candidate see the whole picture.

In any case, making an offer is a delicate issue. Do not be arbitrary and simply look at the candidate's existing salary to make a percentage increase. His existing salary may be out of line - either too low or too high. Do not rely on national data processing salary surveys, but rather look at what fits **your** region and cost of living. The best way to

determine this is to review the salaries of the candidates you're interviewing. This will tell you what other companies are paying and what you have to pay in order to remain competitive and attract the top talent.

Most importantly, look carefully at what skills and experience the candidate is bringing to your company. Will he need training, or will he be able to "hit the ground running"? Perhaps he will be bringing some skills that he can pass on and teach to other members of your staff. All of these factors should be weighed in order to make a "fair" offer.

The Candidate:

If you get an offer while you're on the interview, GREAT! Don't be afraid of this and fear that the company is expecting an answer on the spot. Feel free to say, "I'll give this very serious consideration and let you have my answer after I talk with my family". If you feel strongly enough that you want the job, it's fine to accept the "on the spot" offer. This will not make you appear desperate, but rather it will let the company know that you are decisive and that you know what you want and recognize it when you see it.

If you wish to discuss it with your family, it's important that you don't drag out your decision. After having done research on the company, having had a thorough phone interview followed by a thorough in-house interview, you should have a good feel for the job and whether it's what you want. If you need more information on which to base your decision, ask for that information immediately and get on with your decision. It's not fair to the company to keep them waiting and it's not fair to you to agonize over your decision.

What I see more often than not in making a decision over a job offer is the candidate "going emotional". This is most common for the first time job changer, but still evident in practically every job change. The important thing to realize is that it's **normal**! Everyone experiences this, yet many still move ahead with their career and make the job change. It's important not to let your emotions rule your head. You must make the best business decision for your career using logical factors, and keep your emotions out of it. Once an offer is made, the reality is now there in front of you and you have a decision to make! You have to decide if you should really leave all of these friendly co-workers where you know you're accepted. What if your new co-workers don't like you? You **know** you can do the job where you're at. What if you don't succeed at the new company? You have wonderful friends and family nearby. What if you don't like the new town and can't make any new friends? This is all "fear of the unknown". Unlike the positive "what if..." slogan that

HP uses, these "what ifs" will immobilize you.

What may help to make a difficult decision easier, and to bring this decision out of the emotional arena into the logical arena is to use the 8-point value survey that I found. Research was done on mid-level professionals and what they considered to be their most important values when contemplating making job changes. The candidates asked themselves the following 8 questions and the survey reported that if they could not answer "yes" to at least 4, they did not take the job. If they could answer "yes" to 6 of the 8, 80% did make the change. I find these questions help to pinpoint the most important issues for a candidate when he has a job offer to consider.

1. Do you like the nature of the work you'll perform?
2. Can you do the job?
3. Is the company stable?
4. Is the chemistry between you and them appropriate?
5. Will the company pay you a fair wage?
6. Is there opportunity for growth?
7. Is the location appropriate?
8. Is their philosophy of doing business compatible with yours?

To summarize, there are many issues involved with interviewing and hiring. The successful candidate and company will take these issues seriously, preparing ahead of time, making their time "count" on the interview in a good balanced exchange of information, carefully extending and considering the job offer, and then everyone comes out a winner. Good Luck!

THE MYTH OF GOOD TRAINING AND DOCUMENTATION (6732)

Tom A. Elliott
Distribution Resources Company
6061 South Willow Drive, Suite 100
Englewood, Colorado 80111
(303) 889-4500

In this paper we will take a look at the feasibility of dealing with training and documentation issues. To begin with we will define "training" to include two broad functions: 1) showing people how and why they are supposed to do their job and 2) teaching people general concepts so they might be able to do their job better. This distinction between 'training' and 'educating' will be emphasized as we go along. The documentation portion of this paper will cover virtually anything that should be put into some type of archives so others might read it and benefit from it. That could include things such as policies, procedures, training materials, etc. We will attempt to identify the reasons companies typically do a very bad job at both training and documentation and try to give some practical guidelines to help us all improve our outlook in those areas.

Throughout our lives we do our share of training and documentation instinctively; we just don't put the labels on our efforts. As our children grow up we are constantly showing them how to do things; tying a shoe lace, swinging a bat, baking a cake. We enjoy our share of successes in these endeavors and feel good about them, even though during the process our patience sometimes runs a little thin. This informal training is natural to virtually everyone. Anyone that proceeds to tell someone else how the company/world should be run is attempting to educate them. Although none of you engage in this type of activity you probably know someone that does. Have you ever written a memorandum designed to make sure the recipient understands your position in a matter - just in case something happens? That's documentation.

You say there's a difference between "that kind of training, educating and documentation" and the formal kind? You are right ... and you are wrong. Let's identify this as a barrier that we have erected. Formalizing a concept for better understanding simply takes more planning and homework. Unfortunately, those facts alone tend to make us stay away from these responsibilities. The point is we know we can do it; we do so much of it automatically in everyday life. That barrier, therefore, can be reduced if we can lessen some of the formal issues.

Now that we know training and documentation can be done, let's get more specific. We need to define more of the barriers that exist and figure out how we can break them down.

The next barrier is "perceived cost". Management tends to think in terms of dollars. However, there are two kinds of dollars, "hard dollars" and "soft dollars". Hard dollars are easily defined. These are dollars that are spend for which we see concrete results and/or product. If we buy a truck for $10,000 we can see the truck. It is a physical asset. The cost justification for acquisition of the vehicle runs along the lines of "How much do we spend on maintaining our old truck? How many more customers/orders can we handle with the new truck?" This line of questioning again deals with hard dollars, speculative though they may be. Payroll, inventory levels, etc. deal with hard dollars.

Training, on the other hand, produces questionable gains in the company. We can take a look at a very simple example. You hire a new telephone receptionist. A customer calls.

Receptionist: "Acme."
Customer: "May I speak to Mr. Jones, please?"
Receptionist: "He's not here."
Customer: "When do you expect Mr. Jones?"
Receptionist: "I don't know."
Customer: "Thank you."

In the entire conversation the voice has been pleasant and no false information was given. Is the customer content with the discussion? Probably not. The next questions are the most difficult to deal with: Will the customer call back and how important is it whether they do or not? The answers are never known, therefore are difficult to quantify into hard dollars. Do the answers make a difference? Probably. Did we miss a $10,000 order due to the abruptness of the conversation? Or was it just a social call? What can we do to make the conversation go a little more like:

Telephone rings.
Receptionist: "Good Morning. This is Acme Distributing Company. How may I help you?"
Customer: "Good Morning. This is Mrs. Crankshaw. I would like to speak to Mr. Jones."
Receptionist: "I'm sorry Mrs. Crankshaw, but Mr. Jones is not available at the present time. May I take a message to let him know that you called?"
Customer: "Yes, please inform Mr. Jones that I called and that I would like him to return my call. My number is: area code 303. The number is 555-5050."
Receptionist: "Can you give me the best time for him to return your call?"
Customer: "This afternoon or tomorrow morning."
Receptionist: "And can you tell me what this call is

regarding so he can be prepared?"
Customer: "Yes, this is in regards to the order we talked about last week."
Receptionist: "Thank you, Mrs. Crankshaw. I'll see that Mr. Jones gets this message as soon as possible."

Is this conversation more beneficial to both the company and the customer? What does it cost to train someone to be able to deal with the customer better? Very little time and very little money. Seminars are available in virtually all cities. Audio cassettes can be purchased for a low-cost learn-on-your-own-time process. Videos are also available quite inexpensively. You choose the level of time and cost commitment you think you are able to make toward training your people to make you look the very best.

Get the idea? Sure, the topic was simple and the solution was obvious. If the solution was so obvious, however, would the situation occur as often as it does when YOU call someone?

Let's face the facts: we're talking matters of degree in finding easily available and relatively inexpensive solutions. What is perceived as a problem to one is not to another. A $10,000 solution may be inexpensive to one and prohibitive to another.

What we need to do is to take a position and address it. The position I would like to take is this: Very few companies have any kind of comprehensive training program established either for new employees or employees new to a job function. In addition, very few companies have documentation of any kind that is anywhere current. Look at your own company. Am I right? Notice that I'm NOT saying you don't attempt training nor that you have no documentation. Nor am I saying that you are not successful in your line of work. What I am implying is that you can do better at both with some degree of effort.

Easy for me to say. What you need are solutions, not accusations. Put up or shut up! OK, here goes!

There are some secrets we need to share. None of the secrets I am about to divulge require anything more than common sense and acceptability. What I want to do is to SELL you that it can be done. Once you BUY that concept (want to buy a bridge?) you are ready to progress into creating training and documentation opportunities.

SECRET NUMBER 1: Know your audience. Sure it's trite. Perhaps that's the reason we keep forgetting it.

SECRET NUMBER 2: Create a foundation the audience can accept. You can't teach until you have the foundation.

SECRET NUMBER 3: Develop a learning process. You will never be able to create a "class" or a document that will answer all possible questions that will ever be asked. A solid learning process will enable the customer (user) to work the problem through to a solution.

SECRET NUMBER 4: Keep it simple. The old KISS concept. The more detailed you become the more difficult it becomes to maintain the process and the quicker it is abandoned.

SECRET NUMBER 5: Tie it to real activities. Don't just talk concepts; make it meaningful.

SECRET NUMBER 6: Know your media opportunities. Some material is readily available, some is inexpensive and some can be learned and used easily.

DOCUMENTATION - WHAT IS IT?

Generally speaking we will refer to documentation as the archiving of reference information whose intent is to provide answers to questions yet to be asked. This paper will discuss the following types of documentation and their media:

1. PC-based applications
2. Operational procedures
3. Non-PC based applications
4. Programming documentation

To better understand the difficulties of documentation we can simply look at the various groups of encyclopedia. Information as a whole is presented in topical fashion, alphabetically. To aid the inquiree there is a special volume dedicated to cross referencing the topics. When was the last time you looked something up in the encyclopedia and found it on the first try? How about the second try? Would you believe the cross reference put you in the general vicinity in the topic and you eventually found your answer? What did you expect?

There are a variety of "secrets" that one must use if they are to have effective documentation and training. To better understand these concepts we should look at the problems that exist with typical thought patterns for both documentation and training.

When embarking on a project of any size we know, either intuitively or because our consultant told us it must be done, that we must fully document the procedures, policies, intent, individual tasks, etc. to the fullest. And so we start. Within a reasonably short period of time we begin to be faced with a dilemma: our project is somewhat behind our beginning schedule and we SEEM to be spending a lot of time documenting. There's only one solution - do less documenting. Using our innate intelligence we determine what is the most important thing we need to have documented when the project is complete and we work toward documenting in a priority sequence. At last the project is done. We have some degree of documentation completed and it is given to all active and inactive participants of the project. We have archived the project and have a binder(s) to prove it. Depending on the nature of the documentation one of three things will happen to it: 1) if the documentation was designed to archive the steps of the project it is put on a shelf and will gather dust. 2) if the documentation is one of procedures and/or policies it is used for initial training (what a novel idea) then it is put on a shelf (closer to the

end-user) where it proceeds to gather dust. 3) if the documentation is for programming it is put in binders and put on a shelf (closer to the programming staff) where it can also collect dust.

The only exception to the rule is documentation that must be created by a software firm. This type of material will by necessity be designed for two levels of people, the technical and the non-technical. If we look even further, applications software documentation can be classified into PC-based and CPU-based (Non-PC). Due to the nature of the PC-based software their documentation has been composed using the "secret". CPU-based software tends to be much more comprehensive in it's scope, making the process of documentation much more difficult; and the companies developing this type of software generally have not found the "secret". As time has developed many computer users have begun to use both kinds of software and cannot figure out why they seem to be able to find answers in the documentation for PC-based applications and cannot in CPU-based documentation. We can get a hint as to why this occurs by looking at what is being documented in the four types defined earlier.

Type 1: PC-BASED APPLICATIONS:

PC-based documentation, by it's very nature, is feature oriented. Each feature stands on it's own as to what it delivers. The number of features is finite and the number of features that are used together are minimal. That means the documentation can dwell specifically on a single feature without regard to the effect of combining features. Upon investigation into the PC-based documentation it looks something like this:

1. Getting Started

This section of the documentation is aimed at how to load the software into the system. The concept is to make the discussion extremely simple, recognizing that the user might not be very technically oriented. Separate discussions on loading to a hard drive, single vs. multiple floppies, backing up the master discs (do people still do that?), etc. are presented. Notice that the discussion is very GENERAL in scope, acknowledging that the software can run on many different hardware brands using different printer brands and using different modem brands and using different tape backup brands. What that really means is that what they explain in the book is NEVER how MY system responds as we install the software. Before I get done installing the software to hard drives and configure the printer and the modem I WILL be more

technically oriented than I was when I started.

2. The Overview

This is the sales pitch. It is a broad discussion on how you might use the software to better your productivity, the productivity of your company and the betterment of the world in general. It contains comments like "You can produce documents like this one to enhance your image with the board of directors without having to go to a professional service and spending a large amount of money.". This is generally followed by a magnificent picture containing pie charts, bar charts, multiple fonts and sizing and a variety of shading. Included both above and below are sections of text. Depending of the number of features in the package this section may be relatively large.

3. The Features

Basically the "how to" portion of the documentation and the area where you will spend most of your time. It is usually quite good and will walk you through examples to show you how to do things. You should note, however, the examples given are typically for the simpler uses of the feature, not the complicated combinations (that are the ones needed to produce the output defined in section two that will make you a hero in the eyes of the world).

Upon analysis we can begin to see what the "secret" might be.

Section 1 is detailed but can be as the process is virtually the same for any system/software package ever available. What that means to the documenter is the section will probably NEVER have to be altered.

Section 2 is very general with broad terms, a lot of pizzazz pictures and a sense of sales. Improvements in the software means that the documenter will probably NEVER have to change this portion of the documentation.

Section 3 deals with the features and as such are typically command driven. Each command is identified and discussed in their simple terms. When the software is improved one of two things will occur: either a new command/feature will be added making the documenter simply add a new section or an existing command/feature will be made "slicker". In the latter instance the documentation might change to include the new slick stuff in the overview section (Section 2) as a one-liner but will probably NEVER alter the feature discussion as the improvement is generally not in the simple

syntax. If it is, the change in this section is MINIMAL.

In retrospect, we might define the "secret" in the form of RULES:

RULE 1: WORK REAL HARD TO DEFINE THE DOCUMENTATION STANDARD.

RULE 2: STICK WITH THE BASICS. DO NOT TRY TO ANSWER ALL OF THE QUESTIONS THAT MIGHT BE ANSWERED.

RULE 3: STAY AWAY FROM THE DETAIL AS MUCH AS POSSIBLE. IF YOU MUST HAVE DETAIL, KEEP IT BASIC (see 2 above).

RULE 4: IF YOU MUST GIVE EXAMPLES, KEEP THEM BASIC (see 2 above).

RULE 5: DEFINE YOUR AUDIENCE. BE CAREFUL NOT TO DEFINE THE AUDIENCE AS IT MIGHT BE BUT THE SINGE AUDIENCE YOU WANT TO APPEAL TO. YOU HAVE NO CONTROL OVER WHO THE AUDIENCE REALLY IS.

RULE 6: IF YOU HAVE ANY QUESTIONS ON THE AUDIENCE TAKE A SINGULAR MIDDLE GROUND. ASSUME SOMETHING!

If the "secret" seems like it is leaving part of the job undone your are exactly right. The part that is being left undone is the part that only will pertain tc the program expert, the one that has mastered the basics and is going on to more exotic uses of the software...like creating the images that are shown in Section 2. Looking further what is missing is the "linking" of the features and the advanced syntax.

Is that all bad? Not at all. What's more, it allows for the single most important concept in creating documentation. If improvements in the software require major changes in the documentation the release date of the software might have to be moved back to allow the documentation to be correct. From a marketing standpoint, this is unacceptable. Therefore, you make your compromise..you DO NOT fully document the new and better improvements, basically saying you have compromised your standards, which says your standards were wrong to begin with.

In short, it is a little like noticing that all of the system passwords have been written down and taped to the terminal. Your concept of security was probably faulty and you have over-secured the system. In the same sense, if you make changes in your software and you must compromise your original documentation standards in order to get the software to market, your standards are suspect. Take another look at

WHY you set your standards the way you did. You MUST REALIZE at this point that the problem WILL NOT GET BETTER! Retreat, reorganize your thoughts on your standards and make plans for rewriting all of your documentation. As a whole you should plan on doing this every 5 years or so anyway as your direction on the software has changed and the old documentation will not reflect your new image, concepts, etc.

If we take the above "secret" and apply it to the other three types of documentation they might look like this:

Type 2: OPERATIONAL PROCEDURES:

When working on a new project this is a must. A wonderful exercise, it allows you to define the difference between the way things work as opposed to the way we THINK things work. This clarification can then be put down on paper and everyone can be given a copy. By the way, you might think of including the policies of the company in appropriate places to indicate why the procedures are designed the way they are. Stay away from detail! NEVER say

> "Suzy fills out the application form. This form consists of 3 parts. The white original goes to Fred in the personnel department. The yellow 2nd copy goes to Whitney in the payroll department. The salmon 3rd copy goes to Sidney for data input. Fred files the copy alphabetically by last name. Whitney files the copy alphabetically by the department the new employee will report to. Sidney uses the program "NEWAPP" TO create the computer file."

What will happen if any of the people mentioned leave or are transferred to another department? What will happen if the personnel or payroll departments decide their filing systems need to be changed? What will happen if the computer software is changed and a new program is used for data input?

> "The payroll clerk completes the application form. Copy 1 (white) goes to the personnel department for filing. Copy 2 (yellow) goes to the payroll department for filing. Copy 3 (salmon) goes to the data processing department for data input."

This is not people dependent, filing techniques dependent or computer programming dependent. Any of those can be changed and the procedure is not altered, meaning the documentation does not need to be changed. In the above illustration it would be just as workable if the colors were taken out just in case the next batch of forms were ordered differently because "we couldn't read the salmon...let's try blue instead...".

What we need to do is to change the documentation only if the procedures change in intent.

RULE 7. IF THE DOCUMENTATION IS WRONG IN ANY WAY, THE USER OF THE DOCUMENTATION WILL LOSE CONFIDENCE IN IT'S CONTENT.

RULE 8. IF THE USER HAS LESS THAN 100% LEVEL OF CONFIDENCE IN THE ACCURACY OF THE DOCUMENTATION THEY WILL QUESTION EVERY ANSWER, NOT KNOWING IF THIS ANSWER IS ANOTHER WRONG ANSWER.

RULE 9. ONCE THE USER BEGINS TO QUESTION THE ANSWERS FOUND THROUGH USE OF THE DOCUMENTATION THEY WILL NO LONGER USE IT.

RULE 10. IF DOCUMENTATION HAS BEEN CREATED AND NO ONE USES IT QUIT PRODUCING THE DOCUMENTATION.

Type 3: NON-PC BASED APPLICATIONS:

There needs to be two levels of documentation for this type; technical and non-technical. Be careful in how you identify the differences between these two entities. If you begin analyzing the customer and who will be using the documentation you will be in immediate trouble. You will be able, for example, to classify the technical person in several ways. The technician who understands the computer's operating system and not the application. The technician who understands the technical aspects of the application but is very light in the computer's operating system. The technician who is the supervisor of other technicians and who might have some insight to the politics of the company. Etc... Who do you aim the technical side of the documentation toward? The technician that understands the application and who will be getting their hands dirty (so to speak) in analyzing problems and resolving them within the confines of the system. (see RULE 5 and RULE 6)

Do the same thing with identifying the end-user audience. Assume they know something or can get help from some other resource.

This type of software also tends to be feature oriented, but the features have a more significant meaning (they might have a direct impact on the financial health of the company) when they are implemented and they tend to be less independent of other features in the system. For instance, if you are going to begin using service charges for past-due accounts receivable you are impacting the cash flow, the nature of

credit collection, and the ulcers of the CFO. How you present the feature in the documentation may will mean the difference between the user understanding the impact BEFORE they implement it as opposed to AFTER. Therefore it makes sense to add a new module to your documentation: The business explanation of features. Be careful, though. The same rules apply!

This new module creates a new problem we must face: When defining a feature, we need to tell the technical person how to implement it. We need different words to define the business reason for the use of this feature. And we must have some kind of documentation for the end-user that will be seeing the actual screens. We probably do not want to put all of these into one section of a manual or even the same manual. That leads us to the area of redundancy. It is virtually impossible not to have information from one area appear in another area.

If you DID NOT follow the rules, you have more detail than you need. This means that, if you change the program/ feature, not only must you change the documentation in one area but you must also review and change the documentation in the other two areas. Will you? If you are normal you only are interested in one phase and will ignore the others. "Let someone else worry about that!". (see RULES 7-10)

If you did follow the rules, you will find that the only place you will need to change the documentation is at the level the where the improvement was made. The other parts will not need to be altered!

RULE 11: IF YOU HAVE TO WORK HARD AT MAKING CHANGES IN YOUR DOCUMENTATION FOR A PROGRAM/FEATURE CHANGE YOU PROBABLY WON'T DO IT, OR YOU WON'T DO IT RIGHT!

Type 4: PROGRAMMING DOCUMENTATION:

This is one of the most discussed and cussed topics in the area of documentation. Observers will find several ways of doing this. They include: 1) commenting the heck out of the source program, 2) very detailed specifications, examples, notes, etc., 3) computer- based flow-charting (when was the last time you used that term?), or 4) none. The strangest thing about those four types of documenting is that each of them may be as good as the others!

Our rules include defining the audience. In this case the audience can easily be defined. It is the programmer that will be called upon to maintain this program. What, then, will the documentation be used for? To help them resolve a

problem.

Looking at the four types of documentation above let's take a look as different scenarios: If we are using a program where we are relying on the comments we will generally find that the program was modified and the comments will not have been updated. Murphy's Law states that: THIS WILL OCCUR ONLY WHEN YOU CAN LEAST AFFORD FOR IT TO OCCUR. The programmer will assume the comments are correct and modify the program accordingly. Whoops! After getting burned once, the programmer will automatically assume the comments are incorrect and will work it out of the code.

If the programmer is going from very detailed specifications, etc., they will soon find it takes longer to update the documentation than it did to make the programming change. Since their perception is they are being evaluated by the volume of programming they do they will shortcut the documentation. The next programmer into the program will assume (one time only) that documentation is correct and will act accordingly. The next time they will assume the documentation is incorrect and will work it out of the code.

Use of computer-based flow-charting is quite good if the programmer charged with changing the program creates the new flow-chart before they get into it. The use of this tool will depend on the size of the program. If the flow-charting takes 2 hours to produce and prints nice neat symbols on 465 pages of 11 inch paper (8 lines to the inch) chances are you will not use it, no matter how handy it might seem to be initially. Whether the programmer looks at the tool or not, eventually they will force themselves to look at the code in detail to get assurance it really is where the flow says it is.

If you do not do program documentation, the programmer will be forced to get into the actual code and figure out the solution to the problem. You will then receive a slap on the hand by your auditors for not having program documentation.

It seems like all instances leads to the same thing: The programmer ends up getting into the detail code. Why have documentation?

RULE 12: FOR PROGRAMMING DOCUMENTATION, IT'S PURPOSE IS TO ALLOW YOUR PROGRAMMER TO GET TO THE RIGHT AREA OF CODE AS FAST AS POSSIBLE.

RULE 13: IF YOUR PROGRAMMING DOCUMENTATION DOES NOT GET THE PROGRAMMER TO THE RIGHT AREA OF CODE AS FAST AS POSSIBLE, CONSIDER YOUR REASONS FOR PROGRAMMING

DOCUMENTATION.

The next area that needs to be included and the area that can be the most critical would be the people that are responsible for documenting. If possible get qualified technical writers that understand the critical nature of standards, can help develop good standards and can work effectively with ALL segments of the company, from the technical to the non-technical. The only area of the company that is not required for interfacing is senior management. This is alright because senior management is generally unable to recognize good documentation if it ran over them. By the way they also cannot recognize bad documentation either. Their measure is the number of pages it consumes. Someone needs to be able to take the technical concept of a program/concept and make it readable by the masses defined as the audience.

RULE 14: HIRE TECHNICAL WRITERS AND GIVE THEM THE ACCESS TO THE COMPANY.

RULE 15: DO NOT TRY DOCUMENTATION WITHOUT QUALIFIED TECHNICAL WRITERS UNLESS YOU AGREE IN ADVANCE THAT THE OUTPUT WILL BE PUT ON THE SHELF AND WILL GATHER DUST.

Finally, we must agree that having documentation that is usable and in a reasonable format for maintaining will get nothing but a rash of complaints until you have a workable index. The index should show only the areas where a word is used in a significant fashion and not just in passing. Although the concept of indexing is simple the creation of same could be a real problem. To begin with, SOMEONE must read the entire set of documentation with the concept of SIGNIFICANCE is consistent. Volunteers?

RULE 16: DOCUMENTATION MUST HAVE A COMPREHENSIVE INDEX THAT CONTAINS ONLY SIGNIFICANT APPLICATIONS OF WORDS.

RULE 17: WHEN YOU HAVE DONE YOUR DOCUMENTING YOU CAN BE SURE THAT ANY ASSUMPTIONS YOU MADE IN CREATING STANDARDS WILL BE WRONG AS IT WILL SEEM AS THOUGH NOBODY LIKES THE DOCUMENTATION AND THEY COULD EASILY DO BETTER.

RULE 18: IF YOU DO NOT AGREE WITH RULES 1-17 FOR DOCUMENTATION YOU HAVE EARNED THE RIGHT TO SUFFER.

TRAINING - WHAT IS IT?

We typically think of training as providing the opportunity for someone to "come up to speed" on a topic in a logical fashion. In this paper we will discuss the differences between "training" and "education" and specifically at possible solutions for:

1. Technical training - the mass infusion
2. Technical training - the continuing saga
3. Educating the corporate management
4. End-user training - the new application
5. End-user training - the replacement technique

There are similarities between the documentation and the training/educating concepts. The paraphrasing of some of the rules of documentation into the context of training/education we might see something like:

RULE 1: STICK WITH THE BASICS. DO NOT TRY TO ANSWER ALL OF THE QUESTIONS THAT MIGHT BE ANSWERED.

RULE 2: STAY AWAY FROM THE DETAIL AS MUCH AS POSSIBLE. IF YOU MUST HAVE DETAIL, KEEP IT BASIC (see 1 above).

RULE 3: IF YOU MUST GIVE EXAMPLES, KEEP THEM BASIC (see 1 above).

RULE 4: DEFINE YOUR AUDIENCE. BE CAREFUL NOT TO DEFINE THE AUDIENCE AS IT MIGHT BE BUT THE SINGLE AUDIENCE YOU WANT TO APPEAL TO. YOU HAVE NO CONTROL OVER WHO THE AUDIENCE REALLY IS.

RULE 5: IF YOU HAVE ANY QUESTIONS ON THE AUDIENCE TAKE A SINGULAR MIDDLE GROUND. ASSUME SOMETHING!

RULE 6. IF YOUR TRAINING HAS BEEN CREATED AND THERE IS NO REQUEST FOR IT, ASSUME YOUR ASSUMPTIONS WERE TERRIBLE.

RULE 7: HIRE QUALIFIED TRAINERS AND GIVE THEM THE ACCESS TO THE COMPANY.

RULE 8: DO NOT TRY TRAINING WITHOUT QUALIFIED TRAINERS UNLESS YOU AGREE IN ADVANCE THAT YOU ACCEPT THE SPAGHETTI APPROACH -- THROW ENOUGH AGAINST THE WALL AND SOME OF IT IS BOUND TO STICK.

To get us into other areas of training let's take the same approach as we did with documentation, namely, by looking at

the five (5) different types involved.

Type 1: TECHNICAL TRAINING - THE MASS INFUSION.

This is really the easy part. There are a host of classes external to your company that will provide your technical with the basic skills needed to survive. Included are the classes taught by HP, PC-based classes available through a variety of suppliers, one and two day classes available through local companies (including audit firms and your local colleges) and even from your handy consultant. The fees/cost for this type of training are reasonable and the associated expenses can generally be held to a minimum. The only downside to this type of training is they tend to use the shotgun approach: teach everything and something should be useful. (see spaghetti approach above)

RULE 9: DO NOT ATTEMPT BROAD-BASED TECHNICAL TRAINING INTERNALLY.

Type 2: TECHNICAL TRAINING - THE CONTINUING SAGA.

We have the marvelous opportunity to go from the problem that is easily solved to the problem that is virtually impossible to solve in a significant manner. The logical position is the same one suggested in rule 9; do it externally. Most of the companies that hold classes for the beginner (Type 1) individuals hold advance classes as well. The difficulty is the level and pace of these advanced classes. Finding the one that is just right for your people is like an experience in playing lotto. It may work and it may not work. Generally speaking you can expect some return on your investment.

RULE 10: SEE RULE 9.

Type 3: EDUCATING THE CORPORATE MANAGEMENT.

This is essentially a job in sales. Before a new application can be successful it must have the full support of management. Their job is to determine the future of the company and the road map on how to get there. In introducing a new applications package, custom or purchased, it is critical for the senior management of the company to understand the fundamental concept behind the software and be able to identify the problems of the company to the capabilities of the software. For example, if a company has a problem with cash flow they should understand if there is software that will help them manage that problem better. Most companies do not attempt to train our senior management because we are afraid the senior management will not accept

that responsibility.

RULE 11: NEVER ASSUME YOU KNOW WHAT THE ANSWER IS TO A QUESTION BEFORE YOU ASK IT.

RULE 12: IF IT IS IMPORTANT FOR SENIOR MANAGEMENT TO GET INVOLVED (AND IT ALWAYS IS) YOU MUST DEMAND THEY GET INVOLVED. THIS MEANS PARTICIPATION IN, NOT JUST AGREEING THAT IT'S A GOOD IDEA.

The obvious problem with rule 12 is the potential consequences that might befall you if you go into the President's office and blurt out that they MUST attend a training session or you are going to quit.

RULE 13: BE TACTFUL IN EXERCISING RULE 12.

RULE 14: IF RULE 12 CANNOT BE CARRIED OUT, EXPECT THE REST OF THE TRAINING TO LAST AT LEAST 3 TIMES LONGER THAN YOU WOULD LIKE AND THE COOPERATION TO BE 3 TIMES LESS THAN YOU MIGHT EXPECT.

Your next challenge will be to sell the middle managers and end-user supervisors. If you were successful in selling senior management you will have the ingredient necessary to deal with this issue: the attention of the middle managers. Senior management will have started the selling process and the key to complete the selling to middle management staff will be organizing the process. If you were not successful in training senior management you will probably not be able to sell middle management either, as there is no incentive for them to learn, other than the obvious things they typically ignore, like the opportunity to become more productive, etc.

RULE 15: IF RULE 12 CANNOT BE CARRIED OUT, PUNT!

The strange thing is that most of the training is done after we have punted and are on the defense. This might also be identified as the survival mode.

Type 4: END-USER TRAINING - THE NEW APPLICATION

The actual training of the end-user in a new application is probably the easiest area of all. The primary objective is to set a positive tone for the application. These are the people that will make or break the application. If you have sold the middle management, these people will be motivated. If they are not motivated your first task will be to get them ready to learn. This requires a set of skills not found in just any individual. (see RULE 7 and RULE 8) Just like any

other important project you must PLAN. Will the training be one-on-one or will the trainer be training several people at a time? Is the training hands-on or lecture/reading? Will there be a student manual and, if so, how will it be constructed? What kind of training is required for the trainer?

RULE 16: CERTAIN GOOD TRAINING CHARACTERISTICS CAN BE TAUGHT IN A SHORT PERIOD OF TIME; OTHERS CAN NEVER BE TAUGHT BUT THEY CAN BE LEARNED, PROBABLY THROUGH EXPERIENCE.

RULE 17: PLAN...PLAN...PLAN...PLAN...PLAN...PLAN...PLAN

RULE 18: IF YOU DECIDED TO IGNORE RULE 17, RESIGN YOUR POSITION AND TAKE UP A CAREER IN UNEMPLOYMENT.

Type 5: END-USER TRAINING - THE REPLACEMENT TECHNIQUE

The basic problem in instituting any kind of training for people who are new to a position is that is done with a different flair than the training that was done initially. In that instance we worked hard to explain how the whole thing worked with trillions of examples, etc. We had built special database (which we have since purged from the system because we needed the disc space) and took the time required to teach someone their job in this new and exciting application. Why we even spent time reviewing the new procedures manual complete with the company policies that we expected to use. The new person gets the benefit of receiving the manual and told to "read it tonight and we'll discuss it in the morning". The next morning we have someone that is doing a similar job show them what they do, using live information on a live system updating live information.

Why do we assume the new person doesn't need the same level of training as the originals? We don't have the time to do it right, it's too costly, reason, reason, reason, (read 'reason' to be 'excuse'). Besides, our original plan was for mass training, not for individual training.

RULE 19: WHEN CREATING YOUR TRAINING PLAN, MAKE SURE IT CAN INCLUDES ONGOING TRAINING.

There are many things you can do to help your training process. Look at such things as using existing media programs available in a wide variety of topics like sales training, HRD topics, telephone concepts, marketing thoughts, etc. Video, PC-based presentation graphics, one-day seminars and other methods make the training concept much easier and cheaper than you might think. Catalogues are now available

to help you begin to identify what might be ready to use at your site.

Special seminars are available to take an individual and help them to become better presenters/instructors/trainers. These seminars key on special "use them now" topics and allow the person to walk away with a vast amount of information on how to become better at their assigned task. Things like:

> How to make a question personal, determine if you are a right-handed or left-handed presenter, use of media in the presentation, what the learner identifies as the most important thing you are trying to get across, etc.

There you have it! Hopefully these "secrets" and "rules" will be of some benefit to you.

Accounting Software Packages: What's New and Who Is Doing It

Robert E. Shelley

Highgate Financial Systems
2000 Powell St., Suite 1200
Emeryville, CA 94608
(415)596-1707

The purpose of this paper is to review accounting software packages, focusing particularly on what is new or different in features or capabilities. From this review I expect that some trends and new directions can be seen which should be of interest to people who have responsibilities in the financial systems area.

Only GL and AP packages are discussed in this paper, since they are the most general ones used by all organizations. The product data presented come from a survey which was sent to each vendor, and from follow-up conversations to confirm the information. To avoid even the appearance of favoring one vendor over another, all listings are in alphabetical order.

With only a few exceptions, the vendors' products must presently have a feature in order to be listed for that feature. When a listed feature is planned for a future release, that fact is noted next to the vendor.

WHAT'S NEW WITH VENDORS

Let's start by taking a look at what is going on in the accounting software vendor community. This section will focus on changes that have some effect on all of the products for a given vendor and that indicate a direction in which some vendors feel the market is going.

Mergers and Acquisitions

As you have probably noticed, there has recently been a lot of merger and acquisition activity in the U.S. business community. The computer industry has been a active participant -- according to a recent issue of *ComputerWorld*, 446 deals were made in our industry in 1988. The HP accounting software market has seen several changes of this kind in the last year, including these:

- Cardinal Data Systems was bought by Ross Systems, a large player in the DEC VAX accounting software market.

- Collier-Jackson was purchased by Compuserve, a computer services company which is expanding into a number of new computer markets.

- Satcom bought the HP line of accounting software products from McCormack and Dodge.

- SOTAS International purchased the HP line of accounting products from Software International (Computer Associates).

Programming Language Used

Historically, most vendors have used Cobol as the language of choice for developing their products. However, in the last few years some existing software suppliers have shifted over to other languages, while new vendors have chosen other languages to launch new products. The motives may differ from vendor to vendor: some want to position their products so they are more portable to other hardware platforms, others want to use newer technology like 4GLs, and so on. Here is a sampling:

- C is the primary language used by Mitchell Humphrey.

- Pascal is the language of choice for Hewlett Packard and Smith, Dennis & Gaylord (SDG).

- PowerHouse is used by Cognos (no surprise there) and VOCOS.

- Speedware is used by Infocentre (no surprise there either).

Data Dictionaries

These days most vendors are supplying a data dictionary of some kind, to make it easy for their customers to do ad-hoc reporting and to add in custom procedures. Including vendor-specific dictionary products, here is what is available:

- Dictionary/3000 -- Hewlett Packard and Satcom
- PowerHouse -- Cognos, MCBA, and VOCOS
- Speedware -- Infocentre
- Vendor-specific -- Bi-Tech, Collier-Jackson, DPAI, Mitchell Humphrey, Schaefer, SDG.

Languages Other Than English

Today's world is becoming more and more multilingual, and so some vendors are providing their products (that is,

screens, reports, and user manuals) in languages other than English. Examples include

- ASK -- French and German
- Bi-Tech -- French
- HP -- More than 15 other languages
- Infocentre -- French
- MCBA -- Spanish.

User-Customizable HELP Messages

On-line HELP, a major innovation only a few years ago is taken for granted today. The next logical step is to allow the user to modify the vendor-supplied HELP messages in order to further explain how a feature should be used within the context of the user's organization. The following vendors provide such a facility:

- Bi-Tech
- DPAI
- Hewlett Packard
- Infocentre
- MCBA
- Mitchell Humphrey
- Satcom
- Smith, Dennis & Gaylord.

24-Hour Customer Support

As organizations become more multinational and continue to decentalize the accounting function, the need for 24-hour support has grown. The following vendors provide this kind of service, but be sure to ask for details -- sometimes the support has to be prearranged:

- Collier-Jackson
- Hewlett Packard
- Mitchell Humphrey
- Schaefer.

Field-Level Security of Data Entry

I do not know of any vendor who does not provide function-level security (that is, batch entry, running reports, closing a month, and so on). However, not all vendors provide field-level security. This can be necessary if you want to ensure that some users are allowed to enter batches only for company A, for example, or to run reports only for company B. These vendors provide field-level security:

- Bi-Tech

- Collier-Jackson
- DPAI
- Hewlett Packard
- Infocentre
- Mitchell Humphrey
- SOTAS.

Customization within Maintenance Agreement

Most vendors discourage or disallow their customers from making changes to screens, reports, or processes within the vendor's software package. The reason is that if the software has been modified, it is more difficult for the vendor to support the customer when problems arise. However, some vendors are opening up their systems in certain ways to allow customers to perform special operations that are unique to their operations. Among those vendors are

- Bi-Tech
- DPAI
- Hewlett Packard
- Infocentre
- Mitchell Humphrey
- Smith, Dennis & Gaylord.

Use of Windows and Other PC Technology

As you know, personal computers have had a dramatic impact on the way accountants do their job and the tools they bring to a given task. Some vendors feel that it is time to recognize that their minicomputer products should be updated to include technology which is now taken for granted in the PC environment, technology such as windows for looking up valid account codes when entering transactions, and even Lotus- or MacIntosh-style pull-down menus. Vendors using this technology now -- and those who have announced a commitment to enhance their product with this technology -- include

- Infocentre (in a future release)
- MCBA (in a future release)
- Oracle (now available).

CPU-Sensitive Pricing

Not too long ago, CPU-sensitive pricing was a topic of some controversy among vendors. Today, however, there is no longer any debate. What is new is that all the vendors surveyed, without exception, offer CPU-sensitive pricing. This should come as no surprise, since all of these products run on HP3000s that range in price from the inexpensive

Micro/3000 on up to the Spectrum 950.

WHAT'S NEW WITH GENERAL LEDGER PACKAGES

On the surface, it would seem that one shouldn't expect much change from accounting software vendors in the area of GL packages. But further investigation reveals that there really are new issues to be addressed and that different vendors take different approaches in resolving them.

Multicurrency Processing and Reporting

As multinational corporations are moving toward decentralized accounting operations, there is increasing need for multicurrency capabilities in minicomputer general ledger systems. The following vendors have this kind of capability:

- Bi-Tech
- Collier-Jackson (2 currencies)
- Hewlett Packard
- Mitchell Humphrey
- Satcom
- Schaefer
- Smith, Dennis & Gaylord
- SOTAS.

Automatic Elimination of Intercompany Accounts

This capability is needed by many larger organizations that have a lot of transfer pricing transactions between divisions or interrelated companies. It allows each division to recognize income or expenses from other divisions, but also to eliminate these entries when consolidating the data for company-wide reporting. These vendors offer this capability:

- Bi-Tech
- Cognos
- Collier-Jackson
- Hewlett Packard
- Mitchell Humphrey
- Satcom
- Schaefer
- Smith, Dennis & Gaylord
- SOTAS.

Selective Archiving

Most GL packages have some ability to remove historical transactions from the system and store them outside of the

main transactions file. This process usually speeds up processing of the day-to-day accounting operation and conserves disk space. Selective archiving goes one step further, allowing you to specify -- at the level of the chart of accounts -- the amount of detail you want to keep on-line. Thus you can hold on longer to details for some accounts than for other accounts. Some vendors offering this feature include

- Bi-Tech
- Cognos
- DPAI
- Infocentre
- MCBA
- Mitchell Humphrey
- Satcom
- Smith, Dennis & Gaylord
- Schaefer
- SOTAS.

Validate Presence of Standard Entries

Many general ledger packages are now accepting data from other systems in the form of files, instead of requiring the data to be rekeyed into the computer. Examples of this might be a payroll file twice a month, a fixed assets update, and similar interfaces. These subledgers could reside on other computers or even at a service bureau outside of the company. Typically the number and frequency of these interfaces are the same from month to month. Hence the need for this feature, which tracks the interfaces done each month by type and frequency to ensure that they have all been completed before the month is closed. Vendors with this feature are

- Cognos
- DPAI
- Hewlett Packard
- Schaefer.

Fund Accounting

This capability is used by many nonprofit organizations and governmental agencies, which often have different funds (sometimes with certain restrictions applied) that can be used to pay their expenses. Typically each fund has its own balance sheet, which is closely watched. The following vendors support some degree of fund accounting:

- Bi-Tech
- Cognos
- Hewlett Packard

- Mitchell Humphrey
- Satcom
- Schaefer
- Smith, Dennis & Gaylord
- SOTAS.

Encumbrance or Commitment Accounting

Encumbrance accounting is like fund accounting but takes the concept to a lower level of detail. Essentially it involves setting up budgets for different projects or funds and recording each purchase order against the budget. It implies a very controlled environment, which is usually only found in government agencies. Vendors with this feature include

- Bi-Tech
- Hewlett Packard
- Mitchell Humphrey
- Satcom.

PC Upload and Download Product

Mini and mainframe GL vendors are finding that most of their customers do their budget on a personal computer using a spreadsheet product. In support of that effort, they need a way to download historical GL data to the PC when preparing for the budget process, as well as a way to upload the approved budgets into the GL for variance analysis in the upcoming year. Vendors who provide a product which supports this activity are

- ASK
- Collier-Jackson
- DPAI
- Hewlett Packard
- Infocentre
- Satcom
- Schaefer
- Smith, Dennis & Gaylord.

Support of Graphics Output

The capability to bring summary data out of the general ledger and present it in the form of a graphic is now available from some vendors, but the method used varies. Here are some examples:

- Bi-Tech downloads to a PC.
- Collier-Jackson transfers data to 20/20, another minicomputer product.
- DPAI downloads to a PC.

- Hewlett Packard.
- Infocentre transfers data to Easywriter.
- Satcom downloads data to Lotus.
- Schaefer has a DSG interface (DSG is an HP3000 product).

WHAT'S NEW IN ACCOUNTS PAYABLE PACKAGES

Other than what has already been reviewed, there is not a lot to report concerning new features of AP systems. The one area where these systems are changing is the way they work with outside systems or process data from banks.

Tape Load of Cleared Checks from Bank

Vendors are now providing an interface facility for a magnetic tape, provided by your bank, detailing all cleared checks. For high-volume check operations, this feature is a must, since the only alternative is to process all the checks manually from the bank statement. Some vendors providing this feature are

- Bi-Tech
- Collier-Jackson
- Computer Financial Services
- DPAI
- Hewlett Packard
- Mitchell Humphrey
- Satcom
- Schaefer
- Smith, Dennis & Gaylord
- SOTAS.

Create Magnetic Tape for the Bank

The flip side of the bank providing you with a tape is for you to give them a magnetic tape detailing the checks that you have issued. This is a new service provided by some banks because many organizations do not want to take the time internally to do check reconciliation -- they prefer to hire their bank to do it for them. Vendors supporting this option are

- Bi-Tech
- Collier-Jackson
- Computer Financial Services
- Hewlett Packard
- Mitchell Humphrey
- Schaefer
- Smith, Dennis & Gaylord
- SOTAS.

Recording Wire Transfers

In today's computer age, wire transfers are being used more and more by the largest companies for paying their vendors. The reason is quite simple -- it's a lot cheaper than printing checks and mailing them. Some of the vendors supporting this new means of making payments are

- Computer Financial Services
- DPAI
- Hewlett Packard
- Mitchell Humphrey
- Satcom
- Schaefer
- SOTAS.

On-line Update of GL When Entering Vouchers for Payment

Historically, most subledgers to the GL have stood on their own, with periodic updates taking place with a file transfer activated by the subledger and imported into the GL. With the emphasis today on completely on-line integrated processing, some subledger products are designed so that as soon as you enter a voucher for payment with its GL distribution, the general ledger is also immediately updated with the information. Here is a sampling of vendors with this capability:

- Bi-Tech
- Cognos
- Hewlett Packard (user option)
- Mitchell Humphrey (user option)
- Smith, Dennis & Gaylord.

IN CONCLUSION...

As you can see, some new and exciting capabilities are being added to accounting software packages. Many of them reflect the general environment (for example, business mergers and the influence of the personal computer), while others, such as the use of data dictionaries, reflect emerging technologies. Whatever the reason, accounting software is changing, just like all the other areas of computer technology; to ensure that you get all the benefits you deserve from your accounting software budget, it is well worth taking the time to stay informed.

One final note -- if any accounting software vendor was not included in this paper but would like to be in future papers of this nature, please contact the author.

Appendix A

SOME HP ACCOUNTING SOFTWARE VENDORS

ASK COMPUTER SYSTEMS, INC.
730 Distel Drive
Los Altos CA 94022
(415)-969-4442

BI-TECH SOFTWARE, INC.
1072 Marauder, Suite A
Chico CA 95926
(916)-891-5281

*CARDINAL DATA CORP
The Hillsite Building
75 Second Avenue
Needham Heights MA 02194
(617)-449-0066

COMPUTER FINANCIAL SERVICES
13425 N.E. 20th
Bellevue WA 98005
(206)-746-2666

COLLIER-JACKSON, INC.
3707 West Cherry Street
Tampa FL 33607-2596
(813)-872-9990

COGNOS INCORPORATED
3755 Riverside Drive
Ottawa ON K1G 3N3
(613)-738-1440

*COMPRO FINANCIAL SYSTEMS, INC.
6025 The Corners Parkway
Suite 200
Norcross GA 30092
(404)-662-8754

DeCARLO, PATERNITE & ASSOC.INC
Rockside Square II
6133 Rockside Road, Suite 400
Independence OH 44131
(216)-524-2121

*GTS COMPUTER SYSTEMS, INC.
3529 7th Avenue South
Birmingham AL 35222
(205)-252-9446

*HIGH LINE DATA SYSTEMS, INC.
716 Gordon Baker Road
Suite 100
North York ON M2H 3B4
(416)-494-2504

HEWLETT PACKARD
5301 Stevens Creek Blvd.
Mail Stop 52U/92A
Santa Clara CA 95052-8059

INFOCENTRE CORPORATION
7420 Airport Road
Suite 201
Mississauga ON L4T 4E5
(416)-678-6880

MCBA
425 West Broadway
Glendale CA 91204-1269
(818)-242-9600

MITCHELL HUMPHREY & CO.
11720 Borman Drive
St. Louis MI 63146
(800)-237-0028

Appendix B

SOME HP ACCOUNTING SOFTWARE VENDORS

*MANAGEMENT TECHNOLOGY INT'L
16 Inveress Place East
Englewood CO 80112
(303)-790-7734

*ORACLE CORPORATION
2929 Campus Drive
Suite 200
San Mateo CA 94403

SATCOM
4521 Professional Circle
Virginia Beach VA 23455
(800)-472-8266

CHRIS SCHAEFER & COMPANY
1500 S. Dairy Ashford Road
Suite 400
Houston TX 77077
(713)-558-2273

SMITH, DENNIS & GAYLORD, INC.
3211 Scott Blvd.
Santa Clara CA 95054-3078
(408)-727-1870

SOTAS INTERNATIONAL
192 Merrimack Street
Haverhill MA 01830
(508)-521-1300

*TECHNALYSIS CORPORATION
6700 France Ave. South
Minneapolis MN 55435
(612)-925-5900

VOCOS
Dewitt Building
P.O. Box 874
Ithaca NY 14851
(607)-272-8464

* = vendor invited to participate, but declined.

APPLYING EXPERT SYSTEMS - PART II

Karen Hopmans
Brant Technologies Inc.
2605 Skymark Avenue
Mississauga, Ontario
Canada L4W 4L5

(416) 238-9790

APPLYING EXPERT SYSTEMS - PART II

Karen Hopmans
Brant Technologies Inc.
2605 Skymark Avenue
Mississagua, Ontario
Canada L4W 4L5
(416) 238-9790

Introduction

At last year's INTEREX conference in Orlando, I gave a presentation called "Applying Expert Systems in the Commercial Environment". At that time I discussed several expert systems in use, emphasizing the cost savings and other benefits the companies that implemented them now enjoy. I talked about companies like American Express, Ford, General Motors and MacDonnell Douglas. And following my presentation, a number of people commented "Expert systems sound great for companies that can afford to spend millions of dollars on a single project. But what good does that do my company?"

Most expert systems provide a considerable competitive edge to the companies using them. As a result, they tend to be very proprietary. The large, successful - and expensive - applications developed by the Fortune 500 represent only a few of the many expert systems that are implemented today. Smaller companies too are reaping the benefits of expert system technology, but these companies are often hesitant to risk losing their edge by "going public" too soon. Some of Brant's own customers, both large and small, have refused to let us publicly discuss the applications we are developing for them. Thus, the representative applications I discussed last year were not intended to discourage small and medium sized companies from initiating an expert system program. My goal, rather, was to generate ideas that can be modified, customized and applied on any scale within any organization.

Nevertheless, in researching this new presentation, I made a conscious effort to find example applications implemented within smaller companies and/or with reasonable development costs. In spite of the "confidentiality roadblock", I managed to find several interesting examples. My basic motive remains unchanged: to foster ideas for expert system applications that can be implemented within any company or organization. Whether founded within large companies or small, ideas are the seeds of future growth and can be applied universally if effectively sown in the appropriate context.

Expert Systems - A Brief Review

Expert systems are a subset of the broader field known as Artificial Intelligence (AI). Basically, expert systems are computer programs that model the reasoning of a specialist in a narrow domain of expertise. In other words, they represent human expertise in software form.

Expert systems are characterized by the separation of the inference mechanism from the knowledge base. Because of this, we can provide an explanation facility that simply traces the rules that contributed to a certain conclusion. Moreover, an expert system's knowledge, made up of rules and facts, can be treated like data. We can add to it, change it and examine it without changing the program that uses it.

Expert systems have tremendous potential in today's business environment. This technology "fits" anywhere in an organization where there is a knowledge bottleneck; where there are too few skilled people to handle too many tasks that demand some level of human expertise. This could be in administration, finance, operation, sales or on the shop floor.

The example applications I will discuss here will, I hope, demonstrate the unlimited range of domains that can be effectively enhanced by expert system technology. The benefits include significant cost savings in terms of time and personnel and, best of all, productivity increases if the "experts" are freed up by a system capable of handling even a percentage of their current workload. This means our experts can concentrate on especially difficult or unusual problems that fall outside the knowledge of the expert system, or they can spend more time on new development or with customers.

Example Applications

E.I. DuPont de Nemours & Company is noted for its proliferation of small, low-cost and highly successful expert systems. To date they total some 500. Most of DuPont's systems are PC-based and are developed for and by the end user with assistance from the internal Advanced Technology Group. The success of these systems is due to the fact that each is aimed at solving one specific task and development takes place very rapidly in order to enable deployment while the "window of opportunity" is still open.

DuPont's Packaging Advisor is an expert system that helps design a rigid plastic food container. It runs on a PC and cost about $50,000 to develop. Given general parameters about the use, shape and dimensions of a container, Packaging Advisor is able to analyze various design alternatives (materials, thickness, special barriers, cost, etc.) and recommend an optimum solution. Since the system's implementation, DuPont has realized a 30% sales increase in new plastic resins. Packaging Advisor is an excellent example of the company's successful expert system strategy.

Metropolitan Life Insurance Company designed an expert system to assist attorneys and paralegals in the closing process for commercial real estate mortgage loans. The system identifies the legal closing requirements by considering numerous features specific to each loan. The benefits include time savings in faster identification of legal requirements, and increased company-wide accuracy and consistency. This system runs on a PC and was developed in nine months for approximately $75,000.

Mediatop in Paris developed the Television Intelligent Mediaplanning Expert System (TIMES) to help the company prepare a television advertising plan. It assists in the definition and construction of a campaign, audience forecast and preparation of a report. The design of a $5 million campaign can now be achieved in minutes, rather than hours. The time savings in the overall campaign preparation is as high as 30% and TIMES also enables quick response to last minute client changes. TIMES was developed in one year and has been in operation since October, 1988. It is an example of what we call a hybrid system, combining expert system technology with conventional programming.

TELESTREAM is an expert sales advisor for distribution management. Developed by GSI Transcomm of Pittsburgh, TELESTREAM serves as an advisor to inside sales and telemarketing personnel during conversations with prospective clients. As a front-end to GSI's order entry system, TELESTREAM supplies the user with up to date information on pricing, promotions, inventory and products, based on the content of the conversation that takes place. The system's primary benefit is reduction in sales costs. It also shortens the time it takes to train new sales employees, has increased margins and ultimately reduces the number of lost sales.

Flygt AB of Sweden developed MIDS (Mixing Design System) as a sales support tool for mixing design, entailing the selecting and positioning of submersible mixtures for different fluid handling situations. The user converses in dialogue fashion with the system

to determine process specification, fluid properties, installation data, etc. The result is presented in a report recommending mixer combinations, installation, positioning, etc. Other potential areas for this type of application include pulp and paper mixing and biological sewage treatment systems.

McDonnell Douglas has developed a Personnel Scheduling System to monitor current and future workloads for about 150 people. The system keeps track of personnel and job completion dates and provides an estimation of future overtime needs. It also matches people's preferences for overtime work slots and produces a schedule.

GM Research Laboratories in Warren, Michigan developed Expert Operator, a system to automate portions of the operation of a large IBM mainframe. Expert Operator monitors the system messages for problem conditions, determines an appropriate course of action and sends a corrective action command to the operating system. Benefits have included increased availability and reliability of the MVS computer systems and a reduction in the demand for the attention of human operators.

My own favourite application is Campbell Soup's "Cooker", affectionately known as "Aldo on a Disk". Aldo Cimino, who was Campbell's resident expert on the hydrostatic and rotary cookers that kill bacteria, was about to retire. Instead of allowing a storehouse of knowledge take years of valuable experience with him when he retired, Campbell and Texas Instruments collaborated with Aldo to create an expert system to diagnose and solve problems in the same way he did.

General Motors' expert system Charley is similar to Aldo, in that through the system the company was able to retain a valued employee's expertise as a corporate asset. Charley performs equipment troubleshooting diagnostics in manufacturing and assembly equipment. The system, based on rules provided by expert Charley Amble, is quite generic and can be used for virtually all kinds of machines with rotating components. Charley can serve as both a diagnostic and predictive tool.

Brant has developed a number of expert systems for a variety of companies. For example, Title Expert System (TES) is a prototype designed to further automate the process of searching public records to ensure that a parcel of land is free from legal encumbrances. TES prioritizes the relevant information in a manner emulating the expert human title searchers. The goal is to make personnel up to

three times more productive and to ensure greater management control. TES currently runs on an HP3000 Series 70 and is a hybrid system, interfacing to Cobol programs. The client's requirements have recently changed to demand faster results and we are currently determining the best way to offload the expert system processing, probably to a PC workstation, to improve performance. The anticipated final cost of the TES prototype is $50,000.

The ROGI financial planning expert developed by Brant is another example of a hybrid application, combining conventional technology with an expert system. ROGI has a database front-end designed in Speedware/MicroSpeedware. The database contains "hard" information about a financial planner's client, including assets, liabilities and cash flow. Once the data is accumulated in the database (complex calculations are performed using 3GL subroutines), ROGI passes the information to the expert system written in MPROLOG. The expert system applies certain "soft" information on client attitudes, past experiences and preferences to come up with a truly "personal" financial plan. The plan is actually formatted by an intelligent text processor which selects appropriate wordings based on the client's combined hard and soft information. Although originally designed for use on the HP3000, ROGI has become primarily a PC-based product. This is due to the fact that the end users of the system - personal financial planners - tend to prefer the portability of a PC.

Brant also developed an expert irrigation scheduling system for an agricultural management company in California. This company decided that there was significant expertise associated with irrigating crops. A numeric model was developed to take into account inches of rainfall, amount of irrigation required by certain crops, etc. The expert system brings into play those "heuristics" or rules of thumb that a farmer also uses when determining his irrigation schedule. The irrigation scheduler was developed using MPROLOG and delivered on the PC. The total cost of this prototype system was approximately $30,000.

APEX (A Personnel Expert) is a system designed to help a large manufacturing firm standardize in the area of job descriptions and evaluations. This company is growing rapidly by acquisition and the goal of the expert system is to ensure corporate consistency in the personnel department, while at the same time placing more responsiblity (ie: actual use of the system) with the divisions. It is anticipated that APEX will cost $50,000 to $75,000 and will take about six months to complete.

Qualifying Candidate Applications

Before embarking on an application, even at the prototype level, you should ensure that the project will likely make a good expert system. Following is a list of features that characterize typical expert systems and this may help you to qualify whether or not a candidate application will lend itself to effective solution by this technology:

- the problem area is characterized by the use of expert knowledge, judgement and experience
- human expertise is not or will not be available on a reliable or continuing basis
- the task is decomposable, allowing relatively rapid prototyping of a small subset of the complete task
- the task is teachable to novices (therefore teachable to and by a computer)
- there are recognized experts solving the problem today
- there is an expert willing and available to work on the project

To increase your chance of success, you should consider these points before attempting to implement your project using expert system technology. You should not try to build expert systems to solve problems people are unable to solve. Nor should you try to solve problems that are effectively solved using other techniques.

Our philosophy at Brant is "don't think of expert system technology as a solution looking for a problem. Rather it is an alternative way to solve a problem domain not effectively addressed by traditional programming methods alone.

Conclusion

As we have seen, expert systems can be applied to commercial applications with substantial payoffs and benefits. Smaller companies as well as large corporations are taking advantage of this key technology. The tools and knowhow exist today to help you begin applying expert systems in your environment. If you start small with a prototype system, perhaps following the tried-and-true guidelines of an AI consulting company, you too can successfully implement expert system technology.

How to Convince Top Management to Adopt An Integrated System Architecture

Amir Farazad
ELM Inc.
1250 S. Washington Street
Suite 603
Alexandria, Virginia 22314

1. INTRODUCTION

Executive decision-makers now perceive information as a key resource to the survival of the organization. Most, however, are growing increasingly frustrated by the inability of MIS departments to provide timely and accurate data in formats which speak directly to the needs of top management.

Thus, the problem is not, in reality, whether top management wants an integrated architecture but rather is MIS prepared to deliver a new system design which can assure data integrity and reflect the business environment in a way that will generate information which can support corporate strategic decision-making.

Based on my experience, I believe that an integrated object oriented design which speaks the language of business is the most appropriate solution. Here I would like to suggest that IS directors re-evaluate their position within the organization from a business perspective and adopt this architecture to overcome current system limitations. Let's review current and alternative methods from both business and system viewpoints.

2. CURRENT ENVIRONMENT

MIS departments first need to admit to the limitation of existing systems.

Figure 1 provides a visual presentation of the current system environment.

2.1 System Limitations

Some of the problems encountered in this multi-system distributed scenario are:

- Duplication of data
- Redundancy of programs
- Data inconsistencies
- Backlogs

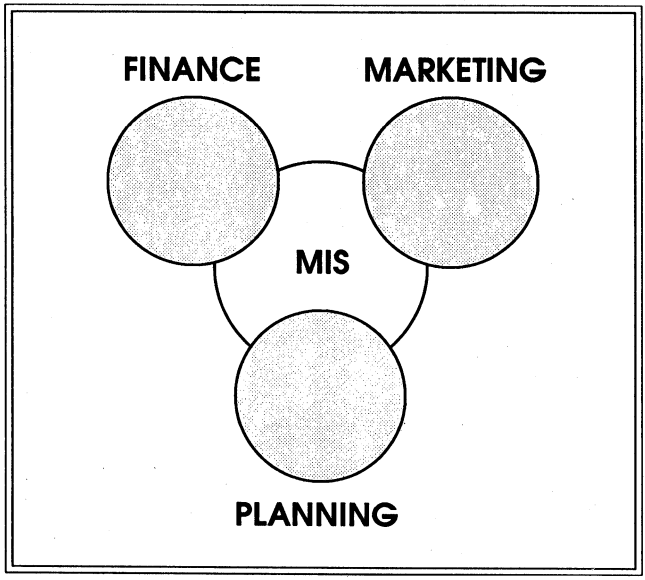


Figure 1.

2.2 Business Limitations

A review of the problems found in the current system environment leads us to ask why and how this system evolved. The root of this problem is executive decision-makers' perception of what MIS is and its role in the organization. MIS units traditionally have never been part of the decision-making loop and have been considered a support group. Thus, a service bureau mentality was implemented and, more importantly, encouraged by the organization's hierarchy. This fundamental problem was further exacerbated by multiple departmental requests for automated systems which led to the problems noted above.

3. PROPOSED ENVIRONMENT

In the new environment, IS directors must communicate in the language of business. Therefore, this author believes it is more appropriate to begin with an analysis of the IS director's role in the organizational hierarchy.

3.1 Business Overview

The information needs for strategic planning are basically a road map for the achievement of the organization's goals. Unfortunately, systems have been developed based on partial information, gleaned from various operating units. Only recently, IS directors have developed the consciousness that, in order to support strategic planning, their involvement is necessary from the outset. A piecemeal approach to system development cannot produce an automated environment capable of addressing the needs of the entire organization. Thus, the customer service orientation of most MIS units is clearly part and parcel of a bygone age which cannot support the integrated environment.

In order to design a system which reflects and supports strategic planning, MIS professionals must fully understand the business environment. To that end, IS directors must lead their units into the development of a partnership with management to better understand the organization's goals, objectives, and operations.

Goals and objectives can best be learned through direct interviews with professionals from various business departments within the organization. An effective way to explore operations is the establishment of internship programs, where MIS staff members spend several weeks to a month in different business units. Actual hands on experience will enable MIS units to establish new policies and procedures, as well as develop generalized systems for the entire organization.

3.2 System Overview

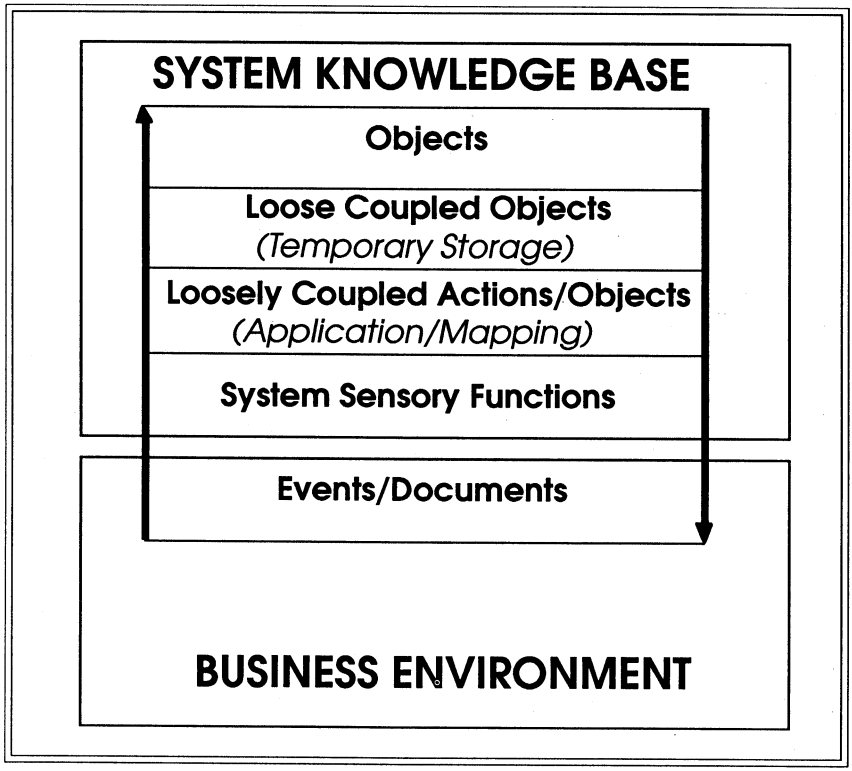
The object-oriented design has been developed from analysis of basic human cognitive processes and is designed to provide a framework for the planning and design of information systems. The architecture reflects the business environment in terms of the basic objects of the business and describes occurrence of events which change the state of those objects. Therefore, the proposed system environment more closely mirrors the business world and, through automation, improves both internal and external communications.

The components of such a design are:

- Global information area which contains all the business objects
- Series of generalized system functions which include programs for ***Sensory, Scheduling, Decision Support, Operations Control, Audit, Attribute Updating/ Edit, Security,*** etc.

- System Knowledge Base which describes the business environment. The System Knowledge Base in its broadest terms describes the organization's objects and business events. Technically, the System Knowledge Base is a set of tables which: (1) define and describe all entities, be they an object, attribute, or action within the system and, (2) describe the object boundaries and object conversions which are used to achieve certain Business Functions.

Figure 2 presents such a system model in dynamic interaction with the real world.



4. MEETING THE CHALLENGES OF THE CHANGING ENVIRONMENT

In order to successfully achieve the desired environment, IS directors must realize the power of the object-oriented design and know that they possess the business and technical knowledge necessary for the creation of such a system architecture. Once IS professionals have this awareness, a presentation to top management can easily be done in a format best suited to the needs of the organization. (The object-oriented architecture is presented in paper number 6741.)

4.1 Benefits of the New System Approach

The following are among some of the benefits which can be expected:

- Dynamic
- Easy to use
- Easy to automate
- Open-ended
- Advanced opportunities for office automation
- Eliminates redundancy
- Leads to optimization of corporate resources
- Supports strategic planning

Dynamic

Generally, the majority of changes in any organization are due to modifications of rules and regulations, either internally or externally. Since rules and regulations are defined as entities in the System Knowledge Base, any changes can be easily implemented by simply redefining that particular entity. Further, any application which utilizes the modified entity will be automatically updated in the system.

Easy to Use

The System Knowledge Base describes the business environment. Therefore, by incorporating extensive sensory functions such as graphics, charts, help menus, etc., user interaction with the system is greatly facilitated.

Easy to Automate

The object-oriented environment follows a standard design based on objects which reflect real world equivalents and their conversion process. This standardization leads to great reductions in man hours during the design phase associated with the automation of any new business function.

Open-Ended

The design of the system functions are based on the concept of loosely-coupled modules which contain specific utilities for the processing of specific business functions. As the organization gains greater knowledge of its objects, their boundaries will expand. The open-ended feature can easily accommodate any extension of the object boundaries by reconfiguring the existing modules or by adding new modules.

Advanced Opportunities for Office Automation

On-line access of business objects in terms of look-up screens, spreadsheets, or mathematical models for the professionals within the organization can easily be achieved which, in turn, increases the quality of decision-making, customer service, operational control, etc. . .

Eliminates Redundancy

The existence of generalized system functions (see Section 3.2) is one of the most powerful attributes of this architecture. The creation of any new system functions can be easily achieved by coupling these generalized functions with a set of specific programs for any single business function. This allows MIS to reduce the man hours spent on generating redundant programs and allow staff members to focus on optimizing system efficiency.

Optimization of Corporate Resources

Understanding the business environment and the stability of this design helps IS directors to assist their partners in balancing the resources spent on automation or its alternatives.

Support Strategic Planning

Top managers prefer to make decisions based on multi-scenario models as integral parts of their strategic planning. Because of the system's inherent technical flexibility, this environment can fully support the needs of top executives.

4.2 Presentation to Top Management

Here the IS director must don his marketing hat and three-piece suit and beard the executive lion in his den. The successful presentation will contain these three major elements:

- Prototyping
- Operational Change
- Timetable

Prototyping

This is the key element of the IS director's presentation. Before beginning any work on the prototype, the IS director and his/her staff should have already gone through their own process of familiarization with the business environment as explained in **Section 2.1**. This process of exploration should lead MIS professionals to identify a specific set of numbers which will answer the questions most frequently asked by top executives. These numbers can be calculated through a conversion process which could range from simple data calculations to more sophisticated statistical or mathematical models.

The prototype must demonstrate how the new architecture can access the type of information required by executive decision-makers, and highlight system flexibility. The presentation can be greatly enhanced by integrating PC tools such as color graphics and charts in an exciting visual presentation. In sum, the prototype should secure the interest of the "buyer" and lead him into the next phase.

Operational Change

The next phase addresses the operational changes which will inevitably occur within an organization when it converts to the integrated environment.

In this environment, object boundaries are of primary importance. By object boundary we generally refer to the amount of information about a specific object which is needed to satisfy a business requirement.

Figure 3 presents the object and the attributes which are utilized by separate business units.

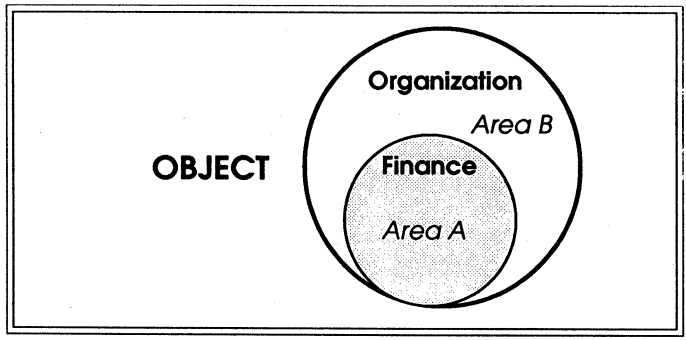


Figure 3.

Here we can see that the Finance Department utilizes the set of attributes from the Area A less than B. These specific set of attributes are less than the organizational object boundary.

If Finance is responsible for transferring the information for this object, then their scope of work will necessarily expand in order to satisfy the organizational object boundary.

Top management must be made aware of the operational change which is inherent in the integrated environment, simply because as object boundaries change and scope of work increases, operational costs are due to rise in proportion to the increase in work.

It is important to note, however, that the benefits derived from expanded object boundaries far outweigh the costs associated with increased operational change.

Timetable

The final phase presents a timetable which lays out a schedule of 1) short-term maintenance plans for existing systems as well as, 2) a long-term plan for conversion to the integrated architecture.

The most critical element in the short-term timetable is the establishment of strict time limits beyond which system maintenance and upgrading will no longer be performed.

The long-term plan requires several months at the outset which will allow the staff sufficient time to explore the business environment in order to design and develop the System Knowledge Base. This is a critical stage in the development of the new architecture because these entities are the building blocks of the entire system. Once the global objects and implementation of the System Knowledge Base is completed, then development of business functions can be achieved within a relatively short period of time.

5. CONCLUSION

The IS Executive must use his presentation to convince top management that the integrated approach will focus on the business environment, marshall the information resources within the organization, and utilize technology in support of strategic decision-making. This new environment will turn mere data into a corporate asset and further strengthen the position of IS professionals within the corporate hierarchy.

Toward An Integrated System Environment

Parvin Rahnavard

ELM Inc.
1250 S. Washington Street
Suite 603
Alexandria, Virginia 22314

OBJECT ORIENTED DESIGN

If one were to peruse the current literature on Object Oriented Design, it would appear that this approach is a very complicated proposition indeed. The reality, however, is quite the contrary. Object Oriented design is simply a method of organizing and grouping information in such a way that it closely mirrors the language used in the real business environment.

Technically, we define applications as input which are processed through a series of programs to obtain the desired output in support of a Business Function. Lets expand the concept in terms of business. We can define a Business Function as a series of Events received in the form of documents *(paper, tape, etc.)* which are processed in order to generate another set of Documents to be mailed internally or externally.

In MIS, we have to think in terms of business and technology. Lets identify these Events/Documents and define them as Business Objects. Therefore, our job is to identify these Objects and their relationship with each other and then to design a data structure to store information as well as identify and develop all possible functions to manage and maintain these Objects.

Architecture Design Overview

1. BUSINESS OBJECTS

Business Objects are the result of an evolving process which takes place in a business environment. Every environment begins with a series of basic elements, and, as internal and external decisions take place, a new series of elements are introduced which are the result of relationships among existing elements. The outcome of those relationships are also a series of elements which in turn, are another form of relationships based on certain rules or procedures.

The creation of Objects in an organization follows much the same process. Therefore, one can describe the Objects as follows:

- Existence of basic elements of the business environment.
- Occurrence of Events which change the state of business and/or environmental elements.

1.1 Basic Business Objects

The basic Objects of a business environment are as follows:

- People — Includes all people who have or may have a business relationship with the organization.
- Organization — Includes all external or internal organizations that have or may have a business relationship with the organization.
- Stocks — Any product or service which is purchased, owned or produced by the Business Organization. Stocks can be subdivided into:

Physical stocks
Financial stocks
Conceptual stocks

1.2 Business Events

The Events are a series of natural language sentences which describe the business activity being recorded. The sentences follow the general journalistic rule of presenting information to the user that chronicles the occurrence of an Event and answers questions such as: ***Who? Where? What? When? How?***

2. OBJECT ORIENTED ARCHITECTURE

Automation of a business function, in its simplest form, is a series of actions which are applied to the input Objects to obtain the desired output Objects. Traditionally, the approach has been to develop separate application programs for each business function. Individual Business Functions own their own information entities and specific programs. The problems associated with this distributed approach are well known to all MIS professionals and documented in Paper Number 6740.

The Object Oriented Architecture embodies an integrated automated framework which supports the organization's strategic planning and operational control needs. It can be characterized as an open-ended, dynamic, consistent design which maximizes the human/system resources within the organization. The major components of this architecture can be defined as follows:

- Objects
- Actions
- Action-Grouping
- System Knowledge Base

2.1 Objects

Each Object has its own table where attributes and their values are stored. There are three types of attributes:

- Attributes unique to that Object
- Attributes pointing to other Objects. This structure is based on the entity-relationship following the *"5th Normal Form."*
- Attributes utilized by the system.

2.2 Actions

Actions are generalized *"programs"* or *"utilities."* An *"Action"* can be described as a series of computer instructions which achieve the desired result once the specific input has been identified. Structure Query Language (SQL) utilities are good examples of *"Action."* They are composed of a series of instructions applied to Relational Tables. These instructions can ***Create, Update, Delete,*** etc. . . any table.

These *"Actions"* can be purchased from vendors or developed internally to fulfill the needs of the following functions:

- Sensory
- Scheduling
- Decision Support
- Operational Control
- Audit
- Attribute Update/Edit
- Security

Sensory Functions

Sensory functions include those system functions involved in the capture and validation of Objects.

Scheduling Functions

These functions include those system functions which pertain to the normal work flow of the business.

Decision Support Functions

These functions define, describe and execute Object conversions in desired document specification.

Operational Control Functions

These functions are in fact, audit functions, in which the system tests the operation of the business according to specified criteria.

Audit Functions

These functions test the integrity of the attributes of the Objects based on the internal or external rules and regulations.

Attribute Update/Edit

These are a series of functions which update or edit the attributes of the Objects with respect to its relationship with other existing Objects.

Security Functions

These functions can be applied on a multi-level basis which includes the Objects, Attributes, or any specified value.

2.3 Action-Grouping

Action-grouping describes the Business Function that changes the state of the attributes of the Objects. An *"Action-Grouping,"* therefore, can be defined as a series of actions performed on a set of input Object(s) that lead to a particular set of output Object(s).

Diagram 1 presents the Input and Output Object creation or conversion.

Diagram 2 illustrates how the action precedence completes the Action-Grouping.

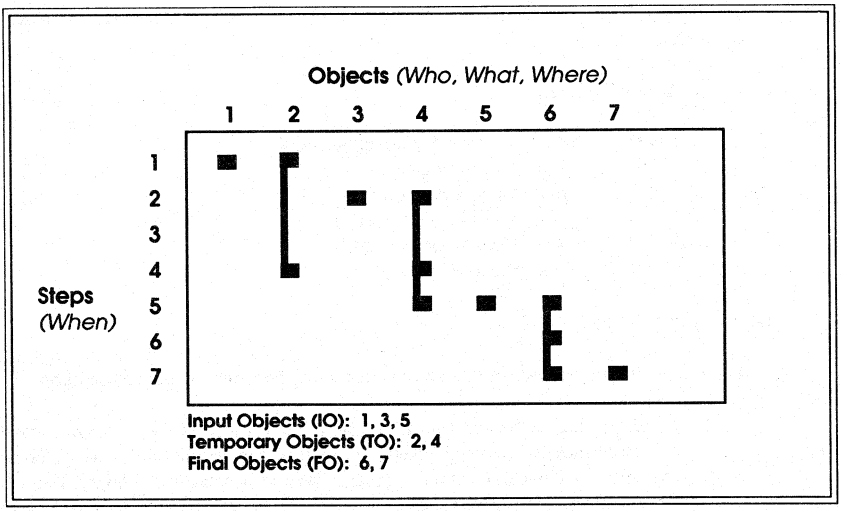


Diagram 1.

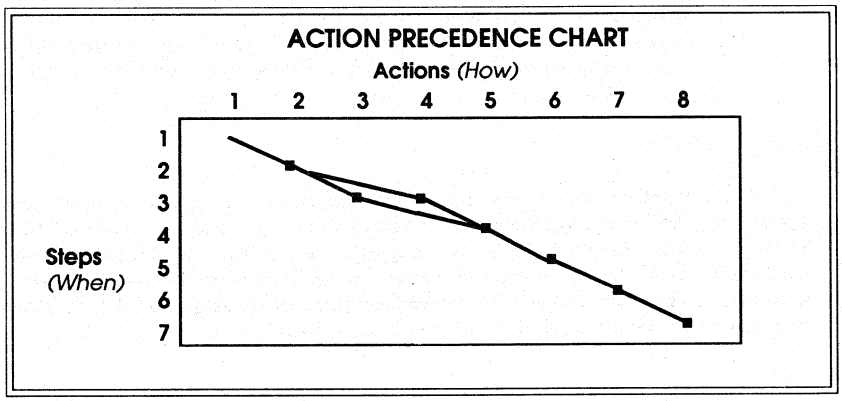


Diagram 2.

Given these descriptions, Action-Grouping, Actions, and Objects can be represented as the following equation:

$$X_o = AG(X_i, A_j)$$

where: **AG = Selected Group of Actions**
X = Specified Object
A = Specified Action
i = Domain of Object(s)
j = Domain of Action(s)
o = Desired output

2.4 System Knowledge Base

This is really the nerve system of the architecture which contains two distinct components:

1. Tables — are the logical representation of the real world business environment that: a) defines and describes all entities, Objects and actions within the system in any desired technical or natural language; b) describes the Object boundaries and Object conversions which are used to achieve certain Business Functions through Action-Grouping.

2. Generation of Executable Code — This utility compiles the Objects and actions based on the instructions in **Diagrams 1 & 2** and generates executable code for any given Business Function. Further modification or upgrading can easily be achieved by inserting the desired changes in **Diagrams 1 & 2** and this utility, in turn, will automatically produce the new executable code.

3. CONCLUSION

In sum, Object Oriented Design is not a mystical approach to system development, but rather a practical method to organize and manage information in terms of real world criteria. This is achieved by structuring the information according to the business Objects and breaking down the functions into their smallest possible code. Further, the flexibility of this approach is characterized by the automated loose-coupling of these entities as described in **Diagrams 1 & 2**, this allows the organization to respond quickly to dynamic business environment.

The Role of 4th GLs in the Integrated System Environment

Parvin Rahnavard

ELM Inc.
1250 S. Washington Street
Suite 603
Alexandria, Virginia 22314

INTRODUCTION

During the last decade, information has become a valuable and high priority item in most organizations. As a result requests for increased automation have piled up on the IS Directors desk. The introduction of 4th GLs to the market and the promises that these tools held; i.e. generation of code ten times faster than 3rd generation language, seemed to be the answer to the MIS dilemma. Although, this was helpful, unfortunately it led to a distributed environment where each business function was addressed independently, regardless of overall organizational needs. The symptoms of this approach are well known to all of us and include:

- Data inconsistency
- Lack of resource sharing
- Backlogs

1. 4th GLs, Blessing or Curse

The quick expansion of the decentralized environment was in large part fueled by MIS professionals in the design phase of application development. They viewed the organization as many departments rather than as a single integrated unit; other contributing factors were 4th GLs and off-the-shelf products. While these tools are indeed valuable for speedy development, they are designed for specific applications and in fact are unfriendly toward integration with other tools. In defense of 4th GLs however, I believe most of us in MIS will admit that these innovative products did allow us to learn more about the business environment and the exigencies of the marketplace.

2. What MIS has Learned

MIS has discovered that management of information in the changing world requires the dynamic system environment. Further, they acknowledged that the users are the professionals who are best suited to extract the raw data and apply the proper methods and techniques of data analysis and manipulation.

If we are to give users the freedom of data manipulation, then we must consider the following points:

a. Information mapping and method of viewing should be dynamic.

b. All data should be fed from one source in order to prevent data inconsistency.

c. All professionals outside MIS must have access to a variety of methods for data manipulation.

To support the joint participation between MIS and professionals the following system criteria must be met.

a. Creation of integrated data area, by integration we mean, data items are expressed once and are independent of business functions.

b. Grouping of the data must be expressed in terms of business language which will lead us toward Object Oriented design and programming.

c. Flexible grouping of data and quick access to information which will be implemented by the use of relational data base structure.

3. Next Generation of 4th GLs

By the end of this decade, the term *"4th GLs"* will, in all probability, be eliminated from software terminology. These days the latest *"hot item"* term is **CASE-Computer Aided Software Engineering.** In recent years, vendors have put together conferences and exhibitions to introduce the CASE concept. As a participant and member of the audience at several of these shows I can testify that these products are still in the initial development stage. Their definition and terminology remains unclear and by the same token MIS professional are more confused about the use of CASE. Here it might be appropriate to quickly review an example of such terminology found in a brochure advertising a conference to be held in October 1989.

CASE tool supports ——

- Information Engineering
- Structured Engineering
- Structured Development Aids
- Non-Integrated COBOL Generators
- Real-Time Applications
- Reverse Engineering and Re-Engineering

CASE diagraming techniques offers ——

- Entity-relationship diagram
- Data flow diagrams
- Decomposition diagrams
- Action diagrams
- Association matrix diagrams
- Property matrix diagrams
- Data navigation diagrams
- Process dependency diagrams
- Data structure matrix
- State transition matrix

Other considerations about CASE tools ——

- Strategic importance of fast development.
- SWAT teams (Specialists With Advanced Tools).
- Utilization of the most powerful available tool set.
- Intensive training of team.
- Maximum motivation for excellence.
- Experience with SWAT teams.

Now, I do not want to be a critic, but during the 6 years of my Ph.D. work in Operations Research which is a Scientific area addressing the study of Mathematical and Stochastic methods and models for real world application, I have never encountered expressions which parallel this outrageous syntax.

3. Simple words, Simple concepts to do our job

Figure 1 represents a communication link between a professional external to MIS and the information area.

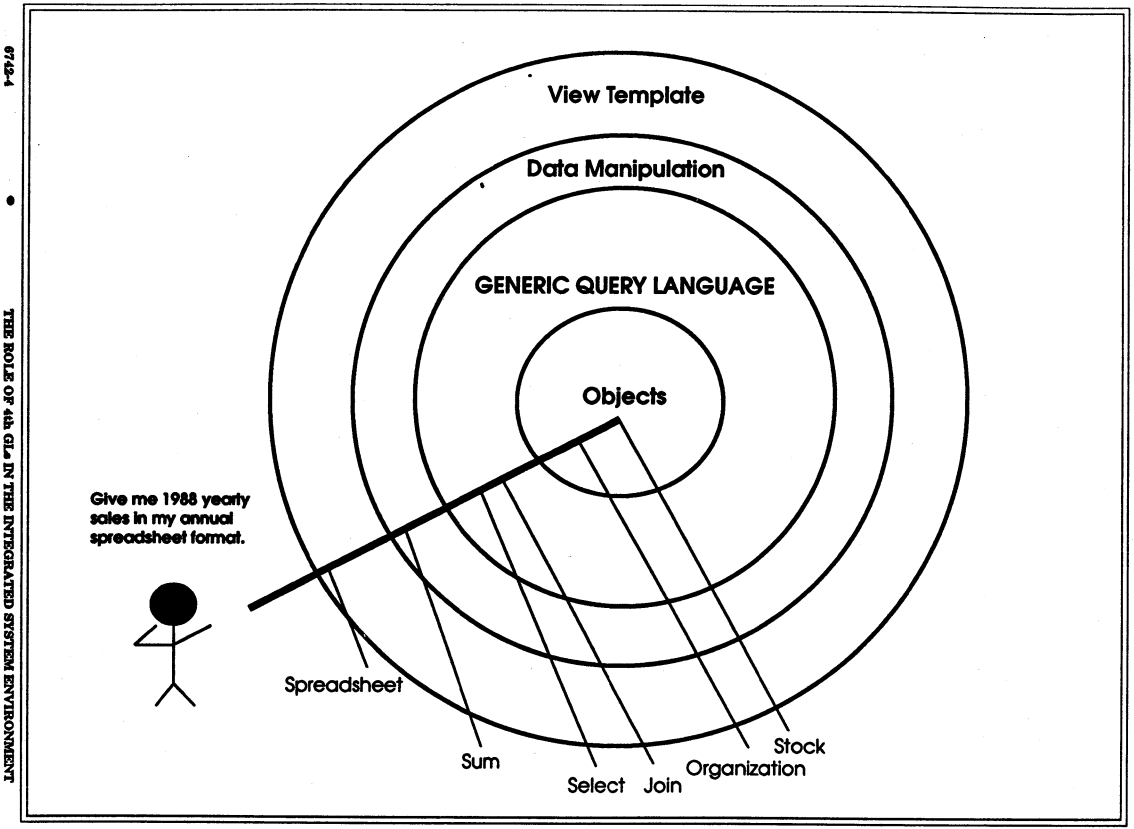

View Template
Data Manipulation
GENERIC QUERY LANGUAGE
Objects
Give me 1988 yearly sales in my annual spreadsheet format.
Spreadsheet
Sum
Select
Join
Organization
Stock

Figure 1.

Definition of the layers are as follows:

Objects where the information resides.

Generic Query Language which is the language to extract or input the information. The term generic will allow MIS to write general language code which relates to a characteristic of a group or class of objects.

Data Manipulation enables us to bring a measure of order to our information as well as to make a prediction about certain aspects of our business decisions.

View Template is a pictorial representation of the window opening (i.e. Graphics, Spreadsheet, etc) to the information area.

The example in **Figure 1** illustrates how the system can respond to a real world business request. The step by step process is as follows:

- Identification of the objects
- Extraction of the data through query language
- Manipulation of data
- Presentation of the extracted data in desired view

4. Open-Ended Conclusion

Lets make this session an open discussion among MIS professionals and Vendors to examine and address two major issues:

- Terminology verses concept—Currently the approach of applying different names to the same concept is misleading and will only lead to future complication and inconsistency.

- The need for greater communication between vendors and MIS professionals for the creation of an integrated architecture. This dialogue will hopefully clarify the responsibilities and expectations of both parties.

Profit Improvement Proposal for Network Implementation

...an objective tool to help companies make informed decisions.

Daniel Strauss
Hewlett-Packard Company
Product Support Division
100 Mayfield Avenue
Mountain View, CA 94043

Introduction

HP's Profit Improvement Proposal (PIP) for Network Implementation is a tool to help review the activities, financial considerations, and critical success factors involved in today's complex network implementations. The PIP makes the necessary activities explicit and eases the analysis of alternative network implementation solutions.

* Installation does not equal implementation:

Whether companies are installing a new network or adding connections to an existing one, starting up a network is a lot more complicated than just installing the data communications products. In addition to staffing, training, and procedural issues, hardware and software components must be correctly configured to work with each other, and each component must be tested to verify that it communicates properly on the network.

Starting up a typical eight node network could easily require working with 35 different vendors. The activities of the different hardware, software, network, and telecommunication vendors, at multiple sites, must be coordinated and the final configuration must be documented. Clearly, the strategic and financial consequences of network implementation errors can be substantial. It is not surprising that companies are now requesting single point of contact solutions for their multivendor network implementations more than ever before.

* The network implementation solution must fit the company and the network:

Selecting the appropriate network implementation solution depends on unique company and network factors. While many companies are convinced that self-service network implementation is more cost-effective than professional

network implementation, many also admit that they have not done any formal financial comparisons. A network implementation analysis is needed to effectively evaluate the benefits of professional solutions relative to self-service solutions. The Network Implementation Profit Improvement Proposal (PIP) is an objective tool to help analyze the relative costs and benefits of alternative network implementation solutions.

This paper reviews key network implementation factors, examines the benefits of both self-service and professional network implementation, and outlines HP's Network Implementation PIP tool. The PIP tool is designed to be used by the customer and HP working as partners in the analysis and design of the solution. A Network Implementation PIP outline is included in the appendix of this paper.

Key Network Implementation Factors

* Strategic versus tactical network implementation:

A clear understanding of the company's objectives is essential to an effective network implementation. Network implementation is either part of a broad, strategic mission aimed at increasing a firm's competitive position, or a narrower, tactical activity that in itself does not change the competitive performance of the company, but does improve one aspect of the network.

Strategic network implementation can take two major forms: a relatively large-scale project or a long-term project (phased). In analyzing a strategic network implementation, it is especially important to review the Network Implementation critical success factors (CSFs). The critical success factors are those activities or situations which must go absolutely right in order to achieve the firm's goals and objectives. For example, if the company's goal is rapid implementation of a new strategic network, then one critical success factor is the availability of an experienced network implementation project team. An obstacle to achieving this CSF might be a lack of adequate in-house resources. One implication of this obstacle might be the loss of a strategic business opportunity. Solutions that the firm could use to overcome this obstacle include training existing personnel, hiring additional skilled personnel, or using a professional network implementation service, such as HP's Network Startup.

The strategic/tactical framework is useful when analyzing the network implementation and developing the PIP. It is a framework that helps to keep both small and large projects in

proper perspective when deciding between self-service and professional network implementation solutions.

* Other significant network implementation factors:

Other factors that may contribute to the decision regarding the appropriate network implementation solution, include:

- o technological complexity of the network
- o number of different vendors involved
- o urgency of implementation
- o cost or financial considerations
- o risk (cost of implementation errors/delays)
- o geographical dispersion of the network
- o in-house IS personnel issues
  - availability
  - network implementation expertise
  - training
  - multivendor capability

Overall, the greater the complexity, time pressure, and multivendor network environment:

--> the greater the chance and cost of implementation errors and delays,

--> the greater the need for a professional network implementation solution.

Choosing the Appropriate Network Implementation Solution

* Going self-service:

Unique company and network circumstances influence the relative importance of the previously mentioned network implementation factors. Common factors that influence a company's decision to "go self-service" include cost or financial considerations, criticality of the network, and the size and technical orientation of the IS staff or company. Financial considerations are typically the primary determinants in the decision to "go self-service".

* Supplementing company capabilities with professional services:

Although the company and network circumstances may be very different, the issues and factors that influence a company's decision to use professional network implementation services

are typically the same as for the company that decides to go self-service: cost, project size, technical capability of the staff, efficient use of manpower, etc. A representative company comment is as follows: "As we get more involved in multivendor networking, the required level of technical expertise increases considerably. It may be more cost-effective to utilize the specialized expertise of outside vendors."

Some companies decide to use professional network implementation services when a project is of a critical nature and the in-house resources are not available (or have insufficient technical expertise) and would not be required after the completion of the project. These companies see using an outside network implementation service as a method of supplementing their staff, especially since trained personnel are such a scarce resource.

* Reducing risks with professional services:

Many companies indicate that they are hesitant to add full-time support staff to address professional services tasks during a period of economic uncertainty. Using an outside professional network implementation service allows these firms to satisfy their network implementation needs without adding to their operation budgets. In fact, one of the biggest advantages of dealing with a professional network implementation team is the elimination of most surprises. Companies want to know how much the network implementation will cost, how long it will take, and how the network will perform. With a professional network implementation service, these issues are resolved upfront.

* Network implementation solution break-even point:

All of the factors mentioned above indirectly relate to financial considerations and suggest there is a break-even point for self-service and professional network implementation service providers. This is an area in which the PIP analysis will be helpful in the decision making process.

HP's Network Implementation PIP Tool

The Network Implementation PIP Tool was developed to facilitate informed company decision making and solution development. By reviewing the activities involved in network implementation with the company, HP personnel can identify customer needs, analyze costs, and translate needs into solutions.

The PIP results will also be useful in presenting the analysis to the company's senior staff. Implementation is a highly visible time period in a network's lifecycle. Educating senior staff about the real costs of network implementation will help to justify the decision regarding network implementation services.

The main point in HP's Network Implementation PIP analysis is that there are some key factors to consider when making network implementation service provider decisions. The PIP involves the company and HP working as partners in the analysis and design of the solution. In this way, the PIP leads to a realistic assessment of network implementation costs and more informed company decisions.

Appendix: Network Implementation PIP Tool Outline

1. Company:
   identifies the primary network user groups.

2. Business Problem:
   describes the company's network implementation risks and concerns.

3. Solution:
   documents the preferred network implementation solution.

4. Company Situation:
   documents significant network implementation factors, including: the strategic or tactical nature of the implementation, technological complexity of the network, urgency of the implementation, financial considerations, implementation risks, personnel issues, etc.

5. Savings:
   describes the relative savings achievable via the various network implementation solutions.

6. Key Assumptions:
   documents company-specific and industry-based assumptions.

7. Investment:
   identifies expenses required to obtain the savings and achieve the solution.

8. Worksheets
   Section 1: analyzes network implementation alternatives.
   Section 2: summarizes quantitative data.

A New Window of Opportunity: Improving User Productivity Through Enhanced Interfaces

Chris Hauck
Jerry Felix
Hewlett-Packard Co.
Cincinnati, Ohio

Introduction

Information is power. In today's business environment, effective use of information can mean the difference between profits and losses, between gaining or losing a competitive edge. Data processing systems have traditionally provided the vehicle for delivering information to decision makers. Obviously, speed is critical. The faster the delivery, the more time available to interpret and act on the information.

The tools for storage and delivery of information are now well developed. In this paper, we examine historical and recent developments in the areas of entry, storage and delivery of data. We also discuss future directions in these areas.

Areas which are not as well developed are the presentation and analysis tools available to end users. We analyze these areas also, as the primary new opportunity for organizations today to improve the return on their current investment in data processing technologies. We discuss tools and methodologies which will reduce the time needed to act upon information, once it is received.

This paper addresses improving a company's information processing and profitability by embracing a cooperative host-to-PC integration strategy. It discusses improving productivity through better interfaces, both system-to-system, and system-to-user. It covers current products as well as new directions which the industry is exploring.

Information Flow

Years of development of computer technologies have led us to today's environment, where systems are available to provide fast access to data, and allow for quick development of new applications to deliver timely data to decision makers. But these years of development have concentrated primarily in the areas of faster hardware and software, and easier-to-use development tools for programmers. The computer industry,

and its users, have focused on the delivery of the data, rather than the presentation or analysis of information.

Table 1 shows the various stages of data passing to the decision maker, where it is finally acted upon. In order to increase the benefit of information systems, we must reduce the overall time required between the time of data entry, and the time when action is taken. The first three stages of this model have been the primary focus for data processing professionals in the past: entry, storage, and delivery of data.

Table 1

Definitions of Data Flow Stages

Stage	Definition
1. Entry	The process of putting data into the system
2. Storage	Insuring data integrity, security, backup
3. Delivery	Retrieval upon inquiry, summarizing, routing to user
4. Presentation	Displaying or reformatting information for analysis
5. Analysis	Manipulating and correlating information
6. Action	Intelligent decision making based on information analysis

Now, the industry is taking a bold new direction. The focus has shifted for many computer hardware and software manufacturers. Getting the data to the decision maker is only half the challenge. We are now at the point where we are inundating the decision maker with data. One of the current challenges before the computer industry is to allow the end user to quickly and effectively turn the incredible volumes of data, which are stored in the computer system, into information. We are making a subtle, but important distinction here: data is virtually useless in its raw format for the decision maker; information is the result of summarization, organization, and manipulation of raw data into a concise and comprehensible format.

Tools are available today which allow the decision maker not only to receive valuable data in a timely manner, but also to better manipulate and interpret the data. These tools are relatively new to the industry, and therefore are not nearly as sophisticated as delivery tools. The delivery tools are well developed, through many years of effort; it is the presentation, analysis, and action tools which are new, and provide the most opportunity for businesses today.

The shift in the industry has been influenced by several related factors. First, advances in personal computer technologies - faster processors, less expensive memory, and high-resolution displays - provide the opportunity to present information to the user in a graphical format. Second, alternate input devices, such as mice, touch screens, and digital notepads have become available which allow the user to comfortably control the workstation. Third, PCs have sprouted throughout organizations, as they became status symbols in Corporate America. Suddenly, middle and upper management began demanding return on the huge investment of PCs, which often sat idle on users' desks. Finally, users have demanded easier-to-use systems, which include built-in help facilities, consistent user interfaces, and iconic access to programs and data.

Unfortunately, even though the computer industry has taken this powerful new direction toward PCs, businesses have been slow to embrace the full potential of these technologies. Many companies are locked into the old way of thinking, and overlook the new opportunities. This paper will explain this new opportunity, the opportunity to improve the profitability of your organization, by improving the time to action: the amount of time it takes to act on data after it is entered into the system.

User Model: An Opportunity Analysis Tool

In order to effectively analyze opportunities for improving the time to action, we must first define the different types of users which are typical in most businesses. Users of computer systems cover a broad spectrum. On one end, there are users whose primary function is entering data. They typically are low in the organization, and have a very microscopic view of the data. Conversely, the decision makers are at the other end of the spectrum. Their function is to draw conclusions based on the information as it exists. Their view is typically macroscopic. Somewhere in the middle of the spectrum is the casual inquiry user. This type of user, whether a low-level manager, or a customer service telephone representative, typically performs occasional inquiries or updates on the data.

Usually, improving the accuracy and timeliness of data can have the most benefit to the organization at its higher levels. It is the upper-level managers' decisions which can benefit the most through the use of accurate data. Most businesses, however, focus there investments and efforts on improving systems for first-level employees. Herein lies the opportunity for improving profitability through more effective use of computer technology.

Typical User Needs:

To examine the data flow model, and better understand areas for improvement, we

begin by categorizing the needs of the various users in the model and highlighting the critical success factors for each type. Once these needs and success factors have been identified, the alternative investment opportunities and their associated returns will become clearer.

In most businesses today, it is the data entry clerk who initiates the process. They are the contact point between the real world and the computer system; therefore, the integrity of the data entered becomes the foundation for all future use and analysis. This type of user's primary request of the system is data storage and validation. The user's view of the data and the system at this point is very narrow and focused only on the data in hand. There is little need for any understanding or interpretation of the data. Typical jobs which would fall into this category are: entry clerks (orders, payroll, and accounting), telephone marketing representatives and other high-volume data entry positions. Speed and accuracy are the two primary measures of success at this level.

The second type of user in our model is the person who performs casual inquiries and updates. They use the system fairly regularly and are familiar with the steps necessary to perform their work. Their primary requirements of the system are ad-hoc inquiries, file updates and maintenance, and reporting. At this level, the user's view of the world opens up, and the need for correlation between sets of data begins to become a factor. The user is more concerned about orders, balanced budgets, and sales - that is, summary information - rather than the details of particular data items. The need for understanding the data, transforming the data into information and knowledge, becomes very important. Typically, this class of users consists of supervisors and departmental managers. Ease of inquiry, reporting flexibility and response time for file access are the primary success factors at this level.

The third class of user in our model is middle and upper level management. Their on-line usage of the system ranges from seldom to never. Although their usage of the system is low, their need for utilizing the data is high. Frequently the data is digested by reviewing reports or through conversations with staff personnel who are using the on-line systems. This leaves the third class of users faced with a major problem: drowning in data but starving for information.

Their view of the world is very broad, and is concerned with correlating data across many business units. They need to be able to manipulate the data at a very high level, organizing and re-organizing the data to try to understand trends, relationships and opportunities. Critical factors affecting the manager's success are: data accuracy, ease of organization, intuitiveness of the system, flexibility of presentation and speed of analysis.

Table 2 summarizes the characteristics of each user in the User Model, including

typical system requirements, amount of system usage, and critical success factors. The next section looks at various tools used historically to meet the primary needs of each user type: data entry tools for one end of the spectrum, inquiry tools for the professional staff, and information analysis tools for the middle and upper level managers.

Table 2

User Model

Typical Job Description		
Clerical Telephone Marketing	Professional Staff Supervisory	Middle Management Upper Management
System Requirements		
Data Entry	Inquiry	Analysis
View of System		
Narrow	Medium	Very Broad
On-line System Usage		
Continuous	Regular	Seldom
Need for Data Interpretation		
Low	Medium	High
Impact of Decisions		
Little Impact	Limited Impact	Widespread Impact
Critical Success Factors		
Response Time Data Accuracy	Inquiry Flexibility Report Flexibility Inquiry Response Time Update Response Time	Data Accuracy Ease of Organization Intuitive Interface Presentation Flexibility Analysis Speed

Historical Perspective: The Evolution of Technologies

Data Entry Tools

The industry has seen an evolution of tools and technologies to meet the needs of the customer. The initial emphasis was on improving those tools necessary to get data accurately into the system. Data processing systems were focussed on automating the routine: accounting functions, customer orders, and production schedules, as examples. This step in the evolution was easily cost-justifiable, even with the huge price tags associated with the early computer systems. The savings in labor alone usually was the incentive for spending vast sums on what can now be done on a PC. Of course, these were logical decisions at the time.

The data entry function has evolved from punch cards to on-line systems. Early on-line systems allowed for character mode or line mode editing, as well as immediate verification of some data fields, eliminating mandatory batching of data. Advances in data communication have enabled system designers to use the screen as the window into the computer system - block mode transfers of screen data are commonplace now. Block mode's popularity was boosted by a number of contributions to the on-line data entry environment: a common "look and feel" for on-line applications, standardization within the on-line programs, and reducing the time-consuming operation of data transfers to the terminals.

In the HP 3000 environment, the evolution began also with punch cards. The card reader was an optional input device on the Series II model. Fortunately, this was quickly abandoned for the much more powerful DEL (Data Entry Library). Eventually, VPLUS evolved (originally View/3000, then V/3000), and exists now as the most common user interface in use today on the HP 3000.

But technologies have changed dramatically since VPLUS came on the scene. Just by examining the advances in data entry hardware, one can see that software development for, and user acceptance of, new technologies have significantly lagged new product introductions. Data entry devices in the HP 3000 environment have evolved from card reader and dumb terminals to intelligent terminals which could handle block mode and forms caching. Block mode marked the beginning of a new era in interactive program design, but soon users grew impatient with the occasional long delays in displaying complex data entry forms. Form caching terminals, like the HP 2624B, addressed this problem by eliminating the redundant downloading of forms to the terminal. After having downloaded a form once to the terminal, a short escape sequence would almost instantly redisplay the terminal-resident form. Developments eventually progressed to co-processing terminals, as in the case of HPWORD. The HP 2626W and HP 2628 terminals would handle the user interface to word processing, while communicating with a host HP 3000 for file serving. This idea may have

been ahead of its time. Current implementations of such cooperative processing environments have worked out many of the restrictions which plagued HPWORD.

Another technology which has been keeping pace with advances in end user devices is data communications. A number of devices were introduced for the HP 3000 to mask the limiting speeds of the communication lines (forms caching, HPWORD terminals). Originally supporting baud rates up to 2400 BPS, the HP 3000's communication capabilities were soon found to be inadequate for many users. In time, 9600 and eventually 19,200 BPS terminal controllers were introduced and the need for some of the caching technology declined. Today, local area networks (LANs) are providing transmission rates which were undreamt of only a few years ago. These communication speeds are allowing the PCs to play a much more integrated role in the HP 3000 environment. By effectively integrating these technologies, the mini- and the microcomputers can form a seamless environment where each processor performs individual and cooperative functions, thus allowing the user to concentrate on the "what", and not the "how" of the problem at hand.

HP is one of a number of companies supplying software for unifying the HP 3000 and PC environments. Third party tools exist which allow application development to take place on the PC platform, and execution on the HP 3000. This is ideal for those shops where there is the constant struggle between development staffs and production users for system resources, since editing and compilation functions are off-loaded to the individual's PC. Other tools allow for sophisticated user interfaces to be developed on the PC using "windowing" and mouse-driven input. Also, a number of tools have been introduced to allow greater transparency in the integration of data located in TurboIMAGE and PC databases.

In certain environments, particularly in management circles, the mouse has not been widely accepted as a comfortable input device. Part of the reason for this perception is that applications have not been written to fully exploit its capabilities, but this does not mean the mouse should be discarded as an option.

The technology of PCs, along with their mice, and various powerful graphical user interfaces (GUIs), provide us with a new area to improve data entry facilities. The opportunities are tremendous. It's up to the consumer to take advantage of them.

Future technologies may reduce the need for manual data entry and clerical functions. Electronic data interchange (EDI) is already in use in limited markets and provides a mechanism for linking the delivery stage of one company into the entry and storage stages of another. Obviously, this technology requires well-defined standards to bridge systems of multiple companies.

Data Inquiry Tools

Most of the major developments in the area of data storage and inquiry tools predate the HP 3000. Early implementations of storage tools include random access files, keyed files (ISAM, KSAM), and the first databases. Each of these access methods became available early on in the HP 3000's life, which is likely the reason for its great success and long life. IMAGE evolved with MPE into a tightly integrated, high performance DBMS. Adequate for the relatively small user load of the early-model HP 3000s, IMAGE's limits soon became shackles around the ankles of application developers on the higher-end models. The key was in loosening the limits by making the move to TurboIMAGE. Now, with the introduction of even larger and faster Precision Architecture systems, further enhancements to TurboIMAGE have taken place, with more anticipated in the future.

Relational databases are increasingly finding their way into applications on the HP 3000. Providing a simplified view of the data, and promising greater industry standardization, relational databases, such as SQL, are giving the user new methods for storage and retrieval of data. This new interface can provide a common "look and feel" to a database, regardless of the actual structural implementation on a particular hardware platform; however, this is only a small step in the right direction toward a true user-friendly interface. Although SQL database management systems have not been as tightly integrated into the MPE operating system as TurboIMAGE and consequently have not achieved as high a level of performance, they are beginning to carve their niche in the less-structured data inquiry areas.

There are a number of newly emerging database technologies which are showing promise for widespread acceptance. Among these are object-oriented database management systems. These databases store and retrieve objects of unknown structure or type, such as documents, spreadsheets, or procedures. The entire area of object-oriented methodologies is very appealing to a wide community of users, since large and complex data structures and files can be manipulated very easily, without the user needing to know the details of how to do the manipulation.

The primary breakthroughs in the storage and inquiry fields in the 1980s have come on the PC. Easier-to-use databases have become available, allowing the business professional a new method of storing and manipulating information. These have become popular not only because of their ease-of-use and low cost, but also because they allow the user to maintain full ownership and control over his or her data. These are ideal for non-shared databases, but of limited value in the more typical world of shared data. However, they have given the industry a good lesson in user friendliness: mini computer database vendors can learn from PC packages. Both the mini and PC database vendors, though, should strive for further improvements in their interface.

Information Analysis Tools

As the computer industry matured, it became apparent that an opportunity existed in the area of information analysis tools. Production managers found that daily reports provided a wealth of information which enabled cuts in inventories or better production scheduling. Financial analysts required their monthly statements to verify their business situations. Throughout organizations, people realized that if they had access to the precious information residing in their computer systems, then they could make better, more timely decisions, and positively impact the profitability of their business.

The printed report became the first means of analysis for the business manager. Tools were developed to enable programmers to better meet the reporting requests of management. But requests accelerated, and MIS backlogs increased. As management received valued reports, more and more were demanded. In the mid 1970s there were predictions that in order to meet the ever-increasing demand for information, the entire population of the world would need to be computer programmers by the turn of the century.

This prediction proved to be not so far fetched. In order to meet demand, the tools for end users have become more sophisticated, enabling novice users to be able to generate their own reports. In effect, as programming tools increased in sophistication, end-users replaced the functionality of the early programmers.

Then came the PC revolution. The primary reason for the success of the personal computer came early in the 1980s: the electronic spreadsheet. This tool finally gave the end-user the ability to manipulate and analyze information in an easy-to-use, WYSIWYG (What You See Is What You Get) package.

Other information analysis tools have become more sophisticated as well. Packages solely based on statistical analysis have become popular. Also, a variety of software exists for aiding the decision maker in organizing and presenting the information from PC databases in simple, easy-to-understand graphical formats. End-users are producing reports and providing pictorial representations of the data never before possible with the corporate mainframes.

So now, the users' appetites have been whetted. They know the awesome capabilities of the electronic spreadsheet packages. They have seen the power of the data manipulation tools of the PC. Now, they can just imagine the endless possibilities, if they only had the ability to easily use these tools to analyze the tremendously valuable information lodged in their mini-computer. In most businesses today, unfortunately, this requires a degree in Computer Science to access the mini-computer data, retrieve the information in a suitable format, transfer the file to a PC, and load it into the spreadsheet package. This shatters the hopes of even the most ambitious, leading to

the common method of retrieval today - keying in the summary information all over, on the PC. We have managed to reduce our highly-paid decision makers to the role of data entry clerk.

This provides us with a huge opportunity for improvement. Although new tools are being introduced to assist us in the transferring of data from the HP 3000 to the PC, we must increase our use of tools which enable the infrequent user to access and analyze information. We must provide easy-to-use systems, with consistent user-interfaces, and self-explanatory command interpreters to the people who can benefit the most from accurate and timely information - our management.

The Foundation for Future Systems

Thus far, we have objectively analyzed the evolution of the computer industry with regard to the uses of information. We have examined the perspectives of data entry, data inquiry, and information analysis. It is appropriate now to discuss strategies to take when examining future investments in corporate data processing systems. Without knowing which way the industry will turn in the future, though, we risk sinking time and money into directions which may become quickly obsoleted. Some companies fell victim to this by committing themselves to pre-MS-DOS PCs and ended up on a dead end path, as the software industry standardized.

A sound strategy is to formulate a single foundation across applications for the entire user base, and build an extension upon that foundation for each user type in areas which require specialization. This is depicted below in Table 3.

Table 3

Model of System Foundation and Extensions

Entry	Inquiry	Analysis
Data Entry Extension	Data Inquiry Extension	Information Analysis Extension
Common Foundation		

Since our goal is to make decisions which will positively impact our companies' profitability, we must base our investments on what will provide the greatest return. Areas in the computer industry which are new and untested could require substantially larger investments than those which have proven themselves with time. The passage of time allows user acceptance and industry standardization, which will reduce the risk of wasting investment money and efforts. In order to reduce that risk, and insure maximum return on investment, you must choose strategies which stay with proven technologies, and appear to be endorsed by industry standards.

Common Foundation Across All Users

Today's technologies of the mini-computer and PC seem to be a sound basis for investments in future systems. The mini-computer has proven itself to be a valuable asset in workgroup computing, and the PC has emerged with strengths of its own. Those companies betting on one or the other would be wise to reevaluate their strategy, and accept that both are here to stay. Now it is important to utilize the strengths of each component, to achieve the best overall end result.

The strengths of each component (the mini-computer and the PC) cannot be evaluated on technical merit alone. User acceptance and standardization must also be weighted into the overall evaluation. We have come to the following conclusions:

The mini-computer has proven itself in the areas of high-volume data management, file- and resource-sharing, data security and integrity (better backups and recovery), multi-user applications, high speed and high capacity peripherals, mature applications and networking tools, and overall hardware component reliability. Although today's PCs are technically able to compete with the mini-computer in most of these areas, the strength of the mini is in its years of industry acceptance. It is not likely to be displaced in the near future by the PC or even the PC acting as a server in a network.

The PC has its strengths primarily in its user-friendly interface. It can provide powerful graphics, quick response to data manipulation, smart user interfaces, alternate input devices (such as a mouse), and low operating system overhead (although the overhead for OS/2 increases). As a dedicated processor, it can provide more predictable response times to heavy computational demands. Future industry directions almost insure faster compatible processors to further its strengths in each of these areas.

A systems strategy foundation should exploit the strengths of each technology, as listed above. Table 4 shows how these strengths fit into the normal flow of data for intelligent decision making. As you can see, the PC is the primary user interface, while the mini-computer acts as a hub, for data sharing, peripheral sharing, and backup.

Table 4

Analysis of Alternatives for Data Flow

Stage	Best Alternative in a PC-Mini Environment
1. Entry	PC: Strong user interface, quick response to entry
2. Storage	Mini: Best for data sharing, backup, large quantities
3. Delivery	Mini: Mature networking, good retrieval / summarization tools
4. Presentation	PC: Strong user interface, quick response to user
5. Analysis	PC: Dedicated to a user, good graphics, WYSIWYG
6. Action	Mixed: Networked mail system, artificial intelligence tools

A set of core functionality is required for each user environment. On the PC, the hardware, operating system, networking services, database services and user interface tools provide this foundation. The HP 3000 also requires the necessary networking hardware and software, database services and tools for interfacing to the PC. With this core functionality in place, the groundwork is laid for implementing the specific functionality required by the end users of an organization.

For today's corporations, choosing a PC platform usually means choosing an Intel compatible processor and an MS-DOS based Vectra-compatible system. This has been the PC of choice for the past few years, and with promises of future processing power from Intel, will likely remain in that position. On the other hand, choosing an operating system for the PC is not as simple as it was a year or two ago. With the release of OS/2, OS/2 Extended Edition and continuing enhancements to MS-DOS, the waters became muddied. MS-DOS is safe. It is established, has a large base of software applications and tools, and is relatively frugal with its resource consumption. OS/2 on the otherhand, is relatively new, more robust, and allows true multi-tasking. While it can run MS-DOS based applications, it takes much more memory and horsepower to drive the system, meaning a larger investment in PC hardware for each user. For the majority of users, the PC of choice will remain the 80286 or 80386 processor running the MS-DOS operating system. It may not be the "latest and greatest" operating system, but it still has a very bright future, and the volume of software available will make it difficult to be replaced easily. Additional accessories to assist in making the user interface as intuitive as possible, would include a mouse and a high resolution color display.

A windowing system also seems to be an appropriate piece of a data processing foun-

dation. This type of graphical user interface provides a consistent method of viewing and interacting with applications, a flexible and consistent screen design, as well as a multitasking environment capable of juggling several user applications concurrently. By standardizing on a single windowing system for workstations, an organization can improve the utilization of systems and the productivity of its users since the mechanics of using any application is identical. The learning curve necessary to use and understand an application is decreased, and real productivity can be achieved much sooner. Microsoft's MS-Windows is the strongest windowing system on the market today, due not only to its technical strengths, but its market acceptance as well.

Presentation Manager is the windowing system available with the OS/2 operating environment. It is very similar in appearance and functionality to MS-Windows, and since it is available exclusively under OS/2, is probably not the ideal solution at this time due to economic considerations.

High speed and compatible networking services are crucial in providing the pipeline for the data exchange between the PC and the HP 3000. Without a fast link between the two, the potential synergy of the HP 3000-PC connection is greatly reduced. In the HP 3000 environment, the interface of choice is the 802.3 local area network and NS (Network Services). In addition to providing very high transfer rates (10 MBPS), this link also provides common services on both the HP 3000 and the PC, a full range of diagnostic and trouble-shooting utilities, and the flexibility of concurrent multi-system access. With this capability available to software developers, the communication limitations which have been a hindrance to many types of cooperative processing applications are being removed. The PC can now realistically access and manipulate the large volumes of data maintained on the central system. Previously, accesses were limited to a small subset of the total available data, since just the transfer time involved made the task impractical. It is now feasible for the user at the PC to analyze and browse large amounts of information, without even being aware of the data's location. It is ironic that originally hardware was developed to mask the communication speeds and now, the communication speeds can be used to mask the hardware.

Since interfacing with the HP 3000 and sharing data across the network will involve the storage of data on the PC, a data storage strategy should also be addressed. With the widespread use of databases on mainframes and minis, it is no surprise that databases have found their way to the micros. Examine the list of best-selling PC software, and their use will become evident. Although we do not make a recommendation on a particular database package in this paper, it is important to consider the following points when examining the alternatives.

- Is the package established and has it been proven in the business community?
- Are there other utility software packages that work in conjunction with the database for report generation and program development?

- Is the data easily accessible from other packages or programming languages?
- Are there utilities for easy importing and exporting of data?
- Is the data compatible with any DBMS available on the minis?
- Is the software networked? Can concurrent users be accessing the same data files?
- Does the inquiry interface allow for the level of diversity required?
- Is there a standard interface language for minimizing user training? (SQL, or query by example)

Data Entry Extension

With the core functionality established, building upon this foundation and implementing the tools necessary to allow each type of user to reach their maximum potential is the next step in the process.

In breaking down the data entry task into its components which impact the speed and accuracy of entry, the following are contributing factors:

- Typing speed of user
- Screen display rate
- System delay between fields and records
- Order of fields on screen vs. paper copy
- Accuracy of original document
- Keystroke errors by the user
- Level of field and record validation by the system

Some of these factors are outside the scope of the system, and others suggest technologies which are not feasible today. For example, optical scanners are available, but are not fast enough or accurate enough for this type of application. However, bar coding provides a fast and accurate means of entering data at a relatively low cost. To alleviate the screen time and system delay problems, it is necessary that the entry form be resident on the PC. This forms package should be able to provide fairly sophisticated edit checks, check digit generation and validation, and ideally provide an interface for external data file cross-referencing. This would provide for the user the ability to key in data as quickly as possible and detect errors immediately. The data entered could either be stored locally in a database file for later batch transmission to the mini, transmitted immediately to the central system, or passed to a background transmission process in a multitasking environment.

Data Inquiry Extension

Since users involved in data inquiry are dealing with a larger volume of data then the data entry class, the need for efficiently managing the data becomes more crucial.

Less rigid constraints on the user interaction with the system, and greater flexibility in the manipulation of the data, begin to play a more significant role in the effectiveness of the user.

By examining the critical success factors from Table 2 for this user, the extensions necessary to support their needs become evident. The use of report writers and 4th generation languages allows ad-hoc access to the databases while at the same time reducing the need for MIS involvement in supporting one-time programming requests. Each user has the ability to apply their own creativity in organizing the data to meet their needs. Some of the best applications for supporting these functions execute on the PC platform; therefore, it is necessary to have some method of seamlessly integrating the HP 3000's data with the PC programs. Tools like Information Access, Data Express, Omniview and Oracle will allow for this transparent data exchange between the HP 3000 and PC applications.

Often, a single tool does not provide all of the required functionality. Many PC applications, especially those written using the MS-Windows interface, can allow for very transparent PC application to PC application data sharing. Using MS-Windows terminology, a "clipboard" is used to temporarily hold data from one application before being "pasted" into another application which is usually on the same screen in a different window. As an example, this clipboarding may allow a user to be organizing data in a spreadsheet, clip a portion of the data, paste the data into a graphical presentation program, and instantly view the data in the form of a pie chart.

Applications such as NewWave, which are giving users a glimpse of the potential of advanced user interfaces, also provide additional capabilities to simplify the redundancy of certain tasks. NewWave's Agent facility allows a user to show the system how to do a specific function by recording the steps involved, then at any time having the system re-play those steps to perform the task. This is ideal for such operations as moving data from a HP 3000 database into a spreadsheet, or taking the weekly sales figures, generating a bar chart from the data and electronically mailing the results to a list of users.

Information Analysis Extension

Users whose primary job relates to information analysis have traditionally been left out in the cold when it comes to on-line computer assistance. The computer industry has been very successful in the concrete handling of data; however, it has been very slow in developing applications for the more abstract manipulation of information. And unfortunately, assisting in the task of analysis for this type of user involves a substantial amount of high-level data control. The degree to which the computer can work with the user on the user's level relates directly to the user's productivity and profitability.

Technologies which have been evolving for years are beginning to find application in these areas. With the introduction of spreadsheets, data became pliable. It could be molded and shaped into a format that allowed users to extrapolate information and perform "what if" modeling. Combining this capability with high quality graphical presentation packages allowed the users to conceptually convey their analysis. Statistical programs also allowed the computer to perform trending and further analysis of the data for the user. With these tools, the users' demands for on-line access to the mini-computers' data increased.

At this time, our recommended spreadsheet would be either Lotus 1-2-3 or Microsoft Excel. Both of these products have wide acceptance in the business community and have a strong future for continued use. Excel has the an advantage over 1-2-3 in that it is MS-Windows based and provides that common look and feel of most other windowed applications. Although 1-2-3's interface does not have the same user-friendly features as Excel, it is very common in the user community, and therefore deserves consideration.

In general, this class of user does not use the system as frequently as other types of users. Therefore, applications and interaction with the system at this level must be much more intuitive and friendly. This is one of the major reasons why environments such as NewWave are prime candidates for providing an interface to the system. The user can be isolated from the details of how to get a task accomplished to a greater degree. Features such as the Object Management Facility allow the end user to manipulate groups of complex data structures as a single entity. For example, by simply moving a mouse cursor on the screen to a spreadsheet icon, clicking on that icon to "pick it up", and moving it to the mailman icon can automatically instruct the PC to electronically distribute that spreadsheet to a specified distribution list. The user is freed from having to know any special commands necessary to handle a PC file in the E-Mail program. The object orientation of this approach is what allows graphics, word processing documents and any number of other file formats to be treated in the exact manner, even though internally the system must work with each object type uniquely.

Since less usage of the on-line systems by these users usually implies a lower degree of typing skills, minimizing the amount of keyboard interaction with the system is mandatory. Products such as natual language interfaces, digital notepads, voice storage devices and voice recognition systems allow for a more natural interface with the system. By allowing a user to respond to an electronic mail message by simply writing his thoughts on an electronic notepad reduces their frustration with battling the keyboard. Again, this example illustrates the potential for an object oriented methodology in application design, since the method for acquiring the data is inconsequential - the user does not need to manipulate the data any differently than if they had entered it via the keyboard or had spoken into a microphone.

For years the term artificial intelligence has been a buzzword in the computer industry, often fueling the debate over the possibility of replacing people with computers in the business world. As business applications of AI become more refined, we feel that its greatest potential for impact will be for the information analysis user. Expert system applications and other fields of AI will allow for preliminary analysis of the data by the computer, thus permitting the decision maker to concentrate on further analysis and recommendations.

As industry acceptance of these types of applications and products increases, their use will serve to free the user from the constraints of the current application environments. This is not meant to imply that every piece of technology that is developed is valuable for each type of user in every circumstance, but that as new technologies emerge, each should be evaluated for its potential impact on the various segments of the user community.

Table 5

Summary of System Foundation and Extensions

Entry	Inquiry	Analysis
Forms Package	Report Writers	Spreadsheets
Bar Coding	Information Access	Graphics Package
Key-To-Disk	Natural Language Interface	Natural Language Interface
	Clipboard Functions	Clipboard Functions
	Agents	Agents
	Statistical Package	Statistical Package
	E-Mail Interface	E-Mail Interface
		Digital Notepad
		Voice Storage
		Object Management
		AI Tools
		Hot Links

Communications	Interface Software	DBMS
Operating System		
Hardware		

Conclusions

This paper presents industry observations regarding opportunities for improving functionality of information processing systems, by taking the approach of improving interfaces. Increases in profitability may be attainable by providing decision makers with valuable information faster, and making it easier for them to interpret, process, and act on such information.

Cost of implementing such systems is relatively easy to quantify. Unfortunately, the benefits are not easily measurable in dollars and cents. It is a safe decision to avoid implementation of a plan when the potential return is unknown, but doing nothing could be even more costly. Unquantifiable costs associated with lack of action include:

- Lost business opportunities due to untimely access to information
- Wasted managerial time and effort attempting to use antiquated systems or reentering analysis information
- Missed opportunities to gain a competitive edge
- Loss of customers' business, due to perception of lack of commitment to new technologies
- Loss of productivity, because of user frustrations with current systems

We have made recommendations for improving interfaces, both system-to-user and system-to-system. Our recommendations are based on our experience and exposure to various customer environments, as of April, 1989. Obviously, your actual situation will determine deviations from our recommendations, as will changes in technologies and industry directions.

In order to apply our recommendations to your current environment, we suggest that you first apply the user model, to determine the potential beneficiaries of timely access to information. Remember to include middle and upper management, who may not currently be on-line users. Creatively imagine the possible gains of implementing a system which provides decision makers with timely information and easy retrieval and analysis.

Next, formulate a strategy for you own site, which includes a common foundation across the user base. We feel that this foundation should include the workstation and operating software. However, it may be appropriate for some companies to provide their data entry users with lower cost workstations (terminals or low-cost PCs), rather than the complete workstation used by the decision makers.

Finally, determine the appropriate extensions necessary for each class of user. Base

your decisions on ease-of-use, integration with other packages, industry standards or acceptance, and reputation of the vendor. Formulate your strategy for the present, as well as the foreseeable future. Then, reevaluate the strategy at least once a year, to include changing technologies and user needs.

This sort of analysis should lead you to an environment which improves access to valuable information already existing in your automated systems, particularly for those who can benefit the most, the decision makers. By improving the access and analysis tools for these users, you will increase the return on your information processing investment, and ultimately improve the profitability of your organization.

Products mentioned in this paper are listed below. These products are currently or were previously available through the company listed.

Hewlett-Packard:
HP 3000, Series II, DEL, VPLUS, HP 2624B, HPWORD, HP 2626W, HP 2628, KSAM, IMAGE, TurboIMAGE, HP SQL, MPE, NewWave, Vectra, NS/3000, Information Access

Microsoft:
MS-DOS, MS-Windows, Excel

Microsoft and IBM:
OS/2, Presentation Manager

IBM:
OS/2 Extended Edition

Intel:
80286, 80386

IMACS Systems Corporation:
The Data Express Series

Dynamic Information Systems Corporation:
Omniview

Oracle Corporation:
Oracle

Lotus Development Corporation
Lotus 1-2-3

INTEREX SPEAKER SHORT BIOGRAPHY

DAVID SWINKIN is a Personal Computer System Engineer with Hewlett-Packard. Mr. Swinkin has extensive experience installing and training corporate personnel on the use of desktop publishing solutions, laser printing and workstation configuration. Mr. Swinkin has presented a paper on local area networks at the 1986 American Inventory and Production Control Society National Convention and has toured with the Hewlett-Packard/Aldus/Microsoft Desktop Publishing Tour in 1987. Mr. Swinkin has taught as a member of the adjunct faculty at Quinnipiac College in Hamden, Connecticut.

Applying Project Management Techniques to the Implementation of Application Software Packages

Jane Fastiggi
Hewlett-Packard Company

Much has been written about project management as it relates to the management of software development projects. But today, many - if not most - HP 3000 customers purchase rather than write their major application software. The implementation of this application software must be managed just as carefully but it frequently is not. This paper deals with the use of formal and informal project management techniques in the implementation of application software packages. What follows are recommendations on how such a project should be managed, and the final section contains examples of what commonly goes wrong and why.

Why Project Management

Project management is a set of formal techniques for the planning, organizing, directing and controlling of a project. Project management has been used for many years by large aerospace and defense companies to manage multi-year and multi-million dollar development projects. It has become increasing popular in recent years in businesses of all types and sizes, and especially for data processing projects. This is because projects have become more complex and - more importantly - because the projects that organizations attempt today cross the traditional hierarchical management lines. It is this situation - the implementation of an integrated manufacturing and financials package or a payroll and personnel application, for example - that makes project management critical to success.

The Phases of a Project

The formalization of project management has two main aspects: a definition of distinct phases, tasks and milestones involved in the project, and a set of techniques for the management and control of the project. Each aspect is equally important.

Formal project management in a data processing development environment involves the following phases:

* Project Definition, in which the project and its scope are formally defined;
* Feasibility Study, in which it is determined whether or not the project can and should be done;
* Systems Analysis, in which the overall project is outlined and planned;
* Design, in which the detail design of the project is done;
* Programming, in which the actual coding and unit and systems testing are done;
* Test, in which the user-acceptance tesing is done;
* Installation, in which the end-user training and data conversion are done, and the system goes into production; and
* Maintenance and enhancement, in which the system is used and changed as the needs of the users change.

Upper management is involved heavily in the first three phases. The end-users are involved heavily in the design, test, installation and maintenance phases, but upper management stays involved throughout in a supporting role.

When an application package is purchased, the same phases exist, but in a different form; these phases are as follows:

* Project Definition is the same - the project and its scope must be carefully defined;
* Feasibility Study becomes the search for an application package that meets the company's needs for an affordable price;
* Systems Analysis is the phase in which the implementation process is defined and planned;
* Design, programming and test are eliminated - they are the phases that take the most time, and this is the main reason that people purchase rather than write their own software;
* Installation includes not only training and conversion but also such things as rewriting the chart of accounts and developing procedures;
* Maintenance and enhancement involves the integration of new features and use of report-writers.

Here again, upper management must be involved throughout, and the end-users are involved in the later phases.

Much attention is paid to this aspect of project management. Support tools for this aspect, like Gantt and PERT charts, are widely available. All too often, project teams focus on the milestones and tasks but forget that the techniques associated with project management are as critical to the success of the project.

Project Management Techniques

The techniques that are used in the traditional data processing project are important. Much attention is paid to how a software development project is managed because the costs are so enormous; too often this is ignored when software is purchased. But the "turn-key" application is just a mythical as the "man month"!

Many techniques are used throughout the software development and installation cycle. Those that need to be applied to the implementation of purchased software are as follows:

* Each phase must be clearly defined, with milestones and tasks identified and responsibilities for each task assigned.
* The phases are not linear but cyclical; that is, new facts discovered in one phase often require that the team go back to the previous phase or phases and make changes. For example, during the feasibility study it is often determined that the project must be redefined; it cannot be accomplished with the time, people and money available, for example.
* The project scope must be carefully defined and priorities set by highest level management at the outset of the project.
* The project team must be identified early on, and the core team must remain intact throughout the implementation, including the maintenance and enhancement phase. This includes assigning a project manager and giving him or her the responsibility and authority to manage the project.
* Upper management must be committed throughout - and must communicate this commitment to all levels of the organization.
* A detailed plan must be in place before any work is done (in the case of purchased software this means before the software is installed). This plan is a working document; it is monitored and enhanced on a regular basis.
* Communication is critical, and this is best accomplished by regular project status meetings.
* Resources must be identified in the analysis phase, and must be kept accountable throughout the project.

Most companies use these techniques in the software selection process. A clearly defined goal is set, software packages are carefully evaluated and the best one selected from a vendor who can and will provide the proper levels of support, training is scheduled and taken, and then the system and the software is installed. Then why do things

seem to go so smoothly for some organizations and so painfully for others? Because that is just the beginning. The follow-through techniques that too many people ignore must be applied. The remainder of this paper focuses on the techniques that are the most valuable - and the techniques that are most commonly ignored.

The Project Team and Project Manager

Selecting the project manager and project team is the first step. The project team must be in place during the first phases of project definition and software selection and remain intact throughout. The project manager must begin managing the project from inception through installation.

The Project Team

The project team consists of a high-level manager, representatives from all departments involved, and the project manager. The high-level manager must be one to whom the department heads of all departments involved report, because his or her main role is goal setting and conflict resolution. This manager must be committed to the project and communicate that commitment to everyone in the company. In other words, this manager is the project sponsor.

Two members from each of the involved departments are part of the project team - a management team member who will be involved heavily in the software selection and a lead user who will be involved heavily in the actual installation. The management team member makes sure that the goals and needs of his or her department are met, and is responsible for allocating the needed resources from his or her group. The lead user is responsible for conversion to the new system, end-user training, the development of procedures, and the maintenance and enhancement phases.

The Project Manager

Care must be given to the selection of a project manager, and his or her responsibilities and authority must be clearly defined and communicated to everyone involved. The main responsibilities of the project manager are:

* development of the implementation plan;
* monitoring and controlling the implementation;
* team status meetings;
* setting priorities;
* organizing all resources.

In other words, the project manager doesn't really DO

anything. For example, the project manager does not attend training, unless the budget allows - the lead users attend the training.

Some do's and don'ts in selecting a project manager are:

* DO select a strong person with good leadership skills; the project manager leads the team.
* DO select someone with good verbal and written communication skills; the project manager leads meetings, documents and communicates progress, and communicates with vendors, which is more effectively done in writing.
* DON'T select a user from one department if multiple departments are involved, especially if the departments do not now cooperate and work together closely and smoothly (many payroll and personnel departments, for example).
* DON'T use a computer programmer; they are detail-oriented, and it will look like MIS "owns" the project.
* If you must go outside the company, DO look for an industry consultant in your industry.
* DON'T use your hardware vendor, unless the person assigned is an industry consultant; hardware vendors generally don't know anything about your business or your software.
* DON'T use your software vendor, unless the person assigned is an industry consultant; software vendors want to get the <u>software</u> implemented, and you want to get the <u>application</u> implemented.

<u>Planning and Controlling the Project</u>

The key to a successful project is a good plan, and the key to a good plan is to start early and constantly enhance the plan. Start developing the plan before the software is selected. Asking for implementation information during the software evaluation phase not only helps get a good and complete plan in place early on, but also can help in the evaluation of vendors. Don't think that planning is not necessary because you are replacing an existing automated system with a new one; this is more difficult than automating a manual system!

<u>Project Planning</u>

A project plan starts with a strong and complete statement of goals. Next, major milestones are identified, each with a start date and an end date. Each milestone is then broken

down into tasks, the tasks into subtasks, and the subtasks into actual work packages. The smaller units also have start and end dates; in addition, the interdependencies are defined and the work packages are assigned to individuals. Remember that the project plan is a working document; when a work package is not completed on time, the entire plan must be reworked. But keep the plan in perspective; no plan can ever be set in concrete, and if completing the plan on-time and as initially defined becomes too important, the result will be the famous "we did it on time and on budget - but it's totally useless". Schedules are made to be broken, and project plans constantly evolve.

Controlling the Project

Projects are controlled and directed through project meetings. The bad news is that project meetings must be held, and held on a regular basis with all members in attendance. The good news is that project meetings do not have to be painful. Project meetings should be held once a week and should last no more than one hour. The only purpose of the project meeting is to review the status of the project - period. No issues are resolved in the project meeting, but they are identified in the meeting. Projects are like sharks - if they don't keep moving, they die. Meetings keep projects moving.

The project manager is totally responsible for the meetings. He or she prepares an agenda, and distributes it at least one day in advance along with the updated project plan. A good meeting format is for each department manager to give a brief status report, along with any issues that are impeding the department's progress. The project manager takes notes, moves the meeting along, and develops an action plan. The meeting is over at this point. The project manager distributes the notes and action plan, and works with the appropriate team members outside the meeting to resolve the issues before the next meeting. This makes for short, effective meetings and also reinforces the roles and responsibilities of the team members - the project manager is responsible for the plan and the team members are responsible for the work.

Identifying and Using Resources

In most projects, all of the necessary resources are available, but are neither identified nor effectively used. The project manager is responsible for coordinating the resources, both internal and external. The responsibilities

of all resources, both internal and external, including the hardware and software vendor, must be clearly identified, monitored and controlled.

Internal Resources

Internal resources are usually clearly identified, and the main issue here is ensuring their availability. This is one of the main reasons that management is part of the team. The project manager must have the authority to ensure that the team members meet their commitments and must be able to resolve any conflicts that arise. The project manager's responsibility is to "run interference" for the team members. For example, if a user from the payroll team needs some information from a user in the personnel department and is having trouble getting it, the project manager is responsible for resolving this. The project manager has the authority to go to the user or the user's manager to get the communication problem fixed.

External Resources

External resources are usually not as clearly defined, especially at the outset of the project. The external resources that should be available are as follows: the hardware vendor, the software vendor, training courses, implementation services, and telephone assistance. The sooner all available resources are identified, the early they can be used. Here again, the project manager is responsible for ensuring that commitments made by and for external resources are met, but first the project manager must understand what resources are available.

The project manager is also responsible for running interference between the team and the vendors. Because the vendors do not work for the company directly, the most effective way of dealing with them is in writing. Identify problems and document meetings and commitments in writing immediately after the problem is identified, the meeting is held or the commitment is made. This can insure that all parties have a common understanding of the situation and the problems are attended to in a timely manner. And yes, your software and hardware vendors can actually meet with the project manager and the team at the same time!

External resources are often ignored or used at the wrong time by the wrong people. These external resources include training, implementation services, and telephone assistance.

Training

The right people must be trained at the right time. The lead users should attend formal classroom training at the beginning of the installation phase. It is often helpful for the project manager to attend training with the users, but it is not necessary. Formal classroom training is recommended because it gets the users out of the work environment and provides an opportunity to learn from the other students.

After the training is completed, the users return to the site and use what they have learned to fine-tune the project plan and to develop procedures for how the software will be used. In the formal training they learn how the software works; in developing the procedures they determine how the software will be used to accomplish the tasks of their departments. These procedures are documented and form the basis for the on-site training of the other users by the lead users.

Telephone Assistance

Telephone assistance is very valuable, but it is no substitute for training. It should not be used until formal training of the lead users has been completed. Each lead user should be the telephone contact point with the vendors. Each call should be handled by the lead user responsible for that area of the application, and should be carefully documented by that user.

Telephone assistance should be used not just to report problems but also to answer questions on software use. Don't wait until you are hopelessly confused and under time constraints to place the call, but also don't expect the telephone consultant to solve all of your problems. You will receive an overwhelming amount of documentation with your system; one of the best services that telephone consulting can provide is to point out where in the many manuals the information that you need is located.

Implementation Services

The implementation services provided by your hardware and software vendors should be clearly defined during the software evaluation process. At the very minimum, the vendors should provide assistance in the development of the initial project plan, including the setting of major milestones with realistic start and end dates. Some vendors provide project implementation handbooks and assign one of

their support people to your project team. These are good things to look for if the implementation is your first.

Take advantage of these services even if you have an MIS group or have implemented software packages before. Also look to the community of users of your software package; other users are normally very willing to share their experiences with you. Take advantage of every source of experience available - at the very least it will confirm that you are on the right track and have the project well under control.

Project Management in a Dynamic Environment

Most businesses are constantly changing, and often the implementation of a new software package accelerates this change. Therefore, it is extremely important that the project team be firm yet flexible. Changes come in two forms - changes in the business impact the software implementation, and the software implementation often changes the business itself.

When Your Business Changes

Often, changes in the business require that the project be modified to take new directions into account. The project team must have an understanding of the business and recognize when this is required. It often means reworking the project plan and adding to the project team. In very dynamic businesses, this should be recognized at the outset of the project, and the appropriate checkpoints for re-evaluation of the goals and scope of the project should be part of the project plan.

The project should not be put on hold unless absolutely necessary, because it will be extremely difficult to pick it up again. If this does happen, the project team must go back to the beginning once the project becomes active again. This means redefining the goals, reworking the project plan, and often changing the project team itself. Here again, a strong statement of goals and active involvement of the project sponsor are crucial. No one should be penalized because the business climate or the business itself changed, and this will be easier if the directive comes from the highest level of management.

When Your Software Changes Your Business

Often the business changes because of the software implementation. Many businesses use the implementation of a new application package to make needed changes to the way that business is done. Many companies have changed outmoded accounting practices or changed the focus of a business (from manufacturing-driven to customer-service-driven, for example) through the introduction of new software. This is often the easiest way to change the business, but it is very difficult on the project team. In this case, a very strong project manager is a must. An outside consultant is a good choice for project manager in this case - everyone can hate and blame the project manager, who will leave after the system is installed.

Changes are a given in today's business world, but the project team must take changes into account at every phase of the project plan.

What Works and What Doesn't

The vast majority of projects succeed - some are just much more successful than others and at a much lower cost. The most common mistakes made delay the implementation and make it very painful for the end-users; these mistakes are detailed below.

* No clear statement of goals.
  This results in a project for which the success cannot be measured; no one is really satisfied, because expectations were never clearly set. It also results in a project that goes off in many different directions, with each group setting and attempting to achieve its own goals. The single most important success factor is to have highest-level management set the goals and communicate these goals to the team and the entire company.

* No project manager or the wrong project manager.
  The project manager must have strong leadership skills, and must have the responsibility and the authority and the resources to control and direct the project. Often the project manager is either too high in the organization (not enough time) or too low in the organization (not enough authority). The lack of a project manager or the wrong project manager results in a project without direction or control.

* The wrong project team.
  The project team should consist of the project sponsor, the project manager, and at least two representatives from each department involved in the implementation. The manager of the department must be on the team, because only he or she can allocate resources and resolve conflicts. A lead user for each area of the application must also be on the team, and must be an individual who understands the day to day job functions and can train and lead the end-user implementation.

* No plan, a too high-level plan, or a plan without defined phases and start and end dates.
  This results in an implementation that drags on forever - half the people don't know what they are supposed to do, and the other half let what they are supposed to do slide because of the more pressing concerns of the day to day tasks. It also results in an unrealistic plan; the milestones can never be achieved in the time alloted.

* Training and the development of end-user procedures ignored.
  This happens most commonly in environments with an MIS department; the users lean heavily on MIS, and MIS is used to training and procedures coming naturally from the design phase. This leads to frustration on the users' part, with the software never really being utilized effectively - the users sort of figure out someway to get their work done, often by fighting and "kluging" their way through the application. This also happens when the wrong person - typically the project manager - is trained instead of the lead users.

If implemented correctly, an application software package can be what it is advertised to be - a quick and easy way to make people more productive. If implemented incorrectly or haphazardly, an application package can cost just as much and take just as long to implement as an in-house developed application.

Use the project management techniques: define your goals, assemble a good team, assign the right project manager, develop a good plan, follow the plan, use all of your resources, and make sure at each phase that the project is meeting the current company goals. But keep in mind that you are working with people, and that most projects make significant changes in how people do their jobs. The success of a project is not that it comes in on time and

under budget but that the time and money spent was worthwhile for the company.

References

Project Management for Small and Medium Size Businesses
Harold Kerzner and Hans J. Thamhain
Van Nostrand Reinhold Company, publisher

Project Management Handbook
David I. Cleland and William R. King, editors
Van Nostrand Reinhold Company, publisher

Winning at Project Management
Robert D. Gilbreath
John Wiley & Sons, publisher

The Payoff from Information Technology

Marilyn Martiny
Hewlett Packard
Cupertino, California

OVERVIEW - THE PAYOFF FROM INFORMATION TECHNOLOGY

Management is turning an increasingly skeptical eye on the benefits of information and computer technology. Executives who welcomed information system projects a few years ago are much slower to embrace today's automation proposals.

Executives are naturally reluctant to invest in information systems without concrete evidence of its payoff. The solution is to measure the benefits, and the contributions of information systems to increased profits, enhanced productivity, and most important, link the benefits from technology to the attainment of the critical success factors and strategic business goals of the organization.

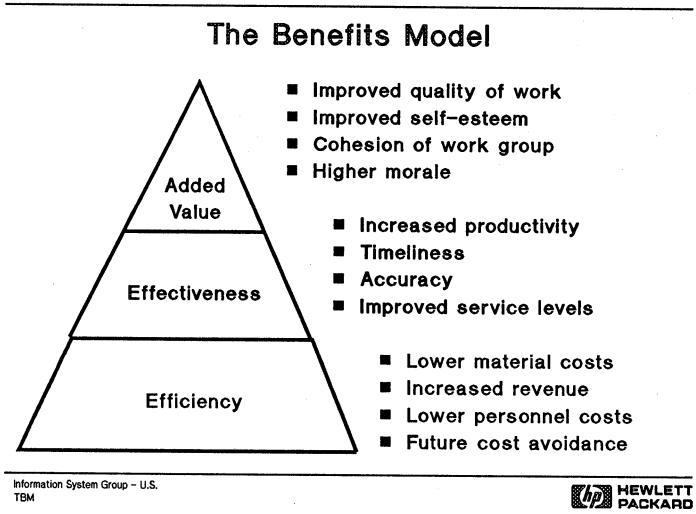
For many years the strategic benefits of information systems were considered to be "intangible". But at Hewlett-Packard we have developed in partnership with Arthur Young a methodology that measures the efficiency, effectiveness and added value benefits of technology in quantitative terms.

We call this methodology a Benefit Analysis and have used it successfully to measure benefits that clients have achieved through the strategic implementation of technology. The Benefit Analysis can also be used to help justify or evaluate "high payoff" applications...applications that will help overcome business obstacles to reach the goals of an organization.

Using the Benefit Analysis methodology we have developed case studies documenting the strategic payoff of information systems. These case studies offer proof to the executive decision maker that information systems have delivered value and made major contributions to the goals of the organization.

THE BENEFIT MODEL

The Benefits Model is the structure that identifies and communicates the projected or already achieved benefits of implementing technology. It provides a framework for presenting business benefits derived through the implementation of information systems on 3 distinct levels: Efficiency, Effectiveness, and Added Value.



Efficiency Benefits:

The Efficiency benefits are derived from investments in technology that impact an organization's costs or revenues. At this level, the benefits have a direct relationship to the investment in technology and it is reasonably certain the benefit would not be realized without the investment.

Efficiency benefits include the following:

- Lower material costs
- Increased revenue/profit improvement
- Lower personnel costs
- Future cost avoidance

Effectiveness Benefits:

Organizations have realized a major impact on their operations in areas identified in the Effectiveness level. These benefits also have the potential of impacting an organization's cost and revenue streams, however, the benefits may not have occurred solely because of the investment in technology.

An example might be that the client projects a productivity improvement over the next year due to the technology plus a reorganization that is to take place. What the client needs to project is what percentage they feel is due to the investment in technology (1%-100%). Whatever the percentage they choose will be reflected in the Effectveness level of the Benefits Model.

Effectiveness benefits include the following:

-Increased productivity
-Improved timeliness
-Accuracy
-Improved service levels

Added Value Benefits:

The Added Value benefits link directly back to the company's goals and critical success factors.

Added Value benefits may be:

-Improved competitiveness
-Improved quality of work
-Improved self-esteem
-Cohesion of work group
-Higher morale

It is important to note that dollar values are derived from the client during the Benefit Analysis process. The quantifiable benefits can be ascribed to every level of the Benefits Model, however, it is more typical to have dollars on the Efficiency and Effectiveness levels.

GAINING PAYBACK

The goal of the Benefit Analysis process is to gain a fair determination of an investment's worth. The keys to a successful Benefit Analysis are:

-High level, executive involvement across functional areas

-Identify high payoff opportunities that are strategic to meeting the critical success factors and goals of the organization

-Analyze the costs and benefits of an investment in technology

-Measure the actual benefits that occurred with a follow-up analysis

Of course a Benefit Analysis for information systems has little meaning without an appropriate organizational response. If information technology is to successfully impact white-collar productivity or improve customer service levels, management must assure that the organization responds to take advantage of the new technology tools. The following is a case study using the Benefit Analysis methodology conducted at Westinghouse Electronics Assembly Plant (EAP).

BENEFIT ANALYSIS CASE STUDY AT WESTINGHOUSE EAP:

Westinghouse, the $11 billion Pittsburgh-based electronics conglomerate, has built a leading computer integrated manufacturing plant, Westinghouse Electronic Assembly Plant (EAP), in College Station, Texas. EAP's corporate goals are to manufacture electronic printed circuit assemblies of the highest quality, at the lowest possible cost, while maintaining consistent on-time deliveries. The U.S. department iof Defense (DOD) is Westinghouse Electric Assembly Plant's (EAP) major customer, and it's a tough one.

In order to achieve these goals, EAP needed an Integrated Business Solution that would help it achieve the following critical success factors:

- improve communications among departments to facilitate timely decision making.

- deliver the right information, in a timely manner, for effective problem solving

- share information throughout the organization to create an atmosphere of teamwork

- create an atmosphere that facilitates people-to-people communication

- enhance work team productivity to ensure timeliness of product delivery

OBSTACLES TO ACHIEVING EAP's GOALS AND CRITICAL SUCCESS FACTORS

EAP runs three shifts a day, seven days a week and the ability to share information among shifts has a direct impact on the cost and on-time delivery of EAP's products to their customer, the Department of Defense. Prior to implementing an electronic messaging capability on the manufacturing floor, information regarding equipment problems, needed maintenance or manufacturing yields was handled with pencil and paper. Problems occurred with accurately reading the information and most importantly, there was no historical audit trail. The result was that the productivity of the work teams suffered.

Sharing information between departments and throughout the organization was also paper-based. Pulling together the right type of information for effective problem solving took a lot of time and impacted the productivity of both the management and administrative staff. Due to the difficulty in easily communicating enterprise-wide, was resulting in barriers to communication between departments. At EAP creating an atmosphere to facilitate people-to- people communications is of utmost importance, so overcoming this obstacle was imperative.

At EAP sharing critical manufacturing data to all levels at the plant posed a challenge. It was determined that "a picture is worth a thousand words" and graphics could play an important role in displaying manufacturing yields, attendance and safety information.

"Our people weren't getting the proper information they needed to perform effectively," states EAP's plant manager. As a result, individual productivity, as well as quality, cost, and on-time deliveries suffered.

THE TECHNOLOGY SOLUTION

EAP chose an integrated information solution from Hewlett-Packard. The HP solution has improved company-wide communication at EAP through electronic messaging capabilities, impacting productivity and playing a strategic role in meeting the goals of the organization. In addition, the solution's time management feature simplifies scheduling, helping to eliminate telephone tag and facilitating people-to-people communication.

EAP also benefited from the graphics capabilities that the HP information system provided. "When business data is graphically displayed," the plant manager states, "it's more easily understood and disseminated." The quality of work has improved significantly, enhancing the plant's image to customers.

EAP's THREE-TIERED BENEFITS

The benefits EAP has achieved with HP's integrated information solution are significant and can be grouped into three categories: Efficiency, Effectiveness, and Added Value.

EFFICIENCY BENEFITS

A shared document production system, electronic messaging, and graphics tools have helped lower costs. EAP has achieved the following Efficiency benefits:

- doubled the workforce without increasing administrative support staff, resulting in a savings of $160,000 per year;

- successfully leveraged previous system investments, avoiding the purchase of additional non-integrated systems at a one-time savings of $75,000;

- eliminated the need for temporary administrative help, leading to a 33 percent cut in personnel costs, saving $80,000 annually

THE EFFECTIVENESS BENEFITS

The HP solution has also improved employee effectiveness throughout EAP by:

- increasing professional staff productivity five percent - saving approximately $58,000 per year - due to the E-mail system;

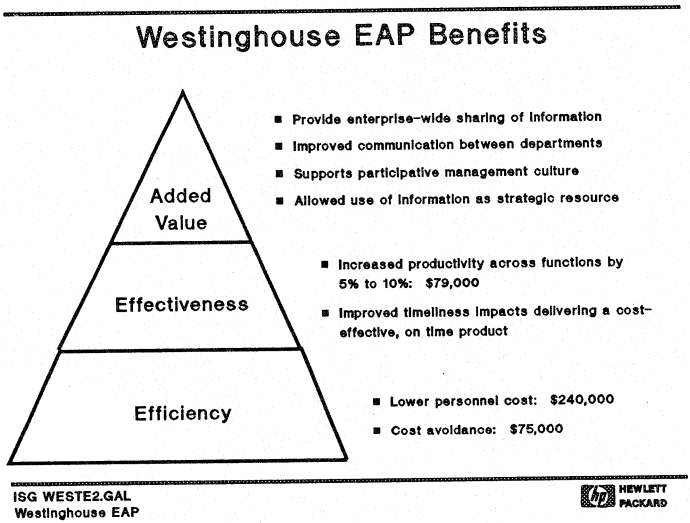
- raising administrative productivity by 15 percent and saving $21,000 annually by reducing the number of internal memos and the time it takes for production and distribution;

- sharing and transferring time-critical information among shifts, increasing the ability to produce and deliver quality products on time

THE ADDED VALUE BENEFITS

Since implementing Hewlett-Packard's integrated information solution, EAP has realized a number of company-wide benefits that directly relate to its critical success factors. These benefits enable the plant to achieve its goals of cutting costs, assuring on-time delivery, and maintaining a commitment to quality. EAP has maintained a competitive advantage through:

- enterprise-wide sharing of information
- improved communications between departments
- greater employee participation in matters affecting company policies
- increased use of information as a strategic resource



Westinghouse EAP was able to achieve a payback on the Hewlett-Packard information system in less than 18 months, much faster than their anticipated two and one half to three years.

The HP solution helped Westinghouse Electronic Assembly Plant overcome its business obstacles to achieve its goals. But, the keys to Westinghouse's successful implementation were:

- Management Commitment
- Critical mass of users across functions
- End user training and support
- Communication valued as key to staying competitive
- Technology solutions directly impacted the critical success factors and goals of the organization