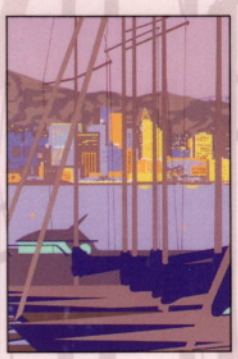
INTEREX



HP Users Conference

August
5-8, 1991
San Diego

PROCEEDINGS

RTE, HP-UX, Workstations

Sponsored by

INTEREX

The International

Association of

Hewlett-Packard

Computer Users

Catch the Wave of HP Computing!

INTEREX

The International Association of Hewlett-Packard Computer Users

PROCEEDINGS

of the

1991 INTEREX HP Users Conference

RTE, HP-UX, Workstations

at

San Diego, California

August 5-8, 1991

Introduction

This volume of the Proceedings of the INTEREX 1991 North American Conference was printed from camera-ready copy prepared by the authors. It contains papers dealing with RTE and with the HP1000 in general and all papers dealing with HP-UX and the HP9000 computers. It also contains papers dealing with the migration of applications from one type of computer or operating system to another.

Because HP-UX has been growing in popularity amongst business users, there are many papers reflecting such use. All papers related to HP-UX have been included here for convenient reference by the reader. This system may not be followed in the future. Papers relating to workstations are included here, since these are small Unix-based systems. At the time of this conference HP's Apollo division produced the Apollo line of workstations and papers relating to those systems are also included in this volume.

Papers were numbered as they were received and in order to present them at the conference in logical groupings the numbers are not necessarily consecutive. The numbers should be considered to be simply reference numbers.

Because the tutorials represent up-to-the-minute information and require much more work than a paper to prepare, it is not always possible for them to be included in the Proceedings.

Thanks go to the authors who met the submission requirements and had their papers in by the deadlines. Thanks also to the members of the paper review committee who read the abstracts and offered criticism and advice to both the authors and the editor.

F. Stephen Gauss
U. S. Naval Observatory
Washington, D.C.

R. Arthur Gentry
Gentry and Associates
Excelsior Springs, Missouri

10 June 1991

Index by Paper Number

Index by Paper Number

Index by Paper Number

Index by Author

Index by Author

Index by Author

RTE

HP-UX

MIGRATION

TUTORIALS

Paper Number : 1003

T : A PACKAGE FOR PROGRAMMING ACROSS SYSTEMS

Jacques Sansdrap, Jean-Louis Matton

University of Louvain, Avenue Hippocrate 55/5560, 1200 Brussels, Belgium
Phone: 32/2/7645561 Fax: 32/27645569

ABSTRACT

"T" is an original package allowing the easy writing and running of "super programs" composed of several programs distributed on a networked system. Thanks to the "T" package, each of these programs, which may provide high level function (such as editing the content of a data base, conducting a dialogue with a user, producing a report, managing a graphic system, etc...) can be seen as equivalent to subroutines of a classical program. These programs, working on different computers, are linked through parameters given to the "T" package and thus could be relocated from a system to another as required for efficiency with local availability of data and resources.
Another key feature of T is its implementation on the RTE-A and on HP-UX. Other systems (MS-DOS, ...) could also be involved. The modularity of these programs promotes the team working. The T package will be available as contributed software.

INTRODUCTION

The hardware that money can buy is now a hundred times more powerful than 20 years ago. By contrast there as been at most a tenfold bettering of the productivity of the software building people and the cost of manpower has increased.
Of course a lot of computer cycles and "core" memory words can now be wasted without any appreciable degradation for the user.
Of course there are now a lot of "off the shelf" software for many common use like text editing or spreadsheet computation.
But often this is not the best fit possible and often it means that the users have to learn the computer ways rather than do their primary job. It also means that often a computer

"just happens" to be there without any plan for the future and much work is needed later to integrate this system in a more general and more efficient system. Thus there is a need to augment the productivity of the programmers to cope with new fields of application and to have a faster response to the user's requests and to the evolving technology. It is also important to avoid being drowned in the maintenance of mature applications.

These goals could be achieved by merging the works of several programmers in a tool-kit of re-usable software modules.
Most components of a new application would be found already made in the tool-kit. If a tool is updated the effect of this update is immediately applicable to all applications using it.

This type of method already exists: it is implemented in the fourth-generation languages (4GL) and in the applications of the software engineering (SE).
The 4GL however lacks some flexibility: basically it always produce the same program with varying options. The SE products are probably efficient in the context of a software house where the job environment of the programmers can be controlled.

The environment in a medical research institution coupled to an university hospital is quite different. There, you will find a number of small highly independent teams. Nearly all teams have at least one PC but there are also powerful computer systems. Often there is only a small number of persons that are programming as a professional activity or as a side activity. The need to integrate all these installations is easy enough to understand, but the means and ways are hard to come by.
This is why we have devised the T package to be able to mix the work of different programmers on different machines.

EXAMPLE

Here follows an example to help to understand what T really is. To fully develop this example we have to use some terms that are going to be defined later: so you may wish to come back here after reading the end of the paper.

An application running on a RTE-A let a user select from a file directory some file containing digitized signals. The terminal could show alternatively a list of file name or a graphic representation of the signals in a selected file. The user could ask for a printout of the graphics if a peculiarity of the signals is observed.
Once the user has make his/her choice, some computation is done on the file and the results are displayed on the terminal.
The user accepts or rejects these results. If they are accepted, they are stored in a data-base for later statistical analysis.

The following programs are used on the RTE-A for this application:

- the T server,
- the main client (APPLIC) that implements this application by scheduling all other clients and performing the computation,
- the terminal manager subcontractor (TERM),
- a client (DIR) that scans a directory for files matching some mask and reports the file names through a window of the terminal by TERM,
- a client (VSIG) that draws the signals contained in a file with a graphic library which end point is usually TERM; the set-up of the graphic could be modified through parameters entered in windows managed by the alphanumeric side of TERM,
- a subcontractor client (PLOT) that could temporarily replace the graphic side of TERM so that the output of VSIG is on a plotter,
- a data-base manager client (DB) that is an interface to an IMAGE-II data-base,
- a storage subcontractor client (STORE) that adapts the results produced by APPLIC to the structure requested by DB.

The user does not perceive that this number of programs are at work for him/her.
Only APPLIC is specific to this application (it is a rather small program); all other programs have some use in other applications.

Now there is a second RTE-A system used for data acquisition. Let us name the first system HARV and the new one LUC. There is a NS-1000 connection between the two systems.
A user of HARV could need the same application for signals stored on LUC. A simple change of configuration is needed and no new software:

- on HARV (where the user is):
    - T server,
    - TERM,
    - PLOT,
    - DB,
    - STORE;
- on LUC (where the data is):
    - T server (used in slave mode),
    - APPLIC,
    - DIR,
    - VSIG.

Later an HP-UX (9000/835) system named CARD is connected to the LAN. The SQL data-base and a statistical package on this system are preferred to the IMAGE-II data-base and to home-made statistical programs.
There is also a new LaserJet-III on this system that is faster for graphic production than the old plotter on HARV.
The configuration is then:

- on HARV (where the user still is):
    - T server,
    - TERM,
    - STORE;
- on LUC (where the data is):
    - APPLIC,
    - DIR,
    - VSIG;
- on CARD (all programs are new):
    - T server (used in slave mode),
    - DB (to SQL),
    - PLOT (to LaserJet).

Now the user is tired of the graphic terminal and wants a PC as everybody else. There is a LAN card on this PC and it runs with MS-DOS and WINDOWS 3.0. It is connected to the LAN. More users are now fluent in UNIX than in RTE: CARD

becomes the usual logon machine.
The configuration is then:

- HARV is only used when the data is stored there;
- on LUC (where the data is in this case):
  - T server (used in slave mode),
  - APPLIC,
  - DIR,
  - VSIG,
  - STORE;
- on CARD (no new program):
  - T server,
  - DB,
  - PLOT,
- on the PC (new program):
  - TERM (alphanumeric and graphic interface to WINDOWS).

How would it be realized without T? That is difficult to say. The current RTE-A user is more knowledgeable that the user of another system so the "point and click" interface implemented by DIR and TERM would not initially be seen as required. This would have discouraged some potential users.
When the data from LUC would need to be accessed one would have to use one of the tools offered by the network system: TELNET for remote terminal connection, TRFAS for remote access to data files or RDBA for remote data-base access. All these tools suffer from a discouraging slow performance and from other limitations as well (no block mode for TELNET in the current software release). One would probably have to re-write the application to circumvent these problems. This application would probably be implemented as a monolithic program designed by one programmer with few parts easily re-usable for other applications. The user could probably not be isolated from the idiosyncrasies of each system and of the tools that move the data from place to place. The arrival of the HP-UX system would probably mean a complete porting of the application to the HP-9000, the RTE-A would be discarded and the users would have to be re-trained.
The application would once more have to be re-designed to use a GUI (PC or workstation) rather than a graphic terminal.

A programmer would have to spend most of his/her work time to follow the evolution for this application in this short time. Any change that the user should ask would be hard to accommodate. A new application, even looking very similar to the previous one, would benefit only from the experience accumulated by the programmer and not from modules shared with the programmer's colleagues.
The user should have to learn as many ways to interact with the system as there are applications.

GUIDELINES

Data and the modules that process it should preferably be encapsulated together so that a change in the structure of the data only needs a localized software modification.
An application written with this tool-kit should be able to get the data or other resources on several systems if needs be.
The access, low-level processing and management of resources like data files or peripherals
should be done by local modules on its system of origin.
An application should need a minimum of modification if one of the computer is replaced by another type of machine or if the data is moved.

The communication between processors should not be fixed on one defined standard: at the present time there are so many standards and variations of them that it would not have been possible to find a common ground for all computers of various make, age and size we have.
The protocol of communication should be simple to implement over any networking standard available between each pair of machines.

The amount of data that has to transit through the networks should be minimized so that the time penalty should not be too visible to the user.
The modules should be independent programs rather than relocatable libraries. It is easier to assure that any required housekeeping is performed. The program could be more easily debugged than a relocatable module that has to be embedded in a program. When some maintenance has been done on a relocatable module all the programs using it have to be re-compiled or at least re-linked; there are tools to automate this task but

it could not be convenient to disturb too many applications at the same time.
A module under the form of an independent program could be easier to replace. It also give more freedom to its designer.

T: A SOCIETY OF PROGRAMS

A working T society is composed of several client programs distributed over various systems and of a "go-between" T server program. A client politely waits for the answer when it has sent a query to another client. Thus the T server listens only to one client at a time and there is a stack of dialogues.
The messages are strings of up to 200 ASCII characters. Each client designer has to define and publish the form and content of messages that his/her program accept and send back. There is no problem of variable data representation with this type of transmission. There are other mechanisms to send greater amounts of structured data to some special clients (explained later): the subcontractors.

A client program can send to T requests other than the transmission of messages. Each client of T is usually scheduled on request from another client. A client to schedule is referred to by a symbolic name. T finds in its environment (following some search path) a client description file for this name. This file supplies the method of communication to use with the new client and its localization.

The nature of a request that a client sends to T is identified by the first 16 bits word of the request (each request has a minimum length of two 16 bits words).
There are some number of categories of request: a category is identified by the quotient of the code word divided by 100. The requests in the category 0 concern the basic working of T and of its link with the client: closing, scheduling another client, sending a query or an answer to another client...

The other categories are of a more general nature and very few of them are actually implemented by T (writing or reading on the terminal,...). A client can elect to be a subcontractor for one or more of these categories of request: it stops then to be addressed like an ordinary client but receives from T all requests of these categories coming from any client. A subcontract for a category can be overlaid by a later

subcontract from another client: the first subcontract will be re-instated at the closing of the last subcontractor.

T LINKS

The links between T and its clients could all be of a different type: the routing and translation are done by T. When a client and T are on the same machine, a local method of program-to-program communication is used (class I/O on RTE; bidirectional pipe on HP-UX). Any applicable networking method could be used when a client is on another machine than its T server (TCP/IP, NS-1000, BSD sockets). It would even be possible to link two computers by their serial ports if no other means were available.

Each link is a stream of 16 bits words: this is easily supported on the byte stream that is transmitted by network protocols and is more efficient on machines (like HP-1000) that have a penalty for byte addressing.
There are subroutines to put or get any type of data on this stream. The communication is of the half-duplex type with buffered or immediate transmission at the choice of the sender: a message of any length could be build element by element in the communication buffer before it is sent. There are subroutines to transmit to T any of the requests it accepts so that the programmer has not to care for the protocol. Some of theses subroutines are in a library that could be adapted as fit for the language and the processor but does not depend on the type of the link. In contrast, the other subroutines are specific to some link. This last group does not represent an extensive programming effort: the subroutines needed to implement the link by bidirectional pipes for HP-UX is contained in a C source module of 400 lines; for the class I/O link of RTE-A it is a 500 lines FORTRAN module.

CLIENT-TO-CLIENT DIALOGUE

At the very beginning of the life of each client there should be an "Open T" call to initialize the communication link. This returns also a run-string received from T: it is the first message. A client should never try to retrieve its arguments in another way.
Just before ending its life the client should call a "Close T" subroutine that sends a parting message to whoever launched it and close the communication link. These two

calls in the program and the presence of a client descriptor file pointing to this program are all that is needed to qualify a program as a T client. A set of subroutines will have to be selected at the link-editing time as appropriate for the method of communication with T.

If a client program is aborted, T detects that its link is down and abort itself ; each client detecting that its server has disappeared has the opportunity to do an orderly shut-down.

A client that is open for T is referred to by a client number. One could ask to T the number of a client with a known name, if it is open and free for a dialogue (not in the client stack or a subcontractor).

T-TO-SUBCONTRACTOR DIALOGUE

A subcontractor client starts its life as any other client and could take part in a dialogue while setting-up its environment.
When it is ready to enter its role, it sends to T an array containing the number of the categories of request it is ready to take charge of and an answer message to send back to its scheduler. Its life as client able to take part in a dialogue is then ended.

What the subcontractor receives then from T are no more messages but the raw request codes sent by some client. T is just a relay and does not knows anything about the structure of the request. The subcontractor has to compute the length of the data associated with the request and asks to T for this amount of data. This could be as long as needed and could be buffered by the transmission mechanism. The structure of the data is specific to the request and is known to the requester and to the subcontractor. There could be in the request an indication of the data representation in use where the client is: the subcontractor could have to do some translation.
The subcontractor could send back as much data as needed to the client but first the amount of data has to be indicated to T.

When a request has been fully processed, the subcontractor send a "cut" command to T to let it take back the control of the communication with the client.

There is some specific request type that has to be reserved to mean an order for the subcontractor to leave the scene.
The subcontractor then performs its last duties and sends T a request to close the link before termination.
This re-instate any subcontractor previously defined for the affected categories.

An ordinary client could ask T the name of the client (if any) that is registered as subcontractor for a request type.

LIBRARY IMPLEMENTATION THROUGH T

A subcontractor usually manages some complex functions like driving a display or accessing a data-base. The designer of the subcontractor writes a library of stub routines that package the parameters they are called with into requests to T and indirectly to the subcontractor. Several subcontractors could be prepared for the same stub routines. A client may select some working mode by scheduling one or the other subcontractor.
This is typically how a graphic workstation system is configured.

T is still in development and at the moment this paper is written (April 1991) only 3 subcontracting client programs with the matching stub libraries have been written: a printout manager, a terminal manager with form library and graphics library and a storage manager.

The printout manager is implemented for the RTE-A in FORTRAN and for the HP-UX in C. It is elementary because it has been mainly used for tests but is routinely used for remote printing.

The terminal manager is written in FORTRAN for the RTE-A only and put the terminal in block mode. The question is still open if it will be implemented for HP-UX by using curses(3x) or blmode(3c).
The form library is nearly identical to the X-FORM package (IUG meeting, Brussels 1989).
The graphic library is modeled on the DGL of Graphics/1000-II. This is sufficient for the type of applications we are developing and should be simple to emulate with Starbase

graphics on HP-UX and with any graphic library on a PC under MS-DOS.

The storage manager implements a structured billboard where a client could store some data that could be retrieved in full or by extract by other clients.
The main use will be for data-base access:

- a client is responsible for any access to a data-base. It does any referential constraint check that is needed and perform any processing attached to some data. It gives a declaration of the structure of its data to the storage manager and updates the content of this structure.
- all other clients that need this data gets it from the storage manager. Some clients could ask to the data-base client to update this data before getting it. Others clients could be designed to process simply what is currently in storage.
- a client could put a mask or model of data in the structure before sending a selection request message to the data-base client.

The stub library sends to the storage manager an indication of the data coding in usage where the client is. The storage manager translates data as required.
The storage manager is currently implemented on the RTE-A, converts the floating point data between HP-1000 representation and IEEE representation and takes care of the word alignment limitations of HP-PA. Its stub library exists for RTE-A and HP-UX.

ENSLAVED T

If several clients of T are on the same remote system it could be wasteful to open several communication links through a network. T could then be configured to schedule an enslaved copy of itself on the remote system. The remote clients would then in fact be local clients of the enslaved T.
The enslaved T and its clients would be running in a session and with an environment appropriate to the user.

Here is how it is done:

- A T program could schedule another remote T as another client with the appropriate link method. Currently this method is through an Ethernet LAN and the Internet daemon INETD that is available with HP-UX and NS-1000 on the RTE-A.

- The remote T starts a session on the logon account and with the password specified by the master T. An appropriate environment is built from the indications in a special session initialization file.
- The master T gives then a slaving request to the remote T.
- Thereafter the slave T receives only requests to open and close local links to clients, to relay the transmission of data to/from these clients and to stop to be a slave (immediately followed by an order to close the link with the master which terminates the session).

When T has a slave T, any scheduling of a client using the enslaved T type of link to a host with the same name is transformed into a request to the slave for scheduling a local client.

The enslaved T is released when the last of its clients is closed. T could have several simultaneously active slaves.

IMPLEMENTATION

The first implementation of T has been done on RTE-A in FORTRAN and has been used for a few applications on a single system or on two systems linked by a LAN and NS-1000.

The basic client library and the link subroutines for NS-9000 (NetIPC) have then been written in FORTRAN for HP-UX. There has been two uses of this software:
- an HP-UX client to test the access to an RTE-A data-base through the storage manager;
- an HP-UX client for printing on a LaserJet connected to an HP-1000.

A two man-week work has been needed for these tasks.

Then the link subroutines for bidirectional pipes and the T server have been written in C for HP-UX. The T server uses the BSD socket for LAN access rather than NetIPC so it should be directly portable to other UNIX machines. About one man-month has been needed.

The client link subroutines are currently developed for a PC under MS-DOS. This software is written in C with the NetIPC subroutines of HP OfficeShare. As MS-DOS does not do multi-tasking there will never be a T server for this type of machine (excepted with add-on software like MS-WINDOWS).

As a client can use a remote T server to talk with remote clients, there is no need to wait until the whole of the T package is implemented on a machine to start developing software using T on this machine. In some cases a T client could also be used as an interface to commercial software.

So we are confident that T could be quickly ported to other machines as needed if the tools for software development and some connection hardware are available.

CONCLUSION

At the moment this paper is written, we do not yet know if T will be successful in its function of teaming the work of the programmers. But we have received some encouraging support.

There is a general trend in the computer world toward standards and heavy systems. At some time in the future, all computers should use the same operating system (UNIX-like), any program could communicate with any other thanks to systems like HP's NewWave or HP Sockets, each programming language would have compatible extensions for object oriented programming (OOP), all data-base management systems (DBMS) would be SQL compatible with OOP extensions, each user would be able to use a graphical user interface (GUI) and distributed computing would be the norm. While waiting for all theses promises to be realized, T is a light, viable, alternative.

Paper # 1007

Adding an X-window User Interface to an HP1000 Application

Bob Combs
Combs International, Inc.
886 Belmont Avenue, Suite 3
North Haledon, NJ 07508
(201) 427-9292

Abstract

Users are now expecting applications to be run from X servers. A windowed terminal does provide some nice benefits, but what about true real-time applications? They can't *just* be ported to UNIX. There is a reason why RTE is still around! This paper reviews the design of a real-time application and how it has been restructured to take advantage of networks and an X interface, while still retaining its real-time (RTE) components.

1. Introduction

We have a package which performs data acquisition and control named **MAXS+**. This product has been around for about seven years, running exclusively on the HP1000 series of computers. **MAXS+** is a real-time software package which is used in factory automation, facility management, laboratory automation, and pilot plant control.

The typical system scans analog and digital points using various data acquisition boxes, and processes this data each scan. The scan processing must take priority over background functions such as reports and user requests. This real-time demand is what the HP1000 was designed to handle.

The draw back of using the HP1000 is that it lacks windows and has a limited graphical capability by today's standards. Enter UNIX and X-windows. Fortunately the HP1000 does have an Ethernet interface and can communicate with a UNIX system.

UNIX does not currently have true real-time response. After several years of propaganda to the contrary, HP is finally willing to sheepishly admit that HP-UX isn't really real-time. UNIX's priority scheduling algorithm, lack of a preemptable kernel, and non-deterministic context switching leave it quite lacking in the real-time department.

Keeping these points in mind, how do provide a system with a Graphical User Interface (GUI) (i.e. X-windows) and still be able to process data in real-time? The answer we came up with was a hybrid approach. That is, keep the real-time scanning and processing on the HP1000, and connect a UNIX box via Ethernet to provide the X-window user interface.

2. Original HP1000 System Layout

The standard version of MAXS+ uses an HP1000 for all tasks: real-time scan processing and user interface. The real-time scan processing consists of modules which perform input/output with the data acquisition box (often called the front end), a master scan processor, and a trend buffer management module. Refer to figure 1. for the original module layout.

The user interface consists of many modules, each of which is a screen application command. The screen modules are called from a command shell module which regulates the CRT screen.

The heart of all information in the MAXS+ system is a collection of tables which are memory resident in Shared Extended Memory Array (SHEMA). The SHEMA tables contain all of the current real-time values of the various variables in the system, and all configuration information. Memory resident tables allow the system to be much faster than if this information were kept on disc or other secondary storage.

One of the great benefits MAXS+ has is the ability to change virtually any portion of its configuration while on-line, having no effect on the other portions of the MAXS+ system. This flexibility allows real-time processes to continue while portions are being configured "on the fly".

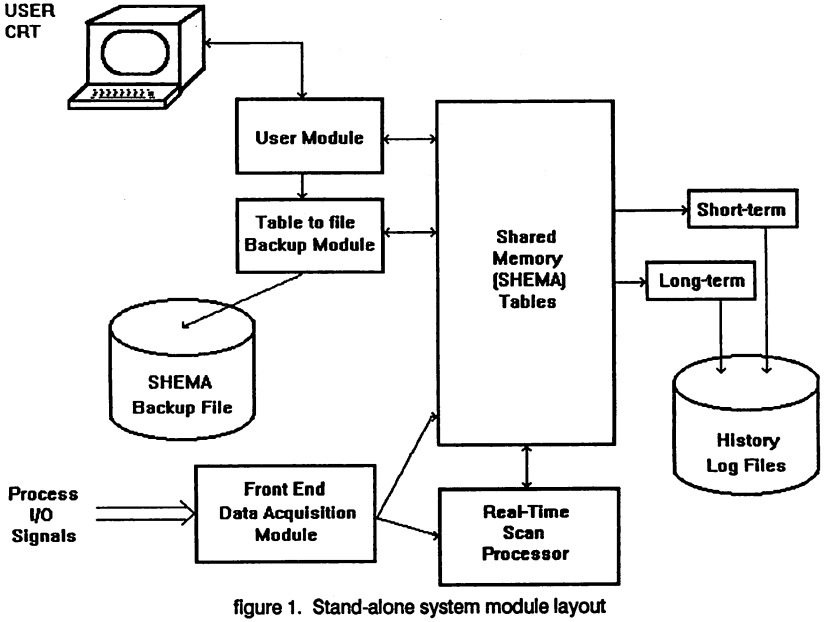


figure 1. Stand-alone system module layout

2.1. Real-Time Modules

There may be multiple front end modules on the system which gather input readings, placing them in SHEMA tables, and take outputs from the tables and send them to the front end. Many front end devices work directly in engineering units these days. Either way, the front ends are responsible for all analog/digital input/output with the real world. The communication with the various front ends may be RS-232, HP-IB, or even LAN.

The master scan processor processes the signals after all front ends have performed their I/O. There are nine different variable types, but each signal has certain stages it must go through: conversion, type dependent processing, discrimination, clamping, and alarm checking. After the processing is complete, trend updating is initiated, and finally data logging.

2.2. History Logging

The data logging is kept in three file types; short-term, hourly, and daily results. The short-term is perhaps a misnomer since it really means all data and events; which may or may not be wanted for long term archiving. The prime reason for keeping the data in these three file types is to allow users to quickly access different picture sizes of data. For example, a user may look at short-term data to see the last 8 hours worth of data, but may access the daily file to get a picture of the last year's worth of values. A year's worth of short-term data would take a much longer time to scan, even with today's faster discs, than daily values. Therefore these three history file types allow a user to select the detail level desired and spent less time waiting for results.

The short-term data is written to disc by SHLOG. Both the hourly and daily results are written by LOLOG.

2.3. User Interface

The user interface utilizes the HP1000 CRT screen as if it were three separate windows. They aren't true windows, but are treated as separate areas by the various user modules. As you can imagine, this was quite a trick since the HP1000 is really only a half-duplex connection.

The top line of the screen is reserved for the exclusive use of an alarm banner module which displays various status states of different components of the system, such as alarms or stalled front ends. The alarm banner line (top line) is written to the terminal's graphic plane to alleviate some screen interactions.

The bottom two lines of the screen are reserved for the command window. All command prompts and error messages are displayed in this window.

The middle screen area (lines 2-22) is utilized for forms with unprotected fields, periodically refreshing data displays, or graphic displays. All of the screen form displays were created with our own screen product, **QFORM**. We tailored certain aspects of **QFORM** to allow easier integration into this window-like environment.

The segregation of these three areas has resulted in a very usable interface, but was somewhat difficult to program.

3. Identification of Areas of Change

There were several pieces of information that lead to the decision to distribute MAXS+. One of the critical ones was that HP had begun to remove the preemption points in HP-UX and would be eliminating all of them by HP-UX 8.0. Add this to the fact that HP was no longer touting 9000/800's as a real-time replacement for RTE. It was obvious to everyone that UNIX was not an RTE system. If MAXS+ were to have a X user interface, the solution would have to be a hybrid one, with both an RTE and a UNIX system.

Once the decision had been made to break apart the functions of MAXS+, we had to determine how to do it easily yet retain the features we wanted. The HP1000 should remain virtually hidden from the user, and require minimal support to keep it running. We had decided the end goal system should have the following features:

- Real-time scanning should remain on the RTE system.
- All user interactions should be moved to the UNIX X terminal.
- History log files should be written to the UNIX system.
- Real-time configuration information should remain on RTE.
- Report setups should be moved to UNIX.
- An X terminal should be able to redirect to different RTE systems.
- An RTE system should be capable of handling multiple X users from different UNIX systems.

Therefore, all user interface modules were moved to UNIX, and given an X look and feel. A linkage between the RTE SHEMA tables and programs, to the X programs was developed. The table backup and real-time scan processing would remain untouched. History logging (SHLOG and LOLOG) must transmit their records to a UNIX system designated for history storage.

Note that while it was useful to allow multiple MAXS+/RTE systems to talk to multiple MAXS+/UNIX X users, each RTE would have to send its history to a specific UNIX system. Also, the history system should eventually allow multiple MAXS+/RTE systems to send their history data to the same MAXS+/UNIX system.

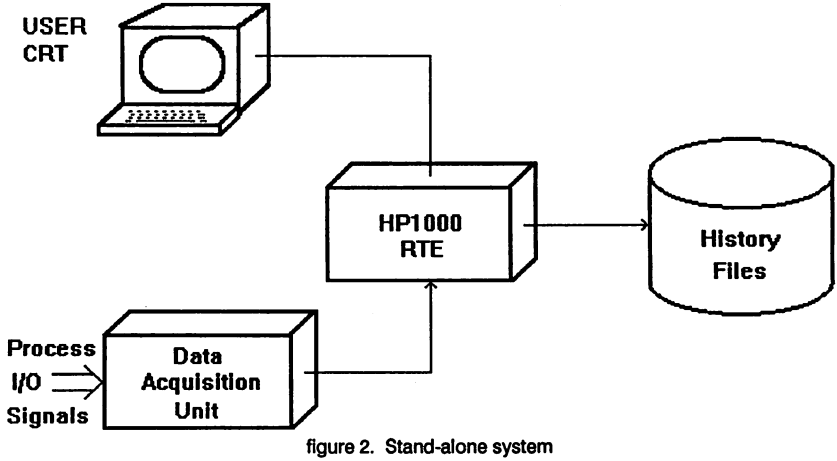


figure 2. Stand-alone system

4. Distributed System Layout

Fortunately, MAXS+ had been written using a high degree of subroutines, with lower level functions focused into few routines. While there were a few exceptions to this, for the most part it allowed functions such as table data access, variable searching, and history writing to be broken and "piped" over to the HP-UX system. Refer to figure 3. for the new system layout.

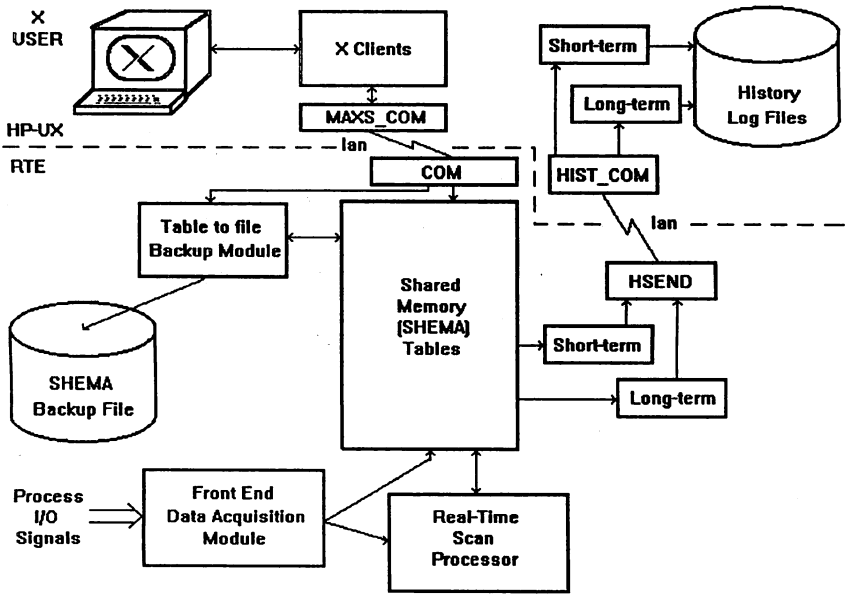


figure 3. Distributed system module layout

4.1. Overview of Distributed System

As figure 3. shows, the SHEMA tables and real-time functions were left untouched on the RTE system. The user interface and history logging functions were moved to the HP-UX system by writing communication modules that carried requests from one system to the other and returned with the data or a response check. The configuration is based upon NS/ARPA services on TCP/IP using Ethernet for the local area network (LAN).

Initially the communication functions were written using NetIPC calls, which locks the UNIX system into being an HP-UX system only. However, we were able to obtain a Beta Release copy of Berkeley (BSD) Sockets for the HP1000 and have since upgraded the communications programs on the two ends to BSD. This means that a user can now use any standard UNIX system that supports X, Motif, and BSD Sockets.

There are two sets of communications programs due to the directions of initial requests between the two machines. History is sent to the UNIX machine, but user module data requests are sent to RTE.

User interface programs initiate requests to read tables and place records back in them from the UNIX system to the RTE system. The RTE user communications module (COM) is the receiver who waits on the socket for a connection request from an X user UNIX system. Requests from the UNIX user interface modules are sent via messages to the UNIX communication module (MAXS_COM) which connects with COM on the RTE system and then honors the requests, sending the responses back to the calling module, again via messages. Each request opens a connection, performs its business, and then closes the connection so that another module or system can access the RTE COM module.

History is written from the RTE system to a specific UNIX system which is waiting for the connection requests. The HIST_COM module on the UNIX system waits on a socket looking for history records to relay to the history modules. The RTE module (HSEND) transfers the short-term or long file requests to HIST_COM. Again each request opens the connection, performs its business, and then closes the request. The history functions did not break quite as easily as the user interface modules did; some of the processing must remain on the RTE system, while the file writing functions had to transfer to the UNIX system. The RTE side of the short-term and long-term history modules had to account for possible link trouble. Therefore a certain amount of buffering or spooling was designed into HSEND. However, the amount of spooling is limited and is only intended to give the operator time to correct the problem.

4.2. Interoperability

Interoperability is a buzz word these days for an application that is distributed over multiple computers on the same network. The user is able to perform any of the functions from virtually any point in the network. This is basically what the X window system has provided the **MAXS+** application. There are a few points that need to be underscored here, to understand how this was achieved.

First, the RTE node name the X window user interface talks to is set in an environment variable. Commands are available to reset the environment variable to another RTE node. This directs all user communication to a specific **MAXS+**/1000 node. Also, since the node is in an environment variable, each user on a UNIX system may direct their command requests to different **MAXS+** nodes. Record locking is handled at the RTE node so that full resource sharing between multiple X users may take place.

Second, history archiving is directed to a UNIX node by setting that node's name on the RTE system side. X users may access the history files on the UNIX system using the X client/server arrangement. That is, the history clients are run on the UNIX system that actually contains the history files. It isn't just interoperability that we gain here; its speed of execution too, since history access is generally disc intensive. Note that an added benefit is that UNIX, with its disc caching, is inherently designed to efficiently process transactions like history access.

Finally, the X user interface module COM has been added to the system in such a way that it does not preclude using the stand-alone RTE **MAXS+** user interface. A user may still work from an RTE RS-232 terminal if they desire. One interesting item here is that with HP's new 2627 emulator window, GFoX, one could open a telnet window under GFoX onto the RTE system and run the

MAXS+ interface, if desired. While this would limit some of the X user interface features, such as multiple windows simultaneously, it would allow a user to open a window onto older MAXS+ systems that have not yet been upgraded to handle the X user interface communications.

The target system is represented simply in figure 4. and a more typical extended system in figure 5.

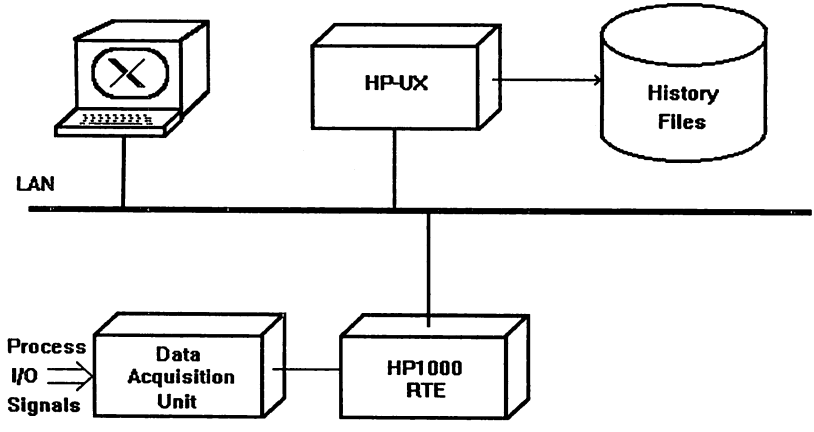


figure 4. A simple X system

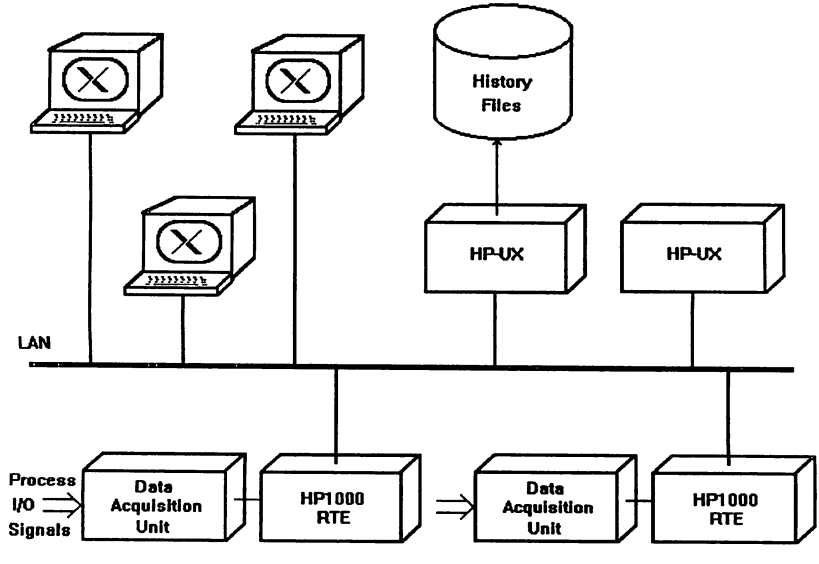


figure 5. A typical X system

4.3. User X forms

The MAXS+ application has a user interface with several dozen screen form programs. This code accounts for at least 70% of the code written for MAXS+. It would have been unrealistic to rewrite all of that code in X calls; not to mention that X is much more complex to program than serial HP-CRT calls require. Recall that we used our own screen forms library subroutines, **QFORM**, for the screen form manipulation. We developed a new product which has exactly the same subroutines and calling sequences, but performs these functions in X. We call this new product **Xfrm** (X-form). **Xfrm** allowed us to quickly and easily move the RTE screen application modules to UNIX with virtually no modifications. This saved many man-months of coding and debugging. Plus, we had a new software tool to offer for sale.

5. Communications Modules

While the communications modules were initially written using HP's NetIPC calls, we quickly converted the communications modules to using Berkeley (BSD) sockets. The prime benefit to using BSD sockets is that the X user interface could reside on virtually any UNIX platform, whereas using NetIPC calls would limit the interface to using only an HP platform. We felt that our customers would be happier knowing they could connect their non-HP systems into their HP systems and still access the MAXS+ functionality on the HP1000. Note that they would still be restricted to using the RTE system for their real-time functions, but this is not so much a restriction as it is a feature set they are provided.

The communication modules were written in FORTRAN on the RTE system and in C on the HP-UX system. These are, of course, the natural languages used on the two machines by most of us. The BSD socket routines were written for C programmers originally, and HP kept their call sequences on the HP1000 identical to the call sequences in UNIX. This was a good decision, but it does create some interesting stumbling blocks for the FORTRAN programmer, particularly on the HP1000. The basic problems arise out of FORTRAN attempting to address C structures in BSD utility routines.

5.1. A FORTRAN BSD socket module.

The following FORTRAN excerpt is from the HP1000 BSD socket program COM:

```
FTN77
$CDS ON
      PROGRAM COM(3,60)
     |,CII MAXS+ <910531.2030>
*     COM - process received LAN requests from UNIX
*
*     This program processes requests from the X user interface
*     modules.  Requests come across the LAN with the 1st word
*     indicating the command type.  Additional words are data
*     for the specific request.  The 1st word of each returned
*     buffer is reserved for error returns.
*
*     910514 - switch from HP sockets to BSD sockets
*
*     startup sequence:
*         RP,COM
*         XQ,COM
*
*************************************************************************
      IMPLICIT NONE

      INCLUDE /NS1000/INCLUDE/SOCKET.FTNI

      INTEGER*2       BUFLEN
      PARAMETER       (BUFLEN=512)
```

```
      INTEGER*2       ADDRLEN
      INTEGER*2       AF
      INTEGER*2       BACKLOG
      CHARACTER*80    CERRMSG
      INTEGER*2       DLEN
      INTEGER*4       FLAGS
      INTEGER*2       IBUFR(BUFLEN)
      INTEGER*2       IERR
      INTEGER*2       IERRMSG(40)
      INTEGER*2       INDEX
      INTEGER*4       IO_RESULT
      INTEGER*2       IRESULT(2)
      INTEGER*4       JLEN
      INTEGER*2       L
      INTEGER*2       LEN
      INTEGER*2       MAXS_COM(5)
      INTEGER*2       MSG
      INTEGER*2       OFFSET
      INTEGER*2       PROTO
      INTEGER*2       R
      INTEGER*2       REQUEST
      INTEGER*4       RTN_LENGTH
      INTEGER*2       S
      INTEGER*2       SD
      INTEGER*2       SERVPTR
      INTEGER*2       SH
      INTEGER*2       SO
      INTEGER*2       SO_TYPE
      INTEGER*2       TRIMLEN

* -- functions
      INTEGER*2       IFBRK

* -- trick buffer for indirect resolution of pointers
* -- (since the BSD socket routines are C compatible)
$ALIAS /MEM/ = 0
      COMMON /MEM/MEM(0:1)
      INTEGER*2       MEM


      EQUIVALENCE     (CERRMSG,IERRMSG)
      EQUIVALENCE     (RESULT, IRESULT)

      EQUIVALENCE (IBUFR(1),REQUEST)

* -- service name (note termination by NULL byte)
      DATA MAXS_COM/'maxs_com',0/



      CALL DTACH

* -- create a call socket
      AF = AF_INET
      SO_TYPE = SOCK_STREAM
      PROTO = IPPROTO_TCP
      SD = SOCKET(AF,SO_TYPE,PROTO)
      IF (SD .EQ. -1) THEN
          CERRMSG='COM:  Unable to create a call socket.'
          GOTO 999
      ENDIF

*-- bind the socket to the service name
      SERVPTR=GetServByName(ByteAdrOf(MAXS_COM,0),0)
      IF(SERVPTR .EQ. 0) THEN
          CERRMSG='COM: service name not found'
          GO TO 999
      ENDIF
      SIN_FAMILY = AF_INET
      SIN_PORT = MEM(SERVPTR+2)
      ADDRLEN = 16                    ! note: IP adrs is ignored
      B = BIND(SD, AddressOf(SOCKADDR_IN), ADDRLEN)
      IF(B .EQ. -1) THEN
          CERRMSG='COM: Unable to bind socket'
          GO TO 999
      ENDIF

* -- set up listen queue
      BACKLOG = 3
      L = LISTEN(SD,BACKLOG)
      IF(L .EQ. -1) THEN
          CERRMSG = 'COM: listen rejected'
          GO TO 999
      ENDIF
```

```
* -- await remote connection request
  100 CONTINUE
      ADDRLEN = 16
      A = ACCEPT(SD, AddressOf(SOCKADDR_IN), AddressOf(ADDRLEN))
      IF(A .EQ. -1) THEN
          SH = SHUTDOWN(SD, 2)
          CERRMSG='COM: accept failed'
          GO TO 999
      ENDIF

* -- fetch input message
  200 IF(IFBRK().NE.0) THEN
          SH = SHUTDOWN(A, 2)
          SH = SHUTDOWN(SD, 2)
          CERRMSG='COM: break detected, shutdown'
          GO TO 999
      ENDIF

      LEN = BUFLEN * 2
      FLAGS = 0
      R = RECV(A, ByteAdrOf(IBUFR,0), LEN, FLAGS)
      IF(R .EQ. -1) THEN
          SH = SHUTDOWN(A, 2)
          GO TO 100
      ENDIF


* -- Now process the command request
      GO TO(1000,1100,1200,1300,1400,1500,1600,1700,1800,1900,
     |      2000,2100,2200,2300,2400,2500,2600,2700,2800,2900,
     |      3000,3100,3200,3300), IBUFR(1)

*--Illegal request code
      IBUFR(1) = -1
      RTN_LENGTH = 2j
      GO TO 9000

* -- 1st Request
1000  CONTINUE
      . . .
      GO TO 9000

* -- 2nd Request
1100  CONTINUE
      . . .
      GO TO 9000


* -- etc.
      . . .


* -- Send reply to caller
* -- RTN_LENGTH is number of return bytes
9000  CONTINUE
      DLEN = RTN_LENGTH
      FLAGS = 0
      OFFSET = 0
      S = SEND(A, ByteAdrOf(IBUFR,OFFSET), DLEN, FLAGS)
      IF(S .EQ. -1) THEN
          CERRMSG = 'COM: Unable to send packet to 9000'
          GO TO 999
      ENDIF

*--go await another request
      GOTO 200

*--ERROR TERMINATION
  999 IF(CERRMSG.NE.' ') THEN
          CALL SYCON(IERRMSG,-TRIMLEN(CERRMSG))
      ENDIF

      CERRMSG = 'COM terminated'
      CALL SYCON( IERRMSG,-11 )
      END
```

Note that the common buffer MEM, in the above listing, is referenced against address 0, and starts with index zero. This is because its sole purpose is to allow the resolution of addresses from pointers. FORTRAN programmers aren't used to dealing with the concept of pointers; its a C concept. But since BSD socket routines are C compatible, they include pointers.

Resolving pointers is further complicated by the fact that the HP1000 is a word addressing machine, whereas C is generally used on byte addressing machines. The C on the HP1000 uses word addresses, except for character variables for which it uses character addressing. Character addresses must be divided by 2 before using as a word address in memory. Fortunately, we can assume that C aligns character strings on a word boundary.

Examine these pieces of HP's BSD include files:

```
/* excerpt from HP's /NS1000/INCLUDE/NETDB.H file */

struct    servent {
    char    *s_name;        /* official service name */
    char    **s_aliases;    /* alias list */
    int     s_port;         /* port # */
    char    *s_proto;       /* protocol to use */
};


C excerpt from HP's /NS1000/INCLUDE/SOCKET.FTNI file

        INTEGER   SERVENT(4)
        INTEGER   S_NAME,S_ALIASES,S_PORT,S_PROTO
        EQUIVALENCE (SERVENT(1),S_NAME)
        EQUIVALENCE (SERVENT(2),S_ALIASES)
        EQUIVALENCE (SERVENT(3),S_PORT)
        EQUIVALENCE (SERVENT(4),S_PROTO)
```

These are both definitions of the same service structure; one for C and one for FORTRAN. The service pointer and the contents of its array are all integers. Some of the values are pointers to character arrays or arrays of character pointers. However, to see this, you must compare the C version of the structure to the FORTRAN array.

A pointer to an integer value, such as the port number, is resolved by

```
I = MEM (SERVPTR + k)
```

where 'k' is some constant offset beyond the pointer, where the value is sitting. Therefore, fetching SIN_PORT becomes

```
PORT = MEM (SERVPTR + 2)
```

If we use the service pointer to fetch the first two characters of the protocol name,

```
PROTO = MEM (MEM (SERVPTR + 3) / 2)
```

This is because the pointer plus 3 points to the S_PROTO value, which is a character address of the protocol name.

Fetching the first two characters of the first alias name,

```
NAME = MEM (MEM (MEM (SERVPTR + 1)) / 2)
```

6. Summary

The project to distribute the **MAXS+** application between RTE systems and UNIX systems provided our customers with the X interface they desired, and with the level of distribution that all the industry is talking about. The salient features provided are:

- Standard X-Window user interface
- Full interoperability (multiple hardware platform)
- BSD Sockets
- History on UNIX
- **Xfrm** product

The standard X interface allows users to open multiple windows for greater functionality. Interoperability means a wider access to real-time information. Due to the Berkeley sockets, intercommunication is able to span multiple vendors' platforms.

Placing the history files on a UNIX system opens up history access for some easy data pipelines into current data bases available under UNIX.

The **Xfrm** product didn't just make the project easier, it provides a tool to allow users to create their own custom application programs and incorporate them into the total environment, as if they were an original part of the system.

1008

USING RTE SYSTEM LIBRARY ROUTINES TO CONTROL AUTOMATED PROGRAM EXECUTION

Wendy King

U.S. Naval Observatory
Time Service Department
34th & Massachusetts Avenue, NW
Washington, DC 20392-5100
(202) 653-0486

INTRODUCTION

The U.S. Naval Observatory (USNO) Time Service generates, maintains, and improves the USNO reference time scale which is used to monitor and control U.S. Air Force, Coast Guard, and Navy time-based navigation and communication systems. It is also used by scientific personnel in laboratories world-wide for time synchronization.

To accomplish this mission, two HP1000 A900s continuously collect data from directly connected satellite receivers; take hourly time interval measurements from 20 to 30 atomic clocks; collect timing data from 20 remote USNO data acquisition systems installed in Hawaii, Alaska, Norway, and various points in between; and acquire data from several Earth Stations. These data are processed and transferred to other systems on the USNO LAN which either process the data further or disseminate it to other users, outside agencies and organizations.

THE PROBLEM

For 11 years, we used an HP1000 F system running RTE IVB to collect and process the data. Eventually, the application requirements exceeded the capabilities of the F, and, in 1988, we installed the first HP1000 A900 and began the process of migration.

In the beginning, the new A900 ran quietly, doing what was asked with a minimum of fuss. The collection and processing programs were slowly migrated from the HP1000 F (RTE IVB) to the A900. Programs which needed to be run automatically were scheduled at boot up from the welcome file by a program which calculated their next run time based on the required start and interval time. This was necessary to enable programs which ran more often than once a day to survive an unscheduled re-boot. All the programs ran in system session. However, as the list of automatic, time-scheduled events grew, this strategy began to disintegrate. Six specific problems emerged.

First, there were silent, unobtrusive, and mysterious failures. A scheduled printout did not appear. There was no indication of problems with the process which produced it; it simply did not get to the printer. Then, an automatic taping process failed. There were no clues as to why TF hadn't run; it just did not make the tape. This was getting more serious because that was a critical automated archive which was very difficult to recover. Several weeks passed, during which users complained that random pieces of time scheduled processes were mysteriously failing to run. I was scrolling back the system console buffer one morning and saw the following message:

```
Program already exists in another session:  PRIN1
```

I called the Response Center. "This is the way it is supposed to be. The system will not execute a program in system session if the same program is already running in another session." The mystery was solved but the failures were increasing as system usage increased:

```
Program already exists in another session:  CIX
Program already exists in another session:  TF
Program already exists in another session:  EDIT
```

Clearly, I needed to revise the strategy for running time-scheduled processes if I wanted automated processes to co-exist with users on the system.

Second, as more hourly automatic processes were added, some began to overlap each other. When program #2 kicked in before program #1 was finished, and program #2 changed the current working directory, program #1 failed to find its files. Many of our automatic processes ran a copy of CI and executed a command file. I did not want to begin a list of things users could not use in their processes, such as the WD command. Also, maintenance would be more complicated if the full path name for every program and every file had to be used instead of switching the working directory.

Third, the application programs themselves were in the system time list. When users modified their automatic programs, they had to "off" them before recompiling and relinking. General users do not have write privilege to the /PROGRAMS directory. Therefore, they also had to ask the system manager to install the revised program(s) in the /PROGRAMS directory. The system manager had to re-schedule the program(s) to be sure it was done correctly. During this software development stage, users revised the code almost daily.

Fourth, if one application failed and went interactive to the system console, the pending read could cause a pile-up of programs that were I/O suspended. One such failure could multiply into many, and recovery could take hours to accomplish. Even though running in batch mode, some RTE utilities (including CI) insist on issuing a pending read to the log lu when they fail to find a file or encounter some other problem. Setting a short time out on the log terminal did not solve this problem sufficiently.

Fifth, after installing NS/ARPA, the "WH" output for system session contained so many programs it was very difficult to find the user applications to do a status check when there was a problem.

Sixth, the volume of output from the automatic applications filled the system console buffer very quickly. Significant error messages mixed in with normal application status messages disappeared from the terminal before I knew there was a problem.

I needed to develop a strategy that would control and monitor the automatic execution of application programs and processes in the following way:

1) remove the automatic applications from the system session
2) separate automatic applications from each other
3) allow application programs to be run automatically without putting the application programs themselves in the time list
4) redirect application output away from the system console
5) provide the capability to automatically kill them when necessary
6) provide control and maintenance tools that anyone could use to alter the schedule in the absence of the system manager.

THE SOLUTION

The strategy which solves all the above problems includes the following tools:

1) An ASCII file with the list of programs, start times, and intervals.
2) The programs in the list which schedule the actual applications.
3) A program which uses the list to schedule the programs.
4) A program which uses the list to re-schedule one or all of the programs.
5) A program which uses the list to "off" one or all the of programs.
6) A command file which compiles, links, and schedules a "list" program.

1) - The List of What and When

The starting point is an ASCII (type 4) file called /SYSTEM/TIMLST.CMD which contains the list of what should be run, how often and when. To change the schedule, all that's needed is to edit the file and run the program which re-schedules the specified program. An example of the file appears in Appendix A. I kept the RTE format for the AT command for the sake of user friendliness and simplicity. The file also includes editing instructions to maintain the required format accepted by the programs which use it.

2) - The Application Scheduler Programs

These are the programs specified in the What-When list. They are all cloned from the template program in Appendix B. They all run in programmatic session 260 with the system console as their log lu. When executed, each one does the following:

- gets a unique session number from the system
- logs on a new programmatic session
- attaches itself to that session
- switches the log lu from the system console to another terminal
- schedules its application program without wait (XQ)
- attaches back to the main scheduler session 260
- switches the log lu back to the system console
- monitors the session it just created to log it off when all active programs have completed, or when the time limit defined for that process has been exceeded.
- returns the application's session number to the system.

These are the programs which actually schedule the applications to run, and monitor their sessions to be sure they are terminated and logged off in a timely manner. This ensures that problems with one application do not interfere with applications which follow.

3) - The Time List Controller/Monitor Program

The third part of the strategy is a program called SCHED. This program is executed in background at boot-up from the welcome file. The source listing for this program appears in Appendix C. SCHED logs on session 260 (I selected this number arbitrarily) if it does not already exist, then attaches itself to session 260 and reads the What-When list. For each program in the What-When list, SCHED checks for an ID segment.

If there is no ID segment, SCHED RP's the program, calculates the next run time based on the start and interval time specified, and schedules the program to run at the correct time.

If there is an ID segment for the program, SCHED checks the ID segment time list bit. If it is set, the program is in the system time list and nothing is done. If not, SCHED calculates the next run time for this program based on the start and interval time, and restores the program to the system time list by scheduling it to run at the correct time.

SCHED repeats the process hourly, restoring any programs which may have accidentally been "off'd" or re-scheduling any programs which have been removed from the system time list.

4 - The Re-scheduler: RESCHED

RESCHED, which re-schedules one specific program, is a clone of SCHED which has been modified to accept a program name in the runstring, read through the What-When list until that program is found, remove it from the time list, re-calculate the next run time, and schedule it to run at the new time. This program is used to change the time a process should be run, or to add a new one to the system time list. The What-When list must first be edited to include the new time or to add the new process. The only routine used by RESCHED not used by the other programs is the Exec 12 call to remove a program from the time list by setting the time interval parameter to zero. For example, "call exec(12,IRpName,0)". This is used before re-scheduling the program at its new time.

5 - The Time List Killer: OFSCHED

When the equipment delivering data to the A900 fails or is shutdown, it is necessary to "turn off" either the automatic collection and/or processing of that data until it is restored. OFSCHED is a clone of RESCHED which removes a program from the time list. OFSCHED accepts from the runstring a specific program name or the key word "ALL". It uses Exec 12 to remove a program from the timelist, leaving it dormant in session 260. When this is done, the What-When list must be edited to put a "*" in column 1 of the line for that program to prevent SCHED from restoring the program on its next hourly time list check. If turning off all the programs, it is easier to "OF SCHED" rather than edit every line in the What-When file to prevent premature re-instatement of the time list.

For emergency termination, the easiest way to stop the execution of the time list completely is to kill session 260. I created a command file called "KillTimeList.cmd" which simply issues the KILLSES command. This makes it easier for users to use as they almost never have to use the KILLSES command and are unlikely to remember it. The file contains the session 260 number so they don't even have to remember that. This file also issues the "OF SCHED ID" command.

6 - The Scheduler Program Installer: INSTALL.CMD

This command file ensures that each scheduler program is linked as a system utility and resides in the /PROGRAMS directory. It takes care of removing the current version if this is not a new program. The file looks like this:

```
if ftn7x $1.ftn 0 -
then
  link $1.rel +su
  of $1/260 id
  co $1.run /programs/ DP
  pu $1.rel
  resched $1
fi
```

All scheduler programs are linked as system utilities so they can not be cloned. This ensures that there can never be two copies running at the same time.

THE RTE ROUTINES

All the routines are listed in Appendix D along with the location of their manual documentation.

Solution requirements 1, 2, and 3:

1 - remove the automatic applications from the system session
2 - separate automatic applications from each other
3 - allow application programs to be run automatically without putting the application programs themselves in the time list

These three requirements are met by the application scheduler programs (Appendix B) using the following routines:

GetSn	-	allocates a unique session number
Clgon	-	logs on a programmatic session
Atach	-	attaches the caller to the specified session
Dtach	-	attaches the caller to system session
IdClr	-	sets a flag to kill the id segment on termination

To logon a programmatic session requires a valid account. I created an account specifically for the automatic programs. The main time list session 260 and all the temporary sessions for the applications use this account.

The first three functions, GetSn, Clgon, and Atach, are simple and have never failed me. If you look at how these routines are used in Appendix B, you will note that I included provision to display the error if one should occur, but do not terminate the process of scheduling the application. The error display will allow me to trouble shoot if necessary but the occurrence of an error with these routines should not be allowed to prevent the execution of the automatic application.

Each separate application has its own scheduler. This allows each application to run in its own unique session number, thus satisfying the requirement that the applications be separated from each other.

The application scheduler first gets a unique session number with GetSn, uses Clgon to log on a session using this number, then uses the Atach routine to move itself into the new session just created. Neither FmpRpProgram nor FmpRunProgram allow you to specify a session other than the one in which you are currently executing. To get the application to run in its own separate and unique session, the scheduler itself must be running in the new session when it schedules the application. Once the scheduler has kicked off the application(s), it uses Atach to move back to its original session, leaving the application running by itself in the new programmatic session. It then monitors the new session to log it off as soon as the application is finished. This requires that the scheduler always use "XQ" to run the application in the background.

Sometimes, while the scheduler program is running in the application session, the time list monitor runs its hourly check of the time list, and finding the scheduler missing from session 260, re-schedules it in 260. To accommodate this occurrence, the scheduler checks the error after attempting to attach back to session 260, and if an error has occurred, it will use Dtach to move to system session, Exec 12 to remove itself from the system time list, then IdClr to set a flag to kill its id segment when it terminates. This enables the scheduler to remove itself from the application session so it can still monitor and kill it appropriately while avoiding the possibility that two scheduled copies of itself will continue to exist. The fact that these programs are linked as system utilities should prevent this, but I did not want to leave anything to chance!

Solution Requirements 4 and 5:

4 - redirect application output away from the system console
5 - provide the capability to automatically kill them when necessary

These requirements are met by the following routines:

AtCrt	-	attaches a crt to a session
LuSes	-	returns the user table address
IxGet	-	returns the contents of an address
Clgof	-	logs off a session
RtnSn	-	deallocates a session number

The AtCrt routine puts the lu of a crt into word 29 of the caller's ID segment. The manual implies that the program must first use the Atach routine and must be a system utility. I did not find this to be true as it worked for all my programs regardless of whether they were system utilities or had used the Atach call. Anyway, the effect of using AtCrt is that all output directed to "1" now goes to the system lu specified in the AtCrt call. This satisfies the requirement to redirect all application output to a terminal other than the system console. All application programs scheduled with FmpRunProgram inherit the father's output lu, so the AtCrt is called before scheduling the application program. The application program then uses the new lu for all its output. The scheduler then resets its own output lu with a second call to AtCrt so that any subsequent output from the scheduler program will appear on the system console. This allows all users to write their status messages to "1", and enables the system manager to control the actual location for the output. Because each application has its own scheduler, different applications can have their output directed to different locations.

LuSes, IxGet and Clgof satisfy the 5th solution requirement to be able to kill the application session. Although the documentation for Clgof implied that if I used the Option 0, it would logoff the session when there were no "active" programs, I found that in this case the term "active" really meant RP'd. Any program used by the application which terminates but remains dormant in the session is considered an "active" program. So if you use Clgof with option 0, the logoff fails if any program's ID segment is not released when the program is finished.

LuSes and IxGet allows the scheduler to determine when no programs are running so the log off can be accomplished as soon as the application has truly finished. LuSes returns the address of the User Table for that session, and IxGet returns the contents of an address, in this case word 13. Word 13 of the User Table contains a "Number of User Programs Counter" which is incremented when programs are scheduled, and decremented when they become dormant. Bit 15 is set only if there is a logoff program or command file defined for this user. As this is not the case for the automatic application account, I need only check for the value of this word to be 0. As soon as this occurs, the scheduler logs off the session.

Each of the applications can be expected to terminate within a specified time limit. The MaxTime variable in the scheduler program represents the maximum number of minutes allowed for the application. The scheduler uses this predefined time limit to determine when to log off the session even though programs are still running. In this way, an interactive CI prompt, or a user program which has entered an infinite loop, can be terminated so that subsequent programs can not get "piled" up behind it. If the session does exceed the time limit, a message is displayed to the system console so that corrective action can be taken. The application scheduler ends with the RtnSn routine which releases the session number back to the system.

Solution requirement 6:

6 - provide control and maintenance tools that anyone can use to alter the schedule in the absence of the system manager.

This last requirement is met by the SCHED, RESCHED, and OFSCHED programs using Clgon, Atach, Dtach, IdGet, IxGet, FmpRpProgram, Exec 12 and ChngPr (See Appendix C).

The first three have already been described as they are used in the application scheduler programs. IdGet returns the address of the Id segment of a specified program in a specified session. To verify the existence of an Id segment for a scheduler program, SCHED uses IdGet with the program name and the session number 260. If IdGet returns a 0, there is no Id Segment for. the program and it must be RP'd and re-scheduled. If IdGet returns an address (anything other than 0), then I need to verify that the program is actually in the system time list.

IdGet has a companion routine called IdInfo which returns information from the Id Segment. However, on the A900, IdInfo does not differentiate between the three possible dormant states. Therefore, SCHED uses IxGet to return the contents of word 18 of the Id segment. Bit 12 of word 18 is the timelist bit. If it is set, the program is in the time list. If not, SCHED re-schedules the program after calculating the next run time.

RESCHED is a clone of SCHED which accepts a program name from the runstring, and after removing the specified program from the time list with an EXEC 12, calculates the next run time and re-schedules the program. This is used to change the time when something should be run. Because it is run manually, and the system could be very busy, this program increases its priority with a call to ChnPr. This ensures that the re-scheduling of the program is done immediately. The Exec 12 routine removes a program from the time list if the time interval parameter is set to 0 (e.g., call Exec(12,ProgName,0). ProgName must be the Rp name in a 3-word integer array.

OFSCHED is the last of the three controller programs. This program removes one or all the programs from the time list. This also uses the Exec 12 to remove the program(s) from the time list. This has been very useful during system restoration after a re-gen, when I needed to inhibit the automatic processes. I can run it manually after boot-up, or put it in the welcome file. To re-start everything, I just "XQ SCHED" to restore the What-When programs to the system time list.

PITFALLS

System release 5.1 and 5.2 both have problems relating to what happens when a CI command file terminates with "EX" if it was originally scheduled with "XQ". When running in background, the EX command to CI results in the total destruction of the session, even if other programs are still running. Thus, a command file which issues a PRINT command and followed by an EX will result in the session being terminated before the file completes printing. The rule I use for ending CI command files is to always specify "EX,B". The application scheduler always takes care of logging off the session.

CONCLUSIONS

In the beginning, I wrote status messages from the scheduler programs to the system console. I could see at a glance what had been run, at what time, for how long, and in what sequence for the past several hours. Once I was confident that the strategy and the code were working as intended, I turned off the system console status displays. Now, the only output sent to the system console is error messages. All the ongoing application status messages are displayed on a second terminal reserved for that purpose. Now, all system errors are immediately obvious. Output to the system console is controlled by a logical flag in the scheduler programs. If I need to turn it on again, I just edit the source file, changing the flag to TRUE, and use the INSTALL.CMD file to install the new version in the time list.

Now that our system is in place, and has been tested numerous times in different situations, I can not imagine keeping this pair of A900s under control without it. It is just one more example of the HP1000 RTE flexibility and easy adaptability to user control requirements. I am sure that I will continue to refine the system as new requirements occur. Any new discoveries will be included in the paper presentation.

APPENDIX A

THE WHAT-WHEN LIST: /SYSTEM/TIMLST.CMD

```
*-------------------------------------------------------------------
* This file is used by SCHED, RESCHED & OFSCHED to control the time list
* The first four fields must conform to a specific format.  Everything
* to the right of the 4th field is ignored so you can put anything there
* Specific format instructions are at the end of this file.
*-------------------------------------------------------------------
*   When     Int  What     Run String executed:            Application:
*-------------------------------------------------------------------
* Hourly
AT 00:00:01   1H SCCLD  ! ru /timesc/cldat.run              CLOCK SCAN
AT 00:08      1H SCDAS  ! ru cicopy /das/calldas.cmd        DIALOUT CALLS
AT 00:15      1H SCMTR  ! ru cicopy /dey/monitor.cmd        FTP DATA TO 835
AT 00:20      1H SCWDG  ! ru /timesc/wchdg.run              CLOCK WATCH DOG
* Daily
AT 00:00:01  24H SCCYR  ! ru /sysprogs/chkyr.run           CHECK YEAR
AT 00:30     24H SCGPM  ! ru cicopy /gps/gps_midnite.cmd   MIDNITE GPS
AT 01:30     24H SCHKP  ! ru cicopy /hskp/hskp.cmd         HOUSEKEEPING
AT 02:22     24H SCBKP  ! ru /mgr/backup/backup.run        BACKUP
AT 05:30     24H SCHLT  ! ru /sysprogs/rboot.run           BOOT SYSTEM
AT 09:05     24H SCWMS  ! ru cicopy /snoopy/ms.cmd         PREPARE DATA
AT 09:30     12H SCGPS  ! ru cicopy /gps/gps_process.cmd   PROCESS GPS
AT 11:35     24H SCMCS  ! ru cicopy /cksteer/ckstr.cmd     CLOCK STEER
AT 12:40     24H SCDAM  ! ru cicopy /scham/scdam.cmd       PROCESS DAS
AT 13:10     24H SCLOP  ! ru cicopy /hc/rdctn.cmd          REDUCE LORAN
AT 18:35     24H SCTTG  ! ru cicopy /hc/wttg/udtv.cmd      TV TIME UPDT
** 21:30     12H SCGPS  ! ru cicopy /gps/gps_process.cmd   PROCESS GPS
AT 22:00     24H SCSID  ! ru /sysprogs/iontr.run           BREAK SID FILES
AT 22:30     24H SCSAT  ! ru cicopy /gps/gpsat.cmd         MORE GPS PROCES
AT 23:30     24H SCXPT  ! ru cicopy /export/export.cmd     FTP ->MATSAKIS
*-------------------------------------------------------------------
* Required format: **THERE MUST BE AT LEAST ONE SPACE BETWEEN FIELDS**
* column 1:  * for comment lines   OR
*            blank if the whole line is blank   OR
*            blank if the next four fields are correct (accidental
*            whole line shift to the right will be tolerated)   OR
*            first character of the first field.
* First    - MUST be at least 1 ascii character; this is really a place
* field      holder since the "AT" is used for user friendliness so the
*            line makes sense.  You could put ZZ there and the program
*            will not care.  The character must be printable.
* Second   - NO spaces; the hour, minute, and second values MUST be
* field      separated (delimited) by a ":"; the first number will be
*            interpreted as hour, the second as minute, the third as
*            second.  Leading 0s are not necessary for the program to
*            work.  There must be at least one numeric digit; Omitted
*            fields default to 0 but do not omit minute if seconds is
*            not also 0.
* Third    - NO SPACE BETWEEN THE INTERVAL NUMBER AND UNIT CHARACTER!
* field      Lower case will be accepted
* Fourth   - The name of the program; if no directory path is specified,
* field      it will default to the /programs directory. The .RUN
*            extension is also not required.
```

APPENDIX B

APPLICATION SCHEDULER PROGRAM TEMPLATE

```
FTN7X,L

      PROGRAM SC<prog name>(), schedule <application description>
*---------------------------------------------------------------------
* Programmer:  Wendy King
* Created:     August 31,1990
* Revised:     <910528.1758>
* Purpose:     Template program; customize to fit application
*              Create a unique programmatic session, attach to that
*              session and run a program, then return to original
*              session and log off the auto programmatic session when
*              all active programs have completed.
*---------------------------------------------------------------------
      implicit none
      character*5  ProgId                  ! name of this program
      character*72 msg0(0:2)               ! getsn errors
      character*72 msg1(0:7)               ! clgon errors
      character*72 msg2(0:5)               ! atach errors
      character*72 msg3(0:2)               ! clgof errors
      character*5  RpName                  ! true program name
      character*80 RunString(1)            ! to schedule program
      integer*2    StdOut,StartLu,SystemConsole,NProgs,I
      integer*2    ITime(15),SesNum,error,Opt,Count,MaxTime
      integer*2    BufLen,buffer(4),Parms(5),FmpRunProgram,StrLen
      integer*2    Clgon,Clgof,GetSn,Atach,RtnSn,UsNum,TrimLen
      integer*2    LuSes,IXGet,Active_Progs,UsrIdTblAddr
      logical      Continue,ConsoleDisplay
*---------------------------------------------------------------------
* CUSTOMIZE THIS SECTION
* Replace the 5 x's with the 5-char name of this program

      data ProgId         /'xxxxx'/

* Replace "prog.run::dir" with the full path and program name to be run
* Duplicate this line for each program to be scheduled; for run w/wait
* use RU instead of XQ.  Set NProgs to the number of programs to be run.
* For additional runstrings, increase the array value in the declaration
* and increment the aray value for each data statement.

      data RunString(1)  /'XQ,prog.run::dir,parameters if any'/
      data NProgs        / 1 /

* Replace n with a number which represents the Maximum number of
* minutes the process should take.

      data MaxTime        / n /

* replace n with the lu of the terminal where you want the
* application program output(s) to be displayed.

      data StdOut         / n /
```

```
* set ConsoleDisplay to false if you want only error messages on the
* system console; set it to true if you want progress/status messages
* on the system console.

      data ConsoleDisplay /.false./

*-----------------------------------------------------------------------
* DO NOT CHANGE ANYTHING FROM HERE ON.

      data buffer        /'AT','/T','IM','E '/    ! account & password
      data BufLen        / 7 /                    ! # chars in buffer
      data Opt           / 1 /                    ! clgof option
      data Count         / 0 /                    ! count log off trys
      data SystemConsole / 1 /                    ! System Console lu
      data Continue      /.true./                 ! flag for logoff loop

* Error messages for getsn
      data msg0(0)(1:)/' 0, No error.'/
      data msg0(1)(1:)/'-1, Cannot get a session number.'/
      data msg0(2)(1:)/'-2, No more session numbers available.'/

* Error messages for clgon
      data msg1(0)(1:)/' 0, No error.'/
      data msg1(1)(1:)/'-1, Internal error, such as no class numbers,
     > or logon not performed.'/
      data msg1(2)(1:)/'-2, No -2 error documented in the manual.'/
      data msg1(3)(1:)/'-3, Too many sessions active.'/
      data msg1(4)(1:)/'-4, No such user.'/
      data msg1(5)(1:)/'-5, Bad or missing password.'/
      data msg1(6)(1:)/'-6, File is not valid user file.'/
      data msg1(7)(1:)/'-7, User configuration file already open.'/

* Error messages for atach
      data msg2(0)/' 0, No error.'/
      data msg2(1)/'-1, session number does not exist.'/
      data msg2(2)/'-2, specified program does not exist.'/
      data msg2(3)/'-3, current session number does not exist.'/
      data msg2(4)/'-4, must be superuser for action requested.'/
      data msg2(5)/'-5, program with same name already exists.'/

* Error messages for clgof
      data msg3(0)(1:)/' (0) Log off completed: no error.'/
      data msg3(1)(1:)/' (-1) There are active programs; Option was 0.'/
      data msg3(2)(1:)/' (-2) Session already logged off.'/
*-----------------------------------------------------------------------
* Identify program and revision number; display time.

      if(ConsoleDisplay) then
        write(1,'(/a)')'---------------------------------------'
        call Ftime(ITime)
        write(1,'(a5,1x,15a2)')ProgId,ITime
      endif
```

```
* save the initial output log lu

      call loglu(StartLu)

* get a unique  session #
      error = GetSN(SesNum)
      if(error.lt.0) then
        StrLen = trimlen(msg0(-(error)))
        write(1,*)ProgId,' GetSn: ',msg0(-(error))(1:StrLen)
        sesnum = 999
      endif

* logon TS/AUTO programmatic session to new session number

      error = clgon(buffer,BufLen,SesNum,error)
      if(error.lt.0) then
        StrLen = trimlen(msg1(-(error)))
        write(1,*)ProgId,' Clgon: ',msg1(-(error))(1:StrLen)
      endif

* atach to new AT/TIME session
      error = atach(sesnum,error)
      if(error.lt.0) then
        StrLen = trimlen(msg2(-(error)))
        write(1,*)ProgId,' Atach: ',msg2(-(error))(1:StrLen)
      endif

* announce output destination

      if(ConsoleDisplay) then
        write(1,*)ProgId,' Running in session number ',UsNum()
        do i = 1,NProgs
          StrLen = trimlen(RunString(i))
          write(1,*)ProgId,' ',RunString(i)(1:StrLen)
        enddo
        write(1,*)ProgId,' Look for output on lu ',StdOut
      endif

* change the output lu to stdout

      call AtCrt(StdOut)

* display status message if standard output device is to be the
* system console for this application and console display is turned
* off, or if the standard output device is not the system console.

      do i = 1,NProgs
        StrLen = TrimLen(RunString(i))
        if( (StdOut.ne.SystemConsole).or.
     >      (StdOut.eq.SystemConsole).and.(.not.ConsoleDisplay) ) then
          write(1,'(/a)')'----------------------------------------'
          call FTime(ITime)
          write(1,'(a5,1x,15a2)')ProgId,ITime
          write(1,*)ProgId,' ',RunString(i)(1:StrLen)
        endif
```

```
      error = FmpRunProgram(RunString(i)(1:StrLen),Parms,RpName)
      if(error.lt.0) then
        write(1,*)ProgId,' FmpRunProgram ',RunString(i)(1:StrLen)
        write(1,*)ProgId,' FmpRunProgram Error: ',Error
        write(1,*)ProgId,' FmpRunProgram Errors: ',Parms
      endif
    enddo

* re-atach (return) to AT/TIME session 260

    error = atach(260,error)       ! try to Atach back to 260
    call AtCrt(StartLu)            ! switch output back to original
    if(error.lt.0) then            ! if error dtach to system session
      StrLen = trimlen(msg2(-(error)))
      write(1,*)ProgId,' atach: ',msg2(-(error))(1:StrLen)
      call Dtach(error)
      if(error.lt.0)write(1,*)ProgId,' Dtach Error: ',error
      call exec(12,0,0)            ! remove me from the time list
      call IdClr()                 ! set flag to kill my ID seg
    else
      if(ConsoleDisplay) then
        write(1,*)ProgId,' Returned to session ',UsNum()
        write(1,*)ProgId,' Waiting to logoff session ',sesnum
      endif
    endif

* terminate the session after active programs have completed;
* if programs remain active past the Max time expected, logoff and
* kill all active programs (assume there is a problem with the
* application).

    do while(continue)

      UsrIdTblAddr = LuSes(SesNum)         !Get User Table  address
      if(UsrIdTblAddr.le.0) then           !Session already logged off
        continue = .false.                 !set flag to quit
      else
       Active_Progs = IXGet(UsrIdTblAddr + 12)! active progs count
       if(Active_Progs.eq.0) then          !Progs finished; logoff now
         continue = .false.                !set flag to quit
       else                                !Programs not finished;
         if(count.eq.MaxTime) then
           write(1,*)ProgId,' Exceeded ',MaxTime,' minute time limit.'
           write(1,*)ProgId,' Killing session ',SesNum
           write(1,*)ProgId,' Check for errors or adjust time limit.'
           ConsoleDisplay = .true.       ! I do want the console to
           continue = .false.            ! display clgof result
         else
           count = count + 1               ! increment counter;
           call exec(12,0,3,0,-1)          ! wait 1 minute; try again.
         endif
       endif
      endif
    enddo
```

```
      error = clgof(SesNum,Opt,error)              ! log off session

      if(ConsoleDisplay) then
        StrLen = trimlen(msg3(-(error)))
        write(1,*)ProgId,msg3(-(error))(1:StrLen)
        call FTime(ITime)
        write(1,'(a5,1x,15a2)')ProgId,ITime
      endif

* return the session number to the system

      error = RtnSn(SesNum)
      if(error.eq.-1)write(1,*)ProgId,' RtnSn: -1'

      end
```

APPENDIX C

TIMELIST MONITOR/CONTROLLER PROGRAM - SCHED

```
FTN7X,L
$FILES(0,1,25)
*---------------------------------------------------------------------
* Programmer: Wendy King
* Site:       US Naval Observatory, Washington DC
* System:     HP1000 A900 RTE 5.2
* Externals:  RTE System and Fortran 77 intrinsics
* Purpose:    Schedule and/or re-schedule Time Service programs
*             which must be run automatically at certain intervals
* Last Revision   <910530.1949>
*---------------------------------------------------------------------
      PROGRAM SCHED(3,30), Restore programs to time list
      implicit none

      integer*2 NextHr,NextMin,NextSec,TmUnit,TmInt
      integer*2 time(5),parms(5),IRpName(3),IMyName(3),TSLogon(4)
      integer*2 IdSegAddr,TimeListBit,Mask,IdSegWord18,IXGet,IdGet
      integer*2 TrimLen,Session,Error,FileLu,BufLen,ios
      integer*2 FmpRpProgram,UsNum,LogonLen,Clgon,SuperUser

      character At*2,Frequency*4,CRpName*5,CMyName*6,Start*8
      character ProgramName*64,InputFile*64
      character CBuffer*80

      equivalence (IMyName,CMyName)
      equivalence (ProgramName(1:5),IRpName,CRpName)

      data parms            / 5*0                    /
      data mask             / 010000B                /
      data Session          / 260                    /
      data TSLogon          / 'AT','/T','IM','E '    /
      data LogonLen         / 7                      /
      data InputFile        / '/SYSTEM/TIMLST.CMD'   /
      data FileLu           / 101                    /
      data BufLen           / 80                     /

      if ( SuperUser(UsNum()).eq.0 ) then
        Write(1,*)'Sorry; You MUST be Super User.'
        call exec(6,0,3)
      endif

      DO WHILE (.TRUE.)                !continue indefinitely

***  if not running in system session, go there;
***  atach to TS session;  create session first if necessary
```

```
      if(UsNum().ne.0)Call dtach()      ! sets loglu to system console
      Call atach(Session,error)         ! try attach to 260
      if (error.ne.0) then              ! if it fails, logon 260
        error = clgon(TSLogon,LogonLen,Session,error)
        if (error.eq.0) call atach(Session,error)   ! attach to 260
      endif

***   Open the list of time scheduled programs

      ios = -1                                      ! file could be in
      do while (ios.ne.0)                           ! use; keep trying
        Open(FileLu,File=InputFile,Iostat=ios,Err=1) ! to open the file
1       if(ios.ne.0) call exec(12,0,2,0,-30)        ! every 30 seconds
      enddo

      DO WHILE (Ios.ne.-1)                          ! while not EOF

      Read(FileLu,Fmt='(a)',Iostat=Ios,Err=998,End=20)
     >     CBuffer(1:)
      if(trimlen(CBuffer).eq.0) goto 20          ! if blank read again
      If(CBuffer(1:1).eq.'*') goto 20            ! comment; read again
      READ(CBuffer(1:),Fmt=*,Err=20,End=20)      !
     >     At,Start,Frequency,ProgramName

*---------------------------------------------------------------------
* Check for current ID segment; if none, RpProgram and put in timelist;
* if RP fails, write error msg and go to next program; if there is an
* ID segment, check that it is in the timelist; if not, put it there,
* if it is, go do next program.
*---------------------------------------------------------------------

      IdSegAddr = IdGet(IRpName,Session)                  ! get ID Seg
      if(IdSegAddr.eq.0) then                             ! if no ID seg
        Error=FmpRpProgram(ProgramName,CRpName,'P',Error) ! try RP
        if(Error.ne.0) then                               ! if error
          Write(1,*)'Sched:  RP ',ProgramName(1:5)        ! write msg
          Write(1,*)'Sched:  Error returned was ',Error
          goto 20                                         ! do next prog
        endif
      else
        IdSegWord18  = IXGet(IdSegAddr + 17)       ! get Id Seg Word 18
        TimeListBit  = iand(mask,IdSegWord18)      ! mask off bit 12
        if(TimeListBit.ne.0) goto 20               ! if bit 12 set ok
      endif                                        ! if not re-schedule

***     Calculate the next runtime based on the interval (how often)
***     and start time.
```

.
.
.

```
**      Time schedule the program.
        call exec(12,IRpName,TmUnit,TmInt,NextHr,NextMin,NextSec,0)

20    ENDDO
      goto 999

998    Write(1,*)'SCHED: TIMLST.CMD::SYSTEM READ FAILURE!!!'
999    close(FileLu)
       call PName(IMyName)
       if(CMyName(1:5).ne.'SCHED') then   ! if true I am a clone;
         call exec(6,0,3)                 ! kill me completely
       endif
       call dtach(error)               ! move to system session
       if(error.eq.-5) then            ! if already exists in system
         call exec(6,0,3)              ! session, kill me completely
       else
         call exec(11,time)                  ! get time now
         if(Time(3).gt.40)Time(4)=Time(4)+1  ! if min>40 inc hour
         Time(4) = mod(Time(4),HpD)          ! mod hour/24
         call exec(12,0,4,1,Time(4),45,0,0)  ! sched next run
       endif
      ENDDO

      END
```

APPENDIX D

REFERENCE MANUAL DOCUMENTATION FOR RTE ROUTINES

HP1000 RELOCATABLE LIBRARIES MANUAL:

CHAPTER 5

IxGet	returns the contents of an address

CHAPTER 6

UsNum	returns the session number
SuperUser	checks if user is super user
GetSn	returns a unique session number
RtnSn	releases a session number
Dtach	moves caller into system session
AtCrt	attaches a crt
Atach	moves caller into a session
Clgon	logs on a session
Clgof	logs off a session
LuSes	returns the user table address

HP1000 PROGRAMMERS REFERENCE MANUAL:

CHAPTER 5

IdClr	sets flag to kill callers ID segment
ChngPr	change the priority of a program
Exec 6	terminate a program

CHAPTER 6

Exec 11	returns the system time
Exec 12	schedules a program now or later

CHAPTER 7

IdGet	returns the caller's id segment address
PName	returns the caller's actual name
LogLu	returns the lu of invoking terminal

CHAPTER 8

FmpRunProgram	schedules a program (no RP needed)
FmpRpProgram	restores a program ID segment

HP 1000 DS and NS over MUX Ports

Donald A. Wright
Interactive Computer Technology
2069 Lake Elmo Avenue North
Lake Elmo, MN 55042 USA
612/770-3728

Abstract:

DS and NS on the HP 1000 are excellent network services. Many people do not realize how robust these services are. They provide far more than NS services on other systems, with such functions as transparent remote Image Data Base access, and complete file transparency. But they can be expensive.

In many applications, the cost of HP's networking hardware is higher than the cost of the DS or NS software to go with it. But every HP 1000 has MUXes, and most have a spare port or two. MUXLINK is a collection of software components which allow full DS and NS services between HP 1000's, using a single MUX port on each computer in a connected pair.

The software emulates the HP HDLC cards and their drivers, so nearly all of the functionality of DS and NS is supportable. Data transfer speeds are limited to the speed of the MUX, so this mechanism is of value where DS/NS functions are desirable but the highest performance is not required. MUXLINK is a commercial product, but the author provides detailed design and internals information. The paper discusses pseudo drivers working in close cooperation with protocol programs, the protocol itself, integration with HP's DS/NS, and related issues.

Requirements:

The need is for a software mechanism providing DS/1000 and NS/1000 services over B, C, and D-MUX ports on both RTE-A and RTE-6/VM. The drivers must appear to HP's software to be HDLC drivers, so that DS/NS will allow the full range of services supported by the HDLC cards. The system should support all normal ND/DS requests, including those from the utility programs DINIT, NSINIT, DSINF, NSINF, and DSMOD. A communications link should require only one MUX or OBIO port at each end, and network traffic should not interfere with the use of other ports on the MUX. It should include a

complete error-detection and retry protocol.

The software should, if possible, include data compression. It would be desirable to allow the use of the MUX ports as terminal ports when not in use for DS/NS, and it will be helpful if normal DSINF and NSINF reports show coherent information when used to query these special links.

Eventually, the software should support all of the special features of the HDLC cards, such as Remote VCP, Remote Program Download, and Forced Cold Load. Autodial and inactivity disconnect will also be useful features.

Design Approaches:

Two different approaches to this problem were considered:

1) Write a special device driver for each type of MUX in both RTE-A and RTE-6/VM, for a total of four drivers.

2) Write a pseudo driver for each operating system, closely linked with a protocol program which communicates through normal MUX ports.

These approaches have very different sets of problems to face:

Four Drivers:

a) Large drivers are required, with lots of programming at the driver level.
b) There is a danger of being privileged too long while doing protocol manipulations.
c) Data buffers must be outside the driver while the protocol work is done, and there is no convenient place to put it.

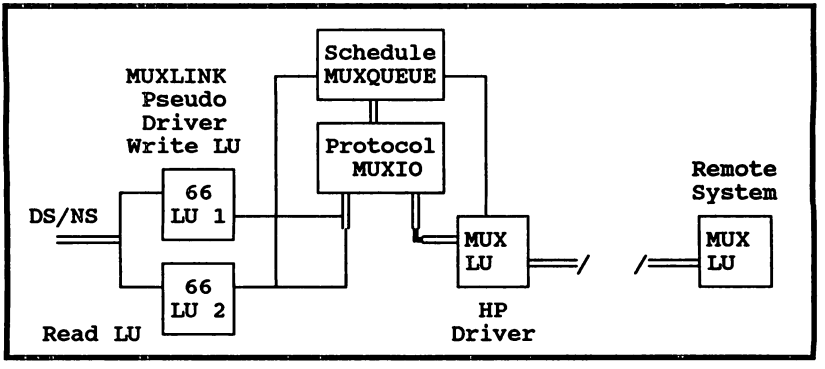
Pseudo Driver and Protocol Program:

a) This approach cannot be as efficient as the other.
b) Unsolicited remote messages and message collisions are harder to arbitrate.

Design Choice:

The pseudo driver approach was chosen. While efficiency will be somewhat reduced, it was felt that this will not affect DS/NS speed at the relatively low MUX data rates. It was also thought that this approach might actually be less intrusive and have less impact on other applications, because most of the protocol work can be done in a normal interruptible program. The drivers will definitely be much smaller, and in RTE-6/VM it will normally be installable without a system generation, using driver replacement. But the biggest advantage is that most of the code can be written and debugged at the program level rather than the driver level. This also improves the liklihood that patches and updates can be installed at the program level without a system generation.

This is a schematic diagram of the resulting design:



Protocol Requirements:

The protocol used between HP 1000 systems cannot actually be HDLC, because that protocol is very compute intensive, requiring a dedicated microprocessor. Instead, we must design a protocol which is much less compute intensive but allows use of a wide variety of connections, at least including direct connection or any lookalike (e.g. shorthaul modems), dialup modems, commercial services such as DunsNet, Tymnet, and Telenet, plus data switches and LANs.

It should support all baud rates available on the MUXes, and must allow for XON/XOFF data pacing. It need not support ENQ/ACK. It must be able to use a 7-bit channel if necessary, and should provide at least run-length encoding as a method of data compression. It must allow for the translation of special characters which may be disallowed by one connection or another.

Normal Sequence of I/O:

When the link is idle, the four major components of the system are in the following states: The Type-66 LU's are idle, MUXIO is waiting on a class GET, MUXQUEUE is dormant saving resources, and the MUX LU is in typeahead mode with a class read on it (B, C-MUX) or enabled to schedule MUXQUEUE when a character comes in (D-MUX).

Here is the sequence of events for a normal write:

1) The driver is entered by RTE with the directive to initiate a DS/NS write to the remote system.
2) The driver makes some cursory checks on the request and then schedules MUXQUEUE, setting a retry if MUXQUEUE is busy, and passes an initiation sequence number along with its own DVT/EQT address.
3) MUXQUEUE determines where the schedule came from, validates it,

and passes it to MUXIO in a class write.
4) MUXIO wakes up from its GET, examines the request for legality, and tells the driver it is complete by setting bits in the Type-66 LU's EQT/IFT and forcing an immediate driver timeout.
5) The driver is entered with a timeout directive, determines that the request is complete, and takes a completion exit.
Note: In this case the driver actually completes the request before the data is transmitted to the remote system. This is unusual, but higher-level DS/NS protocols protect against a lost message, and in fact this is exactly how the HDLC drivers work as well.
6) MUXIO then converts the message's raw data into one or more encoded 'frames', complete with headers and checksums and in-stream special characters to implement the protocol.
7) MUXIO flushes the pending read on the MUX port and begins writing the message to the remote computer, handshaking it over according to the protocol.

A read follows this sequence:

1) The first incoming frame completes MUXIO's pending read on the MUX port.
2) MUXIO wakes up from its GET and handshakes the rest of the message across.
3) MUXIO converts the incoming frames back into a DS/NS message.
4) MUXIO schedules QUEUE (HP's DS/NS incoming-message program) with details about the incoming message, giving it the Type-66 Pseudo LU number.
5) QUEUE places a class read on the Type-66 LU.
6) The Type-66 driver schedules MUXQUEUE.
7) MUXQUEUE informs MUXIO by way of a class write.
8) MUXIO goes privileged, finds the location of QUEUE's class buffer in SAM, and cross-stores it right in. Then MUXIO sets flag bits in the LU's IFT/EQT and forces an immediate driver timeout.
9) The driver detects the completion by MUXIO and takes a completion exit.

The sequences described above apply to all normal DS/NS requests. However, there are some unusual requests that must be handled as well. The most difficult of these is the Special Status Read. In this case, a program such as DSINF issues a normal I/O request (not class I/O) to the driver, and expects to get 10 or 12 words of status or statistics information back from the "card".

The drivers pass this request to MUXIO in the same fashion as all other requests. MUXIO then locks itself in memory, sets status bits in the IFT/EQT to tell the driver that special action is required, and also places there the absolute page address and the relative word offset of the Special Data in its local map. The driver then maps the data directly into an alternate map and cross-stores it to the requester's buffer.

Efficiency:

While the sequences described above do seem complex compared with the notion of performing all of the protocol and I/O directly in the driver, they take advantage of the very facilities that the HP 1000 was designed to do well. In an actual test, the Special Status Read, described above, was executed repeatedly on a Type-66 Pseudo Driver LU from a test program, with a 1024-word buffer rather than 10 or 12 words. In both RTE-A and RTE-6/VM, the test program was able to perform 100 reads per second, which is the maximum possible number when timeouts are used to pace an event in the sequence.

Protocol:

The following is an overview of the protocol which MUXLINK uses to encode data, form packets, and transmit it on RS-232 circuits.

Port-to-Port:

1) An initialization negotiation tells each side the important properties of the other, e.g. max read size.

2) When a channel is idle, C-MUX ports have a read posted and D-MUX ports have program scheduling enabled. When one side wishes to initiate transmission it just sends a packet to the other side.

3) Channel contention is always resolved in favor of the same side, determined by the initialization negotiation.

4) A single message may be broken into two or more frames.

5) After the first frame, channel contention is resolved before additional frames are sent.

6) Errors are detected by checksums and other protocol checks, and are corrected by retries.

7) DS/NS messages are currently limited to 4096 words. The programs written to implement this protocol may have such a limitation, but the protocol itself can handle at least 1 megabyte in a single frame.

8) Multiple unacknowledged frames are also supported by the protocol, though not necessarily by program implementations.

Encoding/Decoding:

1) Run-length data compression for 8-bit data is performed as an integral part of encoding, if enabled.

2) For transmission on the maximum number of possible services, the allowable encoded character set is configurable at startup. At the minimum, it may be reduced to the 64 most common characters plus up to 8 reserved management ("special") characters plus the carriage-return or other EOL character.

3) The encoded data must fit on a 7-bit channel if necessary. This will be indicated in the initialization negotiation.

4) Message frames are sent as variable-length character strings terminated by a hardware-recognizable EOL such as CR.

Initialization:

1) There are four message frames exchanged in the channel initialization process, in this order:

   I) Initialization Request
   J) Initialization Response Data
   K) Initialization Request Data
   L) Round-Trip Time Interval message

2) Basic assumption: it doesn't matter which side is primary and which is secondary. That distinction is used later only to arbitrate channel collisions.

3) When I start up I declare my side to be in an uninitialized state and begin sending primary initialization request messages or invalid frames to the other side at predetermined intervals.

4) When in the uninitialized state, I can recognize only two things: 1) Proper responses to my initialization sequence messages, or 2) The other side's primary initialization request. If an unexpected message is received before the full 4-message exchange is complete, I will execute a random delay and then read to see if a primary initialization request is present from the other side. If so I will respond to that request; if not I will send another primary initialization request myself.

5) Reception of a primary initialization request at any time will invalidate previous initializations and will cause an initialization response to be sent. The only exception is the first message received after sending my own initialization request.

6) I keep track of the time I sent an initialization response and the time I received the acknowledgement. The length of that interval is sent back to the primary side in the round-trip time interval message. This information will be used later to appropriately adjust timeout and retry intervals and may be sanity checked in actual channel use.

7) The side which sends the primary initialization request which actually succeeds is called the primary side.

Encoding Method 1:

This method employs bytewise run-length encoding, high-bit prefixing, translation-table prefixing, and special-character translation. Except for special handling of the high bit, it does not do any bitwise manipulation of the data. It is used to encode the data in C or D frames.

Reserved (special) characters are defined in the startup file. If they appear in the original data with or without the high bit set they are translated to different characters, so they never appear in the encoded output unless they really are special characters. Allowable special characters are as follows: (mnemonics represent single ASCII characters):

TTE - Translation Table Escape (single byte state change).

TTT - Translation Table Toggle. Change state and remain until another TTT or TTE.

RL0 - Precedes the data character for a run of minimum length.

RL1 - Precedes the data character for a run of minimum length +1.

RL2 - Precedes the data character for a run of minimum length +2.

RLC - Precedes the count character(s) for a counted run. The count character(s) are Basic Digits, where the 4 LSB's of each digit indicate the count (0-15) and the MSB indicates whether additional count digits follow. The data character follows the last count character. There is no limit to the number of count characters, so the length of a run is limited only by the message or frame size.

HBE - High Bit Escape (single byte state change).

HBT - High bit Toggle. Change state and remain until another HBT or HBE. If HBT and HBE are not defined, an 8-bit channel must exist and the high bit will never be prefixed but will be sent along with the data, whether translated or not.

EOL - Reserved hardware-recognizable End of Line character, such as Carriage Return. Must be translated to another character so that it never appears in the data stream either with or without High Bit set. Must be the same for both sides.

A minimum length run is defined as a run of three identical characters if any of RL0 - RL2 are specified, otherwise four characters.

In addition to EOL, at least ONE special character must be specified, either TTE or TTT. All others are optional, but their use may improve encoding efficiency. HBE and/or HBT are required for transmitting data over a 7-bit channel, and should not be used otherwise because they will reduce efficiency.

Encoding/Decoding Modes:

7-BIT MODE: If either HBE or HBT is defined in the initialization data from the transmitting side, the receiver uses the channel as if it were a 7-bit channel. The high bit of all received characters is set to the current value of the High Bit Mode, and changes in that mode will be allowed when HBE or HBT is seen.

8-BIT MODE: Neither HBE nor HBT has been defined. The 8th bit (bit 7) in each character is taken literally. If the received character with high bit forced to 0 is a translated one of any kind, its actual received high bit is merged in with the character resulting from the translation.

The transmitted byte stream is encoded so that the following two modes are switched on and off in the receiver (decoder) by special characters found in the byte stream. Both modes are initialized OFF before the first byte of each frame is decoded:

TRANSLATION TABLE MODE: When off, received characters (except special characters) are taken literally. When on, received characters are translated via a 128-character lookup table provided by the neighbor side as part of initialization. As an example, that table may translate ASCII control characters, the EOL, and all special characters to something else. Translation Table Mode is toggled by TTT and switched for one character only by TTE.

HIGH BIT MODE: If High Bit Mode is on, the 8th bit is ignored in received characters and set true on all resulting characters whether translated or not. High Bit Mode is toggled by HBT and switched for one character only by HBE.

Frames:

DS/NS messages are wrapped into one or more Protocol Frames, which contain checksums and other protocol validation mechanisms. The design of the frames themselves is beyond the scope of this paper.

Frame Types:

A = Ack
C = Continuation (preceded by D or C)
D = Data frame
H = Hang-up (disconnect physical line)
I = Initialization request
J = Initialization response data
K = Initialization request data
L = Round-Trip time interval
N = Nak
P = Poll message
S = Stop

Pseudo Drivers:

The Type-66 Pseudo Drivers ID166 and DVP66 are the key to the process. Type-66 LU's are generated in pairs, exactly as HP's ID*66 and DVA66 drivers are used. The first LU is normally used for writes, and the second for reads, although there are some exceptions.

In RTE-A there is one IFT for each DVT pair, with a 20-word extension, and the key information for both LU's is kept in that IFT extension. In RTE-6/VM, one EQT is generated with a 12-word extension and the other with none. The key information for both is kept in the first EQT's extension. This is important because some DS/NS programs actually check the IFT/EQT extension length of the drivers in order to confirm that the LU's are DS/NS LU's.

As described above, most normal driver entry directives are handled by passing the request on to MUXIO. These directives are handles right in the driver:

1) Abort. Several situations are handled. In RTE-6 this is important, but in RTE-A the driver is unlikely to be entered with an abort directive because it always exits with the HOLD bit set unless it completes the request.
2) CN 22B. Set timeout. This is the only user request that the driver handles directly.
3) Continue. Illegal, treated the same as a timeout.
4) Timeout. This is handled differently depending upon the state of affairs. In one case we may be retrying the schedule of MUXQUEUE. In another MUXIO may have set completion bits and forced an immediate timeout. In a third case, an actual timeout may have occurred and a timeout status is reported by the driver.
5) Power Fail. The driver treats this the same as a timeout.

MUXQUEUE Program Design:

MUXQUEUE is a 3-page program which has a normal state of dormant, saving resources. It has no need for and no access to the formatter or the file system. It operates at a reasonably high priority (29) and performs all tasks immediately, so it spends very little time executing. Its job is to sit and wait to be scheduled by one of the following:

1) MUXQUEUE, which initially passes its own class number by way of a schedule. In some circumstances MUXIO may also ask MUXQUEUE to be its alarm clock to wake up MUXIO at a particular time.
2) MUXLINK, a management program, which may request MUXIO's class number or tell MUXQUEUE to shut down.
3) A Type-66 LU driver with a new request initiation.
4) A D-MUX driver with an unsolicited incoming message.

MUXIO Program Design:

MUXIO communicates almost exclusively through class I/O, with no need for or access to the file system. It operates at a modest priority of 50 and does all of the protocol conversion. MUXIO rarely puts itself in the timelist. While it can spend significant time executing, it still spends most of its time waiting on a class GET. It can receive class I/O messages from the following sources:

1) MUXLINK initializes MUXIO's table of LU's and protocol specifications through class writes.
2) MUXQUEUE reflects its schedule requests from MUX LU's and from Type-66 Pseudo Driver LU's.
3) MUX LU's complete I/O requests.

MUXLINK Program Design:

MUXLINK is used to control the MUXLINK system. It issues startup messages to MUXIO based upon a startup command file, and allows other manipulations of the system including shutdown. Here is a list of its commands:

MUXLINK Action Commands:

?	[keyword]		Request help for keyword
CN	lu CW [pram]		Issue control request to specified lu
DI			DIsplay local LU66 and TABL tables
ECHO	on/off		TRansfer-file echo on or off
EX			Exit MUXLINK program
HE	[keyword]		HElp, same as ?
LU66	lu66	*	Begin Defining Type-66 MUXLINK LU
MAXR	maxwords	*	Define maximum MUXIO class-read size
SD		*	Shut Down MUXIO and MUXQUEUE now
SEND	[LU/TA/MR/AL] [#/AL]	*	Send LU66, TABL, or MAXR to MUXIO
SHOW	[LU/TA/ST/AL] [#/AL]		Display MUXIO's internal tables
SS			Suspend Self (use GO to resume)
ST			Display current STatus (brief)
SU		*	Start Up MUXLINK progs & send tables
TABL	tbl	*	Begin Defining Encode Table
TR	[filename/1]		TRansfer to command file

TABLES

LU66 Table Commands (need at least LU66, MUX, and TBL):

LU66	lu66	* Begin Defining Type-66 MUXLINK LU
BAUD	baudrate	MUX port baudrate
BRG	0/1	C-MUX port baud rate gen
MUX	lu	MUX LU for pending LU66
PORT	0-7	MUX Port number
TBL	1-4	Encode TABL used with pending LU66
TOM	ticks	Reset timeout for specified MUX port
XON	on/off	Specify XON/XOFF for pending MUX

TABL Encode Table Commands:

TABL	tbl	* Begin Defining Encode Table
Ival	itran #pairs	Numeric Translation (? Numeric)
EOL	ival itran	EOL Character (default CR)
HBE	ival itran	High-Bit Escape character
HBT	ival itran	High-Bit Toggle character
RL0	ival itran	Runlength char, minimum runlength
RL1	ival itran	Runlength char, minimum runlength + 1
RL2	ival itran	Runlength char, minimum runlength + 2
RLC	ival itran	Runlength char, counted length
TTE	ival itran	Translation Table Escape character
TTT	ival itran	Translation Table Toggle character

All numeric values entered in MUXLINK commands are interpreted as octal values if they have a trailing 'B', otherwise they are assumed to be decimal in all cases.

Commands with an asterisk (*) before the description require superuser capability. This includes any commands which are capable of modifying MUXIO's operating parameters.

Performance:

Extensive testing was done between an A400 running RTE-A 5.2 with DS/1000 and an E-Series running RTE-6/VM 5.2, also with DS/1000. Both systems had C-MUXes, D-MUXes, and HDLC cards. In all tests there was no significant difference between Cand D-MUX test results, so those have been combined below.

9600 Baud: Effective rate in characters per second:

	MUX	HDLC
Uncompressible file	427	
Type-6 file	648	
Large relocatable ($BIGLB.LIB)	736	
Large text file (CONNECT Manual)	770	

19,200 Baud: Effective rate in characters per second:

	MUX	HDLC
Uncompressible file	585	725
Type-6 file	795	718
Large relocatable ($BIGLB.LIB)	1213	1823
Large text file (CONNECT Manual)	1252	1471

38,400 Baud: Effective rate in characters per second:

	MUX	HDLC
Uncompressible file	641	
Type-6 file	846	
Large relocatable ($BIGLB.LIB)	1347	
Large text file (CONNECT Manual)	1400	

230,000 Baud: Effective rate in characters per second:

	MUX	HDLC
Uncompressible file		1823
Type-6 file		1737
Large relocatable ($BIGLB.LIB)		8442
Large text file (CONNECT Manual)		7930

Summary:

The system described can handle DS/NS messages over MUX ports, providing most DS/NS services at the full speed of the MUX.

DOWNLOADING FROM THE HP-1000 TO FACTORY FLOOR MACHINES

PAPER# 1010

Bill Donze
Reliance Electric Company
6065 Parkland Boulevard
Cleveland, Ohio 44124-8020
(216) 266-7619

1. ABSTRACT

The automated machine tools of today's factory are directed by Computer Numerical Controls (CNC's) which accept ASCII instructions to produce the desired machine motion. In the past, most CNC's were equipped with punched paper tape readers to input these instructions. The instructions were generated by a remote computer connected via a modem to a terminal and a tape punch located in the programmer's office. Although this method worked, it was subject to telephone transmission problems, mechanical failures, and was very time consuming. Reliance is installing HP-1000 A-Series Systems and custom software at its plants to implement local control of the shop. This paper describes the MACRO program that downloads machine instructions from the local HP-1000 to the shop floor CNC's. Although this software is proprietary, the paper's in-depth discussion of the process will provide sufficient information for implementation.

2. BACKGROUND

The instructions for a CNC must be created by a Parts Programmer based on an engineering drawing of the part to be produced. The task of conveying these instructions to the CNC has evolved from a manual operation, through a remote computer-assisted solution, to an efficient local computer-assisted process.

In the original manual process, a Parts Programmer would interpret an engineering drawing and write the needed CNC instructions on paper. This can be likened to programming in assembly language without the aid of a computer. The instructions from the hand-written paper would then be typed into a Tape Preparation machine producing a listing and a punched paper tape. The paper tape and the listing then had to be hand-carried to the CNC on the shop floor where the paper tape would be read into the CNC's memory by a mechanical tape reader. This process could take anywhere

from 4 to 40 hours for one part! Any problems such as human error, tape punch failure, or tape reader failure could even lengthen the process. Notice also that the Parts Programmer had to walk to three different locations to complete the job and considerable storage facilities were required to maintain the listings and paper tapes for possible future use.

In the next step in the evolution, several major changes were introduced to the process in an effort to ease the programming task. The addition of a mainframe computer, the APT Processor (Automatically Programmed Tools) program and machine-dependent Post Processor programs provide valuable tools for the Parts Programmer. The APT Processor can be thought of as a compiler which accepts a high level language from the Parts Programmer to produce an intermediate meta-language. The Post Processor then converts the meta-language to CNC instructions. With this approach a part could be programmed in 30 minutes to 8 hours. However, the problems of tape storage, listing storage, and reader/punch failure are still present. Furthermore, with the addition of a 300 baud modem and telephone line transmission, some new problems have been added.

Note that in this approach there is a single host computer which supports, in addition to factory floor operations, other tasks such as payroll, work-in-process, inventory, etc. The parts programmers from many plants must compete for execution time as well as for modem access in some cases.

In the present process, the single remote computer has been replaced by a local HP-1000 A-Series System at each plant, removing the contention and modem/telephone transmission problems. The APT and Post Processors from the remote mainframe computer have been ported to the HP-1000. NCMGR (a user interface and job management program) has been added to maintain listings, punched tape images, and all other pertinent information in database-managed disc files so that the listing and tape storage problems are eliminated. Finally, the tape punch and tape reader mechanical and environmental problems have been eliminated with the addition of the electronic transmission of the tape image data. The data is moved directly from the HP-1000 computer to the CNC by the program DLOAD, a process known as *downloading*.

Part program creation is now possible in 10 minutes to 2 hours which translates to an annual savings of $80,000 to $125,000 at each Reliance plant where this system is installed. In addition, the parts programmer can now create or modify a part entirely at his or her desk; the only reason to go to the shop would be to observe the first run of a new part if it was warranted. Any problems uncovered during a test run can be corrected by the parts programmer

at the shop terminal since the same capabilities available at an office terminal are also available in the shop.

3. THE HP-1000 HOST OPERATING SYSTEM

At present, HP-1000 Systems with downloading capabilities are installed at eleven of Reliance's manufacturing plants throughout the Eastern United States. The hardware and software requirements of these systems are described in the following sections.

3.1 Hardware Components

A typical HP-1000 A900 System which provides the platform for downloading is shown in Figure-1.

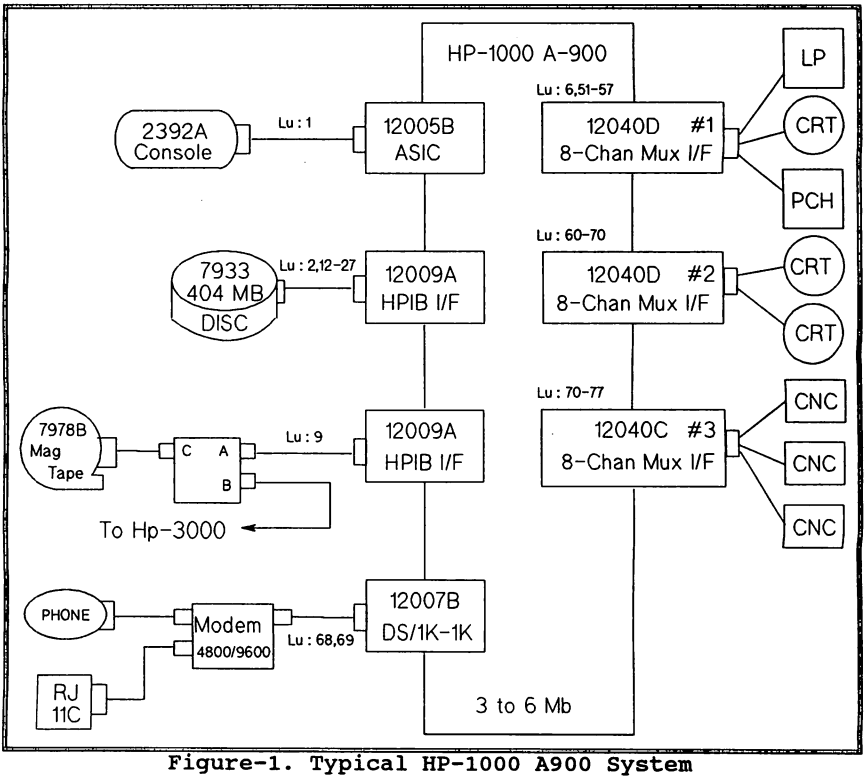


Figure-1. Typical HP-1000 A900 System

All systems are very similar in configuration. Each has the System Console on an ASIC card, the disc and mag tape on separate HPIB cards, a DS/1000-IV dial-up modem link to the

Corporate HP-1000 System, and two or more 8-channel multiplexer cards for the printer, punch, office terminals, shop terminals, shop printers, and machine CNC's. Disc capacities range from 404 Mb to 1212 Mb and additional disc drives are interfaced using separate HPIB cards for improved performance. A 7978B mag tape is used for faster system backup time and is shared with the HP-3000 to offset the cost. The DS/1000-IV modem link provides a cost effective means of system maintenance and software upgrades from the Corporate A900 System in Cleveland, Ohio.

The office terminals, system printer, backup tape punch for the Parts Programmers, and the shop terminals for the machine operators are interfaced using one or more Rev-D 8-channel multiplexers. The D revision is used to reduce the table space in the system generation and to take advantage of the more reliable operation. One or more Rev-C 8-channel multiplexers are used to interface the machine-tool CNC's. The Rev-D mux would be more desirable for the CNC's, but the DLOAD program has not yet been upgraded for D-mux support. The main obstacle is the difference in Xon/Xoff handling between the Rev-C and Rev-D multiplexers. The Rev-C mux has uni-directional Xon/Xoff protocol with the ability to force an Xon state while the Rev-D mux has only bi-directional Xon/Xoff control. Each of the 16 different supported downloading protocols will have to be tested on-site with the Rev-D mux to determine the effect of the Xon character sent to the CNC when Xon/Xoff is enabled.

3.2 Software Components

The typical HP-1000 A900 System for downloading support includes the standard HP software products: RTE-A, VC+, IMAGE/1000-II, and DS/1000-IV. The only non-HP module included in the system generation is the named common block D_RVT, located in System Common, which is explained in more detail below.

3.3 MUX Port Configuration

Each machine tool CNC controlled by DLOAD is interfaced to the HP-1000 via a port on the 12040C 8-channel multiplexer. The Mux is included in the system generation just as if it were being used for interactive terminals as shown in Figure-2.

```
* 12040C: 8-Channel Mux for Shop Machines.              #2
* ======
*
IFT,/SOFTWARE/A92077/%IDM00,SC:33B,TX:20
*
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:60,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:61,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:62,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:63,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:64,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:65,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:66,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
DVT,/SOFTWARE/A92077/%DD*00,M26XX,LU:67,TX:57,DP:1:20004B,-
    DP:5:PR:OM:TX
```

Figure-2. System Generation for CNC Machine Ports

Each CNC machine tool port is further configured at bootup time by a command file similar to the one shown in Figure-3. This sets the port attributes which do not change during a download such as the baud rate, stop bits, parity, etc.

```
* /CmdFiles/Mux2C_On.Cmd    <910219.1523>
*
* Enable Mux #2: Shop Machines
*
* BULLRD Bracket Cell  7-bits, No Modem, Brg0, 1 Stop,
*                      Even Parity, No Enq/Ack, 2400 Baud
*                      No Dc1 Trigger
CN 60 30B 043510B
CN 60 23B
CN 60 45B 0
CN 60 27B 0
          :
          :
          :
* CSTEP DRILL Cell     7-bits, No Modem, Brg1, 1 Stop,
*                      Even Parity, No Enq/Ack, 2400 Baud
*                      No Dc1 Trigger
CN 67 30B 053517B
CN 67 23B
CN 67 45B 0
CN 67 27B 0
*
Echo ` Mux-2 Ready. `
Return
```

Figure-3. CNC Port Initialization at Bootup

3.4 HP-1000 To CNC Wiring

The distance from the HP-1000 to the shop is always greater than the RS-232C limitation of 50 feet. In some cases, small line powered short-haul modems are used. These devices are available in male or female 25-pin configurations with a switch to swap pins 2 and 3, but they require +12V on pin 4,5,6 or 20. The short-hauls are connected by a shielded cable with two twisted pairs for transmit and receive. In other cases, 8,16 or 20 channel multiplexers are used which require one cable with two twisted pairs from the HP-1000 to the shop for each multiplexer pair. Since the distance from the multiplexer to the CNC is also limited, some shop layouts do not lend themselves to this approach.

Most modern CNC's have an RS-232C interface option which is plug-compatible with the HP-1000 and no additional hardware is needed. However, older CNC's may not have this interface option or the cost of retro-fitting the CNC with the option is prohibitive. For these CNC's, a small shop floor computer such as the NUMERITRONIX 1501 is added between the HP-1000 and the CNC. This device has local edit, storage capabilities, and a full keyboard with display. It functions as a solid-state tape reader, but is rather expensive if the edit and storage features are not needed. When only the RS-232C interface is required, the much simpler and less expensive RYBETT CAMSTORE unit is used.

4. THE DOWNLOAD PROCESS

A typical download operation is illustrated in Figure-4 using sequence numbers adjacent to the arrows to show the sequential steps of a download.

The download process begins when the machine operator enters a command on his CRT (1) which requests a download of a certain part to a specific machine in his cell via the NCMGR program. NCMGR allocates a class# (2) for the completion response from DLOAD and then writes a start download request on DLOAD's class# (3). DLOAD retrieves the start request using a class get (4) and sends a 'Download Initiated' message to the operator's CRT (5). A similar message is written to the System Console (6) if DLOAD's logging feature has been enabled. Next, DLOAD reads the tape image file specified in the start request and sends each record to the operator's CRT (7) and to the CNC (8). Steps (7) and (8) are repeated until all of the tape data has been transmitted at which point a 'Download Completed' message is sent to the System Console (9), again if logging has been enabled. Finally, DLOAD finishes the download by writing the download completion status (10) to NCMGR's class#. All the time DLOAD has been performing the download, NCMGR has been waiting for

a completion response from DLOAD via NCMGR's class# which now completes (11). NCMGR finishes the download process by reporting the success/failure result to the operator (12).

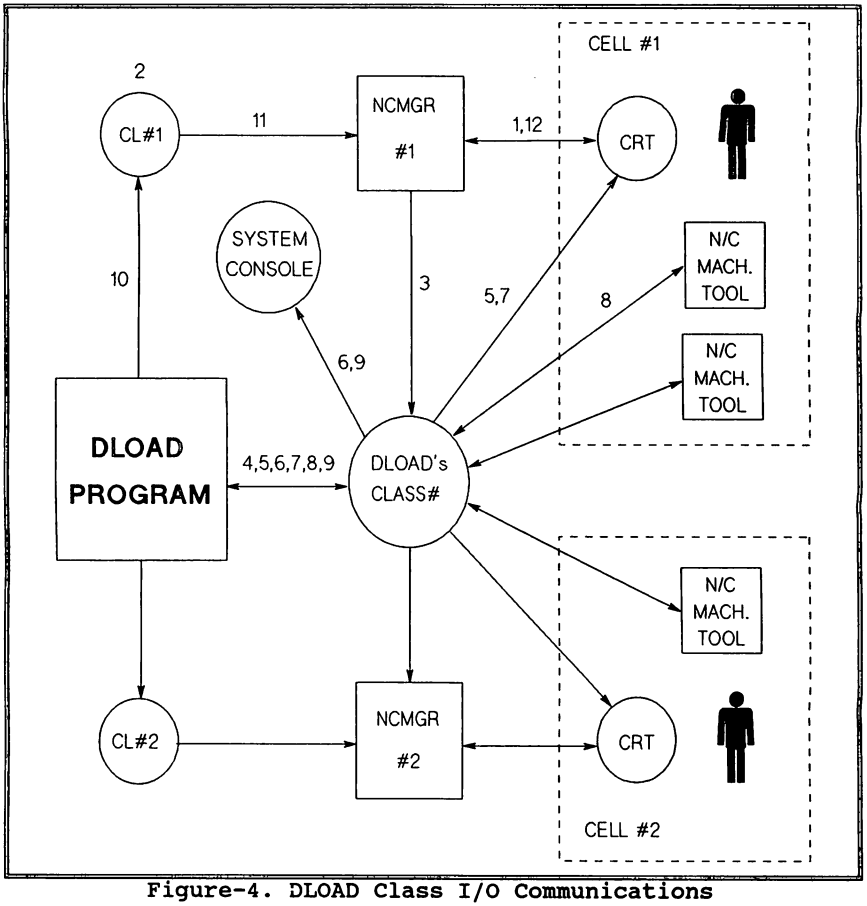


Figure-4. DLOAD Class I/O Communications

A keypoint in the above process is that DLOAD receives all download requests and performs all of its I/O operations using class I/O via a single class#. This technique provides I/O without wait and thereby enables DLOAD to perform asynchronous downloads to multiple CNC's at the same time.

5. THE DLOAD PROGRAM

5.1 Features

The DLOAD program is configured at assembly time to handle up to 5 concurrent downloads and an additional 15 pending downloads in a wait queue. These two limits can be adjusted by parameters in the source code.

DLOAD is designed to handle only Tape Image files which are standard ASCII source files of type-3 or type-4. The tape image records are not modified in any way and are merely transmitted to the CNC with one exception. If the first record in the tape image file begins with "PARTNO", then the record is ignored and is not transmitted to the CNC. The PARTNO record is required by NCMGR, but would be rejected by the CNC.

During the download process, DLOAD will echo each Tape Image file record to the machine operator's terminal as a visual indication of progress. This echo is done such that each line displayed overlays the previous one, i.e., the display does not scroll.

The DLOAD program is a non-CDS program which is loaded with access to System Labelled Common (LC) and as a System Utility (SU) to prevent cloning. It is initiated at system bootup in the WELCOME1.CMD file by the following command:

XQ,DLOAD [*LogLu, ErrLu*, DE, *DebugLu*]

LogLu specifies the device where download initiation, completion, failure and abort messages will be printed. *LogLu* may be in the range of 1 to 100. If not specified, *LogLu* defaults to the System Console. Logging can be disabled by specifying *LogLu* as -1.

ErrLu specifies the device where error messages will be printed. *ErrLu* may be in the range of 1 to 100. The default is *LogLu* or the System Console if *LogLu* was specified as -1. These messages can not be suppressed.

If the characters DE are specified, then the Debug Trace feature will be enabled. This option will display the I/O status, buffer length, lu# and mode variables returned when the class Get in DLOAD's Control Section completes. If a start download request was received, the contents of the request are displayed. If a CNC read request has completed, the input data buffer is displayed in octal bytes and ASCII. For any CNC I/O completion, the CNC Lu# and the continuation address are displayed.

DebugLu specifies the device where Debug and Trace messages will be printed. *DebugLu* may be in the range of 1 to 100. The default is *LogLu* or the System Console if *LogLu* was specified as -1.

5.2 Control Structures

To implement the asynchronous, multiple CNC downloads described previously, DLOAD utilizes three control structures. An external table in System Common is used to make its class# available to application programs which desire to initiate a download and two internal tables are used to manage all active downloads and all queued download requests.

5.2.1 Named System Common

Figure-5 shows the System Common table D_RVT used by DLOAD to hold the global class# (word-2) that the user interface program NCMGR uses to send download requests to DLOAD. This table must be included in the system generation so that it is global to the system.

```
MACRO,L
      HED * RELIANCE VARIABLE TABLE * <910415.1509>
      NAM D_RVT,30 Reliance Variable Tbl  [MAC] *
*
      ENT D_RVT
*
D_RVT NOP       =  1: Spooler Class# (SPOLA).
      NOP       =  2: DLOAD'S Class# (NCMGR).
      NOP       =  3: ULOAD'S Class# (NCMGR).
      BYT 0,0   =  4: Upper: Formats (1=A,2=E,3=AE)
*                     Lower: System Punch Lu#.
      DEC 56    =  5: Lu#: System Plotter.
      NOP       =  6: Ds/1K-3K Class# (D3Mst).
      NOP       =  7: Ds/1K-3K Resource# (D3Mst).
      NOP       =  8: NCACS's Class#.
      NOP       =  9:
      NOP       = 10:
*
      END
```

Figure-5. Named System Common Block D_RVT

5.2.2 The Active Table

This table contains an entry for each active download. The number of downloads that can be active at any one time is limited only by System Available Memory (SAM) and the number of entries in the table which is set by an assembly time parameter. The first two words of the Active Table contain the negative number of entries in the table and the length of each entry in words. The rest of the table consists of

repeated entries, each consisting of 182 words as shown in Table-1.

WORD#	CONTENTS	POINTER
1	CNC System Lu#	CURAD
2	User CRT System Lu#	CRTLU
3	Requestor's Class#	CLAS2
4	Control Flag Bits	CNTRL
5-36	Tape File Descriptor (32)	FNAME
37	Continuation Address	PHASE
38	Current Record#	RECNO
39-182	Fmp DCB (144)	DCBAD

Table-1. Active Table Entry Format

A new entry in the Active Table is created when a start download request is received, and removed when the download completes.

The CNC System Lu# defines the machine CNC to be downloaded and is also the key to finding the table entry. The User CRT System Lu# defines the terminal from which this download was initiated; it is used by DLOAD to send status and error messages to the user. The Requestor's Class# is allocated by the user interface program NCMGR and is used by DLOAD to send the download completion status back to NCMGR. The Tape File Descriptor provides the full filedescriptor of the disc file containing the tape image data that is to be downloaded. The Continuation Address is set initially to the starting address of the specified protocol processor and thereafter maintains the location where execution is to resume when the current Class I/O operation completes. The Current Record# field keeps track of the number of records that have been downloaded. The last 144 words of the entry comprise an FMP Data Control Block (DCB) for reading the tape image file. And finally, the Control Flag Bits are used to control various internal conditions as shown in Table-2.

BIT#	USAGE
0	PARTNO Skip Flag
1-12	unassigned
13	EOF "%" Record Detected
14	Abort-In-Progress Flag
15	Xon/Xoff Flag

Table-2. Control Flag Meanings

The first record in the tape image file is expected to begin with the word PARTNO and the PARTNO Skip Flag enables testing for this so that the record is not sent to the CNC. After the first record, the flag is set to disable further testing.

The "%" character is found in some tape image files and represents a rewind stop code to the CNC. There may be a "%" preceding the tape data, terminating the tape data, or both.

DLOAD is only concerned with the terminating "%" and only for certain protocols. The EOF "%" Flag is set when a terminating "%" is detected and depending on the protocol, the record may be ignored, sent to the CNC, or indicate an end-of-file condition.

The Abort-In-Progress Flag is set when a download is aborted and is used to ignore I/O errors that may result.

The Xon/Xoff Flag is set whenever a protocol processor enables the Xon/Xoff feature of the mux port. If a download terminates abnormally, this flag is used to force the mux port to a known initial condition of Xoff in preparation for the next download.

5.2.3 The Wait Queue

Requests to start a new download which are received when the Active Table is full are placed in the Wait Queue to be activated as soon as an entry in the Active Table becomes available. The number of entries in the Wait Queue is also set by an assembly time parameter. The first two words of the Wait Queue contain the negative number of entries in the table and the length of each entry in words. The rest of the table consists of repeated entries, each consisting of 36 words as shown in Table-3.

WORD#	CONTENTS
1	CNC System Lu#
2	User CRT System Lu#
3	Requestor's Class#
4	Protocol Type Code
5-36	Tape File Descriptor (32)

Table-3. Wait Queue Entry Format

Notice that the CNC System Lu# is stored in word-1 of both the Wait Queue and the Active Table entries. In conjunction with the number of entries and entry length, a single search routine can be used to search either table. The search routine is called at the beginning of each of the major processing sections (START, STOP and IOCOM) to set up the eight pointers shown in Table-1. These pointers are then used throughout all processors to reference the variables for the specific download being handled.

The contents of a Wait Queue entry is simply a copy of all of the information from the start download request. The entry is created by a start download request from NCMGR when the Active Table is full and is removed when a download completes, moving the Wait Queue entry to the Active Table.

5.3 Program Overview

A simplified state diagram of DLOAD is shown in Figure-6. With the exception of the ERRx state, each state is explained in more detail in the following sections.

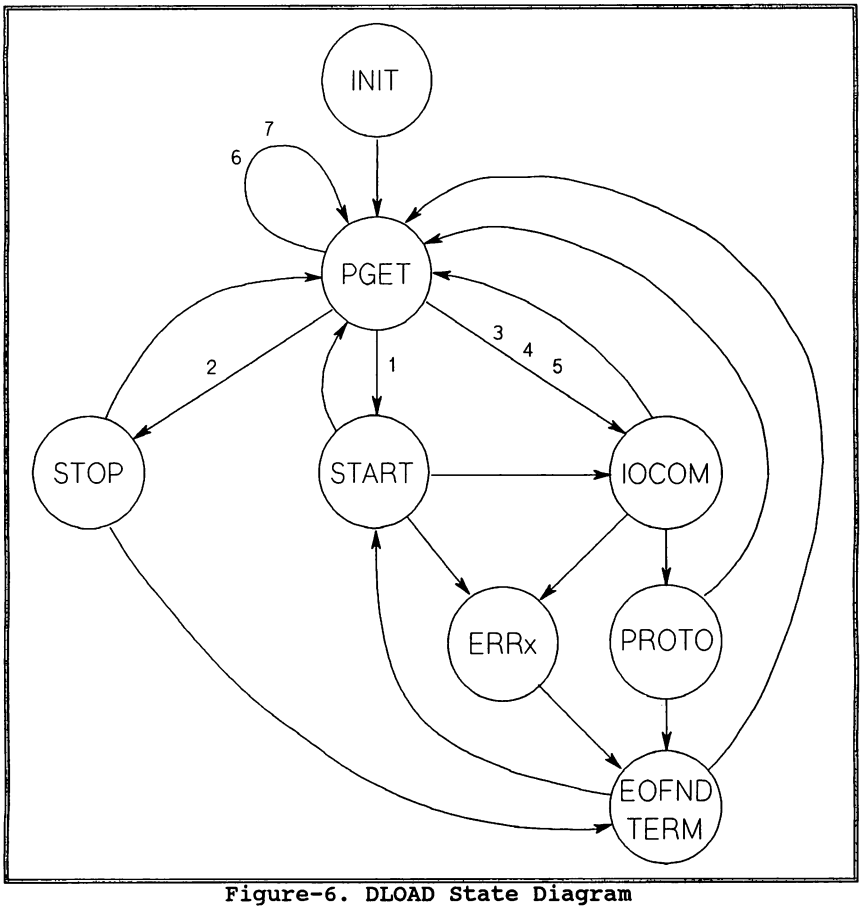


Figure-6. DLOAD State Diagram

The state names are also labels in the various sections and are referenced in the text and flowcharts of each section to explain how DLOAD moves from state to state. Notice that once DLOAD is invoked, it never terminates; it is either in a Class Get suspension at PGET or executing one of the other states.

5.4 Initialization Section

This section of DLOAD is executed only once when DLOAD is first started and performs several initialization tasks.

First, the routine RMPAR is used to retrieve the runstring arguments which are range-checked or defaulted and then used to setup device lu#'s for logging, error reporting, and debug displays.

Next, the routine DTACH is called to detach from any user session in which DLOAD may have been invoked. When debugging a new CNC protocol, it is often necessary to abort, modify, and restart DLOAD several times and the DTACH call insures the proper system environment without having to re-boot the system.

The next task is to setup the global class# in the System Common table D_RVT. If word-2 of D_RVT is non-zero, it means that DLOAD has been re-started without a system re-boot, so a call to CLRQ is made to flush and deallocate the old class# from D_RVT. Another call to CLRQ allocates a fresh class# for DLOAD which is then saved in D_RVT.

Word-1 of each entry in the two internal tables, the Active Table and the Wait Queue, is now set to zero to indicate that all of the entries are empty.

The last task of the initialization section is to display the message:

```
/DLOAD: Rev-2.30 Ready on 4-23-91 16:30:02 PM
```

on the Log Device or the System Console.

5.5 Control Section

The Control Section begins at label PGET with a Class Get on DLOAD's class# as shown in Figure-7. When DLOAD is not executing, it is suspended on this Class Get waiting for an entry to be placed on the class queue for its class#. Entries are placed on this queue by NCMGR (start or stop download request) or by the completion of a previous class I/O by DLOAD (CNC read, CNC write, CNC control, user CRT write, or log/error write).

The purpose of the Control Section is to determine which one of the above 7 request types has been received and to branch to the section meant to handle that request.

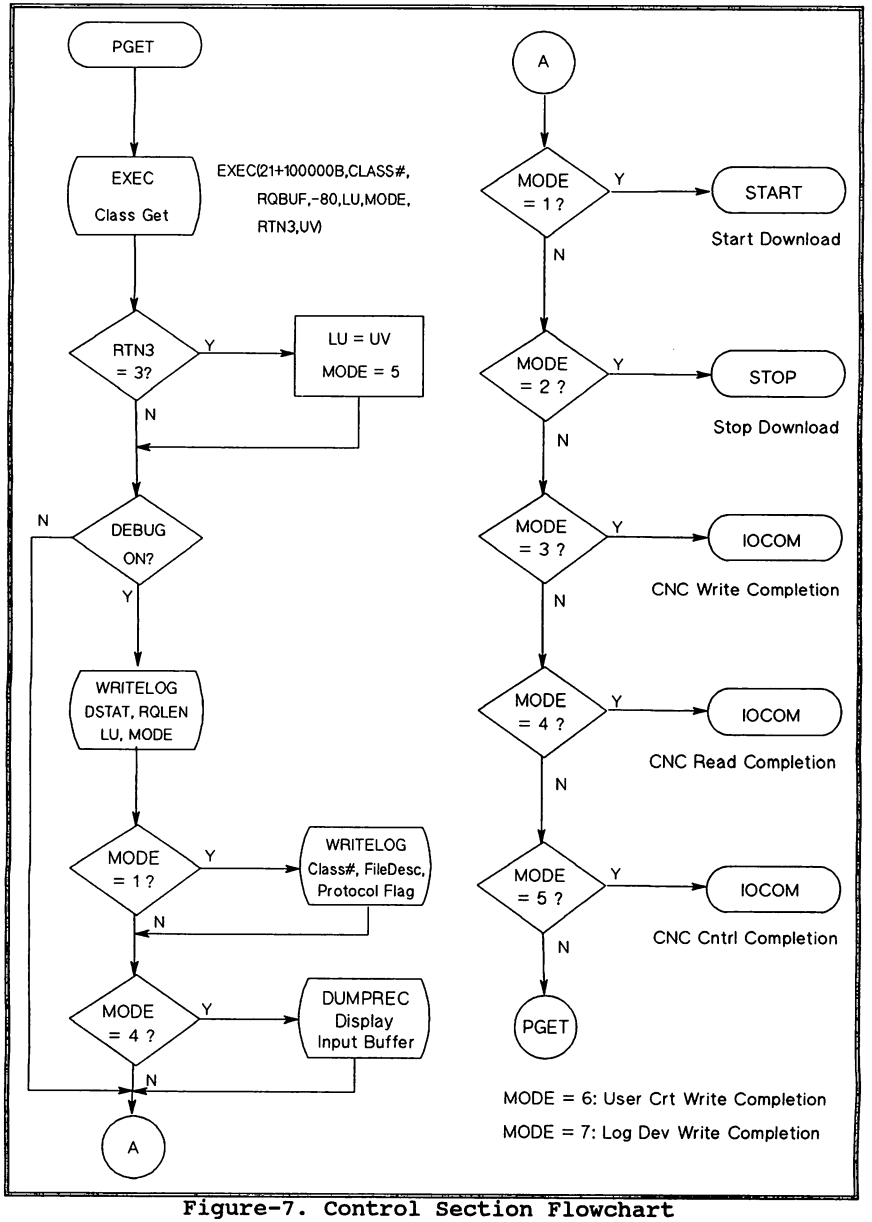


Figure-7. Control Section Flowchart

```
Pget     Jsb Exec       Get w/suspend.
          Def *+9
          Def Rc21n     = No Abort.
          Def DClasGet  = Our Class#.
XRqBuf    Def RqBuf     = Buffer.
          Def Mn80      = Buffer Len.
          Def Lu        = Rtn1.
          Def Mode      = Rtn2.
          Def Rtn3      = Rtn3.
          Def Uv        = Uv.
         Hlt 77B        Fatal Error!
         Sta Dstat      Save Dvr Stats
         Stb RqLen      Save + #chars.
```

Figure-8. DLOAD's Class Get

Figure-8 shows the source code of the Class Get call from which the returned variables LU, MODE, RTN3 and UV provide the means of identifying the request. RTN3 is set by RTE and identifies the original class call as a read or write/read (1), a write (2), or a control (3) call. The LU, MODE, and UV variables return the PRAM3, PRAM4 and UV arguments from the original class read, write or write/read call shown below.

```
CALL EXEC(ECODE,CNTWD,BUFR,BUFLN,PRAM3,PRAM4,CLASS[,UV])
```

For each of these requests, PRAM3 is set to the Lu# and PRAM4 is set to a number which identifies the type of operation as shown in Table-4. The UV argument is not used in this case.

Operation	Ecode	Buffer Contents	Bufr Len	PRAM3	PRAM4
Start Download	20	1: User CRT Lu# 2: Class# 3: Protocol Flag 4-35: FileDesc	35	CNC Lu#	1
Stop Download	20	1: User CRT Lu#	1	CNC Lu#	2
CNC Write	18	1-n: Data	n	CNC Lu#	3
CNC Read	17	1-n: Data	n	CNC Lu#	4
User CRT Write	18	1-n: Message	n	CRT Lu#	6
Log/Err Write	18	1-n: Message	n	CRT Lu#	7

Table-4. DLOAD Class I/O Request Formats

This provides enough information to identify all requests except the control request which has a format different from all of the other class calls. The format of a class control request is shown below in which DLOAD passes the CNC Lu# via the UV argument.

```
CALL EXEC(19,CNTWD,PRAM1,CLASS[,PRAM2,PRAM3,PRAM4,UV])
```

When DLOAD receives a class control completion, RTN3 will have a value of 3. DLOAD detects this special case and stores UV into LU and sets MODE to 5. At this point, LU and MODE have been set correctly for all received requests.

If the Debug/Trace feature is enabled, DLOAD will display DSTAT, RQLEN, LU, and MODE from the Class Get. DSTAT and RQLEN were set from the A and B registers when the Class Get completed and contain the Driver Status and the length of the data received, respectively. If this is a Start Download Request, then the contents of NCMGR's request are displayed; if this is a Class Read completion, then the input buffer is displayed.

Finally, the Control Section branches to the START Section (MODE=1), the STOP Section (MODE=2) or the IOCOM Section (MODE=3,4,5). For User CRT, Log or Error write completions (MODE=6,7), no processing is necessary so control goes back to PGET.

5.6 Start Download Section

This section is entered at the label START (Figure-9) when a Start Download Request (MODE=1) is received from NCMGR. The CNC Lu# (LU) from the request is used to search the Active Table for an entry with a matching lu#. If an entry is found, then this is a restart and the message "Download Aborted..." is logged, the previous tape file is closed, and at label START1, the new request is moved to the Active Table overlaying the previous entry.

If an entry was not found in the Active Table for LU, then DLOAD checks if the Active Table is full. If it is not full, then the request is moved to the Active Table entry at START1.

If the Active Table is full, then DLOAD searches the Wait Queue for LU. If an entry is found in the Wait Queue, then a waiting download is being restarted; the new request is moved to the Wait Queue entry overlaying the previous entry, the message "Busy, Request Queued" is sent to the user, and program control returns to PGET and awaits the next class completion.

If the request LU is not found in the Wait Queue, then DLOAD checks if the Wait Queue is full. If it is not full, the new request is moved to the Wait Queue entry, the busy message is sent to the user, and control returns to PGET. If the Wait Queue is full, then the ERR1 processor sets the completion status to 1 for a busy condition and goes to the Termination Section (TERMS) where the status is sent back to NCMGR.

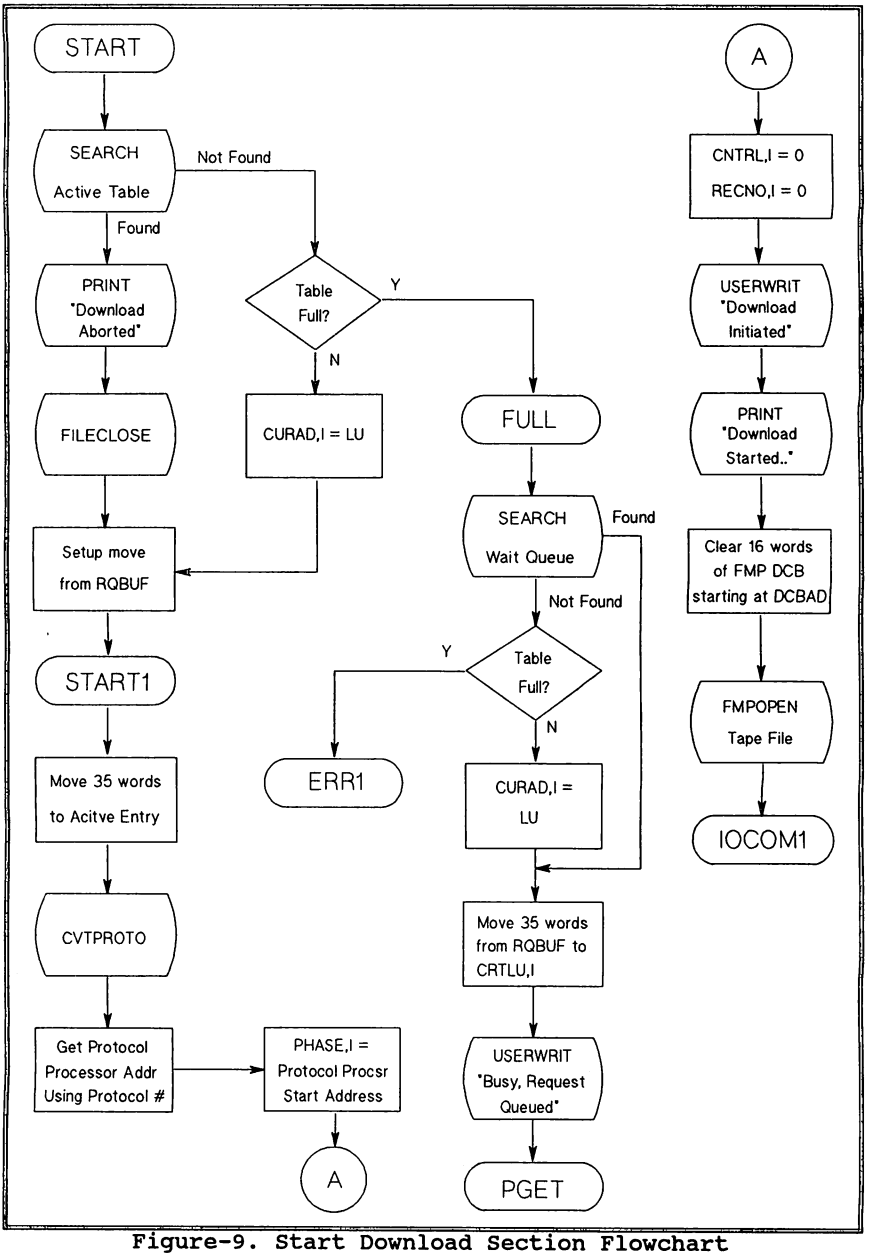


Figure-9. Start Download Section Flowchart

The processing of a new download or a restarted download request continues at label START1 where the request is moved to an Active Table entry. The desired protocol is specified in the request as a two digit ASCII string, so the routine CVTPROTO is called to convert the string to an integer. This integer number is then used as an index into the Protocol Processor Table shown in Figure-10.

```
XProTbl  Def *+1           Protocol Start Addresses.
         Def P00.00        00: No Protocol.
         Def P01.00        01: AB7360 Protocol.
         Def P02.00        02: GN & FANUC Protocol.
         Def P03.00        03: NUMERITRONICS "L" Protocol.
         Def P04.00        04: CINCINNATI 850 Protocol.
         Def P05.00        05: GE 1050 Protocol.
         Def P06.00        06: NUMERITRONICS NB/NC-ASCII
         Def P07.00        07: NUMERITRONICS "LE.
         Def P08.00        08: K&T CNC Series D Cntl-RDC3.
         Def P09.00        09: Cimpoint Fdl-500 Btr I/F.
         Def P10.00        10: Dgv Rs-232 Btr I/F.
         Def P11.00        11: Okuma OSP-5000L-G Proto.
         Def P12.00        12: Cincy Grinder Protocol.
         Def P13.00        13: NUMERITRONICS NB/NC-EIA
         Def P14.00        14: Rybett Camstore 2 Proto.
         Def P15.00        15: G&L 8000B Proto. "D" Mux
MaxPro   Abs XProTbl-*+1   = (-) Max Proto #.
```

Figure-10. Protocol Processor Table

The starting address of the selected protocol processor is extracted from the table and stored in the Active Table entry at word-37 (PHASE,I). Next, the entry's Control Flags in word-4 (CNTRL,I) and the Current Record# in word-38 (RECNO,I) are cleared. A download initiated message is now sent to the user and to the log device. Finally, the first 16 words of the FMP Data Control Block (DCBAD,I) in the entry are cleared, FMPOPEN is called to open the Tape Image file, and program control passes to the I/O Completion Section at label IOCOM1 to initiate the selected protocol processor.

5.7 Stop Download Section

This section is entered at the label STOP (Figure-11) when a Stop Download Request (MODE=2) is received from NCMGR. The CNC Lu# (LU) from the request is used to search the Active Table for an entry with a matching lu#. If an entry is not found in the Active Table, then the Wait Queue is searched in the same manner. If a Wait Queue entry is not found, then the download to be stopped has already terminated, no action is needed and the program returns to PGET in the Control Section. If a Wait Queue entry is found, then the waiting download request is aborted by setting word-1 of the entry to zero to make the entry available. No messages are displayed in this case since the download had not yet started.

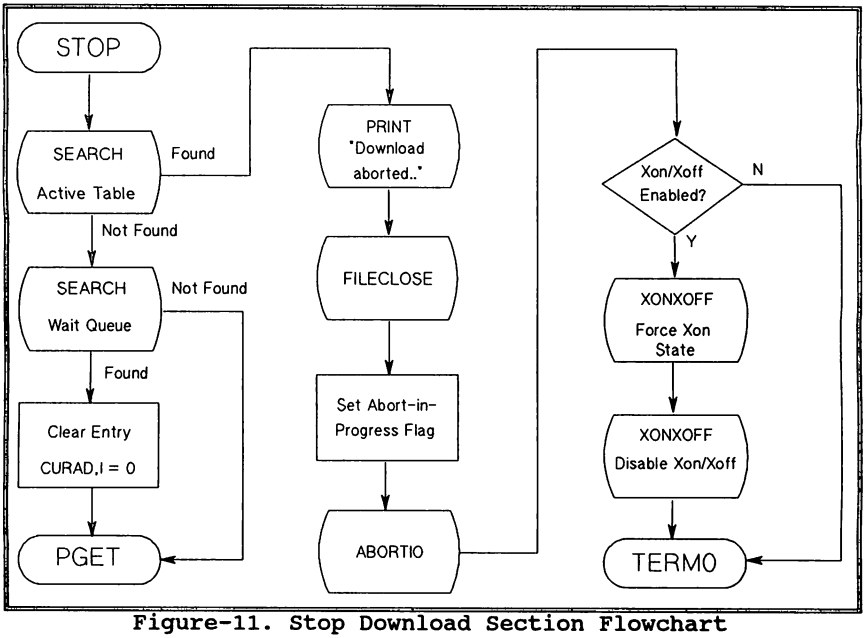


Figure-11. Stop Download Section Flowchart

If an entry was found in the Active Table, then an active download is to be aborted. The message "Download Aborted..." is printed on the log device and the tape file is closed. Next, the Abort-In-Progress flag is set in the Active Entry and the routine ABORTIO is called to cleanup all the pending I/O for this Lu. If Xon/Xoff had been enabled for this download, then the routine XONXOFF is called to force an Xon state to clear the Mux buffers and then a second call disables Xon/Xoff for the CNC's port. Finally, the program proceeds to the Termination Section (TERM0) to release the Active Table entry for this download.

5.8 I/O Completion Section

When the Class Get in the Control Section receives a CNC I/O completion (MODE=3,4 or 5), execution is directed to this section at the label IOCOM to continue an on-going download. Also, the Start Download Section will come here to initiate a new download by entering at the label IOCOM1.

As illustrated in Figure-12, the first task is to search the Active Table for an entry matching the current CNC Lu# (LU). If an Active Table entry is not found, it means that this is an I/O completion for a download which has been aborted so

it can be ignored by returning to the Control Section (PGET).

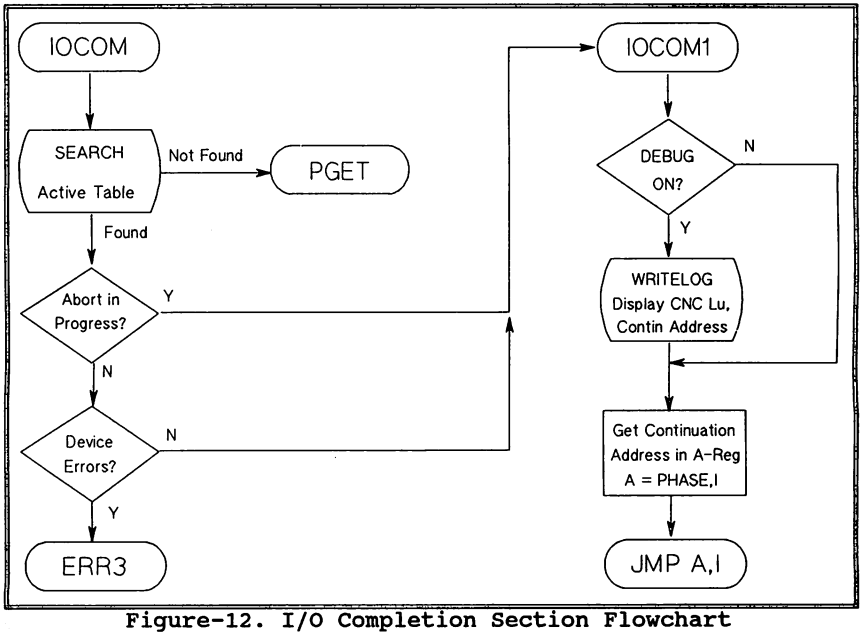


Figure-12. I/O Completion Section Flowchart

If an Active Table entry is found, then the Control Flags are checked to see if this download is being aborted. When a download abort is initiated, there may be one or more class I/O operations to the CNC still pending. When these pending I/O requests eventually complete, they may have device errors due to the abort. The Abort-In-Progress test will bypass device error checking so that the abort operation is guaranteed to finish successfully. During a normal download, the Abort-In-Progress flag will be off and any device errors will produce an error message and stop the download.

A successful CNC I/O completion continues or a new download start request enters at label IOCOM1. If the Debug/Trace feature is enabled, the CNC Lu# (CURAD,I) and the continuation address (PHASE,I) from the Active Table entry are displayed.

The final task of this section is to branch to the continuation address specified in the Active Table entry which will transfer program control to either the beginning or somewhere in the middle of the selected protocol processor. The continuation address is extracted from the Active Table entry and put into the A-Register by the

instruction LDA PHASE,I. The branch is achieved by the instruction JMP A,I which transfers control to the address in the A-Register. This operation and the way the Protocol Processor Section stores the continuation address in the Active Table entry (see below) are the reasons DLOAD is written in MACRO assembly language.

5.9 Protocol Processor Section

This section is different than the other sections in that it consists of many separate routines which can be divided into two groups. The first group consists of 16 routines which implement the 16 currently supported protocols. The second group is comprised of the support subroutines which the protocol routines reference. The subroutines are explained first to establish an understanding of the building blocks for the protocol routines.

5.9.1 Subroutines

These subroutines can be separated into the categories of CNC I/O and Support. The CNC I/O subroutine functions and names are shown in Table-5.

Subroutine Name	Subroutine Function
CncRead	Read from CNC (EXEC 17)
CnCWrite	Write to CNC (EXEC 18)
XonXoff	Enable/Disable Xon/Xoff (EXEC 19)
Read1	Read 1 character from CNC
ReadAck	Read 1 character; test for ACK
ReadDc1	Read 1 character; test for DC1
SendAck	Send ACK character to CNC
SendDc2	Send DC2 character to CNC
SendDc3	Send DC3 character to CNC
SendDle	Send DLE character to CNC
SendEot	Send EOT character to CNC
SendRec	Send data record to CNC
SendSrt	Send "S" character to CNC

Table-5. CNC I/O Subroutines

The first three subroutines (CNCREAD, CNCWRITE and XONXOFF) in the table are the only routines which issue class read, write or control calls to the CNC. They are also unique in that they are called by the standard JSB instruction, but they never return to the caller via the standard JMP <sub>,I instruction. Instead, these routines get the return address from the subroutine entry point and save it in Active Table entry in word-37 (PHASE,I) thus providing the continuation address which is used by the I/O Completion Section to resume the download when this class request completes. To illustrate, the code for the CNCWRITE routine is shown in Figure-13.

```
* SUBROUTINE TO WRITE A RECORD TO THE CNC.
*
* Call:   (P-3) Cle/Cce  Std Cr-Lf/NO Cr-Lf
*         (P-2) Lda <+ # chars to write>
*         (P-1) Ldb <Buffer Address>
*         (P)   Jsb CncWrite
*         (P+1) **** DOES NOT RETURN ****
*
* This routine saves its return address in PHASE,
* initiates the desired operation, and then goes
* to PGET to wait for completion. When the request
* completes, control will be directed to (P+1)
* using PHASE.
*
CncWrite Nop
         Stb CncWrite0    Save buffer address.
         Clb,Sez          Suppress Cr/Lf?
         Ldb M2100        -Yes: Get Suppress bits.
         Stb Cnc_Cw1+1    Set Control Bits.
         Cma,Ina          Make # chars negative
         Sta WriteLen       and save.
         Lda CncWrite     Get return addr and
         Sta Phase,I        save in Tbl entry.
         Lda Lu           Get Cnc's Lu, set
         Ior Bit15          No Sst Map bit,
         Sta Cnc_Cw1        and save.
         Jsb Xluex        Class Write to CNC.
          Def *+8
          Def Rc18n       = Class Write, No-Abort.
          Def Cnc_Cw1     = A(Lu/Control Wd).
CncWrite0 Def *           = A(Data Buffer).
          Def WriteLen    = A(Buffer Length).
          Def Lu
          Def .3          = Mode (Cnc Write).
          Def DClas       = A(Our Class#).
         Jmp Err2         -Rtn: Class I/O Error!
         Jmp Pget         -Ok: Go wait.
```

Figure-13. CNCWRITE Subroutine

The balance of the subroutines in this category are short, simple subroutines which read or write a single character using the CNCREAD and CNCWRITE routines. An example is the SENDEOT subroutine shown in Figure-14.

```
* SUBROUTINE TO SEND AN 'EOT' TO THE CNC.
*
* Call:   (P-1) Cle/Cce    Do/Don't append Cr/Lf.
*         (P)   Jsb SendEot
*         (P+1) -Return-
*
SendEot  Nop
         Cla,Ina          Set for 1 char.
         Ldb XEot         Get bufr addr of char.
         Jsb CncWrite     Send EOT char to Cnc.
         Jmp SendEot,I    -Return-
*
XEot     Def *+1
         Byt 4,0
```

Figure-14. SENDEOT Subroutine

The subroutines in the support category are shown below in Table-6.

Subroutine Name	Subroutine Function
EofTest	Test if EOF "%" Flag Set
FileRead	Read File Record/Echo to User
PctTest	Test if Record is "%" Record
Pct2Test	Test if Record is "%" or "%%"
Sleep	Time Suspend for 'n' seconds

Table-6 Support Subroutines

These routines are rather straight-forward except for the PCTTEST and EOFTEST routines. Because of the re-entrant nature of the protocol processors, two routines are necessary to check for and act upon the "%" record. The first routine PCTTEST (or PCT2TEST) checks the current tape file record for the "%" record and if found, sets the EOF "%" Flag in the Active Table entry. Later, the EOFTEST routine tests the flag since the "%" record is no longer in the record buffer.

5.9.2 Protocol Processors

Since there is no standard in the industry for CNC to host computer communications, a special handler usually must be implemented for each CNC vendor. Sometimes even different models from the same vendor require separate protocol processors. Protocol Processor #0 is shown in Figure-15 and illustrates the simplest protocol.

```
*     ********************************************
*     * NO PROTOCOL PROCESSOR                     *
*     ********************************************
*
P00.00   Jsb FileRead      Read Rec/Echo to user.
          Jmp EoFnd        -Rtn: Eof detected.
         Cle               Set for Std Cr-Lf.
         Jsb SendRec       Write Rec to Cnc.
         Jmp P00.00        -Go get next record.
```

Figure-15. Protocol #0 Processor

This processor reads the next record from the Tape Image file and echoes it to the operator's CRT using the FILEREAD routine. It then sends the data record to the CNC using the SENDREC routine. This read/write process continues until the FILEREAD routine detects a physical end of file condition at which time program control transfers to the Termination Section (EOFND) to send the completion status back to NCMGR and to release the Active Table entry. The SENDREC routine sends the data to the CNC using the CNCWRITE routine which does not return. After CNCWRITE has issued the class write call, it goes to PGET and DLOAD suspends until the class write completes. When the write completion occurs, the

Control Section branches to IOCOM which in turn branches into the SENDREC routine using the continuation address (PHASE,I) from the Active Table entry.

All protocol processors have Protocol #0 as their core and differ only in what is necessary before and after the data records have been transmitted. This is illustrated by Protocol #8 which is shown in Figure-16.

Before the data is transmitted, the READDC1 routine is used to read characters from the CNC until a DC1 is received. Then the XONXOFF routine is called to enable Xon/Xoff pacing by the MUX port. Next, the tape image records are sent to the CNC by SENDREC until a physical end of file or a terminating "%" record is detected.

```
*     ******************************************
*     * PROTOCOL #8:  KT-CNC-Series D With RDC-3 *
*     ******************************************
*
P08.00   Jsb ReadDc1      Go read DC1 char & verify.
          Jmp P08.00      -No: Go read again.
         Clb,Inb          Set for Enable.
         Jsb XonXoff      Go Enable Xon/Xoff.
*
P08.01   Jsb FileRead     Read Rec/Echo to user.
          Jmp P08.02      -Rtn: Eof detected.
         Jsb PctTest      Test for Eof "%" Record
         Cle              Set for Std Cr-Lf.
         Jsb SendRec      Write Rec to Cnc.
         Jsb EofTest      Was Eof "%" found?
          Jmp P08.01      -No: Go get next record.
*
P08.02   Cce              Set for NO Cr-Lf.
         Jsb SendEot      Send Eot to Cnc.
         Jsb Read1        Go read Eot char, binary.
         Clb              Set for Disable.
         Jsb XonXoff      Go turn off Xon/Xoff.
         Jmp EoFnd        -Go to common Eof rtn.
```

Figure-16. Protocol #8 Processor

Lastly, an EOT character is sent to the CNC to signal end of data, the CNC responds with an EOT as an acknowledgement, and the Xon/Xoff pacing by the MUX port is disabled. Program control then passes to EOFND in the Termination Section to conclude the download.

5.10 End Of File/Termination Section

The purpose of this section is to terminate a completed or an aborted download by closing the Tape Image file, sending a completion status to NCMGR, and releasing the Active Table entry as illustrated in Figure-17.

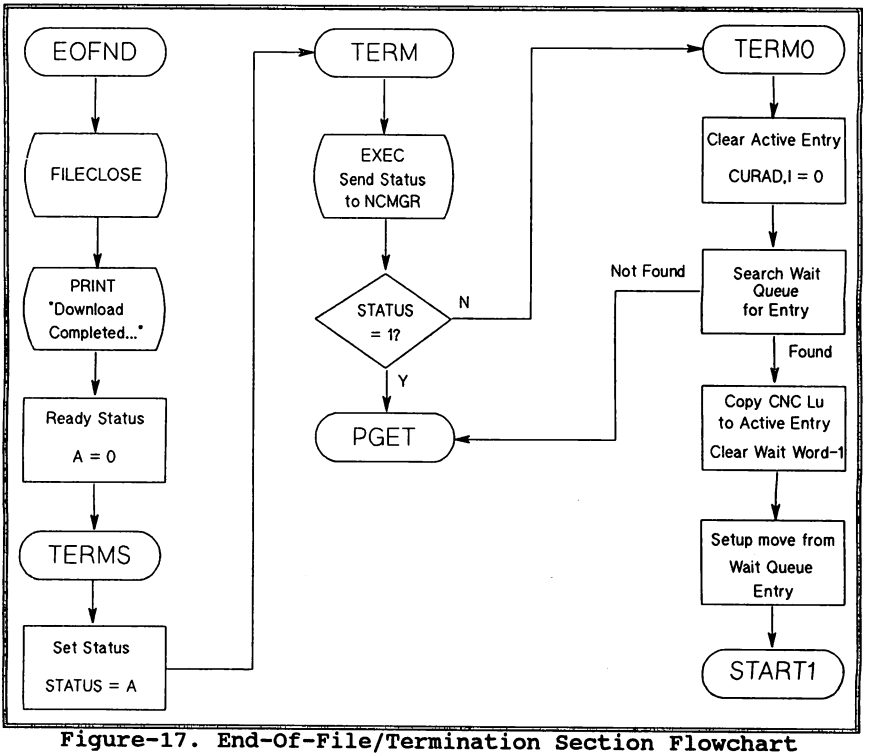


Figure-17. End-Of-File/Termination Section Flowchart

All protocol processors conclude by branching to the label EOFND in this section to complete a successful download. The routine FILECLOSE is called to close the Tape Image file and then the message "Download Completed..." is displayed on the log device by PRINT. The A-Register is set to zero to represent a success condition which will be sent to NCMGR at TERM.

TERMS is also entered from the Start Download Section via the ERR1 processor with the A-Register set to 1 when both the Active Table and the Wait Queue are full. The A-Register is simply stored in the variable STATUS to be sent to NCMGR.

The label TERM is entered from above or from the other error processors which have already set STATUS to an appropriate failure code. Next, the download completion status in the variable STATUS is sent to the application program NCMGR using a class write/read on NCMGR's class# contained in word-3 (CLAS2,I) of the Active Table.

The possible completion status values and meanings are shown below in Table-7. When NCMGR receives the status, it displays a success, failure or busy message to the operator.

Condition		STATUS
Success	(EOFND)	0
Busy	(ERR1)	1
Driver Error	(ERR3)	2
Device Down	(ERR2)	3
I/O Error	(ERR2)	4
Fmp Error	(ERR4,ERR5)	-Ierr

Table-7. DLOAD Completion Status

If the STATUS just sent to NCMGR was 1, it means that the Active Table and the Wait Queue were full. In this case, there is no Active Table entry to be released so control branches to PGET and awaits the next request; otherwise, DLOAD continues at label TERM0.

TERM0 is also entered from the Stop Download Section when NCMGR has requested a download to be aborted. This part releases the Active Table entry by setting the first word of the entry to zero. Next, the Wait Queue is searched for any waiting requests. If none are found, control returns to PGET in the Control Section. If a Wait Queue entry is found, the CNC Lu# from word-1 is copied to word-1 of the Active Table entry just released. Finally, the balance of the Wait Queue entry is setup to be moved to the Active Table entry and control passes to the Start Download Section at label START1 to activate this waiting download.

6. FUTURE ENHANCEMENTS

The most pressing problem confronting the HP-1000 Systems at Reliance's manufacturing facilities is the size of the RTE Operating System caused by the use of Rev-C multiplexers for CNC interfacing. One such plant has the hardware for 9 mux's but only 8 of them can be included in the system generation. Therefore, the first enhancement will be to modify all of the protocol processors for Rev-D multiplexer compatibility. A different solution to this problem could be to use a LAN for CNC interfacing.

Another enhancement will be the use of the new Signal/Timer facility of RTE to replace the timed suspension used in the SLEEP routine to achieve delays. Some protocols require up to 5 second delays during which all concurrent downloads are suspended.

And finally, the distributed nature of today's computing power will probably result in DLOAD being ported to a PC to implement a Cell Controller concept with the HP-1000 or a workstation as a file server.

7. CONCLUSION

DLOAD was originally written in May of 1983 and has changed very little over the years with the exception of new protocol handlers. It is a small part of the overall system, but has been a key component in achieving a more automated and efficient shop floor operation.

DISKMAIL INTERPROCESS MESSAGE SYSTEM

Donald A. Wright
Interactive Computer Technology
2069 Lake Elmo Avenue North
Lake Elmo, MN 55042 USA
Tel: 612/770-3728

DISKMAIL is a memory- and disk-buffered interprocess message system for the HP1000. It represents a substantial improvement over normal class I/O in both functionality and ease of use, providing an elegant way for programs to send data to each other.

Each mailbox has a pre-established ASCII name, and may employ any combination of memory buffering and disk buffering. If both are used, the disk buffer will begin to fill only when a preset memory queue limit is exceeded. All message queues are FIFO, with any memory portion logically nearer the destination than any disk portion. Requests are also provided for priority messages, purging a queue, changing buffer limits, examining limits and queue depths, and many more functions.

Class I/O is used internally for memory buffering, and variable-record-length circular files of fixed size are used for disk buffering. Disk-buffered messages are nonvolatile.

DISKMAIL was developed as part of a warehouse management system and is not a commercial product, nor has it been contributed to the INTEREX CSL. This paper is offered as an example of a way of enhancing program-to-program communications on the HP 1000 and other systems.

System Requirements:

These are some of the overall requirements for the message system:

1) The "Named" mailboxes are described by character variables up to 22 characters in length.

2) Messages may be both Memory and Disk buffered.

3) Configurable limits on both memory and disk, including zero space for either or both.

4) Disk-buffered messages are non-volatile and automatically recovered.

5) Efficient disk usage:
   a) Speed
   b) Space

6) Messages always FIFO:
Sender —> Disk —> Memory —> Destination.

7) Exception: Priority messages always go to the front and are not subject to memory limits.

8) The Disk buffer is not to be used unless the allocated memory space is full.

9) The Disk buffer will not be used if it is currently empty, there is no allocated memory space, but the destination program is waiting to read.

10) Memory messages may be forced back to disk by a special request.

11) The entire FIFO (queue) may be purged with a single request.

12) Any program may write to any mailbox.

13) Any program may read from more than one mailbox.

14) Only one program may read from any one mailbox.

15) Use of the package will not attach FMP routines, or the Fortran formatter, or Image routines to the calling application.

16) There are no limitations on the nature of the data in a message. It must be in an integer array when passed to the DiskMail subroutines.

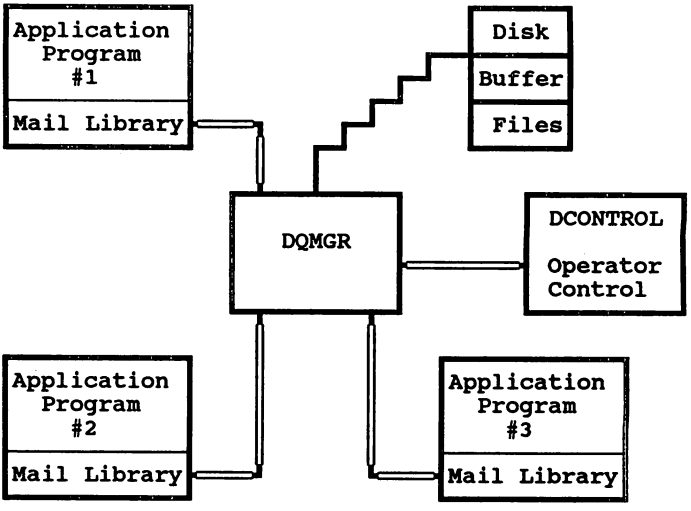
17) The package does not distingush between message types, number the messages, or perform any other type of message management. These things are done at the application level.

Design:

The system consists of four primary components: 1) A library of DiskMail Application Subroutines callable by application programs; 2) A central message management program (DQMGR); 3) an operator control program through which system management is done; and 4) Disk buffer files created for this system.

The schematic drawing on the following page illustrates the software implementation:

Schematic Drawing of Software System:



Library of Application Subroutines:

The SENDMAIL and GETMAIL subroutines are described below to show just how data is sent and received, and to show the options available to the caller:

SENDMAIL:

CALL SENDMAIL (CMDS, IBUF, LENGTH, MAILBOX [, QDEPTH])

(Optional QDEPTH)

FUNCTION: Send Class Mail to a named Mailbox.

SENDMAIL passes a specified number of bytes in an integer array from the calling program to a mailbox. The message is sent directly to the mailbox via class I/O if a program is waiting for mail there. Otherwise it is sent to DQMGR via EXEC 14, and that program decides whether to put it in the mailbox's class queue or buffer it out on disk. If no disk buffer file has been created for the mailbox, and the memory queue depth limit would be exceeded by this message, DQMGR will optionally instruct SENDMAIL to wait on a resource number until the memory queue is reduced.

If the queue-depth limit for the specified MAILBOX is zero and there is no disk buffer for the mailbox, then the actual queue depth is measured but not

compared with the limit, and locking is not done.

INPUTS - Formal parameters:

CMD - a character variable containing option characters. If neither is supplied, a string containing at least one blank character is required:

'N' - No-wait: SENDMAIL will not wait on the resource-number lock if there is no disk buffer file and the class queue depth has been exceeded, but will instead return with the QDEPTH parameter set to the 2's-complement (negative) value of the current actual queue depth.

Note: 'N' has no effect if the mailbox does have a disk buffer file. If the file becomes full an error is returned.

'P' - Priority: The message is immediately sent to the FRONT of the mailbox's class (memory) queue. No queue-depth-limit checking is done. This facility should be used with extreme care, as it puts messages out of sequence and has the potential to flood SAM.

IBUF - an integer array containing the data to be sent.

GETMAIL:

CALL GETMAIL (CMDS, IBUF, LENMAX, LENACT [, TYPE [, STAT [, MAILBOX]]])

(Optional TYPE, STAT, MAILBOX)

FUNCTION: Get a Mail Message from a Mailbox.

GETMAIL performs a Class GET to obtain one class message either from the mailbox assigned to the calling program or from another specified mailbox. If there is no message in memory, GETMAIL will request it from DQMGR and optionally wait on a class GET for it.

INPUTS = Three formal parameters:

CMD - a character variable containing option characters. Two are currently defined:

'N' - No-wait: If supplied, GETMAIL will not wait on the class GET if there is no mail, but will always return immediately with or without mail. If 'N' is not supplied, GETMAIL will return immediately with mail if at least one message is there, else it will wait on the class GET until mail appears.

'S' - Save-data: If supplied, an exact copy of the mail message will be returned, but the original message will remain in the class queue so that a subsequent GETMAIL request will return the same data. If not supplied, the message is returned to the caller and deleted from the class queue.

LENMAX - An integer*2 value specifying the maximum number of CHARACTERS that IBUF can accept.

MAILBOX - An optional character variable specifying the name of the mailbox in which to look for a message. If MAILBOX is blank or not supplied, the program's own mailbox name (i.e. MAILBOX-PNAME) will be used.

OUTPUTS = Four formal parameters:

IBUF - an INTEGER array which will receive the data. IBUF may be equivalenced to a character variable.

LENACT - Integer*2. The actual number of CHARACTERS returned to IBUF. The remainder of IBUF is undefined. If the 'N' option is set and there is no mail, or if an error occurs, LENACT = 0. When a valid message is received LENACT will always be greater than 0.

TYPE - an OPTIONAL integer*2 variable which returns the message type, as follows:

0 = Standard mail from another program.
1 = Job Control Message.
2 = Data received from a device (logical unit).

STAT - an OPTIONAL integer*2 variable containing the A-register (status) return from the class GET. The value depends upon the TYPE parameter, as follows:

0 or 1: Not significant if a valid message was returned. If the 'No-wait' CMDS character was supplied, the actual A-register return is converted so that STAT contains the positive number of still-pending device reads on all LU's for this mailbox (class) number.

2: Device status word (DVT word 6) of the device after the read completed. Can be tested for such information as timeout, EOT found (ctl-D), device errors, etc. Driver dependent.

The following is a complete list of the subroutine calls available to an application program:

GUTFORCEMAIL	Force a Memory Queue to Disk
GUTGETMAIL	Get a Message from Mailbox
GUTINQUIREMAIL	Inquire About a Mailbox
GUTPURGEMAIL	Purge Mail in a Mailbox
GUTRESETQLIM	Reset Mailbox's Memory-Queue Limit
GUTSENDMAIL	Send a Message to a Mailbox

DQMGR:

The bulk of the work in the DiskMail system is done by the DQMGR (disk-queue manager) program. Most messages go through it, and it knows about all messages.

DQMGR is normally dormant. Application programs sending or receiving disk mail schedule DQMGR (queue schedule with or without wait), and receive back a response via DQMGR's termination PRTN parameters and/or class I/O. DQMGR receives the mail messages via EXEC 14 and writes them either to memory or disk, depending on the particular mailbox. A program requesting mail checks its memory mailbox first and schedules DQMGR if there is none.

It accepts no run string and requests no operator input. Some control functions are available using the DCONTROL program.

At startup it inventories /DISKMAIL/BUFFERS looking for buffer files (type 3434) and keeps the results of that inventory in its own memory. There is no other startup file. DQMGR always terminates saving resources except at shutdown. Before terminating it always checks first for advisory messages in its own class queue.

DQMGR maintains a push-down log file called /DISKMAIL/DQMGR.LOG. All startups, shutdowns, and other significant events including disk buffer errors are logged there.

DCONTROL:

The DCONTROL program provides both diagnostic and maintenance functions. It is used to analyze problems during installation or operation, and to set or change DQMGR operational parameters while DQMGR is running.

Within its command list, DCONTROL allows use of every one of the GUTMAIL subroutines. These permit sending and getting mail, opening and closing buffer files, inquiring about a mailbox, and resetting its memory queue depth limit. None of these operations require shutting down the DiskMail system.

This is the list of available DCONTROL commands:

CL	[mbox]	*	Close Buffer File (prompt OK?)
CR	mbox [size]		CReate a new Buffer File
ER	integer		Describe Integer Error Code
FO	[mbox]	*	FOrce Memory Queue to Disk
GM	[mbox] [opts]	*	Get Mail from Mailbox
HE	[pram]		Interactive HElp
IQ	[mbox]	*	InQuire About Mailbox Params
LI	[mbox] [file] [word]		LIst the Buffer File
MF	file	*	Specify Monitor Mode File
MM	on/off	*	Switch Monitor Mode on/off
OP	[mbox]	*	OPen Buffer File
PU	[mbox] [opts]	*	PUrge Data in a Mailbox
RQ	[mbox] [qlim]	*	Reset Memory Queue-Depth Limit
RS	[mbox]	*	Reset a Mailbox's Statistics
SD		*	Shut Down DQMGR (prompt OK?)
SE	mailbox		Set Default Mailbox Name
SH	[mbox] [file]	*	SHow DQMGR Statistics
SM	[mbox] [opts] [data]	*	Send Mail to Mailbox
SS			SuSpend Self
ST	[mbox]	*	Send STop Message to Mailbox

An asterisk (*) denotes a command which will usually require processsing through DQMGR.

Where [mbox] is specified as an optional parameter, the default mailbox name set previously by the SE command will be used if none is supplied in this command.

DISK BUFFER FILES:

Each named mailbox having a disk buffering capability has an associated buffer file. These files are in the directory /DISKMAIL/BUFFERS/ and they have the same names as the names of their associated mailboxes. They are created using the DCONTROL program.

Buffer files are organized as circular fifos with variable-length records. The records are in the same format as standard FMP files with minor additions. The files are type-3434 to distinguish them from normal linear files. They are not 'open' to the FMP system while DQMGR uses them - DQMGR opens them once to determine necessary parameters and then closes them and accesses them directly with EXEC. They are fixed in size and never extended.

The idea is to build a circular file with intrinsic pointers to the beginning and end of valid data. The pointers are essentially the records themselves, with the expected result that access will be faster than it would be if separate pointers were kept on disk in another location. This file survives a system shutdown with data and pointers intact. It is always possible to inventory the file by reading contiguous records from both ends.

Variable-Record-Length Circular Files (see figure on following page):

1) Each record is bounded by length words, exactly like FMP records. The length words are a character count rotated right 1 bit, the same as FMP length words.

2) Message Flag. The first word of each record (what would be the first data word for FMP records) is a flag. ASCII 'OK' means the record is still valid, ASCII 'PU' means it is part of the 'purged zone', and ASCII '--' means it is empty but may have valid records on either side of it. The flag word adds two more characters to the record size, beyond the actual length of the message data.

3) A Message Attribute word. Word 2 of each message contains data equivalent to class-buffer parameter UV. This is passed through transparently in case it is needed at a later time for purposes of identifying a message's priority, or the sender's mailbox, or whatever. It adds 2 more characters to the record size.

4) The first word of the file (word 0) is always the first length word of a record, or EOF. This provides a guaranteed starting point for inventorying the file at startup.

5) As records are written to the file they are added linearly exactly as they would be added to a standard FMP file. A standard FMP 'EOF' mark (-1) is always written after each successive record. A forward search from word 0 will always find an EOF mark.

6) As these records are used (read), they are marked 'purged' by changing the flag word to 'PU', but are not actually deleted. The purged records together form a contiguous 'purged zone'.

7) When the file size would be exceeded by the addition of the next record, a 'splice-record' is created and the next data record is written in the purge zone at word 0 of the file.

8) The splice-record is a dummy with these characteristics:

   a. Its last length word is the last word of the file.

   b. The flag word is ASCII '--'.

9) Records are written into the purge zone until a new splice record is again required.

10) A record with the '--' flag can exist anywhere in the file as a placeholder record. It is included in queue-depth calculations because it reduces the space available in the file.

11) A record is kept of the amount of data in this buffer's memory queue. When a new record written to disk plus all of the records in memory might overwrite the trailing OK (unpurged) record, the write is not allowed and an

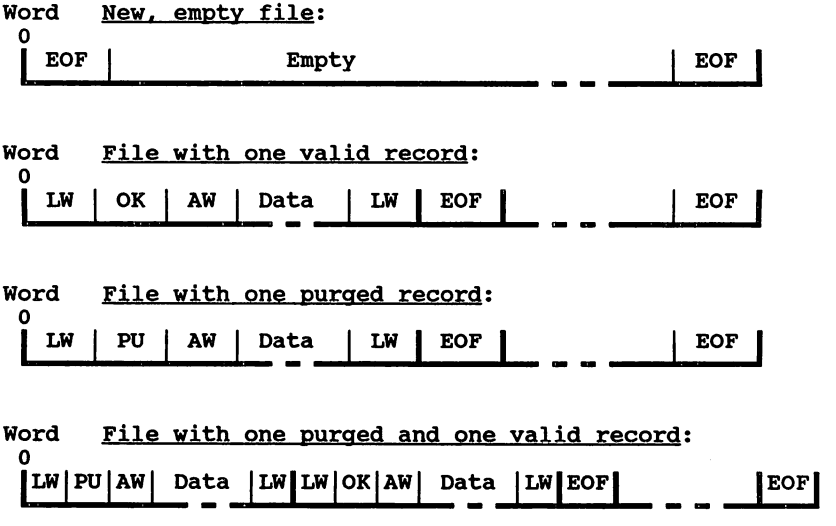
error is reported. Thus a flush of the memory queue to disk will always succeed.

12) Startup inventory: When DQMGR starts up, it searches for all files on /DISKMAIL/BUFFERS with a type of 3434 and examines each one for the bounds of the non-PU messages:

    a. Beginning at word 0, scan forward until the EOF is found.

    b. If the record at word 0 is non-PU, search backward until a 'PU' record or an illegal record length is found.

13) A new (empty) file is created by DCONTROL with the size specified by the operator, an EOF mark (-1) in word 0 and in the last word of the file, and directory information specifying that the EOF is in the last word of the file (file is full).



Circular files are normally designed with fixed-length records and with pointers to the first and last valid records kept within the file somewhere, or even within a management file somewhere else. Access to a record usually consists of a read or write of the information itself, and then another write to update the pointers.

The variable-length record files described above have some advantages over those conventional files, and no disadvantages. Advantages are:

1) Variable-record-length files can be significantly smaller for the same number of records if the record size does vary significantly.

2) Access is actually faster, because the management information regarding valid records is kept with the records themselves and not in a separate location, thus reducing access time.

In all cases except the splice, access to the variable-record-length circular file consists of one read followed by one write.

Tricky Parts:

Development of the DiskMail system was relatively straightforward. The design employs normal, documented RTE functionality except in the one case where DQMGR must determine the current depth of an existing class queue. In this case some privileged code is required, to chase the class queue and count the number of outstanding buffers and the total number of words of SAM required. This is the subroutine which returns that information:

```
MACRO
      NAM COMPLQUEUE 880107 ICT Scan Completed Class Queue

*  COMPLQUEUE finds the class number in the class table, then searches
*  that class number's completed-class queue, counting any completed
*  buffers and the amount of SAM they use.  It is called from Fortran
*  as follows:

*     CALL COMPLQUEUE (CLASS, NUMBFS [, NUMWDS [, MAXBUF ]])

*  Where both parameters are INTEGER*2 values:

*     CLASS is the class number in question.

*     NUMBFS is the current number of completed class buffers (zero or
*        positive).  INTEGER*2.  If the class number is not in use or
*        the queue is corrupt, NUMBUF returns -1.

*     NUMWDS is the total number of SAM words used by all completed
*        buffers in the queue, including header overhead.  INTEGER*2.
*        If there are no completed buffers, NUMWDS = 0.  This parameter
*        is optional.

*     MAXBUF is the largest single buffer in the queue, including header
*        overhead.  INTEGER*2.  If NUMBFS = 0, MAXBUF returns Word 1 of
*        the requested class entry.  If bit 14 of that word is set, then
*        a program is waiting on a GET on the class number.


*  Note that this subroutine is for RTE-A only.  It first finds the
*  class number in the system map and, if the queue is not empty, it
*  chases the list in the SAM map.

*  COMPLQUEUE goes privileged while it executes so that the operating
```

```
*  system will not change the queue while COMPLQUEUE is chasing it.
*  It temporarily sets the DATA2 map to the SAM map while privileged.


        EXT $CLTA,$LIBR,.ENTP,$LIBX,.SWMP,.LWD2
        EXT .XLB1,.XLA2,.XLB2

        ENT COMPLQUEUE



CLASSA  NOP         Entry parameter addresses
NUMBFA  NOP
NUMWDA  NOP
MAXBFA  NOP

COMPLQUEUE NOP      Entry to COMPLQUEUE
        JSB $LIBR   Shut down the op sys (go privileged)
        NOP
        JSB .ENTP   Recover caller's pram addresses
        DEF CLASSA   beginning here

*  Save the current working map (WMAP), set the DATA2 map to 4 (SAM):

        JSB .SWMP   Save WMAP
        DEF SAVEMAP Keep it here
        JSB .LWD2   Reset DATA2 map
        DEF -D4      to SAM Map, number 4

*  Preset all return variables (note: the optional ones point back to
*  the A-register if not supplied):

        CLB
        STB @NUMBFA Set NUMBFS = 0, default
        STB @NUMWDA Set NUMWDS = 0, default
        STB @MAXBFA Set MAXBUF = 0, default

*  Find the class number in the table, error it it's zero (not in use):

        JSB .XLB1
        DEF $CLTA   B = address of class table
        LDA @CLASSA A = class number from caller
        AND -B377   Strip off just the class# index
        ADB A       B = address of our class number
        JSB .XLB1
        DEF @B      B = Word 1 of requested class entry
        SZB,RSS     If that word is zero,
        JMP ERROR    we have an error to report
        SSB,RSS     If the queue isn't empty,
        JMP FIRST    go chase it
        STB @MAXBFA Else make it the MAXBUF return value
        JMP EXIT    And leave
```

```
*  Now chase the list, incrementing both variables for each buffer and
*  testing for lost (something wrong with SAM linked list):

FIRST   STB NEXLINK Save initial pointer into SAM
CHASE   SSB         If pending list word is negative, we
        JMP EXIT     are at the end of the list
        ADB =D7     B = address of next buffer word 8
        JSB .XLB2   B = buffer length including header
        DEF @B
        LDA B       A = B
        ADA @NUMWDA A = cumulative SAM utilization
        STA @NUMWDA NUMWDS = cumulative SAM
        LDA B       A = buffer length again
        CMA,INA     A = - buffer length
        ADA @MAXBFA A = largest so far - pending length
        SSA         If A is negative, this one's larger,
        STB @MAXBFA  so make it the largest
        JSB .XLB2   B = next buffer's list linkage wd
NEXLINK NOP          Next becomes pending, B = new next
        STB NEXLINK Save the link word
        ISZ @NUMBFA Incrm NUMBFS counter & test for lost
        JMP CHASE   Not lost, go see if we're done

*  Error exit for unused class number or corrupted linked list:

ERROR   CCA
        STA @NUMBFA NUMBFS = -1, error flag

*  Exit here after restoring the original DATA2 map:

EXIT    LDA SAVEMAP A = original value of WMAP
        RRR 10      Move DATA2 map number down 10 bits
        AND =B37    Select just the 5 DATA2 map bits
        JSB .LWD2   Load the DATA2 register
        DEF A        from the A-register

*  Scram, using $LIBX:

        LDA @NUMBFA A = NUMBFS on return
        CLB
        STB NUMWDA  Reset optional pram pointers
        STB MAXBFA   for next call
        JSB $LIBX
        DEF COMPLQUEUE

*  Local variable:

SAVEMAP NOP         Value of WMAP upon entry
        END
```

SoftBench Link/1000 Encapsulation

A State of the Art CASE Environment for the HP1000

Hilary Feier
Hewlett-Packard
11000 Wolfe Road
Cupertino, Ca. 95014 M/S 42UN

Overview

SoftBench Link/1000 is an encapsulated tool that runs on SoftBench, therefore this paper will first outline the components of SoftBench and the significance of SoftBench as a CASE (Computer Aided Software Engineering) tool. It will then present a simple tour of SoftBench Link/1000 Encapsulation from a user-level perspective. Finally, the components of SoftBench Link/1000 Encapsulation will be discussed in further detail.

I. Introduction

In the past several years CASE has grown from a concept to an industry. Integration of tools, such as the editor, compiler, and debugger, into a homogeneous, window-oriented environment lead to a more productive and better quality software development environment. With SoftBench, Hewlett-Packard provides HP-UX developers a software development environment consisting of both an integrated set of program development tools and a tool integration platform. SoftBench is embedded in a window-oriented environment, based on the industry-standard X11 Window System with OSF/Motif appearance and behavior. SoftBench consists of five tools which cover the construction, test, and maintenance phases of software development (see Figure 1):

1. **Development Manager** manages all file oriented tasks (e.g. version control). Execution of all other tools can be initiated from the DM as well.
2. **Program Editor** edits source files, load files, etc.
3. **Program Builder** activates compiler and linker using automatically generated makefiles.
4. **Program Debugger** tests the execution behavior of a program.
5. **Static Analyzer** provides information regarding the structure of a program. (for HP-UX only)

These tools communicate with each other via a Broadcast Message Server (BMS). This allows for a task oriented and partly automated working environment in the edit-compile-link-debug cycle.

Besides being a software development environment SoftBench is also a tool integration platform. Due to the Broadcast Message Server, SoftBench is designed to integrate other available software packages. For example, software analysis packages or documentation packages could be integrated into SoftBench. Additionally, user written tools can be integrated via the HP Encapsulator package.

SoftBench Link/1000 Encapsulation transparently integrates the HP1000 A-Series into the SoftBench environment. SoftBench Link/1000 Encapsulation, in conjunction with SoftBench, provides a core set of tools for RTE-A application construction, testing, and maintenance. The programmer edits and administrates HP1000 source code on an HP9000 HP-UX platform with the standard SoftBench functionality. SoftBench Link/1000 Encapsulation provides the link to the HP1000 so that the architectural dependent tasks (such as compile, link, and debug) are executed transparent to the user. For the first time the HP1000 user can utilize standard techniques in software development such as revision control, automatic generation and usage of makefiles, as well as tool communication functions to automate tasks in the edit-compile-link-test cycle. With improved windowing capability and a simple user interface, SoftBench Link/1000 Encapsulation results in increased productivity and quality in the software development environment for the HP 1000 A-Series system.

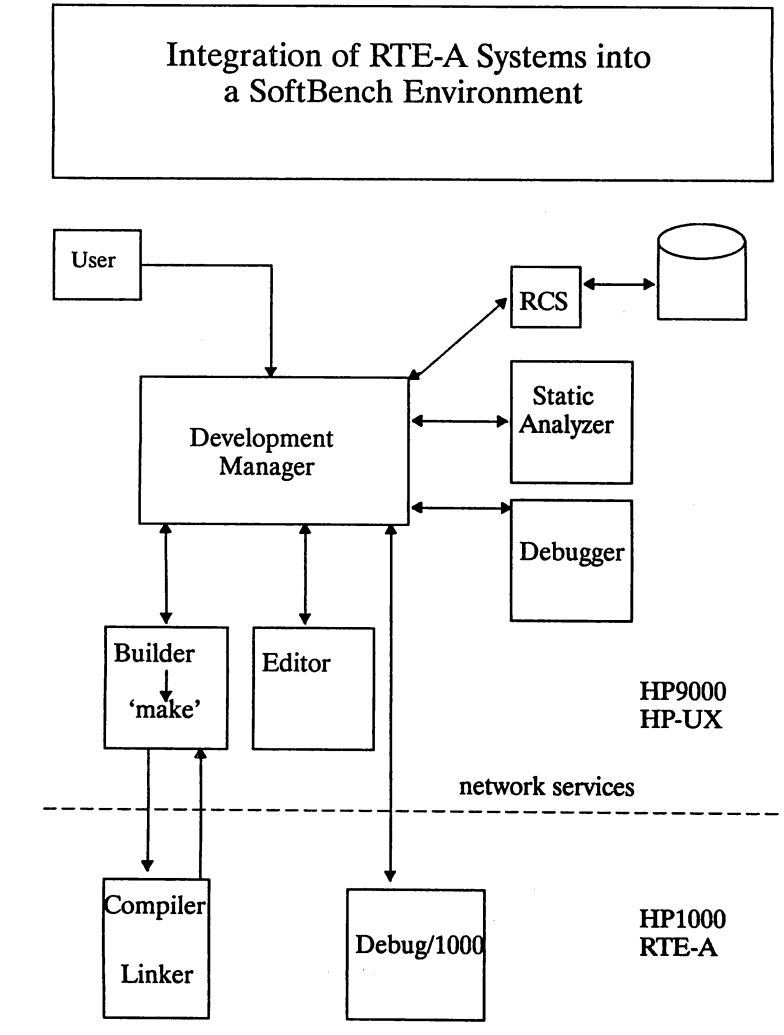

Integration of RTE-A Systems into
a SoftBench Environment
User
RCS
Development
Manager
Static
Analyzer
Debugger
Builder
'make'
Editor
HP9000
HP-UX
network services
Compiler
Linker
Debug/1000
HP1000
RTE-A

Figure 1

II. Using SoftBench Link/1000 Encapsulation

After SoftBench has already been installed on your HP9000 workstation, SoftBench Link/1000 is ready to be installed. Installation of SoftBench Link/1000 software on your HP-UX system fully integrates HP1000 RTE-A development tools into SoftBench. Therefore, simply running SoftBench, after installing SoftBench Link/1000, allows the user to access all HP1000 RTE-A utilities while maintaining the full functionality of SoftBench.

Connecting to the HP1000

As soon as a user starts a SoftBench session, the C_1k utility is started. The C_1k utility transparently connects to the HP1000. RTE-A system configuration information is defined by the user in the configuration file, $HOME/.SBL_conf. If this configuration file has not yet been defined, for example the user is a first time SoftBench Link/1000 user, then a window pops up and the SBL_config tool runs automatically, prompting the user for configuration information. The required configuration information is:

- nodename of the RTE-A system
- account information for the user on the RTE-A system (user-id,password)
- base directory on the HP-UX system

After this information is entered and the SBL_config tool is closed by the user, C_1k will automatically try to make a connection to the specified nodename. C_1k pops up a window that displays all messages being passed back and forth between the HP9000 and the HP1000. This connection is maintained the entire time SoftBench Link/1000 runs, allowing the user to constantly monitor input and output activity on the HP1000.

Moving Software from the HP1000 to the HP9000

Once the user has successfully connected to the HP1000 RTE-A machine, he is ready to set up the development environment on the HP 9000 HP-UX machine. For already existing development environments on the HP1000, SoftBench Link/1000 provides an update utility (UPDATE) to transfer all the source and load files from the HP1000 RTE-A platform to the HP9000 HP-UX platform while maintaining the directory structure. This window driven utility offers a simple user interface to start the environment transfer.

Building RTE-A Software

After moving all the software over to the HP-UX platform the user can now take advantage of the SoftBench application constructing environment. The build utility (BUILD) can be used to build RTE-A software. Software builds take source code, include files, and any other application specific information and compiles and links them to create executables. BUILD will look for a makefile in the current directory and use that makefile to build the software. If no makefile exists, a makefile can be automatically generated by holding down the MAKEFILE menu button and choosing "create program" or "create library", appropriately. An additional window will pop up prompting the user to enter further information, such as the desired executable name, compiler options, load flags, etc. Once this information is entered the makefile will be generated and the builder can be run to build the program or library.

If the library or program encountered errors during the build, the BUILD window will list all of the error messages. By simply clicking on the error message, the file containing that error will be opened by the SoftBench editor and the cursor will be positioned on the line generating the error. The user can quickly fix the problem, save the file, and rebuild.

Debugging RTE-A Software Using SoftBench Link/1000

A successful build doesn't always mean functional code. Logical errors and runtime errors often occur in a program development environment. SoftBench Link/1000 provides window driven functionality to remotely run Debug/1000 on a telnet window connected to the HP1000. By pulling down the UTILITIES menu from the DM and clicking on one of the two DEBUG options, "debug1" or "debug2", the user can start the debugger.

Using Revision Control

Once the user's software has been transferred to the HP9000 and makefiles have been created, the user can check his source code into source control. SoftBench integrates the standard HP-UX revision control utility, RCS, into a menu driven utility. From the Development Manager the user can pull down the VERSION menu and "create initial versions" for the software. Additional editing of these source files will require that these modules be "checked out" of Source control. This will help maintain file integrity and revision control by allowing only one user to have one version of the software checked out at any one time. (For more information on revision control, RCS, see the RCS man page).

III. Components of SoftBench Link/1000 Encapsulation

Tool Communication

Broadcast Message Server (BMS)

The Broadcast Message Server is the heart of SoftBench. On the HP9000 the SoftBench tools communicate in a networked, heterogeneous environment via a broadcast communication facility designed to support close communication of independent tools. Message requests allow one tool to invoke the functionality of another tool. For example, when C_1k was unable to connect to a remote HP1000 RTE-A host because no configuration information had been set, a trigger was initiated to start the SBL_config utility. Notification messages allow tools to define triggers which respond to events and initiate other actions. Triggers are cause/effect relationships. They can be caused by system events, and in turn cause a user-defined action to occur. In this manner, triggers link one or more tools together to support a task or process.

RTE-A Remote Execution and File Transfer Daemons for HP1000 Communication

Using the SoftBench platform for RTE-A program development requires that some tasks be activated on the RTE-A system (compiler, linker, debugger). These actions are initiated through appropriate statements in the makefiles which are handled by the SoftBench Builder. Remote command execution of RTE-A tools is implemented with a client-monitor concept based on the telnet protocol. Results produced by RTE-A tasks will be copied to the local SoftBench environment and output to the builder's window. All HP1000 functions appear completely transparent to the user. The necessary file transfer between HP-UX and RTE-A systems is handled by a file transfer daemon based on the NS dscopy tool. From the users point of view all architectural dependent actions are performed transparently and efficiently.

Development Manager (DM)

The Development Manager is the responsible tool for all file management actions as well as for the activation of other SoftBench tools. The Development Manger has been enhanced by SoftBench Link/1000 Encapsulation to include a Utilities menu that contains all HP1000 specific RTE-A programming tools, such as:

- programming language compilers (FTN7x, Macro/1000, C/1000),
- link,
- Debug/1000, and
- library utilities (merge, lindx)

These tools, when invoked, will set up a connection to the HP1000 system and run the tools remotely.

Additionally, the DM automatically alters the actions menu to list only possible actions on a given RTE-A specific file type extension. For example an xxx.ftn can only be edited and compiled whereas an xxx.run can only be executed or debugged.

One of the most important tasks of the DM is the handling of the revision control system. Revision control provides features like multiple revisions, audit trail, access control, efficient storage, and flexible retrieval. RTE-A files, in addition to HP-UX files, can be checked in and checked out from the DM, and then can be operated on by other SoftBench tools. This is an important feature to the RTE-A developer since there is no Revision control on the HP1000.

Program Editor

The Program Editor is an easy-to-learn, programming language sensitive, mouse/menu based standard SoftBench file editor. It automatically synchronizes file views. If a file is modified by a tool in one window, the file is updated in the other windows where it is also being viewed. It will automatically adjust for programming language specific indentation requirements. The editor is highly customizable (For example, different keyboard accelerators can be specified).

Program Builder

The Program Builder, based on the HP-UX utilities *mkmf* and *make*, automates the process of compiling and linking an RTE-A program composed of many different source files. This leads to efficient builds: only the source files that have been modified are recompiled. The dependency information required for efficient builds can be automatically generated with the make makefile generator. SoftBench Link/1000 enhances the SoftBench make makefile routine by providing an RTE-A makefile template that recognizes RTE-A specific suffixes, compiler options, etc, therefore allowing for automatic makefile generation for HP1000 RTE-A applications. The resulting makefile contains all commands to either create a program or a library for an HP1000 system. All architectural dependent actions (such as, recognizing that a Fortran program must be compiled on the HP1000) are integrated into the makefile and are performed transparently to the user.

When the makefile is executed all tasks for the program or library development are performed automatically (compile/link or compile/merge/lindx). First, the source file is transferred to the RTE-A system. Then the compiler (e.g. FTN7x) is started remotely from the HP9000 on the HP1000 system. (All required file transfers and subsequent RTE-A tool invocation are handled by the SoftBench Link/1000 daemons.) The resulting relocatable remains physically on the HP1000 RTE-A system.

When creating a library, the compiled modules are merged into a library which is lindxd afterwards. For the creation of a program the linker is invoked. The resulting executable is also kept on the RTE-A system whereas on the HP9000 system a dummy executable and a dummy relocatable are created in the user's working directory to satisfy the HP-UX 'make' mechanism. This dummy executable can be invoked which then triggers the actual executable on the HP1000 to either be run or debugged remotely, depending on how it was invoked.

Additionally, the program builder allows browsing on compiler errors and warning messages in the related RTE-A source code files. This results in automated invocation of the SoftBench editor which is positioned on the relevant line in the source file.

Program Debugger

The Debugger is invoked from the HP9000 but executed directly on the RTE-A system. This utility can be run from the UTILITIES menu in the DM or by clicking on "debug1" or "debug2" from the actions menu. When invoked, this utility automatically sets up a telnet connection to the HP1000 RTE-A system and runs Debug/1000 on the specified executable. (For more information on how Debug/1000 works, see the *Symbolic Debug/1000 Reference Manual,* part no. 92860–90001.) Optionally, the user can choose the "debug2" menu option. This allows for debugging in two windows. Two telnet connections are made to the HP1000 RTE-A system and output is redirected to the second window. In other words, one window will display debug information while the other will display the output of the program. This is very useful for debugging Graphics programs. These windows are configurable, therefore allowing a GFoX[1] window to be used for Graphical display.

Static Analyzer

The Static Analyzer provides information regarding the structure of a program. It provides cross-reference queries such as: "where declared," ,"where defined," "where used," or "where modified". This tool is particularly valuable while maintaining code or porting code. However, the generation of static information is a function of the HP-UX compiler. Therefore, only HP1000 source code which can be compiled on an HP-UX system can be used for static analysis.

1. GFoX is HP's Graphics and Forms Terminal Emulator for X11.

Additional SoftBench Link/1000 Encapsulation Tools

To allow trouble-free interaction between the SoftBench development system on HP-UX and the RTE-A target system, additional tools are provided which can be started from the development manager.

RTE-A-Configuration (sbl_config)

The configuration file, $HOME/.SBL_conf, contains information necessary for command and data exchange between the HP-UX and RTE-A systems. This information includes:

- HP1000 nodename, login, and password,
- base directory for the HP-UX system, and
- directory name for "make" template files.

These fields can be configured using the SBL_config utility. In addition to the above information, DSCOPY and C_1k pipe information is contained in this file. (This information is not configurable.)

C_1k Communication Server

The C_1k communication server controls all SoftBench Link/1000 communications processes. It checks the configuration file, links the HP-UX system to the HP1000 by initiating the dscopy daemon, and establishes a session on the HP1000. The connection must be maintained throughout the development process since the dscopy daemon is started and maintained by the C_1k utility. If this connection is closed then the pipe will be closed and all subsequent file transfers will fail.

Environment Transfer and Source File Consistency Check (sbl_update)

If a user has an existing software project on the RTE-A system which he wants to maintain under SoftBench Link/1000 Encapsulation, UPDATE (sbl_update) creates the necessary directory hierarchy on the HP-UX system and copies all source and load files to their related HP-UX directories. UPDATE will copy all files and subdirectories under a specified global RTE-A directory, as defined by the context and base directories, to the HP-UX system.

The HP-UX Base Directory and the Context Directory

The base directory is set by the user using the SBL_config utility when he sets up his development environment. All directories and files that correspond to the RTE-A system must reside below the HP-UX base directory. The corresponding RTE-A directory is then defined by the context directory. If the context directory is NOT set below the HP-UX base directory then subsequent file transfers will fail.

When the context is correctly set below the HP-UX base directory, SoftBench Link/1000 interprets the context in the following manner:
It parses off the HP-UX base directory and sets the corresponding RTE-A global directory to the next directory in the path.

For example, if the HP-UX base directory is set to */users/basedir* and the context directory is set to */users/basedir/global*, then the UPDATE utility will transfer all source and load files from */global/@.@.s* on the HP1000 to */users/basedir/global/* on the HP9000. (See Figure 2)

HP-UX Base Directory: /users/basedir
Context: /users/basedir/global

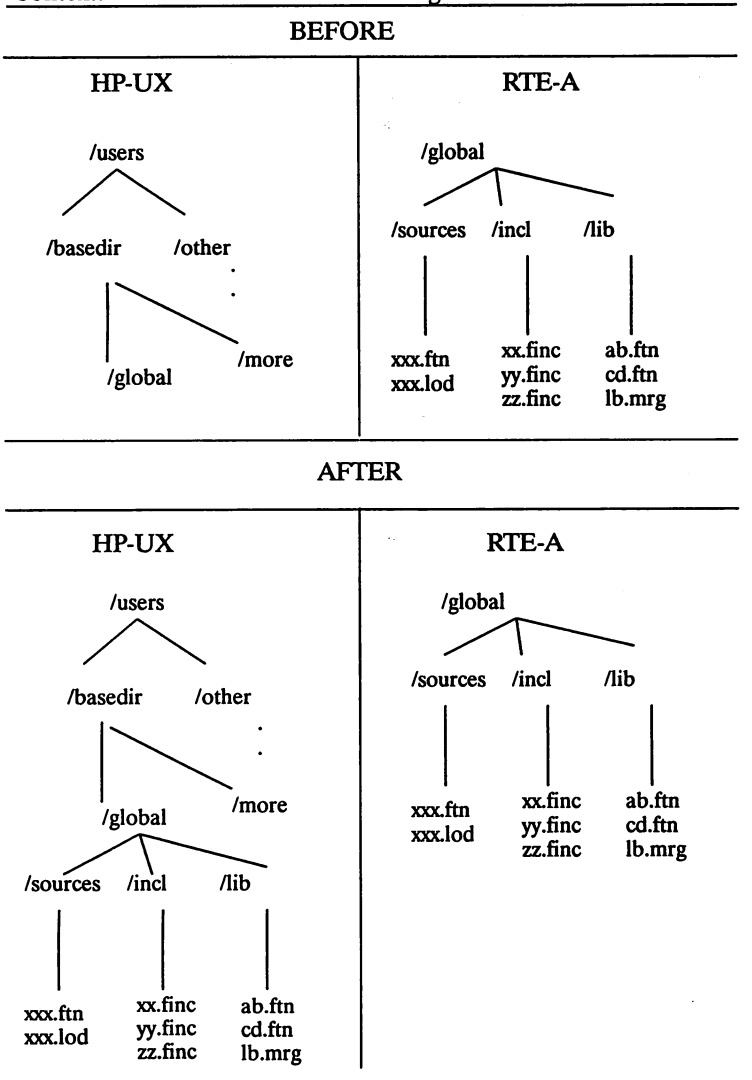


Figure 2

A second major function of UPDATE is the possibility to perform a source file consistency check. As the user is working in two different file systems (HP-UX and RTE-A) it is useful to check the consistency of all related directory information; this tool checks the availability as well as the content of related files in the two file systems. For example, this could be used on a nightly basis to check that all files are consistent on both machines. If the files aren't consistent, it is likely that a file has been changed but the software hasn't been rebuilt. Note that if the developer makes changes directly on the RTE-A system and then rebuilds, the changes will be overwritten. Therefore, it is recommended that all development take place on your HP-UX system so as to not run into file consistency problems.

IV. SoftBench Link/1000 Encapsulation Configurability

In addition to the flexibility and functionality that SoftBench already provides, SoftBench Link/1000 specific functionality is configurable as well. SoftBench Link/1000 has been designed to be configurable to every individual program environment's needs.

- All RTE-A specific utilities running on the HP1000 have configurable window options. Beyond color schemes, this allows a user to substitute GFoX for an HPTERM window if so desired.
- Accepted include file type extensions are configurable such that SoftBench Link/1000 recognizes any RTE-A include file.
- RTE-A makefile templates are customizable. (Full understanding of *make* is required, therefore this is recommended for advanced users only.)
- Alternate configuration files can be specified when running individual tools allowing a user to access another development environment on the same HP1000 (not under the local base directory) without reconfiguring.

For more information on how to customize the SoftBench Link/1000 Development Environment, see the *SoftBench Link/1000 Encapsulation Reference Manual.*

V. Summary

SoftBench Link/1000 Encapsulation integrates the HP1000 programming tools into the SoftBench environment. In summary the following features are implemented:

- integration of HP1000 systems via LAN into the standard SoftBench environment.
- source code administration under revision control resulting in safe development and modification of source code and the possibility of restoring previous software revisions.
- automatic generation and usage of makefiles for the HP1000.
- transparent use of RTE-A tools (FTN7x compiler, linker, Debug/1000 etc.)
- debugging of blockmode applications (e.g. Graphics/1000) over the network in a window oriented programming environment (in conjunction with GFoX).
- automation of tasks during software development and testing.
- common window-oriented user interface, based on the X11-standard, for all tools in the software development system.
- ease of extension and integration of other tools using the HP-Encapsulator
- possibility to integrate software documentation tasks (e.g. integration of HP-FrameMaker).
- multi-window graphical user interface and integrated on-line help functions for ease of learning.
- highly configurable development environment.

These features optimize software development time and improve the quality of the software development process.

Acknowledgements

I would like to thank the following people for their support and their help on this paper: Wolfgang Oskierski, Scott Glover, Kristin Anderson, Doug Fisher, and Carolyn Krieg.

TITLE: How DSO Develops Software and Hardware

AUTHOR: Alan Tibbetts

Hewlett-Packard Co./Consultant

3498 Gibson Avenue

Santa Clara, CA 95051

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 1013

TITLE: Using and Controlling Dialup Modems for Remote Data Acquisition

AUTHOR: Wendy King
US Naval Observatory; Time Service, Bldg. 78
34th & Massachusetts Avenue, NW
Washington, DC 20392-5100
202-653-0486

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 1014

TITLE: HP 1000 Networking Strategy and Future Directions

AUTHOR: Lynn Rodoni

Hewlett-Packard Co.

Cupertino, CA 95014

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 1021

BSD IPC ON THE HP1000

Ramesh Radhakrishnan
Hewlett-Packard Company
11000 Wolfe Road
Cupertino, California 95014

INTRODUCTION

In the early 1980s, DARPA (Defense Advanced Research Projects Agency) funded the implementation of a protocol suite to interconnect heterogeneous networks. This protocol suite is now widely known as the TCP/IP protocol suite. One of the first implementations of the TCP/IP protocol suite appeared in the 4.1 BSD UNIX release. Network interprocess communication (often referred to as IPC) was made possible by providing a programmatic interface called sockets. The concept of sockets not only defines the data structure of a communication endpoint but also the various operations that can be performed on it. The type of communication endpoint provided is generically known as the application programmer interface (API). Several APIs exist today, such as Berkeley Sockets (defined by the Berkeley Software Distribution), TLI (defined by AT&T) and XTI (defined by XOpen). In addition, there are several proprietary APIs such as NetIPC provided by HP on all its platforms. Despite this plethora of APIs, the Berkeley Sockets (henceforth used interchangeably with BSD sockets and BSD IPC) interface dominates the TCP/IP world as the interface of choice and is generally regarded as the de facto standard API for UNIX interprocess communication.

NS-ARPA/1000 release 5.24 now offers BSD sockets along with NetIPC, as a network interprocess communication interface on the HP1000. This not only reiterates HP's committment to standards (open or de facto), but with the imminent release of the C compiler on the HP1000, network portability across HP platforms is a distinct possibility. With the addition of the BSD socket interface to NS-ARPA/1000, interprocess communication among the HP9000, HP3000 and HP1000 using Berkeley sockets is now available. In general, BSD sockets on the HP1000 exhibit the same behaviour as that of BSD sockets on HP-UX 8.0. This paper will attempt to clarify the differences wherever they arise. However, note that there may exist operating system dependencies, such as the fork() call, that makes complete portablity a trifle more difficult.

This paper describes the use of the Berkeley socket interface and the associated utility routines on the HP1000. The next section of the paper will deal with the Berkeley Interprocess Communication paradigm in detail. This is followed by a section devoted to Berkeley socket and Pascal/FORTRAN interface design issues that are specific to HP1000. It is advisable that Pascal/FORTRAN users skim over this section before perusing the next section. The descriptions of the function calls are based on C language semantics. It is assumed that the reader is familiar with the syntax of the C language, especially that dealing with pointers.

BSD INTERPROCESS COMMUNICATION

Overview

The Berkeley interprocess communication model (henceforth, used interchangeably with Berkeley IPC) is based on a client/server paradigm. The accompanying figure shows a typical scenario for a connection oriented communication between two processes.

Client		Server
socket()	Client and Server establish communication end-points	*socket()*
	Server binds to a well-known port address	*bind()*
	Server indicates readiness to accept connection requests	*listen()*
	Server waits for connection requset	*accept()*
connect()	Client sends connection request to server	
send() *recv()*	data <=====> transfer	*recv()* *send()*
shutdown()	Close communication endpoints	*shutdown()*

Creating a communication endpoint

The basic entity for communication is a *socket*. A *socket* is a data structure that is associated with a given process. Two processes may communicate by each creating a *socket* data structure and then performing the necessary actions to establish a connection between the two sockets. As a further level of abstraction, users will only deal with socket descriptors, which can be regarded as a reference to a specific *socket* data structure. A *socket* is created and associated with the calling process by the following call:

```
sd = socket(int domain,int type,int protocol);
int sd;
```

Domains: *sd* is the socket descriptor that should be used in all subsequent calls to reference the *socket* that was created by the above call. Note that once a socket has been created with the *socket()* call, only BSD socket routines can be used on the *socket* created by this call. By the same token, NetIPC routines can only be used on sockets created with the *IPCCreate()* call.

Now, in order for sockets created by two independent processes, perhaps on different machines, to communicate with each other, a connection needs to be set up between them. To set up a connection, there needs to be a way to name the sockets so that each one can refer to the other. Again, names are generally translated into addresses. The space from which an address is drawn is called the *domain* and this is the first parameter to the *socket()* call. There are several address domains (those with the AF_ prefix) defined in the include file *<socket.h>* file. However, the only ones supported for NS_ARPA/1000 are:

AF_UNSPEC
AF_INET

Internally, AF_UNSPEC defaults to AF_INET domain. The AF_INET domain is also known as the Internet domain. In this domain, a socket address consists of an Internet address of 32 bits and another 16 bit address called the port address. We will delve into this further when we talk about binding a socket to a specific address.

Types: The second parameter specifies the *type* of socket created in the *domain* referred to by the first parameter. This type really refers to the communication style. Although several communication styles have been defined in the *<socket.h>* header file (constants with the SOCK_ prefix), the only one supported by NS-ARPA/1000 is the "stream" style represented by the constant SOCK_STREAM. Stream communication implies that communication takes place between two sockets who have already established a connection between themselves. It also means that the communication is full-duplex, and reliable. The data is received in sequence and that no message boundaries are maintained. Thus, a *recv()* call at one end of the connection may return data due to several *send()* calls at the other end of the connection, or only part of the data from a single *send()* call because all the data has not yet been received or if there isn't enough buffer space on the receiving side for all the data. The protocol implementing such a style is responsible for retransmitting lost or error data, ensuring sequenced delivery of data and returning error messages when a connection has been broken.

Protocols: The third parameter is the protocol family. Usually, there is one protocol for each socket type in each domain. A protocol is simply a set of rules that controls the transfer of data between two sockets. In addition, it also keeps track of the socket names or addresses, sets up connections between sockets and cleans up resources after a connection is shut down. All the constants starting with PF_ in the *<socket.h>* header file represent protocols. However, the only ones supported by NS-ARPA/1000 are:

PF_UNSPEC
PF_INET

Usually, it is sufficient to specify the default protocol or PF_UNSPEC. The *domain* and *type* parameters should be sufficient to provide the right protocol. For NS-ARPA/1000, since the only acceptable values for *domain* and *type* are AF_INET and SOCK_STREAM respectively, the underlying protocol provided by the system for interprocess communication is always TCP.

Binding a socket to an address

A socket is created without being bound to any specific port address. Until an address is bound to a socket, other processes have no way of referencing it and hence no connection can be set up between it and any other socket. A connection in the Internet domain is identified by the following quintuple:

<protocol,local IP address, local port address, remote IP address, remote port address>

This quintuple is unique over the entire Internet domain for all connections. The *bind()* call allows a process to specify one half of this association, *<protocol, local IP address, local port address>*. The *connect()* and *accept()* calls will complete the quintuple when a connection is established.

In the client/server paradigm of communication, it is essential that the server bind its socket to a specific port address. It is usually not necessary for the client to bind its socket to a specific port address since the *connect()* call will automatically bind an address to the socket if used on an unbound socket.

The bind system call is used as follows:

```
int bind (int sd, struct sockaddr_in *addr, int addrlen);
```

The *bind()* call binds the socket, *sd*, to an address that is provided in the *sockaddr_in* structure. *addrlen* is the length in bytes of the relevant information in the *sockaddr_in* structure. This structure is declared in *<in.h>* and has the following fields:

```
struct sockaddr_in {
  short          sin_family;   /* address family */
  u_short        sin_port;     /* port address */
  struct in_addr sin_addr;     /* host IP address */
  char           sin_zero[8];  /* unused */
}
```

sin_family should always be set to AF_INET. The *sin_port* field is filled with the port address that the socket is going to be bound to. This port address can either be hard coded into the program or can be obtained by programmatically accessing a network database file called */etc/services*.

For ease of use and HP-UX compatibility, BSD IPC provides four network databases and a set of utility routines to extract information from them. These databases and routines are discussed below.

Berkeley network utility routines and databases : There are four BSD network database files maintained by the network administrator. They are :

```
/etc/hosts
/etc/networks
/etc/protocols
/etc/services
```

/etc/hosts is an ASCII file that contains information about the mapping of a hostname and its aliases to its IP address. The IP address is in dotted decimal format. */etc/networks* is an ASCII file that maps the name of a network and its aliases to the IP network address. */etc/protocols* is an ASCII file that maps the name of a protocol and its aliases to its official Internet number. */etc/services* is an ASCII file that maps the name of a service and its aliases to the port that this service is going to be listening on. There is also a provision to specify the protocol that this service uses.

Note that these are all static files. The NS-ARPA/1000 library now contains several routines that access the information in these files. All these functions return a pointer to a static structure containing the requested information or a NULL on error. The memory for the structure is allocated from the "C heap" via *malloc()*. C/Pascal/FORTRAN users will have to use the *free()* routine (part of the C library) in order to reuse this memory.

The library routines do not attempt to contact any name servers. The lookup is purely static. The file *<netdb.h>* contains the definitions of the structures returned by the routines and must be included when using any of these routines.

Host name mapping: The Internet name to IP address mapping is done by the routine *gethostbyname()* while the reverse mapping is done by *gethostbyaddr()*. The database file for these calls is */etc/hosts*. The *hostent* structure returned is described below:

```
struct hostent {
    char    *h_name;
    char    **h_aliases;
    int     h_addrtype;
    int     h_length;
    char    **h_addr_list;
}
```

h_name is a pointer to an ASCII string (terminated by a NULL character) that is the official name of the host. *h_aliases* is a pointer to an array of pointers, each of which points to a character string, that is an alias of this particular host. The last entry in the array of pointers is a NULL pointer. This serves to indicate the number of valid alias pointers. *h_addrtype* is always AF_INET, since NS-ARPA/1000 deals only with the Internet domain. *h_length* is always 4. This represents the number of bytes needed to represent a host address in the Internet domain.

h_addr_list is a pointer to an array of pointers, each of which points to a 32 bit IP address. As in *h_aliases*, this array is ended by a NULL pointer to indicate the end of valid pointers to IP addresses. There is a crucial point to note here. Although the declaration indicates that the array of pointers that *h_addr_list* points to is an array of pointers to type *char*, they are really pointers to type *u_long* on the HP1000. The 32 bit IP addresses themselves are guaranteed to be word (16 bit) aligned. On most UNIX machines, this is not significant. However, because the HP1000 is a word addressable machine, special care must be taken to extricate the 32 bit IP address from this structure. As an example, if *h* is the pointer to a hostent structure returned by one of the calls above,then the code in C may be as follows:

```
u_long  ipaddr;

/* First get the array pointer,
 *              h->h_addr_list
 * Then dereference it to get the first char pointer.
 *             *h->h_addr_list
 * Assuming this is not NULL, cast it to a unsigned long
 *   integer, which is what the IP address really is !
 *          (u_long) *h->h_addr_list
 * Then, dereference this pointer to get the 32 bit IP address
 *          * (u_long) *h_addr_list
 */

ipaddr = * (u_long) *h->h_addr_list;
```

Pascal/FORTRAN users will have to resort to similar machinations to obtain the 32 bit IP address. In Pascal, it is as follows:

```
cptr := h^.h_addr_list^; { get the char pointer }
i32ptr := cptr DIV 2 ;   { The IP address is guaranteed to be word aligned }
ipaddr := i32ptr^;       { dereferece pointer }
```

Similar routines exist called *getnetbyname()* and *getnetbyaddr()* to extract information from the networks database file called */etc/networks*. The information is returned in a structure called *netent*, described below:

```
struct netent {
   char    *n_name ;    /* official name of net */
   char   **n_aliases;  /* alias list */
   int      n_addrtype;  /* network address type */
   u_long  n_net;       /* network address */
}
```

The only notable item here is that the *n_net* field is defined as *u_long* on the HP1000 while it is an *int* on the HP9000 machines.

Protocol names: For protocols, *getprotobyname()* and *getprotobynumber()* extract information from the protocols database file called */etc/protocols*. The information is returned in a structure

called *protoent*, described below:

```
struct protoent {
    char  *p_name;      /* official protocol name */
    char  **p_aliases;  /* alias list */
    int   p_proto;      /* protocol number */
};
```

Service names: In the Internet domain, a service (or a server process) is expected to wait on a specific port address and employ a specific communication protocol. The only protocol NS-ARPA/1000 supports currently is TCP. The database file for service name lookups is called */etc/services* and the routines that operate on it are *getservbyname()*, *getservbyport*. The information is returned in a servent structure described below:

```
struct servent {
    char  *s_name;      /* official service name */
    char  **s_aliases;  /* alias list */
    int   s_port;       /* port address */
    char  *s_proto;     /* protocol to use */
};
```

The port address that the server needs to bind itself to could be obtained by using the *getservbyname()* call and extracting the information from the *s_port* field of the *servent* structure that is returned.

There are two restrictions in the use of port addresses. Only a superuser can bind a socket to a port address in the range 1-1023. Also, no two sockets may be bound to the same local port address unless a *setsockopt()* with the SO_REUSEADDR option has been issued to the socket prior to the *bind()* call.

Establishing a connection

Connection establishment in a client/server model is assymmetrical. Both processes need to follow a standard set of conventions before service may be rendered.

Server side: The server process first creates a socket and binds itself to a well known (or at least one that is known to the client) port address. Then, it issues the *listen()* call to enable a queue on the socket for incoming connections. If this were not done, then all connection requests made to the server would be summarily rejected by TCP. Now, the server can wait for connection requests via an *accept()* call or a *select()* call. The *select()* call is covered in a later section on multiplexing. The *accept()* call is defined below:

```
int accept( int sd, struct sockaddr_in *addr, int *addrlen);
```

When an incoming connection request is accepted, a new socket descriptor is returned from the

call. This new socket descriptor is the one that the server should use for subsequent communication with the client. A maximum of *addrlen* bytes of the client's socket address is returned in the ubiquitous *sockaddr_in* structure pointed to by *addr*.

If the socket is in blocking mode, the *accept()* call will not return until a connection is available. There is no way for NS-ARPA/1000 to screen incoming connections. It is entirely upto the server process to consider who the connection is from and to close the connection if it's an undesirable one. Using the *select()* call, the server process can wait on more than one socket for incoming connection requests.

Client side: The client process first creates a socket using the *socket()* call. It is not necessary that it bind itself to a specific port address. Once assured that the server process is listening for connection requests, the client process can initiate a connection using the *connect()* call described below:

```
int connect( int sd, struct sockaddr_in *addr, int addrlen);
```

On input, the *sockaddr_in* structure should contain information about the server socket with whom the client wants to establish communication. Referring to the *sockaddr_in* structure described earlier, the *sin_port* field should contain the port address that the server is listening on. As for the server, this may be hard coded into the program or may be obtained by a call to *getservbyname()*. In the latter case, it is essential that the */etc/services* file on both the client and the server contain commensurate information. The *sin_addr* field should contain the 32 bit IP address of the server machine. Again, this IP address can be obtained via the *gethostbyname()* call or can be hard coded into the program. In addition, there are many utility routines for manipulating dotted decimal IP addresses. They are:

```
struct in_addr   inet_addr(char *c);
char             *inet_ntoa( struct in_addr ipaddr);
u_long           inet_network( char *c);
struct in_addr   inet_makeaddr( u_long net, u_long host);
u_long           inet_netof( struct in_addr ipaddr);
u_long           inet_lnaof( struct in_addr ipaddr);
```

The in_addr structure is defined in <*in.h*> and is the structure to use when dealing with Internet IP addresses. The *inet_addr()* function takes a character string in dotted decimal notation and returns the 32 bit IP address. *inet_ntoa* returns the complement. The other functions manipulate the 32 bit IP address and return some portion of it such as the network portion, the host portion,etc.

The *connect()* call normally blocks until a connection is established or it times out. However, by setting the socket to be non-blocking with a *fcntl()* call, connect will return immediately with a EINPROGRESS error. The client can proceed to do other things before waiting on a *select()* call for the connection to be completely established.

Data Transfer

Once a connection has been established, data can be exchanged between the two processes. The connection is full duplex and hence either side can send or receive at the same time. Calls are available for sending and receiving scalar as well as vectored data. Vectored data operations are also known as scatter/gather operations. In order to send scalar data, the *send()* call is used:

```
int send( int sd, char *buf, int buflen, u_long flags);
```

buf is the character pointer to the beginning of the data that is to be sent. *buflen* is the amount of data, in bytes, to be sent starting from where *buf* points to. *flags* is a 32 bit parameter that does not have any options supported currently. The amount of data (indicated by *buflen*) to be sent is independent of the socket buffer sizes. Each socket has a send buffer and a receive buffer. These represent the maximum amount of data that can be outstanding on the socket. These buffers are intermediate repositories for data whose ultimate source and destination are the user data space on the sending and receiving side respectively. Before a connection can be established, the socket's send and receive buffers can be modified by using the *setsockopt()* call. Once a connection is established, however, the socket's buffer sizes can only be examined with the *getsockopt()* call, and cannot be modified. If the socket is in a blocking mode, the *send()* call will block until all the data can be transferred from the user space into the socket buffer space. By setting the socket in a non-blocking mode , via the *fcntl()* call, the *send()* call will return as soon as it transfers as much data into the socket buffer as is currently possible. Note that simply returning from the send call does not imply that the data has already been tranferred to the receiving side. It is merely in the socket buffer for now and it is the responsibility of the underlying protocol (in this case, TCP) to ensure the smooth transfer of data to the other side of the connection.

Scalar data is received using the *recv* call :

```
int recv( int sd, char *buf, int buflen, u_long flags);
```

buf is the starting user space address where data will be received. *buflen* indicates the maximum amount of data that the process wants to receive right now. The following *flag* is supported currently:

MSG_PEEK

Normally, once data has been transferred to the user space as in the *recv()* call above, the socket's receive buffer is freed up by the amount received so that it can accept more data from the sender. However, setting this flag enables the *recv()* call to transfer data to the user space and yet retain the same data in the socket's receive buffer. Thus, the next *recv()* call would return the same data. The *recv()* function itself returns the amount of data that has been transferred to the user space from a minimum of one byte to a maximum of *buflen* bytes. Note that unlike NetIPC, there is no way to force the *recv()* call to wait for a specific amount of data before returning.

Vectored data is sent and received using the following two calls:

```
int sendmsg( int sd, struct msghdr *msg, u_long flags);
int recvmsg( int sd, struct msghdr *msg, u_long flags);
```

The *msghdr* structure defines the starting addresses of where the data received on the socket will be disbursed.

Blocking / Non-blocking

At any time of its existence, a socket can be set in blocking or non-blocking mode with the *fcntl()* call.

```
u_long fcntl( int sd, int cmd, u_long flags);
```

The *cmd* parameter can be either F_SETFL or F_GETFL. All these constants are defined in *<fcntl.h>*. The only *flags* parameter supported currently is O_NONBLOCK. F_GETFL can be used to obtain the current values of the specified flag bits set in the *flags* parameter. The *fcntl* call itself returns the value of the flag bits in a 32 bit entity. If the option corresponding to the flag bit is currently turned on for the socket, then this bit is set in the return value. The F_SETFL *cmd* is similarly used to control the setting of the socket options. By setting the socket in a non-blocking mode, subsequent BSD socket calls will return a -1 with *errno* set to EAGAIN if the socket cannot service the call immediately. For example, assume the socket is set in a non-blocking mode and the process does a *recv()* call. If there is no data on the socket's receive buffer, then a -1 is returned with *errno* set to EAGAIN indicating that the socket could not service the request right away.

I/O Multiplexing

One another important function provided enables the process to wait for events on multiple sockets. This function is the *select()* call. Thus, if a process has many connections open, it can wait for data on any of these sockets by using the *select()* call. It is used as follows

```
int select( int nfds, fd_set *rmap, fd_set *wmap, fd_set *emap, struct timeval *timeout);
```

fd_set and *timeval* are datatypes defined in <*types.h*>. *fd_set* is a bitmask that represents the socket descriptors that the *select()* call should wait for events on. The following utilities are provided for manipulating the bitmasks:

```
FD_ZERO(fd_set *bitmask)        /* clear the bitmask */
FD_SET (int sd, fd_set *bitmask) /* set the bit for sd in bitmask */
FD_CLR (int sd, fd_set *bitmask) /* clear the bit for sd in bitmask*/
FD_ISSET(int sd, fd_set *bitmask) /* test the bit for sd in bitmask*/
```

These utilities are implemented as macros in C and as procedures for Pascal/FORTRAN. Use of these is highly recommended for manipulating the bitmasks.

Let's step through an example here. *sd1*, *sd2* and *sd3* are three socket descriptors. *sd1* is a socket that is connected to a remote socket and is waiting for data. *sd2* is a socket on which we have initiated a non-blocking *connect()* and want to await confirmation of the connection establishment. *sd3* is a socket on which we want to accept incoming connection requests. So, we want to wait for the following three events :

READABLE signal on socket *sd1*
WRITABLE signal on socket *sd2*
READABLE signal on socket *sd3*

Let *rmap*, *wmap* and *emap* be the bitmask structures for reading, writing and exceptional respectively. They would be declared as:

```
fd_set rmap, wmap, emap
```

First, clear all the bitmaps.

```
FD_ZERO(&rmap);
FD_ZERO(&wmap);
FD_ZERO(&emap);
```

Set the readable signal for *sd1* and *sd3*.

```
FD_SET(sd1,&rmap);
```

```
FD_SET(sd3,&rmap);
```

Set the writeable signal for *sd2*.

```
FD_SET(sd2,&wmap);
```

It is always advisable to set the exceptional signal on all the sockets just to be notified of catastrophic events.

```
FD_SET(sd1,&emap);
FD_SET(sd2,&emap);
FD_SET(sd3,&emap);
```

Now, if the *select()* call is issued, it will wait until one of the events that we have selected on occurs.

```
ns = select(nfds,&rmap,&wmap,&emap,(struct timeval *) NULL)
```

nfds should be set to the maximum socket descriptor value that the select call should check for events on. Thus, in our example, it should be set to the maximum of *sd1, sd2* and *sd3* before the *select()* call. Notice that the *timeval* parameter is set to a NULL pointer. This requests the *select()* call to block until an event has occured. Fields in the *timeval* structure can be set to indicate a maximum time limit that the *select()* call would block while waiting for any of the specified events to occur.

On a successful return, the three bitmasks will be modified to indicate which socket descriptors have events recorded against them. The FD_ISSET call can be used to determine whether a particular socket descriptor was selected.

Signals on an HP-UX system are slightly different in the sense that a process may "catch a signal" in several different ways. With NS-ARPA/1000, the only way to be notified of events is to indicate through the *select()* call above that a process is interested in a particular set of events. Also, on the HP-UX system, the exceptional signal is not set by the socket routines. Instead when a broken connection is detected the HP-UX kernel sends a signal to the user process that causes the process to die if it hasn't set up a signal handler. For NS-ARPA/1000, however, exceptional signals are the only way to detect broken connections from a *select()* call.

Terminating the connection

A connection may be terminated by the *shutdown()* call. Since, TCP connections are full-duplex in nature, *shutdown()* may be used to terminate the receiving and sending sides of the connection independently. Once a socket is shutdown for receive, then it cannot receive any more data from the peer socket. Similarly, if a socket has been shutdown for sends, then it cannot send any more data to the peer. A socket can be completely shutdown by setting *how* to 2 in the *shutdown()* call described below:

```
int shutdown( int sd, int how);
```

With how set to 2, the socket can neither send nor receive data. In addition, the socket descriptor *sd* is rendered invalid and liable to be associated with another socket with a subsequent *socket()* call. On the HP-UX system however, a *shutdown()* call does not release the socket descriptor. Although nothing useful can be done with the socket descriptor once a *shutdown()* with *how* = 2 is done on it, the HP-UX process will still possess a valid descriptor until a *close()* is performed on the socket descriptor.

Irrespective of the difference noted above, both NS-ARPA/1000 and the HP9000 provide the graceful release mechanism when the *shutdown()* call is used on a socket descriptor. This means that TCP will try its best to ensure that all outstanding data in the pipeline between the two peer sockets reach their respective destinations before the socket resources are released. Thus, if a local socket does a *send()* and immediately follows it with a *shutdown()*, the TCP at both the local and remote ends co-operate to deliver all the data to the remote peer before indicating to the remote peer process that the local process has done a *shutdown()* and that there won't be any more data.

Socket options

The *setsockopt()* and *getsockopt())* functions enables setting and examining the socket's characteristics (also known as options) respectively. These options may exist either at the socket level or the protocol level. The functions are called as follows:

```
int setsockopt( int sd, int level, int optname, char *optval, int optlen);
int getsockopt( int sd, int level, int optname, char *optval, int *optlen);
```

Although *optval* is of type *(char *)*, in reality, it is a pointer to a character array that contains the value of the option but is not terminated by the conventional "\0" character that terminates C-style strings. *optval* needs to be appropriately recast (in most cases, to an *int*) in order to obtain the value of the option.

There are two kinds of options: boolean and non-boolean. To set a boolean option, any non-zero value in *optval* will suffice. To examine a boolean option, the *getsockopt()* call returns with a zero value (the convenetional successful return) if the option is set and -1 if it is not set. The most important boolean option is the SO_REUSEADDR option. The use of this option enables a socket to be bound to the same local port address as another socket. However, the underlying protocol (TCP in this case) will ensure that only one of these sockets will be allowed to call *listen()* successfully and this is enforced on a first-come first served basis. TCP will also ensure that before a connection is set up, the quintuple of *<protocol,local port address, local IP address, remote port address, remote IP address>* is unique for each socket on the system.

For example, let *sd1* be a socket that is bound to local port address 10000. The SO_REUSEADDR option must be set individually for sockets *sd2* and *sd3* to be bound to the same local port address, 10000. If *sd1* has already done a *listen()*, then the *listen()* call performed on *sd2* and *sd3* will return unsuccessfully. Now, let's say *sd2* establishes a connection with peer process whose IP address and port address are 15.1.13.208 and 15000 respectively. If, *sd3*, does a *connect()* call specifying the remote IP address as 15.1.13.208 and the remote port address as 15000, then the connect call will return unsuccessfully with *errno* set to EADDRINUSE.

The most important non-boolean options are the SO_SNDBUF and SO_RCVBUF. The use of these options controls the sizes of the send and receive buffer sizes of the socket. Although these buffer sizes may be examined at any point, the *setsockopt()* call to change buffer sizes may only be used prior to establishing a connection on the socket.

Miscellaneous

In addition to the functions described above there are several other calls such as the *ntohs, htons*,etc. the use of which will ensure network portability across different platforms. Also, *getsockname()* and *getpeername()* can be used to obtain the local and peer socket address in the *sockaddr_in* structure .

HP1000 CONSIDERATIONS

The BSD socket interface is supported only on the RTE-A systems and the programs using this interface will have to be CDS. This section describes issues that a programmer has to contend with on the HP1000 platform and the differences between using this interface in a Pascal/FORTRAN environment and the C environment. The BSD socket interface has been designed to conform to HP-UX usage and be as natural as possible for C users. Hence, users of BSD sockets and its associated utilities should be aware of the following points when programming in a non-C environment.

Byte and Word addressing: The HP1000 is a word oriented machine. Most UNIX machines, including the HP9000 series, are byte oriented. An example will illustrate the difference. Consider the following C declarations:

```
char *c;
int  *x;
```

Let us assume that c is pointing to a character stored at address 2300 (decimal). Thus the value of *c* is 2300. Now, let's say we want to store this value in *x* also. One simple way might be,

```
x = (int *) c;
```

The *(int *)* casting operator converts a character pointer **c** to an integer pointer **x**. On UNIX systems, we would expect the value stored in x to be 2300 also. However, on the HP1000, the casting has the effect of storing the word aligned, word address of 2300 in *x*. In simple terms, the value stored in x is (2300 DIV 2), since a word on the HP1000 is 2 bytes long. Thus, x would

contain the value, 1150. Even if *c* were to contain the value 2301, the casting operation would render *x* to contain 1150 (because of word alignment !). For C programmers, the casting operator makes the transformation from byte to word addresses and vice-versa extremely simple and transparent. Pascal/FORTRAN users have to use the following functions in order to obtain the same effect.

AddressOf(*parm* : int) , which returns the word address of parm.
ByteAdrOf(*parm,offset* : int) , which returns the byte address of parm modified by adding the value of offset to it.

The relevance of all this is that in a C environment, the equivalent to passing a parameter "by reference" is to pass the address of the parameter. Since several BSD socket functions expect to receive pointers and were implemented trying to maintain the portability of C programs, Pascal/FORTRAN users will have to make use of the above mentioned functions in order to generate the appropriate pointers. In addition, programmers have to be extra careful in passing the right kind of pointer to the BSD socket library routines, viz. byte or word pointers. In general, when a parameter is typed as *(char *)* in the C declaration of a function, then Pascal/FORTRAN users must use the *ByteAdrOf()* function. Other pointer types would use the *AddressOf()* function.

Header files: All the Berkeley socket type declarations and constant definitions are provided in separate header (also known as include files) files for Pascal, FORTRAN and C users. The table below lists the include files for the different languages. The contents of the C header files mirror that found on the HP-UX systems. Note that Pascal needs an additional include file, *ext_calls.pasi*, for the external function declarations because of the Pascal syntax.

C	Pascal	FORTRAN
types.h	socket.pasi	socket.ftni
socket.h	ext_calls.pasi	
fcntl.h		
in.h		
netdb.h		

All the Berkeley socket structures in Pascal are declared as variant records. One of the variant records is a simple 16 bit type. This comes in handy when obtaining the word or character address of the structure itself. This is necessary since the *AddressOf()* function is declared as accepting only an *int* parameter. For example, consider the *sockaddr_in* structure:

```
TYPE
  sockaddr_in = RECORD
    CASE INTEGER OF
      1: (int1        : int);
      2: (sin_family  : int;
          sin_port    : int;
          sin_addr    : in_addr;
          sin_zero    : PACKED ARRAY [1..8] of CHAR);
    END;
```

For getting the word address of the structure, the following function call is used. Let *saddr* be of type *sockaddr_in*.

```
AddressOf (saddr.int1);
```

errno and errno return values: *errno* is a global variable that is declared in the C library. It is referenced by including the header file <errno.h>. On UNIX/C systems, any BSD socket system call that returns an error puts the value of the error into *errno* and programs generally check the value of this variable when trying to determine the type of error encountered. For NS-ARPA/1000, the symbolic *errno* values returned by the BSD socket routines conform to those defined in the include file, <errno.h>, on the HP-UX systems.

String pointers: String pointers in C are pointers typed as (char *) which point to a character string ended by a NULL character , which is "\0". Pascal/FORTRAN users must note that C-style strings are different in character (pun intended) than the ones in their respective languages. BSD socket routines on the HP1000 expect and return character pointers to C-style strings only, viz. character arrays that are terminated by a "\0".

int / longint / u_long: The size of *int* on the HP1000 is 16 bits, while on the HP9000 systems, it is 32 bits. Parameters that are typed as *int* on the HP9000 systems, but which need to be 32 bits long, such as the flags parameter on some BSD socket routines, are typed as *longint* or *u_long*.

malloc / free : Several of the BSD socket utility routines on the HP1000, such as *gethostbyname()*, *gethostbyaddr()*, etc. return pointers to structures that are dynamically allocated using a C library call, *malloc()*. In order to reclaim this space, use the *free()* call in the C library.

SUMMARY

The Berkeley socket interface is provided on the HP1000 as part of NS-ARPA/1000 at release 5.24. It is intended to be functionally equivalent to the one offered on the HP-UX platform release 8.0. By providing this functionality, network portability of programs across the different HP platforms in now possible. In addition, use of this standardised interface promotes intervendor connectivity to other systems using the same model for network interprocess communication.

REFERENCES

1. *UNIX Network Programming*, W. Richard Stevens
2. *4.3 BSD UNIX Operating System*, Leffler et al.
3. *An Introductory 4.3 BSD Interprocess Communication Tutorial*, Stuart Sechrest
4. *An Advanced 4.3BSD Interprocess Communication Tutorial*, Leffler, Fabry, Joy, Lapsley, Miller, Torek

TITLE: Distributed Computing GUI's and the OSF/MOTIF

AUTHOR: Mark Brown

Workstation Systems Group

SAS Campus Drive

Cary, NC 27512

(919) 677-8000

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2001

Paper#: 2003

Identifying CIM Opportunities Using Structured Analysis Models.

Wayne R. Asp
Hewlett-Packard Company
2025 West Larpenteur Avenue
St. Paul, Minnesota 55113
(612) 641-9601

Computer Integrated Manufacturing (CIM) is a term which has been grossly overused within the manufacturing industries to generally indicate the application of information technologies to some manufacturing problem(s). In most instances, these CIM automation opportunities were evolved into point solutions, or islands of automation. These islands were, in many cases, totally ignorant of other islands of automation which existed within the same manufacturing facility -- sometimes within mere yards of one another. At best, these islands were able to share only the most rudimentary information. The islands were designed independently, implemented independently, operated independently, and unfortunately, processed their information independently. Most of these islands of automation have paid for themselves over and over again, since they were installed in manufacturing areas which were ripe for automation.

Now we are in the 1990's. Most of those high payback CIM automation opportunities have already been addressed with many islands of automation. The next obvious step is to tie the islands of automation together. But why? What will tying the islands together do for the manufacturing processes? How will it affect product quality? Profits? What should be tied together first?

Of course. we should never automate just for the sake of automating. The application of information technology should always be closely and clearly tied to the goals, objectives, and plans of the organization. If not, why do it in the first place? The direction of the organization must dictate the identification and prioritization of any CIM opportunities, and any subsequent information technology solutions.

An Enterprise Wide CIM Framework

The best method to achieve this high level of CIM integration is to develop an enterprise wide CIM framework. This framework serves two purposes. First, it provides the framework, or blueprint, across the entire manufacturing enterprise for current and future technology applications. This is important in that a comprehensive plan must be developed to integrate not only current islands of automation, but also how future products, manufacturing systems, and technologies will interact within the framework. Second, the framework must correlate directly to the business practices and objectives of the organization. As such, the framework must be flexible and easily adapted to the ever changing business environment.

When setting long range CIM goals, strategies, and planning implementations, it is often very difficult to identify, much less implement, a workable enterprise wide CIM framework. However without such a framework, any solution implementation continues to be an island of automation, unable to tie directly into the CIM infrastructure and ultimately to the organization's goals and objectives.

It often puzzles me why an organization would forge ahead with some CIM automation project not directly tied to their long range goals and objectives. But it has happened time and time again. Perhaps it is that the planning process in building this type of framework is too painful or time consuming. However, thorough planning cannot be effectively accomplished after the fact. Perhaps it is that the planning process is too expensive. However it is generally more expensive to retrofit something after it has been built poorly. There is an aphorism which states that "Plans may be useless, but the planning process is always indispensable". Planning is indeed the key to long term success.

Using Structured Analysis

Structured Analysis (SA) techniques can be used successfully to build and create an abstract representation of the CIM framework. This representation can then be manipulated and expanded to include existing islands of automation, and proposed information technology solutions as the business goals and objectives change.

SA methodologies allow one to methodically decompose a system (in this case, a manufacturing system) into many less complex component parts. Because these component parts are easier for the human to comprehend, SA is ideal for describing most complex manufacturing systems. One can easily examine the descriptions of the component parts to quickly ascertain the effects of proposed changes within the system(s). The collection of these descriptions is called the model, and consists of both textual and graphical representations of the part, or activities within the system.

The easiest way to understand the role which SA techniques can play within a manufacturing organization is to draw upon the analogy of constructing an office building. A construction company would never dream of constructing a building until the architect provided finalized plans and drawings. Yet, in the business world, material handling systems, manufacturing systems, information systems, etc. are regularly constructed and installed with the barest minimum of in depth planning. In fact, many new installations are designed not on the drawing board, but in real life trials and failures.

There are many SA techniques being practiced today. Among them are Structured Analysis Design Technique (SADT), IDEF0, Yourdon Structured Design, Hierarchical Process Modeling (HPM) and many, many others. Each technique has its own strengths, weaknesses, and followers. Since the purpose of this paper is to describe the benefits of using SA, the analysis of the applicability of a given technique for a given purpose is left as an exercise for the reader. HPM will be used within this paper for purposes of discussion.

A major contribution of SA technique applications is their ability to build consensus within the organization regarding the actions or proposed actions of the model (system) under study. Most techniques use an iterative process. First, initial data is gathered regarding plans and desires. An initial model is then drafted and reviewed by a core committee of previously trained personnel. Modifications are then made to the model and a new version is then published to a wider review committee. This build/review/modify/publish cycle continues until all committee members agree that the model is accurate. A final model is then published and distributed. Although many opportunities might exist within the manufacturing organization for automation, sometimes we are more limited by people's attitudes than by opportunities. SA model consensus building helps to address and change attitudes toward the implications of a enterprise wide CIM framework.

HPM - A Structured Analysis Methodology

Hierarchical Process Modeling, or HPM for short, is a structured analysis methodology which was developed by Hewlett Packard and is used both internally within the company and with HP customers. HPM provides a graphical and textual notation for breaking down and analyzing both simple and complex real world situations. An HPM model can easily be read and understood by anyone, with a minimum of training. This means that an HPM model can serve as the focal point for discussions as situations change or support systems are put into place.

HPM was developed by the HP Corporate Manufacturing group to help document HP manufacturing techniques and help leverage technologies throughout the company. In the mid 1980's, HP investigated several structured analysis techniques and packages, but found none that were flexible or thorough enough to meet the requirements. Combining several existing techniques, such as DeMarko Structured Analysis and IDEF0, along with other ideas, including TQC methodologies, HPM was developed by HP. HPM is still unique in the marketplace, gathering the best of structured analysis techniques within one methodology.

HPM is a formal specification language and methodology for describing systems in terms of integrated collections of processes. Processes are defined in terms of other processes hierarchically. The methodology provides a structure which allows rapid breakdown of a system into its component parts, each of which can in turn be analyzed and broken down further, if required. It is these component parts (processes) which make the model extremely readable and easy to comprehend at any level. The processes represent the fundamental activities within the model. They are a controlled collection of actions which transform inputs into outputs, using resources. Processes are related to other processes in three ways: by Control, by Exchange, and by sharing Resource(s). One process may: command another process to perform some action or assist some other process in performing its action (a Control relationship); produce something which another process consumes (an Exchange relationship); or provide some resource or service, such as access to a database or machine, which another process requires (a Resource relationship). Ports are used to define the interfaces of processes. Flows transfer entities, such as materials and information, between ports.

What is an HPM Model?

An HPM model can provide the in depth plans necessary to support both new and existing manufacturing functions. The model can be tailored to any level of detail, from general overview to extreme detail.

Architectural plans are structured much the same as an HPM model. Many pages of drawings are provided, but each one serves to identify different areas or features. The first page of the plans show the overall building, the facade, the roof lines, etc. There are no design details here, only a flavor of what the building will look like. This drawing is of use to anybody who wants to know what the building will look like.

The next several pages of the plans show the floor layouts floor by floor. This includes where the various rooms are located, how large they are, where the mechanical and electrical closets are located, etc. These pages are of interest to prospective tenants, and also the construction workers who must install the walls and doorways.

Following the floor layouts are the detailed room plans, one for each room. These plans show where electrical and mechanical fixtures should be placed, what special features need to be built or installed (like a water fountain or planter), and the finishing details of the room, like trim work. The room plans are used by the tenants for specifying special finish details, and by the construction workers who will complete the room.

The last part of an architectural plan set are the mechanical and electrical plans. These plans rough out the heating and air conditioning duct work, the heating plant locations, and electrical service locations and sizing. Generally, these plans do not contain all the detail necessary to complete the installation. Some of the extreme detail work is planned and executed by the construction workers themselves, working within the specification given in the plans.

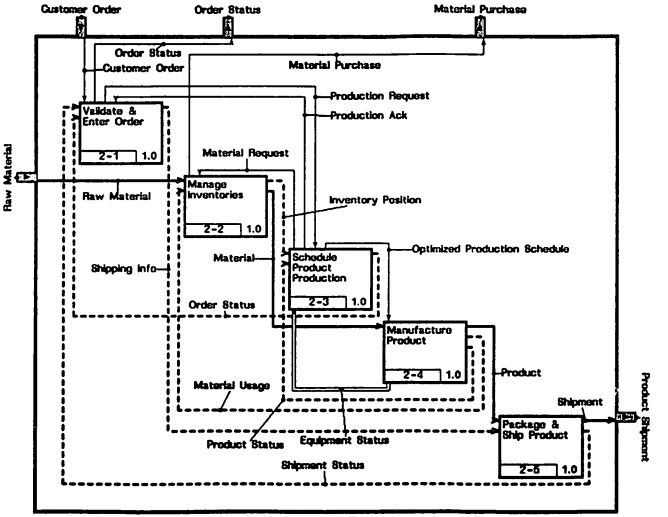
An HPM model is structured much like an architectural plan. The most basic component of a model is called a process. A process performs some activity. A process takes inputs, processes them, and creates some outputs. A process can contain other processes which define how it works, much the same as the room plans, taken all together, make up a floor plan, and the floor plans taken all together make up the building plan. It is processes that make the model extremely readable and easy to comprehend at any level. The processes represent fundamental actions within the model.

Synonymous with the mechanical/electrical plans are HPM flows, which model how entities (like materials, orders, products, information, etc) move between processes. One process may command another process to perform some action. This is called a control flow. A process may produce some output which another process consumes as input. This is an exchange flow. Or a process may provide a service, like a tool crib or a database. This is a resource flow. HPM flows then can transfer control, exchange, and resource entities between processes.

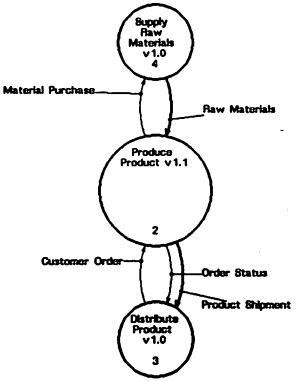
Pages 5 and 6 present the highest level of an HPM factory model.

The final part of an HPM model is the term glossary. This is much like the material list for a new building. It describes all the processes, flows, and other constructs used within the HPM model.

Identifying CIM Opportunities
Using Structured Analysis Models.
2003-5



Identifying CIM Opportunities
Using Structured Analysis Models.
2003-6



Context of Produce Product v1.1

Produce Product

Produce Product represents the high level viewpoint of fulfilling an order within the WIDGET manufacturing facility. The order is actually received by the manufacturing facility via FAX machine, the product is scheduled, manufactured, and shipped to the distributor. Resources used to manufacture the product are explored in more detail within the *Manufacture Product* process.

Produce Product v1

Two specific connections to functions outside this model's scope are shown.

The distributor sends an order to the manufacturing facility, and receives the product once is has been produced and shipped. The distributor is also sent an Order Status when the order is received by the WIDGET facility.

The raw material supplier receives a materials purchase request from the inventory management group based upon current material requests and usages. The supplier then ships the materials directly to the manufacturing facility warehouse.

Produce Product v1.1

Orders are received by the manufacturing facility and entered into the system. Production is then informed of the order, and adds it to the production schedule, taking into account raw material availability and equipment status. The production schedule is sent to manufacturing, along with a materials request to the allocate the appropriate raw materials to manufacturing at the appropriate time. The WIDGET is then manufactured and the product is moved to the packaging area for packing. A shipment is then made to the distributor based on the shipping information from the original order. The order status is then updated with the new shipment information.

Without an architectural plan, a building site would be total chaos. 2x4's might be used were 2x6's should be for structural integrity. Roof supports might be constructed inadequately. Electrical outlets and heating ducts are retrofitted, rather than built in. The electrical service is underrated and must be beefed up later. A reputable contractor would never even dream of building anything more complex than a shed without a complete set of building plans and drawings.

Yet within the business world, literally billions of dollars are spent annually for business ventures without more than a rough idea of what the facade and the roof lines look like. Time and time again, projects grossly exceed their budgets and reduce their scopes because inadequate planning was done up front. Many times, businesses simply do not want to spend the money necessary up front to provide a complete plan. It matters not. They will spend the money anyway after the project is "complete" to clean things up. Another common downfall is that "we have done that before" so "we do not have to plan so much this time". However, oftentimes the documentation for the previous project is missing, inadequate, or out of date. HPM can help bridge the gap in all these instances by providing an analysis framework within which the planning can successfully be accomplished and documented.

Constructing a CIM framework

A Structured Analysis methodology such as HPM can be used as a technique to build the enterprise wide CIM framework. Through examination of this framework, the organization can efficiently and effectively analyze the effects of future information automation projects and identify CIM opportunities across the entire manufacturing enterprise. As such, the framework serves as a highly effective communications tool, serving not to directly identify and solve problems, but providing a focal point for discussions and decisions pertaining to CIM automation issues.

As an example, walk into any manufacturing facility and ask for complete documentation on their manufacturing process. In most instances, you will be provided with many binders of printed, or formal documentation. However, if you were to walk out onto the manufacturing floor and talk to the supervisor, you would find many informal mechanism at work which are not documented. This presents two problems; first, in all likelihood these informal mechanisms are not very efficient. How can they be improved if they are not documented? SA techniques can be used to document them. Second, if they are not documented, how can any CIM implementation or framework possibly succeed to the level expected. Again, SA techniques can be used to capture the informal mechanism at work within the manufacturing processes.

The Four Step Approach

Since each manufacturing enterprise is different, it is next to impossible to make specific recommendations within this context. However, there is a basic four step approach to building the CIM framework which can be presented. Naturally, these steps can be combined, modified, or customized based upon the needs and timeframes of the manufacturing organization.

The four step approach to putting together a CIM framework:

* Benefits Analysis
* Business Modeling
* Solution Modeling
* Implementation

This four step approach reflects HP's broad experience in tailoring advanced technology to the individual needs of an organization. This has proven successful for a wide variety of industries and applications.

Benefits Analysis

Benefits Analysis is a technique used to identify and prioritize high payoff opportunities that are strategic to your organization. The process measures the benefits of implementing information technology in these areas. Interviews are conducted with key personnel in the organization to identify the "critical success factors" facing the enterprise. These factors are then discussed with focus groups made up of the personnel within the organization who really understand the area of concern. As the process continues, people within the organization understand the obstacles that stand in the way of achieving these critical success factors and what the enterprise must do to overcome these obstacles.

The outcome of the Benefits Analysis is sometimes referred to as breakthrough objectives. The objectives, or "critical success factors" provide a very high level map for the next steps in the approach. This (hopefully) avoids the pitfall of addressing one aspect of the process, when the real obstacles exist elsewhere within the process.

Business Modeling

The Benefits Analysis step identified the strategic areas to implement information technology. The Business Modeling step documents and simplifies current business practices, both formal and informal, to provide the most functional framework for information technology application.

Business Modeling is an iterative process. First initial data is gathered regarding business and manufacturing practices, desires, and system requirements. An initial model is then drafted and reviewed by a core committee of previously trained personnel in the enterprise. Modifications are made to the model and a new version is then published to a wider review committee. This review/modify/publish cycle continues until all committee members agree that the model is accurate. A final model is then published and becomes the basis for all future requirements definitions and activities.

Using a SA technique such as HPM, which is predominantly graphical in nature, provides a description of the plants processes, which is easy to understand by people not normally familiar with structured analysis processes. It functions as the tool to bring all departments into fundamental agreement with the overall CIM framework.

In many instances, a significant portion of the current business practices and requirements may have already been done and documented using other techniques. This existing documentation can generally be used and leveraged within the Business Modeling step.

During the Business Modeling step, the objectives can be accomplished in two phases. Phase I will create a high level business practices model which will document current manufacturing processes and practices. This high level model, combined with the strategic objectives from the Benefits Analysis, will provide strategic guidance for possible information technology implementations.

Phase II of Business Modeling will simplify and add a deeper level of detail to specifically targeted processes within the Phase I High Level model. This model provides a more detailed understanding of the processes that make up the areas identified and forms the foundation of any successful project. It describes WHAT a particular system must do to address specific functionality. This phase is typically executed over a two to four month timeframe, and requires much more interaction and time commitment from key personnel.

Solution Modeling

Once the Business Modeling step has provided the "road map" toward a successful CIM framework and technology project, the elements of integrated software, hardware, and "peopleware" solutions can be identified. The Business model(s) will distinctly show the "tie in" requirements necessary to integrate a particular technology solution in the overall manufacturing framework.

Essentially, one can visually this process the same as using tracing paper. The original model serves as the template to match. Each solution has it's own trace, and is sequentially layered over the original template. The "goodness of fit" of the trace determines the applicability of the solution proposed to the model. This process helps to quickly pare down the number of solutions under consideration to those which can match easily with the requirements documented by the SA model.

While the actual process of identifying manufacturing solutions is certainly more difficult than using tracing paper, the model framework does help to focus attention on the specific areas of concern in a very structured and logical way that is easy to comprehend and evaluate.

Implementation

A formal project plan document is generally produced during the Implementation step which incorporates the information developed during the first three steps outlined above along with the specifics for implementing all elements of the framework. The project plan is the controlling document which defines the technical and managerial project functions, activities and tasks necessary to satisfy the requirements of the project. This plan requires detailed involvement in the development, review, and approval process. The project plan is a living document that requires modification as the project itself changes, but at the same time is closely monitored and reviewed for adherence to targeted objectives.

After completion of the project plan, the projects and solutions identified begin to follow the more traditional project implementation phases. Typically, these phases are practiced by the engineering and/or MIS organizations, and generally consist of as many as eight sub-phases.

The eight sub-phases that typically occur during project implementation are as follows:

1. Detailed Design
2. System Development
3. Integration and Test
4. Documentation
5. Training
6. Site Installation
7. Warranty
8. Support

Most certainly, the Implementation step is where the benefits of building the enterprise wide CIM framework are realized. The implementation must insure that each major functional area of the Business model and Solutions model are properly addressed. This is best accomplished by contracting professional project management services, effectively managing subcontractors, and providing appropriate review meetings of the overall project status at appropriate times.

Conclusion

It has been our experience that the Structured Analysis techniques and four step approach presented herein provide a workable method for developing an enterprise wide CIM framework. This is the key to successful integrated CIM projects. Only by utilizing an overall methodology such as this for implementing and achieving long term goals and objectives can an organization effectively combine the islands of automation into a single, workable manufacturing entity.

No manufacturing process is stable. Continual improvements and changes are constantly being made. SA models, therefore, become living entities. They are constantly being improved and provide continual feedback into the methodology for continual improvements within the manufacturing processes and CIM framework. Thus, CIM opportunities are constantly being identified, evaluated, and implemented as the manufacturing processes change, tying directly back into the CIM infrastructure and framework.

* The author wishes to thank Dan Lee, Allen Otte and George Subiti for their contributions to this paper.

Paper #2004
Beyond Interprocess Communications: Strategies for Linking MPE XL and HP-UX Applications

By

Frank Leong, Jr.

Software Engineering Systems Division
Hewlett-Packard Company
1266 Kifer road
Sunnyvale, California 94086
(408-746-5368)

Abstract

Increasingly, today's corporate and manufacturing computing environments require the integration of multiple applications running on heterogeneous platforms in a distributed environment. Such an integration problem is complicated by the existence of previously written or purchased software which cannot be easily modified. To fully address application integration, a strategy for linking new and existing applications is needed.

This paper will first examine the current strategies available for linking applications on MPE XL and HP-UX systems, including file transfer, program to program communications, remote process control, and remote procedure calls. Next, a new strategy that goes beyond the breadth and depth of traditional interprocess communications elements is examined. This new strategy involves the use of HP Software Integration Sockets.

1.0 Introduction

Information access, once considered a luxury, is now a key factor in the success of any company. As the size of a company grows, so does the quantity, complexity, and importance of the information flow. The ability to manage the quality and timeliness of intra-company and inter-company information access ultimately translates directly into the ability to meet customer needs and therefore greatly affects the bottom line.

At the center of information access are the computer and communications technologies that drive it. During this past decade, explosive growth has been seen in the use of both Local Area and Wide Area Networks (LANs and WANs). No longer are computers providing single point solutions within "islands of automation". Instead, computers are linked together providing the POTENTIAL for global access and control of information. Yet, in spite of the advances made in networking, this potential remains largely untapped by many companies. Why is this so?

Much of the problem can be attributed to issues faced when trying to integrate software applications. These applications can be viewed as the embodiment of the information we are trying to access. The success or failure in dealing with application integration issues will decide the degree to which global information access can be achieved. This paper explores some of those issues, especially as it pertains to integrating applications on HP-UX and MPE XL systems. Past and present strategies for application integration are illustrated, and integration technologies reviewed. Finally, a solution involving the combined use of good methodologies and a new tool is proposed.

2.0 Integration Strategies Vs. Technologies

In this paper, the term *integration strategy* will be used to refer to a particular plan for attacking the software integration problem. This can involve the use of one or more *integration technologies*, which are implementations of software and typically available as products. Integration strategies can also involve stated methodologies, best practices, and company standards in developing integrated software systems.

3.0 A Review of Integration Strategies

To gain a better understanding of the integration dilemma, we need to examine its origin and evolution. The following scenario is typical of what companies have endured through in computerizing their information systems. It might not apply to your particular situation, especially if your company is relatively new and has a much shorter history of application development, but it does serve to expose some common issues in application integration.

3.1 A Distribution System Scenario

The Ace Widget Company has been producing world class widgets for over 25 years. A key to their success has been the ability to respond quickly to customer orders. Not only are their products superior, but Ace can fill an order faster than any

of their competitors. Many people think Ace Widget's success has been due to innovative designs and a manufacturing capability second to none. But George Howard, the President of Ace, knows the real reason for their success can be attributed to a distribution system comprised of order-entry, shipping, and inventory subsystems. Things were not always this smooth, however, and a lot of obstacles had to be overcome by the MIS department to reach this pinnacle. John Kramer, head of MIS, remembers those dark days.

3.1.1 Stage 1: One Computer, One Company

It all began in 1974. John, fresh out of college, was assigned to tackle the problem of tying the order-entry and shipping systems back into the inventory control system resident on the company's mainframe. John worked with a team of analysts and software developers. The solution they came up with was hardly innovative, but it was adequate and in keeping with other information system projects completed in the past. Ace had only one computer, an old IBM workhorse. But that seemed adequate for this small, but growing company. All operations were located in one large building. The software John's team developed ran on the company's sole computer. It relied on two principle kinds of input --- punched cards and magtape. As orders came in, they were manually entered on magtape or punched cards, one transaction at a time. These inputs were then hand delivered and fed into the mainframe, which in turn generated a shipping list and tracked the order. Once the shipping department successfully shipped the product, another manual entry occurred on punched cards or magtape. Finally, the company's computer was updated with these hand delivered inputs to reflect the shipped products.

3.1.2 Stage 2: Islands of Automation

By 1978, Ace had outgrown its single computer approach. Customers were beginning to complain about the long lead time between ordering the product they wanted and the time it took to actually receive the product. Even worse, some orders were simply lost --- the result of mangled cards or bad spots on tape. None of this escaped the attention of George, who promptly sent a memo to the head of MIS urging him to do something. Again, John was assigned to the project. His recommendation was to distribute the computing among 3 systems. Hewlett Packard, a pioneer of distributed computing systems, was the logical choice. Two new HP3000 minicomputers would be used: one dedicated to order entry and another to shipping. (See Figure 1.) The company's central computer would still handle inventory, but the new minicomputers would off-load much of the record validation and other preprocessing. The minicomputers were tied to the mainframe using modem lines. At the end of each day, the order entry computer uploaded transactions to the mainframe. Once the orders were processed, the shipping system received a download of transactions from the central computer. The shipping system also uploaded shipping status to the central computer. During the day, the three computers functioned independently. The order entry manager and the shipping department manager were ecstatic over the new system because of its ability to produce departmental reports and provide information on demand.

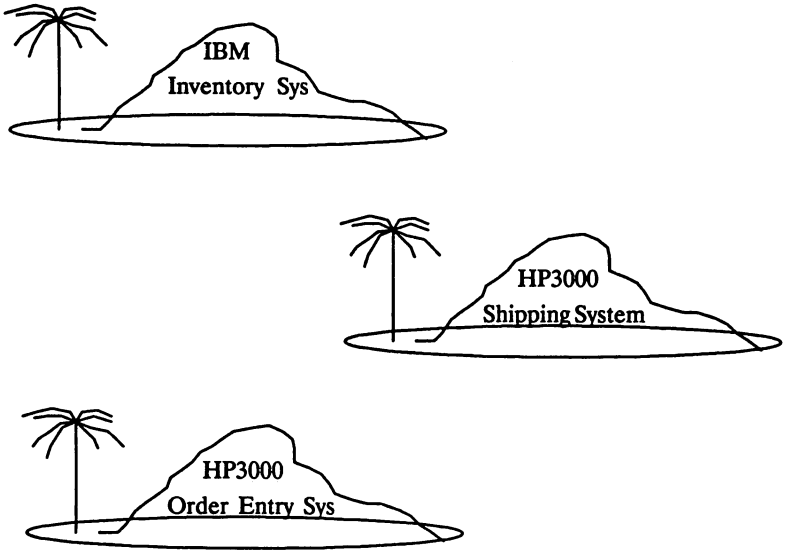


Figure 1: Islands of Automation

3.1.3 Stage 3: Networks Have Arrived: No Computer is an Island

By 1983, Ace had expanded its operations to include 5 sites scattered throughout the country. The idea of departmental computing had propagated throughout the company and there were at least two dozen such computer systems. The strategy of each department having their own computer and using nightly modem uploads and downloads to synchronize and share data had worked well. But now, everyone on the information chain from top management, to department managers, and even operators, wanted consolidated information to be more accessible and on a more timely basis. Nightly updates were no longer sufficient to keep pace with the fleet footed competition. Customers inquiring about their orders wanted up-to-date status. John now managed a project to provide a networking backbone for the company's two dozen computers. They standardized on an IEEE802.3 network, with bridges and routers to gain access to LANs in other cities. Now, instead of one day turn-around times, information could be shared and accessed in a matter of seconds and minutes.

3.1.4 Stage 4: Honeymoon's Over, Where's the Info?

It was now 1988 and five years had elapsed since Ace first installed its computer network. Ace was now a multi-national corporation with hundreds of computers, including mainframes, minicomputers, workstations and PCs. Although networking had

provided such innovations as e-mail, network file access and file transfer, and graphical user interfaces like the X Window System, George's dream of global information utopia had not been realized. After spending a fortune on building and maintaining a sophisticated networking system, the distribution system which had begun years ago was still having difficulties expanding with the company. George knew that without a responsive distribution system to keep his customers happy, his company's dominance in the marketplace would eventually evaporate. Acknowledging this possibility, George issued an ultimatum to the MIS department. John was the logical choice to head up a task force to investigate what went wrong. He had set up the distribution system 10 years ago, but had since moved on to managing the networks group. He was eager to find out what had gone awry with his pet project.

John launched his investigation by interrogating employees from the order entry and shipping departments. It was clear their workload and sophistication had increased substantially and the system they were using had not kept pace with their escalating needs. Shipping had updated their single HP3000 with a distributed system involving several HP9000 workstations running HP-UX. (See figure 2.) Barcode scanners, weighing stations, and label printers were connected up to these workstations. As boxes of widgets came through the shipping department, they were identified using the barcode scanners. A workstation would then show the parcel's shipping information on an X Window display. The weight of the parcel was automatically recorded and its postage computed. An operator viewing the shipping information could override any of the defaults shown. Once correct, a shipping label and postage were generated.

Next, John talked with the MIS group in charge of maintaining the distribution system. In spite of all this new hardware, John found very little had been done to integrate the order entry, shipping, and inventory control subsystems together beyond what he had done 10 years ago! They were still using files to upload and download information, only now the updates occurred twice a day instead of once. Part of the problem was attributed to the complexity of maintaining existing software. Although source code was available, the original developers had left long ago. No one wanted to touch that code for fear it would bring everything to a grinding halt.

John's recommendation was to completely revamp the system. He knew this would be disruptive, but he saw little choice. The team assigned to perform the renovation proceeded with caution, trying not to change existing software unless it was absolutely deemed necessary. Various network access software, such as network interprocess communications, remote procedure calls, network file access and transfer, were examined and used where appropriate. After an exhaustive two year effort, the renovation was complete. John was congratulated on his work and was ultimately promoted to head up all of MIS.

(Note: all names used in this story are strictly fictional. Only the resulting lessons are real.)

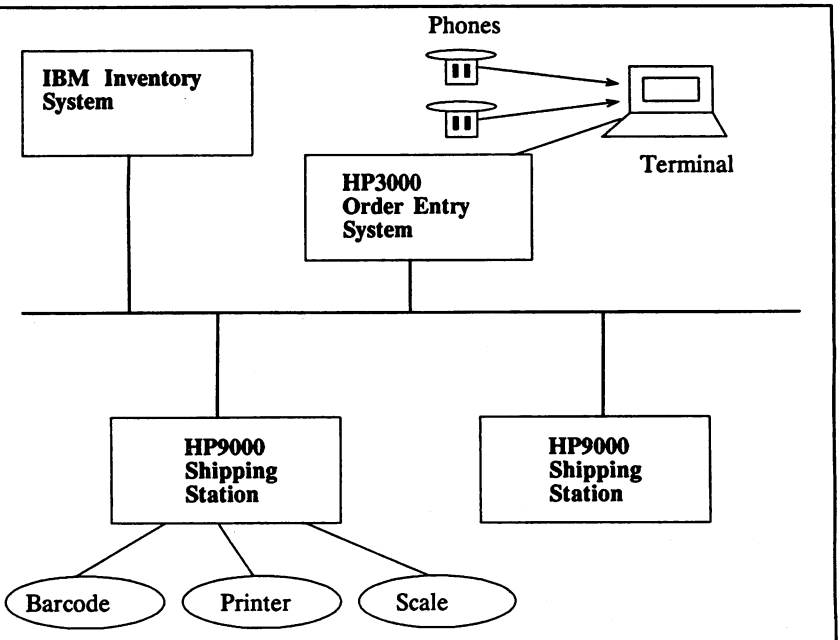


Figure 2: A Distributed System

3.2 Issues in Integrating Application Software

Some of the problems encountered by the Ace Widgets Company might sound more than vaguely familiar. Although this scenario involved an MIS system, it could easily have involved software integration in manufacturing, engineering, or other areas. Let us examine some of these issues:

3.2.1 Integration and Maintenance of Existing Software

When a typewriter or other piece of business equipment becomes obsolete and difficult to maintain, it can simply be replaced with new capital equipment. This is true even for most computer hardware. But with existing software, it might not be desirable or even possible to replace it with new software. Software is more evolutionary than revolutionary in nature. A piece of software placed in service is likely to remain in operation, untouched, or evolve through incremental changes. As in the case of Ace Widgets, a company's operations might be so highly dependent on a piece of software that it can ill afford to change or replace it. The cost of retraining personnel, along with the risk of disrupting operations, makes companies very wary of change even when much superior software technology is available.

Rather than change or replace software, a decision is usually made to build systems around existing software components. Now the problem becomes one of integrating existing applications with newly written software. To complicate matters, the original authors of the current software might have left the company long ago and the new staff might be very resistant to making changes in the old code in order to integrate new software. If the existing software had been developed by an outside third party, source code might not even be available and the third party might no longer be in business to make necessary changes . A company might overcome these obstacles and have some success in developing an "integrated" system. However, as in the case of Ace Widgets, this could result in a poorly or partially integrated system. Despite the new technologies at Ace's disposal, their Stage 3 solution amounted to nothing more than simple, infrequent file transfer. It did not take advantage of the potential provided by the new technologies because the staff was reluctant to take on the task of revamping portions of the distribution system and learning new technologies.

3.2.2 Obtaining and Maintaining Expertise

Expertise is required to maintain software as well as track ever changing software technologies. The former determines to what extent a company's base of existing software can evolve. The latter provides the basis for incorporating new software technologies and methodologies. Recruiting and keeping a qualified staff to deal with software integration problems poses a challenge. In some cases, you might find a staff exuberant about new technology but shuns even the thought of dealing with old application software, also known as *legacy software*. In other cases, the staff might be comprised of people who were the original developers of a software application and are reluctant to change because they are unaware of or uncomfortable with new technologies.

To lessen the impact of inevitable personnel changes, companies must find ways to maintain expertise, and to acquire new expertise when new technology arises. Later in this paper, we will see how software tools, which encapsulate new integration technology and promote the use of good methodologies, can be used to solve part of the problem.

3.2.3 Changing Requirements and Growth

Application software is usually intimately tied to processes --- the way a company does business. In the Ace Widgets example, software for the distribution system embodied the processes used in the physical distribution system. Indeed, as a company's operations change and grow, so too must the underlying application software systems that support it. The inability for application software to evolve can seriously jeopardize the expansion of a company. Ace's distribution system could not keep pace with the increased demands placed upon it by a growing customer base. The obstacle was inflexible legacy software components never designed for interoperability with new software components.

Today, the availability of networking technology makes it attractive for companies to adopt a strategy of incremental growth using a distributed architecture. Newly developed applications can be distributed anywhere on the network, and new computer nodes can be added whenever a company exceeds its computing capacity. Ac-

companying this incremental growth strategy should be an equally important software integration strategy. Without such a gameplan, incremental growth can go out of control, hampering the flow of information and the ultimate objective of global information access.

3.2.4 Custom Solutions Vs. Standard Products

Custom solutions for software application integration can be a mixed blessing. On one hand, you get exactly what you want. On the other hand, you might not really know what you want. It could take several iterations of trial and error before you converge on the right integration strategy for your company. It takes a keen understanding of your particular needs, both current and future, and an understanding of technology and its application to those needs.

Custom solutions require significant investments of time and resources. (Ace Widgets took two years to come up with an integrated software solution.) Not only are there up front costs, but ongoing support costs. "Is what we developed adequate for future needs?" "Is there adequate documentation?" "Who is supporting the integration software?" "Will it break with a new operating system release?" These are just some of the questions company's must face when dealing with custom solutions.

The alternative to building your own solution is using standard products. Unfortunately, few standard products exist for integrating software applications. Those that do exist do not address all the issues associated with integrating software. Given these realities, what is the right solution? The next section of this paper will look at some emerging technologies that show promise.

3.2.5 Technology Alone Is Not Sufficient

Perhaps someday, linking together software applications will be as easy as linking together computer hardware in a LAN or WAN environment. Software will have standard interfaces just as hardware currently does with ICs, backplanes, and network access. Software purchased from one vendor will be able to easily be integrated with software from another vendor. Software integration tools and standards will exist for developers wishing to create software with standard interfaces. For now, those standard interfaces do not exist. A company can, however, adopt corporate wide integration strategies that formalize software interfaces for the company, make use of good methodologies, and use software integration tools as they become available.

3.3 Linking HP-UX and MPE XL Applications

Part of the fictional Ace Widgets example was based on an actual order entry and shipping system being implemented by a group within Hewlett Packard. The design team investigated various ways of linking their shipping system, running on HP9000 workstations, to their ordering system which ran on an HP3000. They narrowed their choices down to two alternatives: (1) develop custom software using NetIPC and Network File Transfer (NFT), (2) make use of a tool designed to help integrate software applications. They chose the second alternative because of the following reasons:

- Development effort and time required for a custom solution

- Resources required to support a custom solution
- Developing a custom integration tool detracted from their mainstream activity
- Using an appropriate tool would encapsulate expertise and make implementation easier

The integration tool they used, HP Software Integration Sockets, will be described in the next section of this paper.

4.0 Emerging Integration Technologies

The previous section served to illustrate some past and present integration strategies. These strategies evolved as newer technologies became available. In the early days, integration involved hand delivered punched cards and magnetic tape. Then modems became available to eliminate the inherent lack of reliability that hand delivered media carried with it. Data communications technology culminated in the development of LAN and WAN network technology. This naturally led to software which provided access to network services, such as network file transfer and access, network interprocess communications such as BSD Sockets and NetIPC, and Remote Procedure Calls (RPC). (Please see Figure 3, "Evolution of Application Integration Technologies".)

The push for *Open Systems* and the pervasiveness of applications running in distributed environments with computers from multiple vendors has led to a need for newer technologies that promote the integration of software applications. Not only do remotely located applications need to communicate, but often they reside on different machine architectures and were written in different languages. The following represents some of the technologies helping to solve these integration problems.

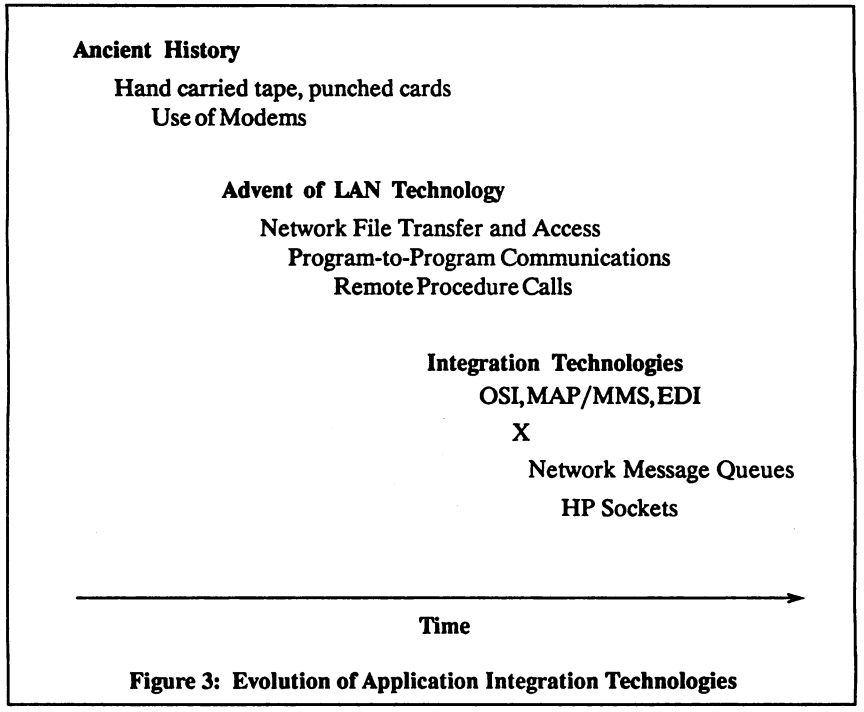


Figure 3: Evolution of Application Integration Technologies

4.1 Open Systems Interconnect

The International Standards Organization (ISO) has developed a seven layer Open Systems Interconnect (OSI) protocol reference model to serve as a framework for defining standards for linking together heterogeneous computers. Although ISO has made great strides in defining standard protocols in layers 1 through 4, protocols in layers 5 through 7 have progressed at a much slower pace. The complexity of understanding the upper layers of OSI has slowed its development as a pervasive technology. Few standards and products are actually based on a full ISO protocol stack. Also, few people know how to create or interface to layer 7, the Application Layer, of OSI. The OSI protocol stack serves as a good framework for defining standards, hence spurring on the development of products which conform to OSI standards. But it does not, in its current form, fully address application integration issues. (For instance, how would someone use OSI to integrate the different software pieces of the distribution system discussed in the Ace Widgets example?) Despite its shortcomings, anyone involved in application integration should be aware of ISO standards and monitor its progress. Two of the more relevant OSI standards are discussed below.

4.1.1 MAP/MMS

The Manufacturing Automation Protocol (MAP) is an implementation specification that makes use of a subset of OSI standards. The Manufacturing Message Specification (MMS) is an application layer service in an OSI/MAP stack. Together, these protocols allow manufacturing application software to communicate with factory floor devices such as PLCs, robots, and vision systems. It also allows some limited communications among different computer nodes in a CIM (Computer Integrated Manufacturing) hierarchy.

MAP and MMS are specifically peaked for manufacturing applications. It might not be appropriate nor cost effective in other application integration situations. It also does not solve the problem of integrating existing software not compliant with the MAP/MMS protocol.

4.1.2 Electronic Data Interchange

Electronic Data Interchange (EDI) is an OSI application layer (layer 7) protocol standard used for the electronic exchange of information between business partners. Documents, which were once physically created for business transactions such as purchase orders and billing, can be electronically transacted using EDI. This results in faster, more reliable, and less costly transactions among business partners.

EDI is used in inter-company communications. In a sense, it defines standard interfaces from which companies conforming to the standard can productively conduct business with one another. It is not typically used for intra-company communications. In fact, EDI vendors do not recommend its use for intra-company communications because of its inherent store-and-forward/batch operation. EDI does not solve the problem of integrating application software because it lacks the facility for real-time communications between software and it requires adherence to the EDI formats.

4.2 X Windows

X Windows was originally conceived of as part of a distributed computing project at MIT. Since its commercial inception in 1986, it is quickly becoming a pervasive technology. X Windows is based on a client-server model, where the server provides a windowed user interface display service to a locally or remotely located client application. By making this logical separation between user interface and the main body of the software application, clients scattered throughout a network can be accessed by any vendor's X display server with the right permission.

X Windows provides global access to information for users by allowing applications running on multi-vendor platforms to be accessed from any display server. From an X display server, a user could have multiple windows, each mapped to a different, remotely located client (e.g. one window for a spreadsheet application, one for a terminal emulator, one for a database application, etc.). X Windows does not, however, provide information access to software applications that need to share data.

4.3 Network Message Queues

A relatively new entry in integration technology is network message queues. This takes the notion of UNIX message queues, which work on a single node, and expands upon it to operate in a distributed environment on many nodes. Network message queues provide a robust messaging system among cooperating applications. Communications can occur synchronously or asynchronously, as client-server, or as peers.

Network message queues address some of the problems posed by software application integration. It can be viewed as a basic building block for creating communication links between applications.

4.4 HP Software Integration Sockets

Introduced in 1990, HP Software Integration Sockets was specifically designed to meet many of the needs faced in integrating application software. It includes the following design objectives:

- A solution to the problem of integrating existing, *legacy* applications
- A network message queue implementation that supports multiple kinds of communication links among applications
- Support of incremental growth
- Support of Open Systems, making use of available standards and committed to evolving with emerging standards
- Heterogeneous operation, including support for multiple platforms and multiple languages
- An easy to use, easy to learn interface which protects a company's investment by encapsulating technologies, standards, and promotes the use of good methodologies
- Data translation and manipulation capabilities to help integrate applications that were not written to communicate with each other
- Centralized administration of the integrated environment

The latest release of HP Sockets supports MPE XL as well as HP-UX. Experiences by several internal HP sites have been extremely positive. Many have pointed out that if a tool like HP Sockets had not been available, they would have had to develop one themselves. One group estimated 8 months to develop an equivalent tool that would have been much more narrowly focused, without HP Sockets' data translation and manipulation capability. With HP Sockets, they were able to reduce their integration time by a factor of 5.

5.0 Criteria For Robust Software Integration

There are many levels or stages to integrating software applications. If application software is evolutionary in nature, then so is the process of integrating software applications. The level at which application software is integrated determines the de-

gree to which they can share access to information and function more effectively as a whole than as separate parts. Figure 4 attempts to illustrate these stages of integration as concentric circles. The innermost circle represents the fundamental beginnings of software integration. As we move outward, attributes of a more fully integrated system of applications are seen. Each successive stage adds functionality to the previous stage. A fully integrated system is represented by the outermost circle.

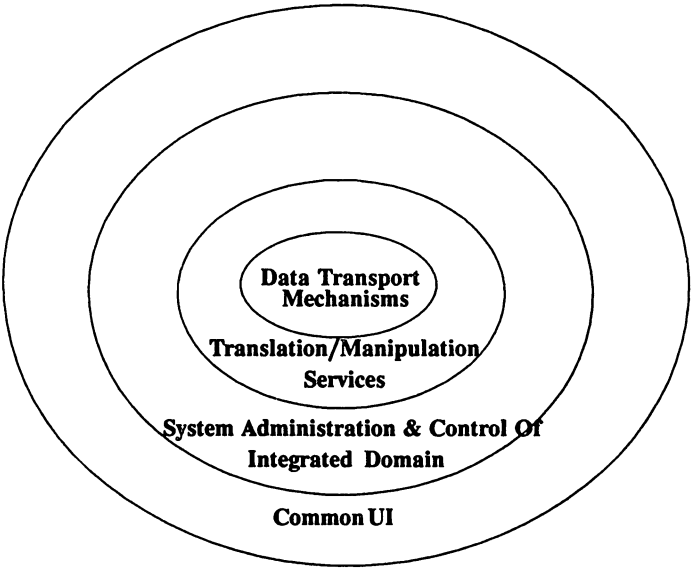


Figure 4: Stages of Software Integration

The emerging integration technologies surveyed in the last section provides us with a glimpse at the direction software application integration is headed. Currently, no single technology can solve all the problems. However, a set of criteria can be distilled from the strategies and technologies examined in this paper. Compare this to your own integration needs when evaluating a particular integration tool or methodology to adopt for your company.

- Ability to integrate existing as well as new applications
- Ability to meet future as well as present needs
  - Support of incremental growth
  - Adequate capacity for handling volume of data

- Adequate response time
- Promotes Open Systems
  - Support of multiple, heterogeneous platforms
  - Support of standards: OSI, OSF, XOPEN, etc.
  - Enforcement or promotion of standard interfaces, using consistent methodology throughout the integration domain
- Quick learning curve and ease of use, especially considering the level of expertise in your development environment
- Highly maintainable, even considering staff turnover
- High level of reliability
- Robust support of different topologies:
  - client/server
  - master/slave
  - peer-to-peer
- Centralized administration and control

6.0 Conclusions

Software application integration will continue to be a challenge for many companies. Currently, no single technology can completely solve the problem. Even if something works today, it may not be sufficient for future integration needs. The solution lies in developing comprehensive software integration strategies, incorporating sound methodologies and integration tools which promote use of those methodologies. This paper has examined the progress of software application integration strategies and technologies, including the use of HP Software Integration Sockets for linking HP-UX and MPE XL applications. It suggests software integration is an evolutionary process with various stages or levels of integration. Finally, a set of criteria for developing and evaluating integration systems was presented.

SCSI: DISK INTERFACE OF CHOICE ON HP WORKSTATIONS

Scott B. May
Hewlett-Packard
19019 Pruneridge Ave.
Cupertino, CA 95014

Why SCSI?

The computer industry in general, and the workstation and PC markets in particular, has been moving rapidly to standards-based systems. Open systems have many advantages. A computer user who invests in standards-based hardware is not limited to the product offerings or pricing policy of a single company, and assuers that his or her investment is protected. Standards-based products are predicted to make up much of the workstation market in the next few years. Microprocessors, graphical user interfaces, operating systems, networks, disk interfaces, and graphics are all seeing significant standards-based development. As standards in these areas become more defined, shrink-wrapped workstation software will become closer to a reality.

Standards-based hardware is getting much attention, especially in the area of peripherals. Users in the PC market have had plug-and-play hardware compatibility. The standards in the PC arena have been set not by an ANSI committee, but by recognition of de-facto standards. The IBM PC set the standard because it had such a large share of the market. Other computer manufacturers utilized the bus architecture making the PC-XT bus the industry standard architecture (ISA) bus. Many bought disks from Seagate, and soon ST-506 became the de-facto standard interface for disk drives. Microsoft licensed MS-DOS, and it soon became another de-facto standard.

The workstation market has evolved standards in quite a different way. Much of the development of UNIX workstation standards has happened in committees of the American National Standards Institute (ANSI), and in industry-wide committees like X-Open and Open Software Foundation (OSF). The subject of this article, the Small Computer System Interface (SCSI-1) was developed by an ANSI committe, and is defined by ANSI standard X3.T31-1986. SCSI was developed using principles from Shugart Associates System Interface (SASI), which was developed in the mid-seventies.

Hewlett-Packard has been a leader in the formulation and adoption of UNIX workstation standards. This is one of the reasons that the SCSI interface is now standard on the HP 9000 Series 300 workstations. In addition, the HP-IB interface was perceived as a performance limiter though in most cases it was not. Yet another reason for the adoptions of SCSI is that it is well adapted to connect a variety of peripherals, like Rewritable Optical, Digital Audio Tape, CD-ROM, and printers. The ability to connect all of the peripherals onto one interface eliminates the need for multiple interfaces, thus saving slots for other uses and minimizing cost. With SCSI, users can connect devices that HP currently chooses not to offer, like Write Once, Read Many (WORM) optical and 8 MM tape.

The SCSI interface is a high-level interface, as opposed to other interfaces like ST-506, enhanced small disk interface (ESDI), IPI-2, and SMD, which are device-level interfaces. Other high-level interfaces include Hewlett-Packard Interface Bus (HP-IB) and Hewlett-Packard Fiber Link (HP-FL), and Intellegent Peripheral Interface-3 (IPI-3). The major conceptual difference is that the disk controller hides many of the detailbads of disk operation from the host computer. The computer does not have to manage the details of where the file is physically located, and does not have to seperate header and trailer information from the data. In addition, the controller presents the disk as one long string of logical blocks, hiding bad tracks. A device-level interface forces the CPU to keep track of bad sectors.

The effect of a high level interface is that the computer is free to do other tasks because the controller handles many details of disk I/O. Single-user, single-tasking machines cannot take advantage of this feature because the computer is not able to go on until the data from the disk arrives. However, a multi-tasking or multi-user computer can take advantage of the higher availability of the CPU, leading to a significant performance improvement.

Time To Market

One of the features that makes SCSI so attractive to peripheral manufacturers is that an embedded SCSI interface lets them bring a new product to market very quickly. For instance, immagine that XYZ Company brings out a new WORM drive. If they do not use an industry standard interface they are caught in a catch 22 situation: nobody will build the host adapter because the volumes are so low. There are no sales because there is no way to use the product. XYZ Company must convince another company to develop host adapters for a variety of computers or do it themselves. Instead, if XYZ company designs their own embedded SCSI controller and puts it on the mechanism, significantly less work is needed to integrate the new product into a computer system. Of course, the computer would also require a SCSI driver compatible with the drive and operating system as well. Therefore, peripheral manufacturers can speed their time to market by using the SCSI interface.

Table I

SCSI Bus Signals

Signal	Description
ACK	(Acknowledge) Data on bus.
ATN	(Attention) Request for message out phase. Initiator has message for target.
BSY	(Bsy) Target is Busy.
C/D	(Control/Data) Indicates whether bus carries control messages or data.
I/O	(Input/Output) Indicates direction of data flow on the bus. I/O is true when data goes from target to initiator.
MSG	(Message) Signals on the bus are a message when this signal is asserted.
REQ	(Request) Requests data on the bus.
RST	(Reset) Hard reset of all devices on the bus.

SCSI Bus Functionality

In order to understand the complexities of the SCSI bus, including the differences

between synchronous and asynchronous data transfer, it is important to have some understanding of the different bus states, signals, and commands. Table I lists SCSI bus signals. Table II lists SCSI bus states.

The commands listed in Appendix 1. are those supported by the HP 9754xS disk that is sold on an OEM basis to other manufacturers. Note that many of the commands are described as "vendor-unique". All SCSI drives support many such commands. The vendor-unique commands are mainly used for diagnostic purposes. Just because a drive uses vendor-unique commands does not mean that it is incompatible with another vendor's hardware. The fact is that the implementation of the SCSI commands of the 9754xS were designed with vendor-independence in mind. In most instances the operating system will never issue a vendor-unique command.

Table II

SCSI Bus States

State	Description
Arbitration	Initiatior negotiates for control of the bus.
Bus Free	No device is using the bus. It may be that a target has disconnected and will soon reselect.
Command	Initiator issues a command like read, write, or format.
Message In	Target will disconnect and then the bus will be free.
Message Out	Initiator identifies itself to the target.

Disk Transaction

The basic unit of disk operation is an I/O. What follows is a description of 2a typical disk I/O. This example holds true for any SCSI device, not just disks. The I/O operation is initiated by the operating system of the computer, which for this example happens to be HP-UX. Note however, that many different operating systems support SCSI peripherals, including MS-DOS. The disk transaction begins with a request from the computer. The request may be for a file system block or for a virtual memory page. This request is passed to the SCSI driver, along with data that the driver will use to locate the data, including the device identifier. Then the driver goes through the ARBITRATION and SELECTION process to gain control of the bus.

The disk responds with a MESSAGE OUT, and then receives an IDENTIFY message from the host. The disk's response to the IDENTIFY message indicates to the host whether the disk supports disconnect/reconnect during the data phases, and also whether the disk supports command queuing (Part of SCSI-2). The host driver then asserts the ATN signal to maintain the MESSAGE OUT phase and determines whether the disk can support synchronous transfer. Then the driver issues a command indicating whether it wants a read or a write. Upon receiving this command, the disk sends disconnect and save data pointers to the host, and then disconnects if it supports that option. At this point the bus is free to service other devices. During this time the controller decodes the request and carries it out. When the disk is finished or nearly finished loading data into the buffer, it asserts the RESELECT signal, and transmits data from its buffer. Depending on the size of the disk buffer and the amount of data requested, the disconnect/reconnect cycle may happen several times during an

I/O. When the transaction is complete, the STATUS message is sent to the host, and the transfer is complete.

Comparisons

Single Ended vs. Differential

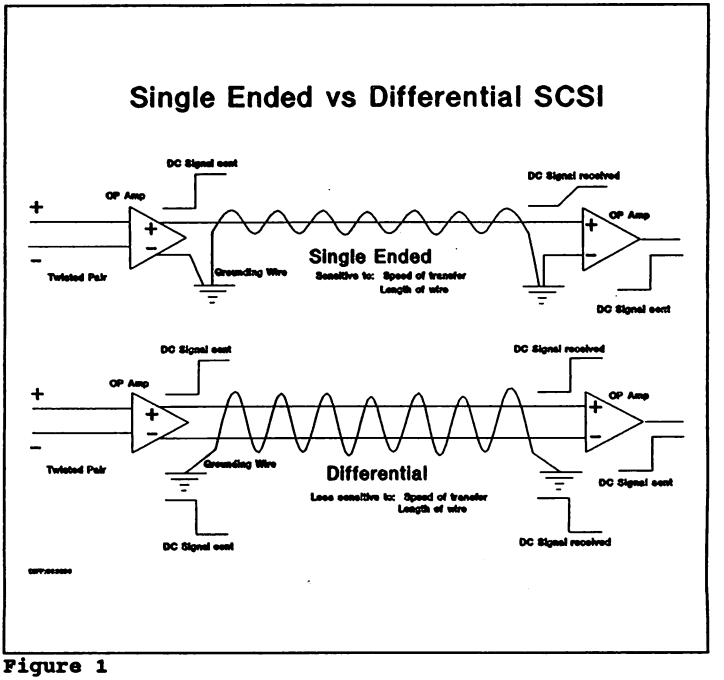


Figure 1

The difference between single-ended and differential SCSI is in the electrical definition of the signal drivers. Differential SCSI uses twice as many wires as single-ended SCSI. In single-ended, one wire of a pair is used for grounding only, whereas differential SCSI uses the second wire to send the complement of the signal in the first wire. In other words, as the first line goes high, the second line goes low. This has the effect of decreasing electromagnetic emissions because the fields generated by the two wires tend to cancel each other out, like a coaxial cable does. This gives a higher signal to noise ratio, which allows faster clock speeds and longer cable lengths. Figure NNN

Synchronous vs. Asynchronous

Synchronous SCSI-1 is rated at 5 megabytes per second, whereas asynchronous is rated at only about 1.5 to 3 Mbytes per second. Note that all commands and messages are

transmitted at the asynchronous rate. Only data can be processed in synchronous mode. When a transfer is initiated by the host computer, either device can issue a "synchronous data transfer request" (SDTR) message. If either the initiator or target fails to issue the message, the transfer defaults to asynchronous. Asynchronous means that for each 1 byte transfer, the target sends an ACK signal and then waits for an REQ signal from the initiator before more data is sent.

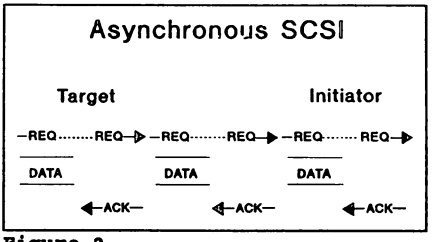


Figure 2

Thus, the ACK/REQ handshake must happen for each byte of data transferred in asynchronous mode. During a synchronous transmission, the target device does not wait for the REQ signal between data transmissions. Instead, data is sent until a number of bytes, determined by the REQ/ACK offset agreed upon at the initiation of synchronous transfer, has been sent. During the transmission, the initiator continues to send ACK's back to the target. The target keeps track of them and knows that when the number of REQ's matches the number of ACK's the transfer is complete. See Figures 2. and 3. for a graphical description.

SCSI-1 vs. SCSI-2

The SCSI-1 interface is rated at up to 5 Mbytes per second for synchronous data transmissions. The SCSI-2 definition allows for data transmission at up to ten megatransfers[1] per second (differential cables only), giving differential SCSI-2 transfer rates up to 320 megabits, or 40 Mbytes per second. There are two different kinds of modifications to the SCSI-1 standard that allow the higher data rate.

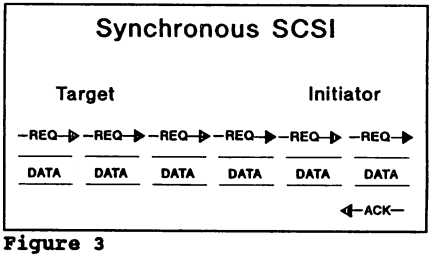


Figure 3

The first modification is called "fast". Fast SCSI-2 requires differential electronic drivers. Single-ended electonics are not capable of the signal to noise ratio necessary to implement fast SCSI-2. The clock speed of fast SCSI-2 is twice that of SCSI-1. Using the same eight bit data path, fast SCSI-2 is capable of ten Mbytes per second. The cable length limit of differential SCSI is 15 meters compared to 6 for single-ended.

[1] As will be explained, each transfer can be eight, sixteen, or thiryt-two bits wide.

The second modification affecting transfer rate is "wide" SCSI. Wide SCSI-2 gives a data path of either sixteen or thirty-two bits as opposed to the 8-bit path of SCSI-1. The space required to connect the cable may preclude the use of wide SCSI on 2.5" and 3.5" disks. By combining wide and fast, SCSI-2 can achieve a synchronous data transfer rate of 40 Mbytes per second.

Implications of SCSI-2

Most applications will not need a wide and fast SCSI-2 bus. Doubling the transfer rate of the bus will not double the throughput of the I/O system. In a random disk transfer, the seek and latency constitute a far larger chunk of time than the channel time. An average 5.25" disk has an average seek plus latency of 24 milliseconds compared to less than four milliseconds for an 8 KByte transfer. Even a 40 Mbyte per second chanel would not necessarily decrease the four millisecond figure because it takes a substantial amount of time to get the data from the disk surface once the heads are over the data.

For a standalone workstation, fast SCSI or even synchronous SCSI-1 provides more bandwidth than a single or dual disk drive configuration can utilize. Very few 5.25" disks used on workstations have a sustained UNIX file system transfer rate of greater than 2.0 Mbyte per second, even for large files.

Certain applications, however, could certainly benefit fron a wide and fast SCSI bus. Examples include file servers with multiple disks, transaction processing again with multiple disks, solid state disks, disk arrays, and processor-to-processor communication.

SCSI-2 has other features that differentiate it from SCSI-1 besides transfer rates. SCSI-1 allows only one outstanding command from an initiator to a target. Command queuing IN SCSI-2 allows the host I/O driver to handle multiple requests. Command queuing allows up to 256 requests to be outstanding from each initiator to each target. SCSI-2 has much more tightly defined electrical specifications. SCSI-1 electrical specifications are loose enough that two devices that both meet the spec could not work together. SCSI-2 closes the gaps.

SCSI-1 vs. HP-IB

The two interfaces currently available on The HP 9000 Series 300 and 400 workstations are HP-IB and SCSI-1. HP-IB is an interface based on IEEE-488 and is rated at a transfer rate on 1 Mbyte per second. The Hewlett-Packard disks available for Series 300 workstations include the Series 6000 Model 670H and Model 660S. The 670H uses HP-IB, while the 660S uses SCSI-1. In order to characterize ther performance differences between these two disk drives, a series of benchmarks were executed.

The Khornerstone benchmark is owned by Workstation Labs, Inc, and the disk portion of the test includes disk intensive tasks, such as reading and writing files both randomly and

sequentially. As can be seen in Figure NNN, the Model 660S has a disk Khornerstone score about twice that of the Model 670H. Figure NNN+1 shows the throughput of these two disks for files of various sizes, both reading and copying (reading and writing).

The Model 660S and the Model 670H share the same disk mechanism and ESDI device-level interface. The differences are:

- The firmware of the Model 660S has been tuned for HP-UX.
- The SCSI channel is much faster than HP-IB.

The net effect of these two changes is dramatic. Whether the tuning of the Model 660S' firmware or the faster channel makes the biggest difference, the performance choice for HP Series 300 workstations is clearly SCSI.

SCSI vs. ESDI

As previously stated, ESDI is a device-level interface. Many SCSI disk drives use ESDI as the device-level interface under SCSI. The difference between SCSI and ESDI is that whereas ESDI defines a device controller, SCSI defines an interface bus. Because SCSI has more intelligence than ESDI, it has more overhead. SCSI does some of the work that the central processor would do for an ESDI device. In some environments moving intellegence from the CPU to the controller does little good. For example, PC's often perform better with an ESDI drive than with a SCSI drive, at least in a single drive configuration running a single-tasking operating system (DOS). The reason is that the added overhead of decoding the SCSI commands has a greater effect than the higher availability of the CPU. In the DOS environment the CPU cannot work on other tasks in the background. However, when a computer runs a multi-tasking operating system and uses multiple drives, SCSI will outperform ESDI.

The SCSI standard allows a device to disconnect from the bus during long operations such as formatting, seeking, and tape positioning. The added availability of the SCSI bus as a result of the disconnect/reconnect feature gives SCSI an advantage over ESDI. Again, the difference between the two is small for single drive configurations. However, when multiple devices share the same bus, the throughput of the SCSI system is much greater than for the ESDI system. Figure 4 shows the results of several sequetial write benchmarks that were executed on an 80386-based PC. The sgnificance of the slide is not the relative performance of the SCSI and ESDI drives tested. The significance of this data is what happens as more disks are added. Notice that the throughput of the SCSI bus increases dramatically as more disks are added. The ESDI channel also shows some improvement, but not nearly as much as the SCSI channel.

In choosing between an ESDI interface and a SCSI interface, the main questions are these: 1) Will the computer be used in a single-user-single-task, single-user-multiple-task, or multiple-user-multiple-task environment? As more processes and users access the disk(s),

SCSI will give better performance than ESDI.

2) Will other types of peripherals be used? Many different types of devices are available that have an embedded SCSI interface. It is possible to have an ESDI interface on a tape drive, but none are available. On the other hand, many different vendors offer SCSI peripherals, including 8 MM tape, 4 MM tape, 1/4" tape, plotters, CD-ROM, rewritable optical, write-once-read-many (WORM), solid-state disk, and even printers.

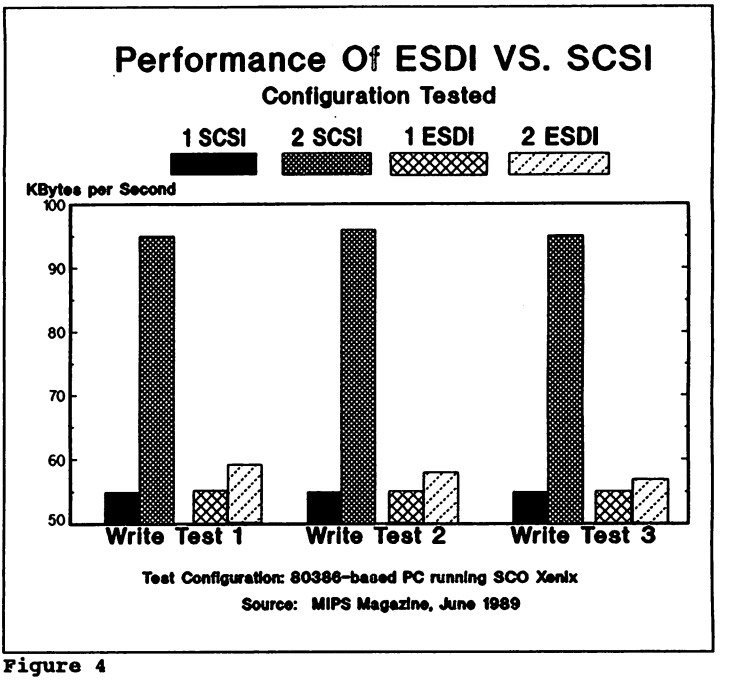


Figure 4

SCSI vs. IPI

One of the main differences between SCSI and IPI is that while SCSI is a peer-to-peer bus, IPI is a master-slave bus. The IPI interface is really a hierarchy set of interfaces. IPI-0 defines a 16 bit parallel cable IPI-1 defines a communication protocol. Together IPI-1 and IPI-2 define a device-level interface with the nearly the same characteristics as ESDI and the SCSI hardware specification. IPI-3 defines a high level interface similar to full SCSI. Most of the IPI systems sold connect to IBM mainframes.

IPI-2 has a transfer rate of ten Mbytes per second, the same as fast SCSI-2, making it a good choice where fast I/O is needed. One of the drawbacks of using IPI-2 or 3 on a workstation is that IPI disks tend to be much more expensive than SCSI disks because almost all of the

IPI disks available are 8" or larger and are dual ported (they have two read and write heads). In addition, no other peripherals are available that use the IPI interface. If other peripherals like a tape drive are needed, another interface card must be purchased. Backup to tape requires another interface. For some machines, this may not make a differennce. If the machine in question happens to be a large file server, then putting a tape on the same interface as the disks will probably cause a degradation in performance while the tape is running.

On the other hand, a disk interface that cannot support tape is not a good choice for a standalone workstation. A workstation on a Local Area Network (LAN), however, can access a tape anywhere on the LAN regardless of its interface.

The IPI interface is best suited for large servers and mainframes. The high cost is justified by very high throughput dual-ported eight inch disks. However, multiple smaller disks on a SCSI-2 interface may give equal performance at a lower cost per Mbyte, especially in a random server environment, where I/O's per second are more important than Mbytes per second. The larger number of spindles would allow concurrent seeks, which are very important to performance in a random multi-user environment.

Summary

Standards are becoming more and more pervasive. The SCSI interface is becoming the standard peripheral interface for Unix-based workstations. Many manufacturers produce mechanisms with embedded SCSI controllers, and as the SCSI standard becomes more tightly defined, incompatibilities among SCSI devices will decrease. SCSI is ideally suited to the workstation environment for these reasons:

- Many different types of devices are being produced with an embedded SCSI controller. Users who have a SCSI interface will be able to make use of them.

- The SCSI bus has high throughput. As wide and fast SCSI products become available, the performance of multi-disk systems can be expected to improve.

- SCSI is better adapted to a multi-user environment than ESDI, yet is more cost effective than IPI.

- By using embedded SCSI controllers, manufacturers can reduce the amount of integration work needed to bring a product to market.

Appendix 1.

SCSI Commands Supported By The HP 9754xS Disk

COMMAND	DESCRIPTION
ACCESS LOG	Vendor-unique command. Requests Target to retrieve information from its maintenance log.
EXECUTE DATA	Vendor-unique command. Executes special code downloaded via the WRITE BUFFER command.
FORMAT UNIT	Formats Target media into Initiator addressable logic blocks. Defect sources include P, D, and G lists (no C list). When formatting, it is recommended that the Initiator not include a D list (FMTDAT=0). However, if the Initiator does include a D list, it must be in the physical sector or bytes from index format. The Target uses an interleave of 1 regardless of the value in Interleave field.
INQUIRY	Requests Target to send parameter information to the Initiator. Additional Vital Product Data (VPD) may be supplied if requested by the Initiator.
INTERFACE CONTROL	Vendor-unique command. Enables ESDI commands to be sent to the disk drive processor.
MANAGE PRIMARY	Vendor-unique command. Used to manage the primary defect list (P list). This command can delete the current P list, install a new P list, or append defects to the current P list.
MEDIA TEST	Vendor-unique command. Used to test the integrity of the disk media.
MODE SELECT	Enables Initiator to specify media, logical unit, or device parameters to the Target.
MODE SENSE	Enables Target to report its media, logical unit, or device parameters to Initiator.
READ (6-byte) (10-byte)	Requests Target to transfer data to Initiator. Relative Addressing not supported in extended (10-byte) format (REL=0).

READ BUFFER	Used with WRITE BUFFER command to test the Target's data buffer. Recommend executing RESERVE command to guarantee data integrity.
READ CAPACITY	Enables Initiators to request information regarding the capacity of a logical unit. Use of PMI bit supported. Relative Addressing not supported (REL=0).
READ DEFECT DATA	Requests Target to transfer media defect data to Initiator. Target returns P, G, or P+G lists in physical sector or bytes from index format.
READ HEADERS	Vendor-unique command. Requests Target to read all the headers on the addressed track and return the requested number of bytes of header information.
READ FULL	Vendor-unique command. Requests Target to return the header, data field and ECC bytes of one physical sector.
REASSIGN BLOCKS	Requests Target to reassign defective logical block to an area on logical unit reserved for this purpose. It is recommended that the defect list contain only one defect location per command.
REFORMAT TRACK	Vendor-unique command. Formats a single track. If HS bit is 0, then it uses normal default header information. If the HS bit is 1, the supplied header information is used for the track logical address and flag bytes.
RELEASE	Release previously reserved logical units. Third-Party Release supported. Extent Release not supported.
REQUEST SENSE	Requests Target to transfer sense data to the Initiator, including: Sense Key (0-6,B,E), Additional Sense Code, Device Errors (DERRORS), and Recommended Actions. The Bit Pointer and Field Pointer fields are not used. Only the Extended Sense Data format is supported.
RESERVE	Reserves logical units for use of Initiator. Unit and Third-Party Reservations are supported. Extent Reservation are not supported.
REZERO UNIT	Requests Target to perform a recalibrate and then to seek to logical address 0.

SEEK	Requests Target to seek to a specified address. (6-byte) Target returns GOOD status when seek is complete. (10-byte)
SEND DIAGNOSTIC	Requests Target to perform specified diagnostic tests. Self-test is supported. If self-test fails, Check Condition status indicates that results are available via REQUEST SENSE command.
SPECIAL SEEK	Vendor-unique command. Requests Target to leave the disk drive selected after execution of a seek. Allows for special testing at the seek address.
START/STOP UNIT	Requests Target to enable or disable the logical unit for further operations. Using the immediate bit on START is supported, but not recommended.
TEST UNIT READY	Checks Target spindle for proper speed. Target returns GOOD status if drive is up to speed.
VERIFY	Requests Target to verify the data written on the media by performing a selectable ECC check or a byte compare. Relative addressing not supported. (REL=0).
WRITE (6-byte) (10-byte)	Requests Target to write the data transferred by the Initiator to the media. Relative Addressing supported in 6-byte format. Relative Addressing not supported in extended (10-byte) format (REL=0).
WRITE AND VERIFY	Requests Target to write the data transferred by the Initiator to the media, then do an ECC verify of the data that was written. Relative addressing not supported. (REL=0, BYTCK=0).
WRITE BUFFER	Used to test Target's data buffer or download code. To avoid possible data corruption, it is recommended that a RESERVE command be executed prior to the WRITE BUFFER command.
WRITE FULL	Vendor-unique command. Requests Target to write one complete physical sector, including header, data, and ECC fields.

Paper 2007

Referential Integrity in ALLBASE/SQL

Amelia Carlson
Hewlett-Packard Company
19111 Pruneridge Avenue
Cupertino CA 95014
(408) 725-8900

5/7/91

Abstract

Integrity constraints are restrictions placed upon tables in a database which limit the legal values of columns in the tables. Two very important types of constraints are the unique constraint and the referential constraint. A unique constraint is used to prevent duplicates in columns, and referential constraints are used to control the values in a table based upon values in another table (or other columns in the same table).

In the past, applications have enforced such constraints manually. ALLBASE/SQL introduces this functionality to allow database designers to ensure that data integrity is never violated in any application, and to inform applications of attempts to violate constraints.

This paper discusses a method of introducing integrity constraints into a database schema, and shows the benefits this provides to the application writer. This paper targets database designers and application developers.

Introduction

In today's applications, unique and referential constraints are being enforced in applications. With ALLBASE/SQL Release E, these constraints can instead be placed in the database schema and enforced by the database.

A unique constraint requires that each row has a unique value for the unique constraint columns. It enforces the same uniqueness as a unique index does today. However, unique constraints require that none of the unique key columns allow null values. This is to permit a referential constraint to reference the unique constraint columns. Unique constraints are refered to as *keys*; one unique constraint in a table can be the *primary key*, and the others are *candidate keys*.

A referential constraint requires that the referencing columns in each row have matching values in the referenced table's referenced columns. The *referencing columns* are columns in the table which defines the referential constraint. The *referenced columns* are columns in the table being referenced. The referenced columns must be unique constraint columns in a

unique constraint in the referenced table. Referential constraints permit nulls to exist in the referencing columns. If a referencing column contains a null value, then that row does not reference any row of the referenced table. Referential constraints are refered to as *foreign keys*.

Example Schema

This paper will introduce constraints using tables from the ALLBASE/SQL sample database. The sample database comes with the ALLBASE/SQL product. On HP-UX, it is located in /usr/lib/allbase/hpsql/sampledb. On MPE/XL, it is located in SAMPLEDB.SYS. The tables that will be considered are listed with their columns. NOT NULL columns are listed as such.

- PurchDB.Parts (PartNumber NOT NULL, PartName, SalesPrice)
- PurchDB.Inventory (PartNumber NOT NULL, BinNumber NOT NULL, QtyOnHand, LastCountDate, CountCycle, AdjustmentQty, ReorderQty, ReorderPoint)
- PurchDB.SupplyPrice (PartNumber NOT NULL, VendorNumber NOT NULL, VendPartNumber NOT NULL, UnitPrice, DeliveryDays, DiscountQty)
- PurchDB.Vendors (VendorNumber NOT NULL, VendorName NOT NULL, ContactName, PhoneNumber, VendorStreet NOT NULL, VendorCity NOT NULL, VendorState NOT NULL, VendorZipCode NOT NULL, VendorRemarks)
- PurchDB.Orders (OrderNumber NOT NULL, VendorNumber, OrderDate)
- PurchDB.OrderItems (OrderNumber NOT NULL, ItemNumber NOT NULL, VendPartNumber, PurchasePrice NOT NULL, OrderQty, ItemDueDate, ReceivedQty)

The indexes created on these tables are:

- UNIQUE INDEX PartNumIndex ON PurchDB.Parts (PartNumber)
- CLUSTERING INDEX PartToNumIndex ON PurchDB.SupplyPrice (PartNumber)
- INDEX PartToVendIndex ON PurchDB.SupplyPrice (VendorNumber)
- UNIQUE INDEX VendPartIndex ON PurchDB.SupplyPrice (VendPartNumber)
- UNIQUE INDEX VendorNumIndex ON PurchDB.Vendors (VendorNumber)
- UNIQUE CLUSTERING INDEX OrderNumIndex ON PurchDB.Orders (OrderNumber)
- INDEX OrderVendIndex ON PurchDB.Orders (VendorNumber)
- CLUSTERING INDEX OrderItemIndex ON PurchDB.OrderItems (OrderNumber)
- UNIQUE INDEX InvPartNumIndex ON PurchDB.Inventory (PartNumber)

This paper will show how to recognize the constraints enforced on these tables and how to develop a schema including those constraints.

Application level constraint enforcement

There are three things to consider when examining an existing database for constraints.

- Unique indexes point toward possible unique constraints.
- Application programs may contain validation procedures for checking that user input satisfies some conditions before placing it in the database. These checks may point to unique or referential constraints.
- Ad-hoc queries which are often performed may point to unique or referential constraints.

We will consider each of these in turn, with the goal of converting the database to use schema defined constraints in place of existing indexes, application level constraint enforcement, or implicit unenforced constraints.

Unique Indexes

The columns in a unique index are a good candidate for a unique constraint. The creation of a unique constraint means that the index can be eliminated; it is possible to create a unique constraint with the CLUSTERED or HASH options, so that performance gained from the unique index will not be lost with its transferral to being a unique constraint.

In order for a unique index to be transformed into a unique constraint, the database designer must ensure that all columns in the unique index are declared NOT NULL, as all columns in a unique constraint must be declared NOT NULL.

In our example, we see that several unique indexes exist:

- PartNumIndex
- VendPartIndex
- VendorNumIndex
- OrderNumIndex
- InvPartNumIndex

Further, each column in each index was declared NOT NULL. We will thus have five unique constraints in the PurchDB database, replacing the five unique indexes.

Since the referenced columns of referential constraints need to be unique constraint columns in an existing unique constraint, other unique constraint may be identified later.

Application programs

Because there was no way to define referential constraints in the ALLBASE/SQL database schema before ALLBASE/SQL Release E, constraints had to be enforced with user applications. Typically, the database designer may have written up a list of requirements which all application developers had to follow.

So, the application programs for accessing a given database may contain certain data validation steps for checking that users' data satisfies the database designer's requirements. We can examine the application programs, or, if available, the database designer's list of requirements, to look for unique or referential constraints.

A requirement for a unique constraint might be phrased in the list as "values placed in PurchDB.Parts.PartNumber must be distinct" or "values placed in PurchDB.Vendors.VendorNumber must not duplicate any existing value in

PurchDB.Vendors.VendorNumber". These two phrases suggest for our example that a unique constraint should be placed on PurchDB.Parts.PartNumber and another on PurchDB.Vendors.VendorNumber.

A requirement for a referential constraint might be phrased as "a value cannot be placed in PurchDB.Order.VendorNumber without verifying that the value is a valid vendor number". Deciding whether a vendor number is valid may be stated ("valid vendor numbers are those values in PurchDB.Vendors.VendorNumber") or may be left to the application developer to deduce. For our example, this would represent a referential constraint on PurchDB.Order.VendorNumber, with PurchDB.Vendors.VendorNumber being the referenced column. This requires that a unique constraint exist on PurchDB.Vendors.VendorNumber.

Other phrases in the database designer's documentation that would suggest referential constraints are:

1. Values cannot be removed from PurchDB.Parts.PartNumber without making sure that no order or inventory currently includes that part number.
2. Values in PurchDB.OrderItems.VendPartNumber must identify existing vendor part numbers.
3. Values in PurchDB.Vendors.VendorNumber cannot be altered if there is any existing pricing or order for that vendor.

These would map to several referential constraints.

The first item points to several referential constraints with PurchDB.Parts.PartNumber as the referenced column. This is because the SupplyPrice and Inventory PartNumber columns are based on the PartNumber value defined in the Parts table. The referencing column in each case would be where PartNumber appears in the other tables, with one referential constraint for each of PurchDB.Inventory.PartNumber and PurchDB.SupplyPrice.PartNumber to PurchDB.Parts.PartNumber.

The second item points to a referential constraint from PurchDB.OrderItems.VendPartNumber to PurchDB.SupplyPrice.VendPartNumber. We can identify the referenced column either from other information in the database designer's list, or by recognizing that we have defined PurchDB.SupplyPrice.VendPartNumber as a likely unique constraint from examining the unique indexes.

The third item, similar to the first, points to several referential constraints with PurchDB.Vendors.VendorNumber as the referenced column. The referencing constraints would exist on each of PurchDB.SupplyPrice.VendorNumber and PurchDB.Orders.VendorNumber.

The database designer's requirements list may not always be available, however. In this case, the application programs can be examined for validation checks of user data. If we examine the sample programs, we find some validation checks. The ALLBASE/SQL sample database C program cex9 (also available in Cobol as cobex9 and in Pascal as pasex9) contains validation checks for Vendor data. Specifically, it contains the code segments shown in Figures 1 and 2.

```
int ValidateVendor()  /* Function that ensures vendor number is valid */
{
  BeginTransaction();

  printf("\n Validating VendorNumber");
  EXEC SQL SELECT  VendorNumber
             INTO :VendorNumber
             FROM  PurchDB.Vendors
            WHERE  VendorNumber = :VendorNumber;

  switch (sqlca.sqlcode) {
    case OK:              EndTransaction();
                          VendorOK = TRUE;
                          break;
    case NotFound:        EndTransaction();
                          printf("\n No vendor has the VendorNumber you");
                          printf("\n     specified!");
                          VendorOK = FALSE;
                          break;
    default:              SQLStatusCheck();
                          EndTransaction();
                          VendorOK = FALSE;
                          break;
    }  /* End switch */
}  /* End ValidateVendor Function */
```

Figure 1.
This code segment verifies that a given vendor exists when an order is being created for that vendor.

The code segment in Figure 1 shows that an order will not be created unless the vendor number is found in the table PurchDB.Vendors. This can be represented with a referential constraint from PurchDB.Orders.VendorNumber to PurchDB.Vendors.VendorNumber.

```
int ValidatePart()  /* Function that ensures vendor part number is valid */
{
  BeginTransaction();

  printf("\n Validating VendPartNumber");
  EXEC SQL SELECT  VendPartNumber
             INTO :PartSpecified
             FROM  PurchDB.SupplyPrice
            WHERE  VendorNumber = :VendorNumber
              AND  VendPartNumber = :PartSpecified;

  switch (sqlca.sqlcode) {
    case OK:              EndTransaction();
                          PartOK = TRUE;
                          break;
    case NotFound:        EndTransaction();
                          printf("\n The vendor has no part with the number");
                          printf("\n     you specified!");
                          PartOK = FALSE;
                          break;
    default:              SQLStatusCheck();
                          EndTransaction();
                          PartOK = FALSE;
                          break;
    }  /* End switch */
}  /* End ValidatePart Function */
```

Figure 2.
This code segment verifies that a given vendor part number exists for a given vendor when that vendor part number is added to an order.

The code segment in Figure 2 shows that an item in an order will not be created unless the vendor number and vendor part number are found in the table PurchDB.SupplyPrice. This suggests a referential constraint from PurchDB.OrderItems.VendPartNumber to PurchDB.SupplyPrice.VendPartNumber. Notice that there is no VendorNumber column in OrderItems, so that column cannot be included in the referential constraint. (This type of constraint, where values from two tables are combined to reference another value in another table, cannot be fully represented via referential integrity; a more general integrity mechanism such as the ANSI SQL3 draft TRIGGER feature is needed.) The referential constraint, then, gives us a first approximation on the requirements that the vendor number and vendor part number co-exist in the SupplyParts table.

Examining the database designer's application requirements documentation and the existing applications uncovers several referential constraints that can be represented in the database schema.

Queries

The database will also be subject to access through ad-hoc queries. These queries may further show implicit constraints. We can examine these possibilities without a record of the ad-hoc queries made on the database. A typical database will contain indexes created to improve the access time of queries. These indexes point to common access paths. If we examine closely the index columns, we can deduce the nature of the queries and of the constraints they might assume.

These constraints might also be listed in a document written by the database designer as constraints the data is expected to maintain, although they were not explicitly defined in the schema.

We already examined the unique indexes in a previous section, to determine the unique constraints. There are four other indexes on the example tables which are not unique.

These are

- PartToNumIndex
- PartToVendIndex
- OrderVendIndex
- OrderItemIndex

The first index, PartToNumIndex, is on the PartNumber column of SupplyPrice. This suggests that the PartNumber column is frequently accessed. Further, we know there is a unique constraint on PurchDB.Parts.PartNumber. It is likely that some of the ad-hoc queries are joins between these two tables on this column. A view included in the sample database, PurchDB.PartInfo, supports this idea. So, we would expect that the PartNumbers in SupplyPrice are a subset of the PartNumbers in Parts. A referential constraint between these two tables on these columns will enforce this constraint within the database. Because the referential constraint creates a constraint index on the referencing columns, we will no longer require the index PartToNumIndex. The syntax for creating constraints allows us to declare the constraint index as CLUSTERING so that this attribute of the old index is not lost.

Similarly, PartToVendIndex suggests that the vendor numbers in SupplyPrice should be a subset of those in Vendors. A referential constraint will enforce this condition and replace the index PartToVendIndex. OrderVendIndex suggests a referential constraint from PurchDB.Orders.VendorNumber to PurchDB.Vendors.VendorNumber. OrderItemIndex suggests a referential constraint from PurchDB.OrderItems.OrderNumber to PurchDB.Orders.OrderNumber.

Examining the typical ad-hoc queries through the indexes on the database has thus uncovered several more referential constraints that we can represent in the database schema.

Introducing constraints in the schema

These are the constraints we have found in examining a portion of the sample database:

- Unique constraints
  - PurchDB.Parts.PartNumber
  - PurchDB.SupplyPrice.VendPartNumber
  - PurchDB.Vendors.VendorNumber

- PurchDB.Orders.OrderNumber
- PurchDB.Inventory.PartNumber

- Referential constraints
  - PurchDB.Inventory.PartNumber references PurchDB.Parts.PartNumber
  - PurchDB.SupplyPrice.PartNumber references PurchDB.Parts.PartNumber
  - PurchDB.SupplyPrice.VendorNumber references PurchDB.Vendors.VendorNumber
  - PurchDB.Orders.VendorNumber references PurchDB.Vendors.VendorNumber
  - PurchDB.OrderItems.VendPartNumber references PurchDB.SupplyPrice.VendPartNumber
  - PurchDB.OrderItems.OrderNumber references PurchDB.Orders.OrderNumber

To define these constraints in the database and retain the original index CLUSTERING characteristics, we would use these CREATE TABLE statements:

```
CREATE PUBLIC TABLE PurchDB.Parts
  (PartNumber        CHAR(16)          NOT NULL
                     PRIMARY KEY CONSTRAINT PartNumPK,
   PartName          CHAR(30),
   SalesPrice        DECIMAL(10,2) )
  IN WarehFS;

CREATE PUBLIC TABLE PurchDB.Inventory
  (PartNumber        CHAR(16)          NOT NULL
                     PRIMARY KEY CONSTRAINT PartNumPK
                     REFERENCES PurchDB.Parts (PartNumber)
                                CONSTRAINT InvToPartFK,
   BinNumber         SMALLINT          NOT NULL,
   QtyOnHand         SMALLINT,
   LastCountDate     CHAR(8),
   CountCycle        SMALLINT,
   AdjustmentQty     SMALLINT,
   ReorderQty        SMALLINT,
   ReorderPoint      SMALLINT )
  IN WarehFS;

CREATE PUBLIC TABLE PurchDB.Vendors
  (VendorNumber      INTEGER           NOT NULL
                     PRIMARY KEY CONSTRAINT VendorNumPK,
   VendorName        CHAR(30)          NOT NULL,
   ContactName       CHAR(30),
   PhoneNumber       CHAR(15),
   VendorStreet      CHAR(30)          NOT NULL,
   VendorCity        CHAR(20)          NOT NULL,
   VendorState       CHAR(2)           NOT NULL,
   VendorZipCode     CHAR(10)          NOT NULL,
   VendorRemarks     VARCHAR(60) )
  IN PurchFS;
```

```
CREATE PUBLIC TABLE PurchDB.SupplyPrice
  (PartNumber         CHAR(16)           NOT NULL
                      REFERENCES PurchDB.Parts (PartNumber)
                                 CONSTRAINT PartToNumFK,
   VendorNumber       INTEGER            NOT NULL
                      REFERENCES PurchDB.Vendors (VendorNumber)
                                 CONSTRAINT PartToVendFK,
   VendPartNumber     CHAR(16)           NOT NULL
                      PRIMARY KEY CONSTRAINT VendPartNumPK,
   UnitPrice          DECIMAL(10,2),
   DeliveryDays       SMALLINT,
   DiscountQty        SMALLINT)
  CLUSTERING ON CONSTRAINT PartToNumFK
  IN PurchFS;

CREATE PUBLIC TABLE PurchDB.Orders
  (OrderNumber        INTEGER            NOT NULL
                      PRIMARY KEY CONSTRAINT OrderNumPK,
   VendorNumber       INTEGER
                      REFERENCES PurchDB.Vendors (VendorNumber)
                                 CONSTRAINT OrderVendFK,
   OrderDate          CHAR(8) )
  CLUSTERING ON CONSTRAINT OrderNumPK
  IN OrderFS;

CREATE PUBLIC TABLE PurchDB.OrderItems
  (OrderNumber        INTEGER            NOT NULL
                      REFERENCES PurchDB.Orders (OrderNumber)
                                 CONSTRAINT OrderItemFK,
   ItemNumber         INTEGER            NOT NULL,
   VendPartNumber     CHAR(16),
   PurchasePrice      DECIMAL(10,2)      NOT NULL,
   OrderQty           SMALLINT,
   ItemDueDate        CHAR(8),
   ReceivedQty        SMALLINT )
  CLUSTERING ON CONSTRAINT OrderItemFK
  IN OrderFS;
```

Since each of these constraints causes the creation of a constraint index, the original indexes created on these tables are now superfluous. We have made the constraint indexes CLUSTERING wherever the original index on that column was clustering. The constraints have been given names similar to the names of the indexes they replace; our convention is to use the suffix PK for primary keys and FK for foreign keys.

Effects of constraints on applications

Other constraints may suggest themselves. For example, PurchDB.OrderItems does not have a primary key defined. However, we must trade off the creation of a constraint with its impact. Do we wish to have a primary key on PurchDB.OrderItems? Do we wish to pay the overhead of the constraint index this would entail? Since no unique index was originally created on this table, the answer to these questions appears to be "no."

Now that we have created the schema with the desired constraints, we can take advantage of the constraints in application programs. The statements affected by the creation of constraints are INSERT, UPDATE, and DELETE.

INSERT

When an INSERT is performed on a table containing a unique constraint, that action may violate the uniqueness of the constraint. Such inserts are not permitted by the constraint.

When an INSERT is performed on the referencing table of a referential constraint, it requires that the referenced table contain the value being inserted into the referencing table. This is because the referencing row will reference the row that matches it in the referenced table. If no such row exists, the insert will not be permitted by the constraint.

When an INSERT is performed on the referenced table of a referential constraint, the referential constraint cannot be violated. This action creates a new row in the referenced table that will not yet be referenced by any row in the referencing table. Thus, INSERTs into the referenced table cannot violate referential constraints.

UPDATE

When an UPDATE is performed on a table containing a unique constraint, the final result must leave all rows unique on the unique constraint columns. If it does not, the update will not be permitted by the constraint.

When an UPDATE is performed on the referencing table of a referental constraint, it requires that the referenced table contain the modified values in the referencing table. This is because the referencing rows will no longer reference the rows containing their old values, but will reference rows containing their new values. If no such rows exist, the update will not be permitted by the constraint.

When an UPDATE is performed on the referenced table of a referential constraint, if it changes values in the referenced columns, then the rows whose values change cannot be referenced by any row in the referencing table. Such an update would cause the referencing rows to no longer have a row to reference, so it would not be permitted by the constraint.

DELETE

When a DELETE is performed on a table containing a unique constraint, the unique constraint cannot be violated. This action removes rows from the table, and thus cannot create a duplicate row. Thus, DELETEs cannot violate unique constraints.

When a DELETE is performed on the referencing table of a referental constraint, the referential constraint cannot be violated. A DELETE removes rows from the table, and thus cannot create a new row in the referencing table that needs to match a row in the referenced table. Thus, DELETEs from the referencing table cannot violate referential constraints.

When a DELETE is performed on the referenced table of a referential constraint, then the rows deleted cannot be referenced by any row in the referencing table. Such a delete would cause the referencing rows to no longer have a row to reference, so it would not be permitted by the constraint.

Conclusion

In Codd's twelve rules for relational databases, rule 10, "Integrity Independence," requires referential integrity. Up to now in ALLBASE/SQL, such integrity has had to be managed by the applications. Beginning with Release E, ALLBASE/SQL now provides the features of referential integrity at the database level.

Providing unique and referential constraints in the schema definition language centralizes constraint definition to give greater application reliability and reduced development and maintenance costs.

The examples presented in this paper have shown how to determine the constraints in existing databases. These methods and more traditional database modeling techniques such as the entity-relationship model can be used to determine the constraints that should be enforced in a database schema.

Other constraints, such as check constraints on tables and views and general integrity constraints like SQL3 triggers, will further enrich the database schema language in future releases of ALLBASE/SQL.

Bibliography

ALLBASE/SQL Reference Manual

ALLBASE/SQL C Application Programming Guide

Batini, C., Lenserini, M., and Navathe, S.B., "Comparison of Methodologies for Database Schema Integration," *ACM Computing Surveys*, Vol. 18, No. 4, Dec 1986, pp. 323—364.

Codd, E. F., "Is your DBMS really relational?" *ComputerWorld*, 14 Oct 85.

Codd, E. F., "Does your DBMS run by the rules?" *ComputerWorld*, 21 Oct 85.

Melton, Jim (editor), *ISO ANSI Working Draft Database Language SQL3*, X3H2-91-55, Mar 91.

Paper # 2009

LAN Management; New Challenges and Choices

Russ McBrien
Hewlett Packard
100 Mayfield Ave.
Mt. View CA, 94043
408-691-5692

Changing Support Requirements

During the 1980s, mainframes and minicomputers were used for the 'mission-critical' and important corporate applications. They were supported primarily by centralized MIS departments with a great deal of concentrated expertise. People came to expect certain levels of up-time and performance from the systems and applications on which their businesses depended.

In general, the following rule applied: *the larger the system, the more critical the application and the more comprehensive the support.*

We at HP call this system-level support. It is a set of expectations for performance and up-time for the entire system. It stretches from user to CPU to networks, even telecommunication lines. And it supports the business functions which directly affect revenue and decision-making.

By contrast, PCs, workstations and the LANs that connect them grew up in a decentralized fashion They were purchased by individual departments in companies and usually run by an administrator or other PC/workstation "enthusiast" who had other job responsibilities besides looking after the computers.

Desktop PCs tended to support office functions such as word processing or spreadsheets. But they were rarely involved in critical daily business operations. Departmental resources were sufficient to deal with routine software, hardware and basic network administration tasks.

Workstations tended to be purchased by technical or engineering departments. Frequently there were departmental "*technical gurus*" who tinkered with the newest software and hardware, and took care of the networks, especially while they were relatively small and local.

In general, the network management and support needs of these PC and workstation LANs were satisfied with a fairly basic level of support. Centralized MIS departments were not interested or did not have time to deal with local issues.

This began to change in the late 1980s. The industry began to experience important new developments in the local area network arena. These developments are changing expectations about local area networks and network management for the 1990s.

PCs and workstations are being linked into larger and larger local area networks at an accelerating rate. PC and workstation LANs are not only growing, but they are growing in importance in the nature of the tasks they perform in corporations, pushing into more important and "mission-critical" applications.

More and more companies are "down-sizing" specific applications by taking them off mainframes and minicomputers, and running them on local area networks. Frequently these are client-server environments where the desktop PC links via the LAN to a server (PC, mini, or mainframe) that performs the "back-end" of computation intensive, database or communication applications. These applications are frequently mission-critical. They are the revenue producing applications on which a business depends for its operations; examples include: telemarketing, financial trading, insurance underwriting, or customer service.

Today the size of the CPU is not the only indicator of customer requirements for network management. As a result, managers must expect system-level support for LANS. With mission critical applications the business depends on system-level performance and up-time.

LANs Bring New Support Challenges

Studies show that LAN up-time today is well below that of minis and shared systems. An Infonetics study (1989) showed that the average network goes down over 23 times a year (twice a month) and stays down for 5 hours . An average campus-wide LAN is disabled 6 percent of the time. It is estimated that an average U.S. corporation loses approximately $3.5 million annually due to LAN downtime.

LANs frequently just evolved, rather than being designed and planned from scratch. Departments went out and bought a wide variety of equipment and software, and LANs just sprang up. LANs involve users who have an active role in the LAN's performance and up-time. Because of the traditional end-user independence associated with PCs and workstations, LAN administrators have to work closely with the individual end users to support them. For example, equipment moves or additions to the network involve users. Even when users just move their equipment around on their desks, they can bring a network down. User errors and carelessness (e.g., forgetting passwords, trashing files, disconnecting cables, printer parameter changes) also add to the network management burden.

Most departments are unprepared to deal with the more complex network management issues. Usually there is no established infrastructure for network support. LAN faults are often difficult to fix because LANs are dispersed throughout the company. When the local administrator is inexperienced and the centralized MIS department is overworked and thousands of miles away, the result is costly downtime and/or costly travel to fix network problems.

The Alternatives

Companies facing up to the growing need to deal with LANs effectively have several alternatives for network management:

Companies can perform all their network management in-house.

This provides the benefit of complete control over the management of the network and may provide a perceived level of independence. It can also become a large and costly commitment, involving the training of MIS and user departments. It means establishing procedures and setting up dedicated personnel and providing network support tools at local and remote locations.

A Gartner Group study shows that one dedicated full time person is required for every 40 nodes of users (one for every 150 if not supporting applications). Typical salaries of such people range between $25K and $50K per year.

Companies can out-source, or turn over their LANs to an external, third party to manage.

The benefits include worry-free network management and direct cost control. But other things may become concerns including: cost-effectiveness, loss of control over important areas of the company's business; the reputation and viability of the support provider; internal personnel and union concerns about eliminating jobs.

More and more companies are finding the answer in a balanced solution that is a combination of internal and external support.But how do you find this right mix .

As a first step companies need to perform a step by step assessment of LAN management requirements and the expense, both hard dollar costs and "soft" factors. To make an informed decision about network management, companies must look at the following areas and perform a rigorous cost-benefit analysis:

- o WHAT functions must be performed to support the LAN environment.
- o WHERE the support resources should reside.
- o WHO will provide the support.

It is important that these questions are addressed in this order.

WHAT?

This illustration breaks out the departmental portion of an integrated LAN. These same functions could be applied to a much broader network application.

LAN Administration	PC System Support/ Change Management	Administrative Support	User Responsibilities
Server operation	PC standardization	Communication to users	Communication
Disk management	Support/TS	HW/SW inventory	Problem diagnosis
Peripheral configuration	Communication	Support contract admin.	Info sharing
Server configuration	Escalation process	SW license admin.	Follow LAN rules
SW upgrades	Ongoing PC opt.	HW/SW ordering	Training
Capacity planning	New PC setup		Product champs
Server backup/rec.	OA purchase plan		Client backup
Security	PC services		Private file management
Config. document	Cheat sheet		
User backup/rec.	User training		
Public information management	User backup/rec.		
Admin. document			

A complete list of required support functions comes from exhaustive attention to the following three areas:

- *Technology*
  Standards
  Tools
  Vendors

- *Users*
  Dynamic, changing needs.
  Perpetual program review.
  On-going, open communications.

- *Business Control Needs*
  Purchase Control
  Inventory Control
  Security Considerations
  (Data and Physical)
  Disaster Recovery

Past experiences shows that companies tend to do a reasonable job of understanding the needed care and feeding of network technology but, are weaker at identifying the user needs and business requirements.

WHERE?

Support functions can be performed at departmental, campus or corporate levels. Network managers must determine at which level the system support functions are most effectively performed.

For most companies the direct and indirect costs of decentralized support are becoming intolerable. As a result I have seen a strong trend toward centralized support. This makes a lot of sense in that it leverages the scarce skills, tools and knowledge required for complex LAN management.

For example: A department of 30 users added a LAN using in-house, departmental support. The LAN required 100 hours of set-up and 20 hours of planning at an internal cost of $18,500. In addition an average of 85 hours per month is required for operations. In a project postmortem it was determined that large economies of scale could be realized by managing all LANs on a site-wide basis to leverage resources.

WHO?

In most cases, organizations implementing network capabilities attempt to reuse existing support resources for the new complex environment. Similar to navigating the seas without a compass, this is a very costly mistake.

For example: While a departmental secretary may grow into the role of "looking after" the LAN and handle setting up new users or granting file access rights, he/she is not trained or prepared for full scale trouble-shooting, security, back-up or recovery on larger LANs. In the case of enterprise-wide networks, the centralized technician may be thousands of miles away -- resulting in costly downtime, travel and repairs.

In-house/Departmental Network Support Personnel

The arguments for in house departmental staff are compelling. First is the familiarity in-house staff have with the network. And MIS managers typically have the valuable large system expertise now required to manage complex desktop computing environments. Second is the immediate availability of internal staff during regular operating hours. Finally there is the strong motivation of internal staff to maintain the network at its peak.

These benefits must be balanced against the high start-up and training costs associated with internal staff. Additionally, qualified people are becoming scarce. There may not be sufficient qualified personnel to meet all needs.

Total Outsourcing

The opposite to complete internal support is complete outsourcing of support. Outsourcing has delivered real financial benefits to some firms. It allows them to concentrate on their main business and devote their energies on staying competitive. It also provides these firms with the latest in support technology without a large up-front investment.

But complete outsourcing does not guarantee cost effectiveness ; especially for companies with large investments in tools and other network resources. Also, it is often too expensive for many small organizations and might be seen as something of a security risk for companies that have mission-critical applications on their networks.

Successful organizations will be those who carefully measure their in-house/departmental capabilities against their business objectives before implementing a network support strategy.

ASSESS THE REAL COSTS

Once an organization has evaluated the What, Where and Who of LAN support, it must assess -- and really understand -- the real costs. To find the right mix of support you must look at the cost of in-house versus vendor-provided support -- taking into account both the "hard" and "soft" costs of internal support.

HARD COSTS

To achieve high quality internal support you must pay the price for high quality personnel. First are the high start up costs. These come in the form of training and recruiting.. In one typical departmental LAN implementation, a Network Planner, a Network Administrator and a Technician were trained with total fees equaling $36,400.

On-going costs are associated with salaries and benefits required by trained personnel and their continuing education requirements. The following are average annual personnel costs (Fully loaded, Includes benefits, overhead, etc.):

Data Communications Manager $100,000 - $150,000
Network Planner $90,000 - $120,000
Network Administrator $90,000 - $120,000
Technician $70,000 - $100,000

These are based upon an informal survey of industry salaries, using an index of 1.5 to determine loaded values.

With self support you must also add in the cost of LAN management equipment and tools, both remote and local, which can be high (e.g., $25K to $30K for protocol analyzer tools.) Typically departments do not have the expertise or the resources for these sophisticated network management tools.

A recent study by Nolan, Norton & Co. (Computerworld - 11/12/90) provides some real world numbers for annual PC LAN cost projections. Hardware and Software purchase costs account for 30% of an organizations annual LAN expenditures. The other 70% consists of training, support, communications, etc.

To really understand the significance of these figures, one company used in the study over-shot their annual LAN budget by $8.0 million. $4.0 million had been allocated for equipment without any consideration for user and management needs.

SOFT COSTS

Next look at the 'Soft Factors' involved in LAN support. This includes the availability of internal staff for network management activities and internal turnover as network management experts are recruited away (even within the same company.) Look at the users and the applications they use; determine what uptime and performance is required; include lost revenue, productivity and opportunity costs in the calculations.

Calculate the cost of downtime. A study conducted by the market research firm Infonetics, found that LANs function properly 94 percent of the time. The other 6 percent - downtime - can cost a large corporation millions in losses each year. For example, a large HP customer recently saved over one million dollars per year by raising their LAN availability number by 1 percent (from 95 to 96 percent). Consider how important your applications are and the cost of network downtime becomes very real.

Finally look at economies of scale: a support provider can pass on the saving achieved from the economies of scale associated with handling LAN management in many different sites. Large vendors like HP already provide support from dispersed field locations around the world. Look for areas where you have low economies of scale. These are good opportunities for out-sourced support.

A firm must look closely at its business strategy and make a decision where to focus its information systems resources. Some companies decide that they must focus elsewhere, and let experts handle the specialized areas of network management; others decide that developing such expertise in-house is a part of their competitive advantage and worth the investment.

THE RIGHT MIX

99 percent of organizations with LANs are not prepared to address all of their support needs through either a total in-house or outsourced support solution.

In order to achieve optimized networks, companies must find a more realistic, balanced support solution that includes the right mix of in-house and external support.

For example: One HP customer with PC LANs linked to minicomputers at multiple sites has selected a support mix in which the internal staff handles problems that are easily resolved in less than an hour. More difficult problems are outsourced to HP. This avoids tieing up the staff in lengthy trouble-shooting activities.

Finding the right mix requires on-going analysis of an organization's business objectives and "real" network costs. Because of this, the most important component of most support program is the network manager. She or he is the single point of contact within the organization to oversee planning and communication activities. This person is responsible for continually assessing the "What, Where and Who" of system level network support in order to ensure maximum departmental productivity and efficiency. It's a tough job that it is only getting more complex.

Indicators of Success

In reviewing many service arrangement I have noticed several common success factors. These exist across industries and are associated with successful system level network support programs.

- Understanding the What/Where/Who of the network management.
- On-going analysis of the support functions.
- Clear employee and management communication.
- Clear expectations on performance and costs.
- An understanding of the scope of external services.
- Interaction between internal and external support components.
- Clearly defined internal staff roles.
- Support plans based on medium to long-term strategies -- not short-term needs.

LANs Are Changing, You Need to be Prepared

Today, LANs address important business needs and are becoming more complex. It can take up to 35 different vendors to implement a single eight node network. Companies that want to be successful with distributed LAN applications must work with their vendors to look closely at the alternatives for LAN management. LAN management technology and products are only a part of the picture.

Businesses must select the support solution that will help their business grow in the face of rapid change and build on existing investments. There is no right or wrong answer -- depending on company needs, cost-effective support can fall anywhere on a spectrum from self-sufficiency to vendor-reliance.

Troubleshooting LANs

Sam Sudarsanam
Applications Support Division
Hewlett-Packard Company
100 Mayfield Avenue
Mountain View, CA 94043 USA

Introduction

Today, many businesses are using personal computers (PCs) and engineering workstations to process data. Most often the data processing is done in a Local Area Networking (LAN) environment. Increasingly, LANs are being recognized as a strategic resource for business operations. E-mail, Electronic Data Interchange (EDI), and distributed databases are some of the essential LAN-based applications for businesses, and it is very important to optimize the network performance. In order to accomplish this strategic function, the network manager requires appropriate skills and tools to operate and maintain a LAN. This paper focuses on the common PC and workstation LAN problems and specific tools that can be helpful for effective network management.

Real Story

On May 13 1986, the London Stock Exchange in England crashed. This crash is commonly known as "Black Thursday." It was not the normal stock exchange crash that people associate with stock exchanges. Instead it was a PC Local Area Network crash. Just before the crash, a PC LAN was installed to support more than 30,000 transactions per day on the options market. On "Black Thursday" all the transactions were lost. The cause of this crash is still a mystery and no one was able identify the specific reason for the crash. Today, this PC LAN has been replaced by a larger Ethernet local area network. This real story illustrates both the vulnerability of a LAN and the dependency of the businesses on LAN for major data processing activity.

LAN Downtime

In 1989 Infonetics, Inc. did a study on the costs associated with LAN downtime. According to this study an average network goes down totally or partially about 23 times a year. Each time, the network is down for about five hours. Taking into consideration the lost productivity of the business, the study reported that on the average, the companies lose $3.48 million a year. Based on the study and their own data, Infonetics, Inc. has determined the average cost per hour of network downtime to be $30,000. Obviously it is very important to keep the LAN running as smoothly as possible to keep the business activities alive. This means that the responsible network managers must be ready and able to detect and rectify problems whenever they occur. The network managers need to be skilled and they need to follow a sequence of diagnostic procedures that will help isolate and correct the LAN

problems. By following the procedures and asking a few questions, a network manager can narrow the scope of the investigation. In some situations, by using some of these procedures in a proactive manner some future LAN problems can be avoided. The following procedures can isolate the causes of current problems and help prevent future problems.

Define the problem: This procedure may appear to be an obvious step but it is the first step to isolate the problem. Most often users of a network report that they are not able to communicate from their computer. To define the users problem, the network manager needs to ask a few specific questions, e.g., what the nature of the problem is and when exactly the problem has occurred.. The information gathered can either help recognize the cause of the problem or provide a basis for further problem isolation efforts.

Identify the symptoms: In a complex LAN environment a thorough examination of a the symptoms may provide the required information to isolate the problem. Determining the symptoms by identifying the occurrence of the problem -- whether the problem is occurring on a random or recurring basis. The network manager needs to correlate the symptoms to the hardware or software component of the network.

Use hardware indicators: Most LAN hardware components are equipped with a set of alert lights (indicators). Most of the time, noting whether a light is on or off can be an indicator.

Use built-in diagnostics: Most LAN devices contain built-in diagnostics that can be used to isolate problems. Some of these diagnostics range from a self test to different types of loopback tests.

Know the interface requirements: The key to identifying interface problems is to understand the interface requirements in a LAN. For example, a PC or a workstation may be connected to a twisted pair hub with a RJ-45 modular plug. The network manager needs to make sure that the interface incompatibilities can be rectified.

Examine event reports: Some of the LAN components are equipped with software utilities that generate reports. These reports can provide valuable information on the exact problem -- when it occurred and what was the nature of the problem.

Currently there are a variety of tools available to network managers to troubleshoot LANs. Some tools help monitor an over all LAN on a daily basis and some of the tools help in isolating specific media, devices, or application problems.

Common Problems

There are four types of problems that commonly bother network managers: hardware problems, addressing problems, configuration problems, and network congestion.

In LANs most of the problems are caused by faults in the physical media: the cables and interface connectors.

The following are some of the Hewlett-Packard Company tools that are very valuable to network managers to troubleshoot LAN problems.

HP 4972A LAN Protocol Analyzer

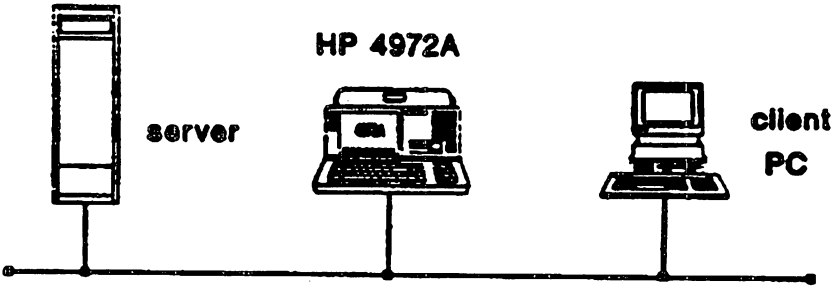


Figure 1

The HP 4972A is a protocol analyzer for IEEE 802.3 and StarLAN local area networks. The analyzer captures and displays frames integral to communications between networked systems and devices. Messages can also be transmitted in order to test system responses, identify active stations, or simulate heavy loaded network conditions. The HP 4972A resolves problems quickly and provides fundamental information for optimizing networked systems. Its many powerful features can be used to

* Resolve communication problems and verify solutions.
* Identify addressing problems and system incompatibilities.
* Analyze the behavior and performance of networked systems.
* Fully test products prior to network installation.

In order to check for errors on the LAN, the HP 4972A performance analysis application can be used by the network managers. The following summary information is displayed on the HP 4972A.

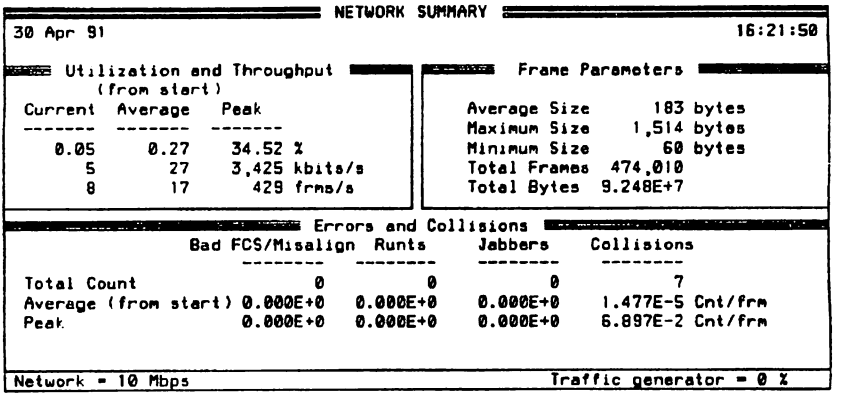


Figure 2

The display in Figure 2 shows the number of bad Frame Check Sequence (FCS) and number of collisions. It also shows valuable information on network utilization, throughput, and frame parameters, such as average size of the frame and total frames. Note that the healthy LANs do have some of the errors occasionally. The collisions that are shown in the display are not errors but, they are part of LAN operations (Ethernet and IEEE 802.3 LAN).

The HP 4972A protocol analyzer has several other features that would help address hardware problems, network addressing problems, and network congestion problems.

HP 4990S LanProbe Distributed Analysis System

The HP 4990S LanProbe Distributed Analysis System provides continuous preventive network management information without the presence of an operator. This system will give the network manager immediate, up-to-the-minute information about the state of the network on an on-going basis. A distributed monitoring system such as HP LanProbe complements the HP 4972A protocol analyzer.

The HP LanProbe distributed system enables a network manager to monitor all critical aspects of a remote or local Ethernet LAN. Completely independent of network equipment or protocols, the HP LanProbe system monitors, tests, and diagnoses every aspect of the network and presents the findings in clear color graphics.

The system consists of one or more HP LanProbe segment monitors and ProbeView™ software running under Microsoft® Windows.

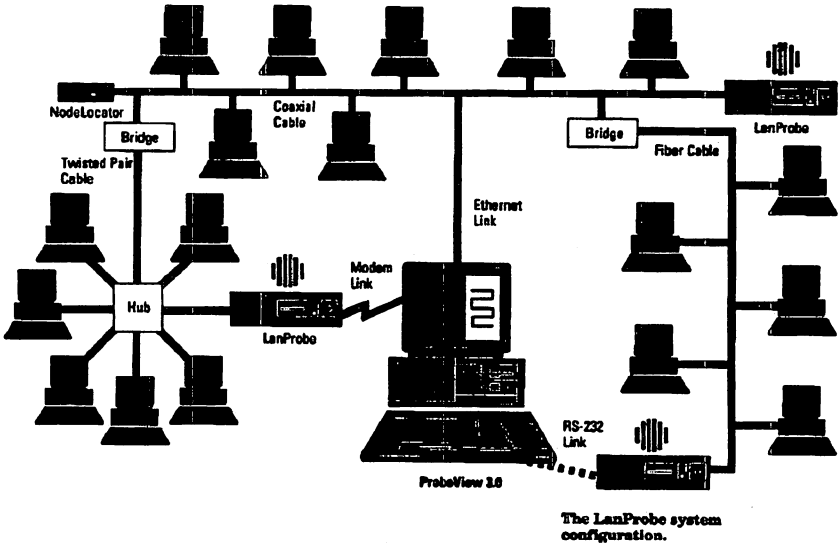


Figure 3

The LanProbe segment monitor (HP 4991) attaches to the end of an Ethernet segment and monitors all traffic. Network data relating to the segment is transferred to a workstation running ProbeView via RS-232-C interface, an Ethernet adapter, or a modem connection. HP ProbeView software, which runs on a PC/AT-class workstation, presents network information in graphical displays.

Figure 4 shows the segment map drawn by ProbeView identifies and displays devices that are active on the monitored segment.

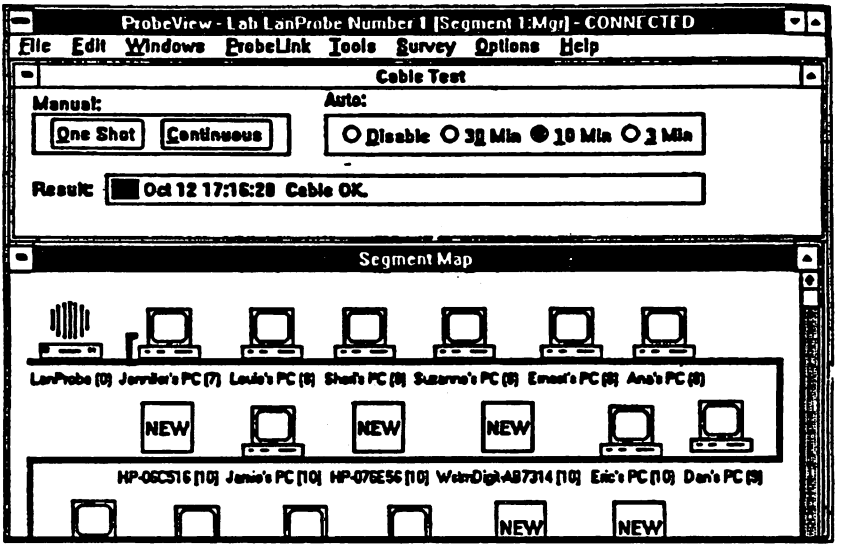


Figure 4

The HP 4992A NodeLocator option attaches to the opposite end of the cable from the HP 4991A LanProbe segment monitor. Using the coaxial cabling schemes, the HP 4992A NodeLocator automatically locates the position of nodes on the Ethernet networks.

HP LanProbe can initiate a call to HP ProbeView when a significant network event occurs. When an alert occurs, the HP LanProbe calls the HP ProbeView to notify the network manager of the alert. The alert is listed in the result box. Figure 5 shows a sample display of the fault description:

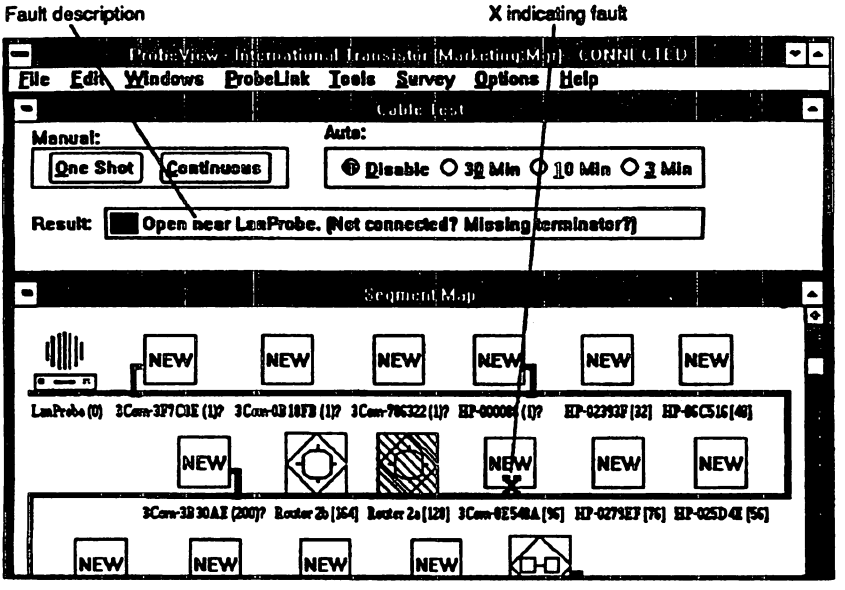


Figure 5

HP 4980A Network Advisor

The Series 386 HP network advisor defines a new class of network troubleshooting tools. It supplies expert system technology to dramatically reduce troubleshooting time. In addition, the HP network advisor offers a comprehensive set of network statistics and protocol decodes to speed problem resolution.

The Fault Finder expert system within the HP network advisor combines the practical experience of troubleshooting experts and computer automation to identify and solve common networking problems.

The Fault Finder's rule-based expert system takes user- furnished symptoms, then iteratively develops hypotheses and performs measurements until a conclusion is reached. The details of the problems, the Fault Finder's reasoning, and the potential solutions are displayed on the screen. In case the problem is not conclusively found, the HP network advisor will list the possible problems and leave a detailed record containing what the HP network advisor has learned about the network. This saves troubleshooting time because the HP network advisor, through deductive reasoning has eliminated possible problems.

The HP network advisor will check the network vital signs: utilization, FCS errors, number of nodes, Ethernet collisions, all Token-Ring MAC error frames and many other factors. The HP network advisor then will compare these vital signs against acceptable limits, and automatically investigate any potential problems.

HP Media Scanners

The HP J2187A Quick Scanner has a built-in Time Domain Reflectometer (TDR) that tells the network managers in few seconds exactly where and what the cable problems is and displays it on the screen. It works on virtually all twisted pair and coaxial LAN cabling systems. Also, it has the capability to indicate network activity on Ethernet networks.

The HP J2177A Pair Scanner quickly isolates the most common problems found in coaxial and twisted pair LAN cabling systems. It has a built-in TDR that tells the network manager in a few seconds the exact location of the fault or break and displays the results on the screen. The HP Pair Scanner adds problem isolation features designed specifically with twisted pair networks in mind:

* Quickly locates breaks, shorts, and bad crimps.
* Measures lengths of cable segments.
* Links test pulse generation for 10BASE-T hub activation.
* Builds a relay for transmit and receive pair testing.
* Detects inverted pairs using optional Multiline Injector.
* Identifies pairs from individual workstations using Smart-T kit.
* Automatically alerts the central PC.

The HP J2181A Cable Scanner is an easy-to-use, handheld tool that quickly helps the network manager to determine the cause of faults in LAN cabling systems. Powerful built-in TDR pinpoints the exact location of the fault or break, and the 32-character display reports the results on the screen. The HP Cable Scanner's color-coded keyboard allows the network manager to select various measurement. The built-in TDR is used to locate shorts or opens, or to measure the length of a cable segment.

The HP 28687A Wire Test Instrument makes twisted-pair cabling verification and troubleshooting easy by measuring the key parameters specified for the type 10BASE-T networks. Key measurements include

* Crosstalk attenuation.
* Signal attenuation over frequency.
* Burst noise.
* Continuity for both 25-pair and 4-pair bundles.

Conclusion

A clear understanding of the systematic procedures and the usage of the network tools will help the network managers isolate the problems quickly. There are various network tools available to solve various network problems. These tools are easy to use and the users need not require a great deal of theoretical knowledge of the technology. Information that is provided by the tools can help isolate the most common problems and keep the LAN always running.

References

1. The Cost of LAN Downtime, Infonetics, Inc., Sept. 1989.

2. Pozzi, M., "Problem Isolation Techniques for TCP/IP Network", Proceedings INTEREX HP Users Conference, Boston, MA, August 20-23, 1990.

3. Tait, P., "LanProbe Makes Diagnosing Ethernets Easy", Info World Vol. 11, Issue 35, August 28, 1989.

Paper #2011

Overview of Capacity Planning UX/VE/XL Systems

DAN STERNADEL
HEWLETT PACKARD PERFORMANCE TECHNOLOGY CENTER
8050 FOOTHILLS BOULEVARD
ROSEVILLE, CA 95678
916-785-8000

As MIS departments' responsibilities expand to managing multiple hardware/operating systems platforms, an understanding of modeling techniques can prove to be critical to the success of their operations. Basic techniques used in deriving input parameters for various platforms will be presented, followed by a brief overview of Mean Value Analysis modeling algorithms. Model results will then be reviewed in the context of computer systems planning.

As mini computers approach mainframe class performance, the importance of capacity planning has become very apparent. In the MPE/VE environment, projection recommendations have resulted in single department buying decisions in the $100,000 range while in the MPE/XL and UNIX environment millions of dollars may be at stake. The relatively small cost of implementing modeling techniques (through consultants or in house expertise) is money well spent considering what's at stake. No business can afford to make guesses about the heart of their operations; capacity planning techniques like those discussed below can assures a company's capital assets are being managed profitably.

Commercial systems MIS departments are generally familiar with the concepts of capacity planning, yet rarely actually implement a comprehensive capacity planning strategy. Often capacity planning is considered too complex, and therefore too costly to integrate into the tight budgets of typical computer installations. There are generally two misconceptions with regard to capacity planning:

1) It is the vendor's responsibility as a part of the sales process to size the system correctly for the given computing environment.

A decade ago it may have been true that vendors would assume responsibility in system sizing issues. But as a result of industry pressures, computers are becoming more of a commodity than a high priced specialty item. As a result, vendors cannot afford to bundle expensive engineering support services into the price of the system. This is the fundamental philosophy driving the UNIX* computing environment. Responsibility for system sizing has been shifting from the vendors to the MIS departments managing the systems. In large mainframe shops there is often a capacity planning staff that does nothing but evaluate system performance levels with regard to service level agreements. Smaller installations often cannot afford to maintain dedicated staffs and therefore rely on consultants to assist in their capacity planning needs.

2) Since computer price/performance ratios are improving, it is acceptable to acquire systems that are larger than necessary, allowing for reserve capacity.

Although there have been significant improvements in price/performance ratios, it is inappropriate to purchase too much excess capacity. In any business it is unwise to invest capitol in an unused resource, particularly one in which the technology is changing so rapidly; this affects the bottom line profitability of the operation. It is far more profitable to invest a relatively small amount in a capacity planning

* UNIX is a trademark of AT&T Bell Laboratories

exercise that assures the right size machine at the right time. An oversized machine ties up company capitol unnecessarily while an undersized machine adversely affects the productivity and profitability of your business. This also results in a self defeating cycle. How do you know how long the reserve will last ?

Before engaging in a capacity planning exercise it is extremely important to understand the relationship of the business objectives to the computer resources required to achieve those objectives. It is not enough to ask: "What will happen if I upgrade my CPU ?" This does not relate computer resource to the business. Perhaps a more appropriate question would be : "What will happen if I move payroll from SystemA to SystemB while adding 200 employees and 4 payroll clerks ?"

To answer capacity planning "what if" questions several techniques can be used, ranging from "gut feel" to elaborate benchmarks. Of course the accuracy of the prediction is commensurate with the effort and cost applied to arriving at an answer. Queueing Network models have been prevalent in the industry for several decades, and have proven to be a very cost effective method to answering capacity planning "what if" questions.

Mean Value Analysis

Mean Value Analysis (MVA) is a technique for solving a Queueing Network Model. The basic concept of MVA is that an average demand on a given resource will result in an average queueing delay. Once a model is correctly parameterized, MVA algorithms can be invoked to provide estimations of response time, utilizations, and throughputs. Figure-1 shows the basic components of an MVA model.

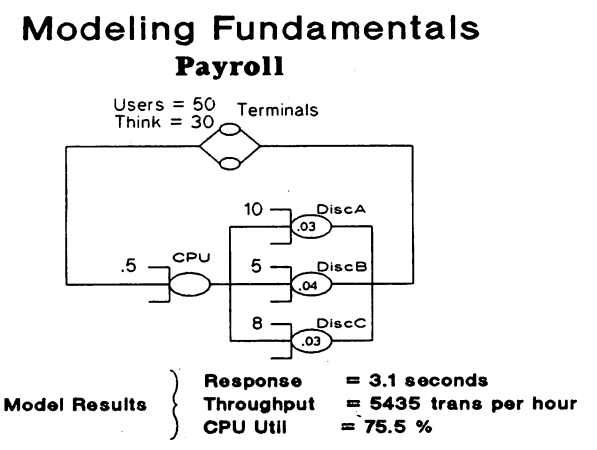


Figure-1

Figure-1 defines a payroll system consisting of 50 users with an average payroll transaction having a think time of 30 seconds, CPU utilization of .5 seconds, and 10 "visits" to DiskA , 5 "visits" to DiskB and 8 "visits" to DiskC. Disk A, B and C have service times of .03, .04 and .03 seconds respectively. Invoking MVA algorithms produces the modeled response time of 3.1 seconds, throughput of 5435 transactions per hour, and a CPU utilization of 75.5 percent. This is a very powerful technique for predicting how a system will react to a given workload.

It is extremely important to realize in Figure-1 that there are no operating system or hardware specific parameters for the model. For example, the model does not care if the CPU is a HP9000 series 850 or a HP3000 series 70 (or any other system). MVA models do not have direct knowledge of system specifics such as device drivers or operating system dispatcher algorithms; rather, they approximate the environment with queuing disciplines such as First Come First Serve (FCFS) or Preemptive Resume (PR). It is up to the modeler or modeling tools to determine which is the appropriate queueing discipline for a given system.

Representative Workload Characteristics

It is extremely important that the capacity planning question you need answered be clearly defined before beginning the exercise. It is almost impossible to answer the "What if I upgrade?" question without knowing what workload characteristics prompted the question. You must have some representative period in mind that is truly indicative of the workload characteristics of interest.

For example, it is not enough to say, "My average payroll CPU utilization for the past year was 30%." That would be like saying that the average temperature in Sacramento, California is 78 degrees. That may be true for the entire year but if that is what you based a moving decision on you would be surprised by the 110 degree summers and the sub-freezing winters. It would be more appropriate to look at the average temperature in January and July to get a true feel for the weather in Sacramento. Similarly the CPU demands for Payroll may be 80 percent when processing checks and only 10 percent during regular employee maintenance. Usually the peak period is used to build models, since this is the period that will experience performance problems first.

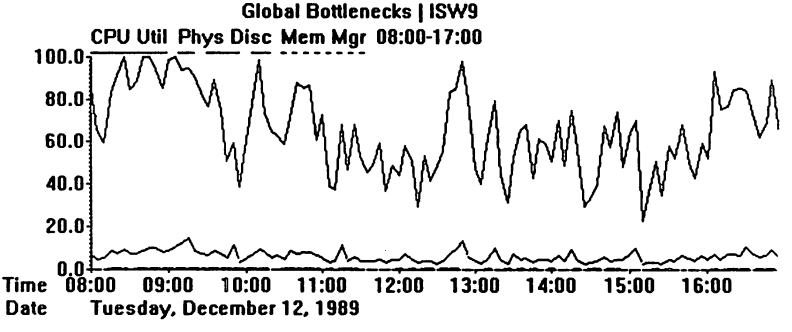


Figure-2

Figure-2 is a graph of an HP3000 running an electronic mail system. From this graph we can see different resource demands depending on the time of day. In the morning the users are logging on to the system and reviewing mail messages from the previous day. In the afternoon, system activity is more sporadic with users creating mail messages using various types of editors. If we were to model this system we would need to decide if we wanted to base our model on the morning workload activity (reading messages) or the afternoon workload activity (creating messages). A model based on the afternoon activity would provide results indicating that there is adequate system resource for additional users. But if those additional users were also going to be part of the users that are in the morning workload, the system does not have excess capacity.

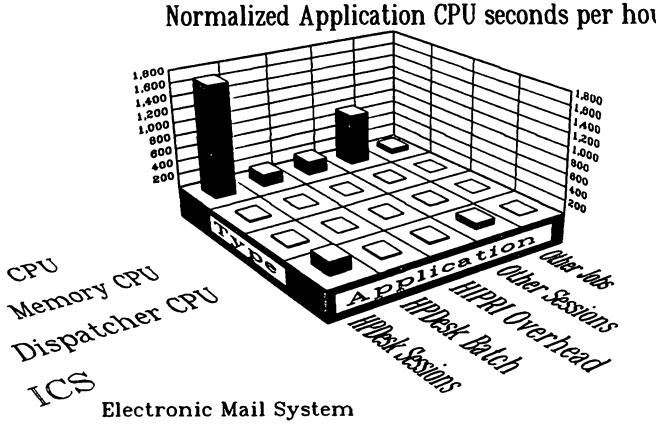


Figure-3

Once the appropriate window has been selected it is important to characterize the system in terms of its workloads. Figure-3 shows the electronic mail system's overall CPU demand characteristics for the morning workload activities.

If the requirement is for more electronic mail users, a model could be based on the morning workloads and then projected onto a larger system. Since there is an imbalance of workload demands throughout the day, perhaps load balancing modeling techniques could be used to take advantage of the excess afternoon system resource. Additional workloads from other systems might be scheduled into the afternoon.

A long term data collection tool is preferable for selecting a representative window. Great care should be taken in selecting a tool that can provide enough information from which models can be built, yet which does not incur significant overhead in the collection process. It is a risky business to base models on very short collection intervals, since you can never be sure you have a truly representative window.

Workload Classes

There are two basic workload types, or "classes", used in MVA models: Terminal and Batch[1]. Each can be used to represent a different aspect of workload characteristics. The modeler (or modeling tool) will determine the best type depending on the attributes of the measured workload on a system.

[1] A third type, Transaction Classes, are extremely complex and are omitted from this discussion.

Terminal Class

This is the easiest class to understand in a typical computer environment. A terminal class can be thought of as a user sitting at a terminal thinking for a period of time, entering some data, and waiting for a response. The input parameters are intuitively obvious:

* Number of Users
* Think Time
* CPU per Transaction
* Disk per Transaction

From these input parameters the model can calculate throughput (number of transactions per hour), utilizations, and response time. In the validation phase these results (outputs) can be compared to measurements for validation tests. In the projection phase any of the input parameters can be altered to reflect anticipated changes in the workload's characteristics. For example, the CPU per Transaction can be adjusted to reflect a different processor. The model can then be re-calculated to predict new response times and throughputs.

The modeler should use terminal classes when interested in modeling terminal type activities. This is not strictly limited to workloads that have terminals. For example, you may have workloads that wait on message files or have long pauses. From a modeling perspective think time can be thought of as the time that a transaction is waiting for some system event to occur. This could be a timer expiring, a read completion from a message file, or a carriage return from a terminal.

Some examples of terminal classes are provided below. Notice that the environments are architecture dependent, while the capacity planning scenarios are not.

In a HP9000/832 "heads down" data entry application the workload would have think times and users to parameterize as inputs. Terminal classes can be used to address "What if ?" scenarios such as:

* What will the throughput be if I increase the number of users?
* How will the response time change if I upgrade the CPU?
* How will throughput be affected if I alter the think time?

Consider an HP3000/949 application that makes inquiries to a parts database. The user at a terminal would check whether or not a part was in stock. This workload would have think times and number of users to parameterize as inputs. Terminal classes can be used to address "What if ?" scenarios such as:

* How will the average response time change if I upgrade my CPU's?
* What affect will reducing the disk I/O's per transaction have on response time?
* What impact will increasing the number of users have on CPU utilization?

An HP3000/70 insurance application program wakes up via a message file to add a new customer to the data base. The user fills out some external screen that processes all field edits. The completed "form" is submitted to the add program via a message file. In this case the think time would be the time between message file activations. This type of workload exhibits all of the characteristics of a terminal type class. Terminal classes can be used to address "What if ?" scenarios such as:

* What affect will increasing the number of submitting programs (users) have on throughput?
* How will CPU utilization be affected by a CPU upgrade?

Batch Class

From a modeling perspective, a batch workload is a terminal type workload with no think time. We assume that when a batch "job" is completed it is immediately replaced by another exhibiting exactly the same workload characteristics.

When we think of a batch job in a MPE environment we usually think of a JOB being submitted by the STREAM command. The problem is that jobs may run for a period of time that extends before and after the window selected for modeling. Unix systems can have several *nice* (background) processes running concurrently. As in the MPE environment it is difficult to align active *nice* processes with the selected modeling window. The modeler must be able to extract from any window without knowledge of when jobs/processes started or stopped. With this requirement we cannot accurately "measure" the actual number of JOBS that were executed in the modeling window.

Effective jobs is a term used to describe the productive time a job spent during the interval selected. A job that is consuming a resource (such as CPU or disk) is considered to be "responding" to the jobs demands, while a job waiting on a system external action (tape mount, console request, etc.), is considered to be non-productive, and therefore "not responding" to the jobs demands.

What we need is an independent indication of effective jobs that existed during the modeling window. Advanced collection techniques can measure the total time an application spent "responding" during an interval. For example, consider JOB A that was launched at 8 AM and finished at 11 AM. If our modeling window was from 9 AM to 10 AM we would see that 1 hour of time was spent responding in that window. In this case we would have 1 effective job in the 9 AM to 10 AM window:

1 Hour Response / 1 Hour Interval = 1 Effective Job

Now suppose that JOB B was submitted at 9:30 AM and did not complete till 11:30 AM. Since our measurement window is from 9 AM to 10 AM we would see we had 1.5 hour response in the window (JOB A + JOB B). We would now have 1.5 effective jobs:

1.5 Hour Response / 1 Hour Interval = 1.5 Effective Jobs

Consider one more scenario where there was a constant stream of JOB C's that were submitted. Each Job lasted 10 minutes and was immediately replaced by another Job C that lasted 10 minutes. This Job environment exists from 7 AM to 5 PM. We would now see that we had 2.5 hours of response in the 9 AM to 10 AM window (JOB A + JOB B + (6 * JOB C)). This of course results in:

2.5 Hour Response / 1 Hour Interval = 2.5 Effective Jobs

The most important point to note is that we are not concerned at all with how many "actual" jobs existed in the system but more with how many "effective" jobs existed in the system. In addition effective jobs only have meaning in the context of the measurement. For Example, 2.5 effective jobs only has meaning when talking about the system we measured. This effective job value is a result of some amount of demands placed on the system and its devices during the interval, resulting in some "response" that was collected for us.

It is possible that a stream of jobs will not consume the CPU if they have significant disk I/O activity. For example, if a job spends 20 percent of its time waiting on disk then it will not consume the total available CPU. If there is more than 1 job being streamed, then the available CPU may be consumed by the other population of jobs. This is basically how MPE/VE systems operated. On an MPE/VE system there was usually a disk bottleneck that provided an advantage to raising or lowering the JOB

limits to allow multiple jobs to have a better chance at getting the CPU while other jobs were waiting on the disk. That was one of the ideas behind the HPPA architecture - remove the disk bottleneck to allow performance to scale with CPU speed.

So now let's look at a job that runs on an XL system that does not have a disk bottleneck. Theoretically it will consume the CPU and therefore not provide an advantage to streaming more than one job at a time (multiple jobs will be waiting on the CPU and not have a chance to take advantage of disk wait times from competing jobs.) Of course, this scenario changes with the introduction of multi-processors, where additional jobs may be scheduled on different processors. Modeling techniques can be used to determine the effects of varying job limits in a multi-processor environment.

How about jobs on an XL system that have disk wait time? It should work the same way as it did on MPE/VE; the competing jobs will have a shot at the CPU while their peers are waiting on disk. If the competing jobs don't consume the CPU, then you need to determine if the cause is some type of locks and latches interaction. For example, if multiple jobs are trying to get at the same data base (or any data management facility that implements some type of multi-access control scheme), then the peers may end up waiting on the "latched" resources until they are available. This situation may result in it appearing as though there is "paused for disk" CPU resource available due to disk waits when the real cause is some sort of impede state.

Batch class examples are provided below. Remember that the modeling scenarios are independent of the system architectures.

Consider a typical HP3000 nightly batch processing environment where the job limit is set at 3 and there is a constant backlog of jobs waiting to be launched. In this case we would measure 3 effective jobs. Notice we do not know how many actual jobs existed, it could be 300 short jobs or 3 long jobs. It is enough to know that there were always 3 jobs running in the system. Batch classes can be used to address "What if ?" scenarios such as:

* What will my relative throughput increase be if the CPU is upgraded?
* What relative affect will increasing the job limit have on throughput?

Consider a UNIX system that has high session activity. The sessions will launch *nice* processes to generate reports. The session users pick up the reports some time later. A user job control facility limits *nice* processes at 5 and there are always report processes waiting to be executed on behalf of the session users. This would result in 5 effective jobs. Batch classes can be used to address "What if ?" scenarios such as:

* Will raising the limit on the number of *nice* processes increase the report throughput?
* What relative affect will upgrading the CPU have on the response time of report processes?

Logical Constraints

Depending on the workload it is not enough to know what it's demands are; we must also know about constraints from shared resources such as data bases. A typical system may have several different applications sharing a common data base. A bottleneck may occur at a logical access level rather than a physical device. Consider the situation in which a bank teller is waiting to update an account while the account data base is locked by another teller's update. The response time that the teller sees is a sum of the time waiting on the other teller's transaction to complete along with the time spent at the physical devices (queueing and service).

The problem is that the user processes within the application are forcing one another to impede (or block waiting for the resource) in order to ensure data base integrity (this happens directly through user level data locks and indirectly through global buffer locks, file system locks, and control block locks). A basic resource model (CPU and disk demands) does not account for the additional delays caused by data base contention. It should be noted that a user who is waiting for a data base is not present at any system device or in the queue of any system device, he/she must wait in an additional queue before beginning to compete for service at these resources.

Several techniques are available to deal with logical contention ranging from a less accurate delay technique to more advanced techniques such as Mean Population Limits (MPL's) and Domains.

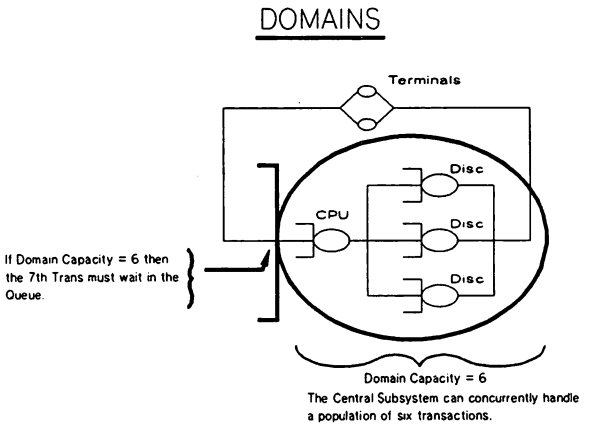


Figure-4

Figure-4 shows the basic concepts of domains. The central subsystem (heavy outlined circle) will have a logical capacity that will limit the amount of concurrent transactions that can exist within. When the number of transactions exceed the capacity of the central subsystem they will have to queue outside until the number of transactions within the central subsystem drops below the Domain's capacity. MVA algorithms can approximate average delay times as a result of Domain constraints.

Since applications often share a common data base, a technique for modeling the interaction of multiple applications sharing a data base would be advantageous. An advanced modeling discipline known as Shared Domains allows multiple applications to share a logical resource, providing the best intuitive model of a system.

Consider an electronic mail system that implements periodic "mail trucks" to send mail to remote systems. Since the truck application shares the same data base as the online users, it is desirable to have a model account for the interaction of the workloads with the data base. Shared Domains could be used to evaluate what effect increasing the number of electronic mail users has on the throughput/response of the electronic mail trucks, along with the data bases ability to support both applications. Modeling application interaction with shared logical resources is as important as modeling physical device interactions.

Validation

Validation is the most important step in a capacity planning exercise. It is the point where the model and the measurements meet. All of the assumptions about the operating characteristics of the system are put to the test. If the model validates, we can feel fairly confident that the mathematical representation of the system is sound.

Your local weather service uses modeling techniques similar to those used in computer modeling. Meteorological model inputs may be temperature, relative humidity, and barometric pressure. Model results could indicate if it was clear, foggy, raining or snowing. If the model predicts rain and a quick look outside reveals a sunny day, then something is wrong with the model. Problems could range from the model theory not being correct (maybe we are on Venus where meteorological theory would be different), to instrument inaccuracy (e.g., the thermometer was not calibrated correctly). It would be fruitless to use modeling techniques to make future predictions about the weather if you can't build a model that represents the current weather.

The same could be said for computer modeling. If your model cannot accurately reflect your current system environment, then making projections from that model would be of no value. For example, if we built an MVA model from system measurements we would expect the model to calculate a response time close to that of the measurements. If the response time modeled is very different from the response time measured then either the model assumptions have been violated or the measurements are not correct. If the model does not validate, the modeler will need to evaluate the significance of the deviations and, based on experience, either proceed cautiously with the model, or explore other capacity planning techniques. It is important that the modeler understand all of the assumptions that are made with regard to the model being able to accurately reflect system characteristics.

Projections

Once the modeler feels comfortable with the validation of a model, the "what if" questions can now be asked. Often single point scenarios such as CPU upgrades, or increasing the number of users for a particular application can be input to models.

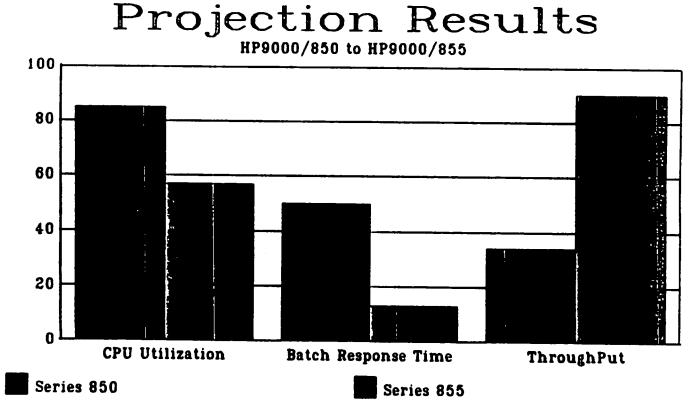


Figure-5

Figure 5 shows the result of a typical single point projection showing the relative changes that could be expected given a processor upgrade from a HP9000/850 to a HP9000/855.

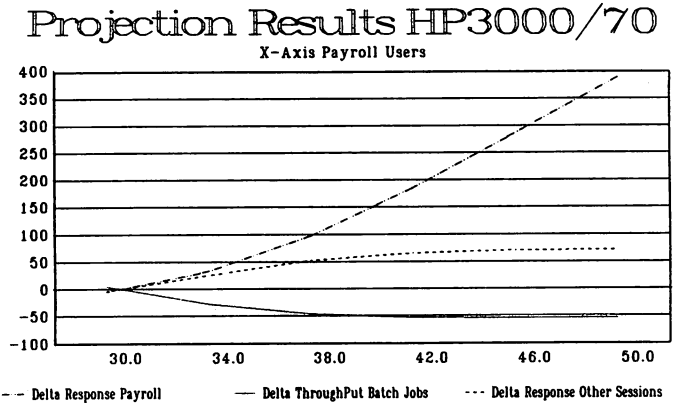


Figure-6

Figure 6 shows a more complex scenario where the number of users were increased in the Payroll Application. It is important to note not only the relative increase in response time of the Payroll application, but the effects on other applications such as the decrease in throughput of the lower priority jobs and the increase in response time of other applications that run at the same priority.

Figure-6 highlights one of the primary advantages of using queueing network models. These models not only make projections on workloads of interest, but also can predict relative effects on other workloads on a system. Other capacity planning techniques such as Bounds Analysis, and Straight Utilization projections cannot predict the interaction of workloads on a system.

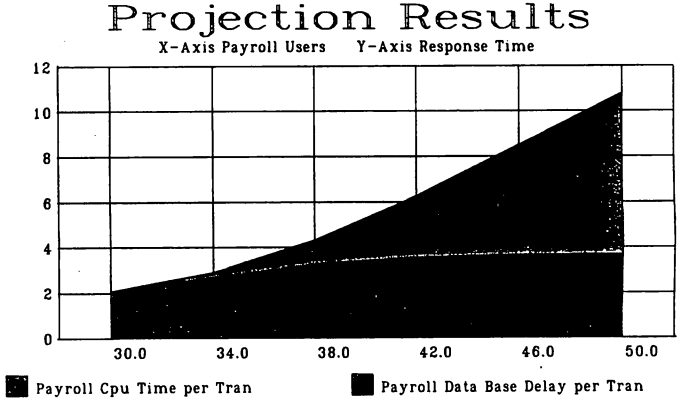


Figure-7

Often it is desirable to show changes in application characteristics over a range of variables. Figure-7 shows the results of a models projection of response times over a range of users. The components of response time are demonstrated here showing the effects of a data base Domain constraint. The lower portion of the graph shows the time spent at the processor, while the upper section shows time waiting in the data base queue. As the number of users increase, greater contention for the data base occurs which results in longer queuing times waiting to gain access to the data base. It is interesting to note that as the data base contention delay increases, CPU utilization begins to stabilize. Since the users are waiting for access to the data base, response may be poor while none of the physical devices are very busy. This demonstrates a significant advantage of analytic models over less accurate techniques.

Summary

It is important to keep in mind that the modeling algorithms are not system specific and can be utilized to model most computing environments. The key to success is getting the right measurement data in the correct form for model parameterization. As more data processing environments shift to a heterogeneous computing strategy, the need for a common system management strategy becomes apparent. Since most computer systems can be modeled generically, a common basis for measuring system performance data will be critical to multi-architecture capacity planning exercises.

Capacity planning can be a very complex subject. Although on the surface it appears to be a science,some consider it an art. In the old days, it was sometimes said that a capacity planner was always safe, since, when it came time to implement recommendations, the customer never really did what they were expected to do. Therefore, one ever had to go through the measurement exercise after changes were implemented to "validate" previous projections. Unfortunately, this meant the loss of an opportunity to learn how to do a better job.

Times are changing in the HP arena of capacity planning and system management. With the new generation of tools available, offering continuous collection and easy to use graphical user interfaces, system managers and MIS directors have a much better understanding of how their systems are performing, and are gaining access to some of the basic data required for capacity planning. Vendors are encouraging their customer's involvement with system performance, and are encouraging customer education in performance areas.

Any business's success depends on careful resource management. Network Queueing Models can be a powerful tool for managing the capacity of one of the most critical resources of any company - its computer systems. Developing and maintaining a capacity planning strategy for your operations (with either an in house staff, or by consulting services), assures profitable management of your company's assets.

For additional information:

Arnold O. Allen, *Probability, Statistics and Queueing Theory with Computer Science Applications, Second Edition,* Academic Press, San Diego, 1990.

Edward D. Lazowska, John Zahorjan, G. Scott Graham, and Kenneth C. Sevcik, *Quantitative System Performance: Computer System Analysis Using Queueing Network Models*, Prentice-Hall, 1984.

Michael K. Malloy, *Fundamentals of Performance Modeling,* Macmillan, New York, 1989.

Paper # 2014

The Impact of Emerging Fast Networking Standards on Document Image Management

Author(s):

Barney Hall
Andy Butler
Hewlett Packard
Pinewood Information Systems Division
United Kingdom
Phone: +44-344-773100

Colin Baker
Hewlett Packard
HP Labs Bristol
United Kingdom
Phone: +44-272-799910

Abstract

Document Image Management Systems, such as HP AIMS (Advanced Image Management System), are fast revolutionizing the ways that computers can manipulate information circulating around an organization. For the first time unstructured information such as correspondence, diagrams and photographs can be managed alongside existing structured information.

The use of image systems will be further enhanced by new technologies that enable easy exchange of image information across large distances. This paper will examine how image management and wide area networking could be combined to improve the efficiency and productivity of companies that are also distributed.

The paper will close by outlining some of the ways that improved image distribution techniques will benefit many industry applications, including healthcare and realty offices.

Introduction

An information explosion is affecting the business world. High volumes of paper-based information are a characteristic of even the most automated of today's offices. This avalanche of paper may contain the information you need, but it is sometimes impossible to access and, at the very best, slow and tedious to retrieve.

An Image Management system, such as the HP Advanced Image Management System (HP AIMS) puts an end to the paper problem by bringing information such as technical manuals, photographs, forms and handwritten correspondence online. Such information can be duplicated electronically and stored with other electronic data such as text and graphics.

A properly implemented image management solution will add the electronic capture, manipulation, storage and retrieval of hardcopy information to existing business information systems; it does not replace them.

The productivity payoff for the customer is the ability to get the right information quickly and easily. With one point of access to all the information they need, the time to respond to customer requests is reduced and better decisions made, giving organizations the leading edge in today's competitive world.

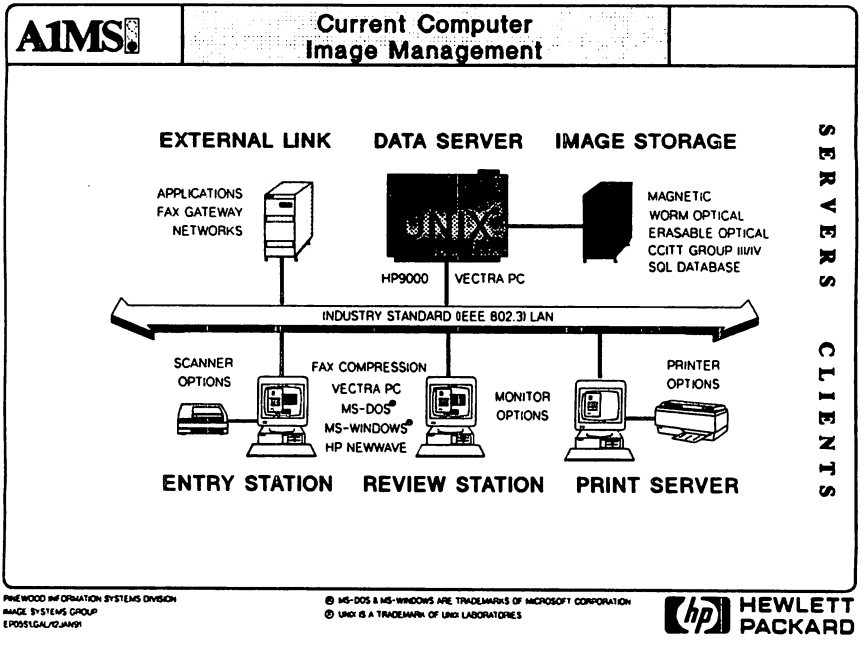
Any department, company or industry storing and processing paper can benefit from an image management system. Today's typical users are to be found in the insurance, pharmaceutical, financial, healthcare, transportation and telecommunications industries, and in government organizations.

For companies that are distributed it is still difficult to make this information available to everyone who requires it. By its nature images and unstructured data tend to be very large. Wide area networks can to be slow and costly. Therefore information that is required by people may not be available when required due to cost or time involved.

New networking technologies that are being developed are intended to be faster and more cost effective. These new technologies will allow companies to make the information available to more people who need it, thus enabling these companies to succeed in a worldwide market.

Current Networking and Image Management

Most image solutions today are based on local area networks, similar to the system shown below. This obviously limits the number of people who can access the information, but more seriously it limits the physical area from within which the data can be accessed.



Most companies today have offices distributed throughout the country if not throughout the world. Therefore they need to share the data between sites, separated by large distances. At present this can be done via external gateways, such as fax, or e-mail. These do not however allow immediate access to the data, thus decreasing the benefits of having the information on-line.

Local Area Network Technical Details

Three popular local area networks (LAN's) are listed below. The performance of these LAN's are variable. They do not need to set up a LAN level connection between the source and destination prior to sending packets. Also each packet of data is routed individually between the source and destination.

These are the networking platforms over which HP AIMS normally runs today.

Local Area Networks

	Availability	Transmission Media (current)	Transmission Rate (future)	Connection Type	Network Technology
Ethernet	Today	10 MBits/Sec	———	Connectionless	Packet Switched
Token Ring	Today	4- 16 MBits/Sec	———	Connectionless	Packet Switched
FDDI	Today	100 MBits/Sec	Research	Connectionless	Packet Switched

Existing Wide Area Networks

Currently, a customer wanting to connect a remote office to a head office can use various networking technologies, such as leased lines or X.25 connections.

Leased lines only allow you to connect the two sites that the line is leased between. If you want to add more sites, then you would have to lease extra lines to connect these into the network.

The line is leased from telecommunications companies for fixed periods of time and the user has to pay for the line at all times. Therefore, this network is normally only used between offices which require to exchange a great deal of information.

Public X.25 on the other hand can allow a user to connect to different sites, via the use of a Packet Assembler/Disassembler (Pad). The number of the remote machine is used to make the connection. Once the connection is made, it remains until the connection is broken by the user.

Public X.25 connections can transfer data at up to 64 KBits/Sec, in the UK and 56 KBits/Sec in the US. This is slower than leased lines, but the benefit is that the user only pays for the connection when he is using it. X.25 is useful if connection to the head office is only required on an infrequent basis, e.g. for batch updates overnight.

At the lower end of the spectrum there is the ordinary public telephone network (PSTN). By using modems computers can connect at speeds of up to 19.2 KBits/Sec.

Existing Wide Area Network Technical Details

As can be seen from the diagram below current wide area networks are slower than current local area networks. They are also connection orientated which means that the two machines that are communicating make a connection and then communicate over this connection. These types of connection are most useful for sustained data transfers as opposed to interactive queries and transfers.

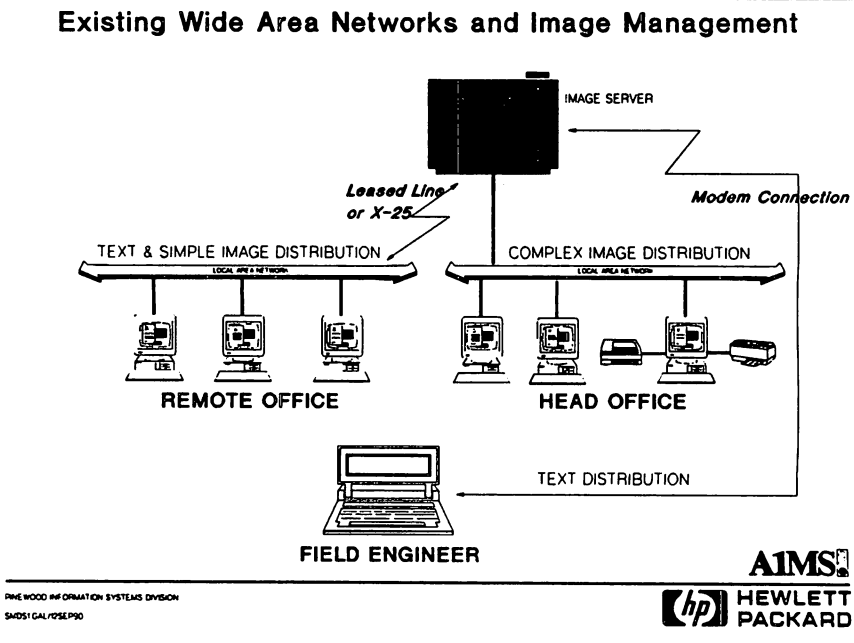
These types of wide area networks are commonly used for e-mail transport and terminal connections. Although these networks can be used for other general LAN uses e.g. file access their throughput can be prohibitively low.

Existing Wide Area Networks

	Availability	Transmission Media (current)	Transmission Rate (future)	Connection Type	Network Technology
Modem (PSTN)	Today	19.2 KBits/Sec	Research	Connection Orientated	Circuit Switched
X.25	Today	56/64 KBits/Sec	Research	Connection Orientated	Packet Switched

An Existing Wide Area Networks and Image Management Solution

The diagram below shows a possible way that organizations could enhance their existing image management configuration to include remote users, using today's technology.



The users at the head office would be connected on a local area network and the remote users would be integrated via a leased line or an X.25 connection, depending on the type of access required. If the remote users required continuous access to the information at head office then they would require a leased line. If on the other hand they only required to access the information on an irregular basis, then an X.25 connection would be more appropriate. However the disadvantage with this is that the X.25 connection is slower than a leased line.

The users in the remote office would probably retrieve less detailed information than their counterparts in head office, due to the time delay and cost involved.

People in the head office could browse through a set of detailed images to find the information they need. Whereas people in the remote office would probably browse through the structured information to decide which subset of the images they required. They would then browse through a group of low resolution copies of this subset, before choosing the image they require. Therefore the people in the remote offices do not get the same benefits from image management as the people in the head office.

Field engineers with portable PCs would be able to hold information on the PC and then use a modem to connect to the server to retrieve textual information. If the portable PC was capable of displaying images then the field engineer might also retrieve images but the speed of the modem connection might make this unusable.

Advantages

People in remote offices and the field can share company information

Disadvantages

Remote users have less functionality than those at the head office
Remote image retrieval is slow except over leased lines
Company needs to administer network
Difficult to add sites to the network

New Wide Area Networks Technology

New networking technology being introduced, will radically change the way people access information remotely. Three of the major new technologies will be ISDN, SMDS and Frame Relay.

ISDN, (Integrated Service Digital Network), allows users to connect two machines using telephone lines. Narrowband ISDN has limited availability today, and is capable of band rates of up to 64 KBits/Sec in the UK and 56 KBits/Sec in the US. Broad band ISDN which will become available later will be capable of higher band widths, probably in the range of 30 MBits/Sec up to 150 MBits/Sec. Once these bandwidths are achieved wide area networks will be as fast, if not faster than local area networks.

ISDN is similar to a leased line in that the user pays a rental charge for the line even when he is not using it. He also pays every time he sends information across the network.

SMDS, (Switched Multi-megabit Data Service), is a public network, which means that connectivity is not restricted within one company. It also means the ends of the connection do not need to be determined at the time of subscription. As new parts of the company wish to communicate it is only necessary for them to subscribe to the service, to have instant access to other entities.

Data sent over SMDS is routed on a per packet basis and as such the subscriber is only charged for the packets he sends. Therefore a high level connection can be maintained, but the subscriber is only charged when data is transferred. This makes SMDS very competitive against a leased line for companies who only transfer data sporadically.

SMDS is managed as a service and as such requires minimal network management by the subscriber. SMDS also has the capability to broadcast information to multiple sites, which will be useful for companies with many remote offices.

Most of the US Regional Bell Operating Companies, (RBOC's), have committed to providing an SMDS service, in the US, during 1992. This will probably run at 1.5 MBits/Sec at introduction.

SMDS will be scalable at a later date, to run at 45 MBits/Sec or perhaps 150 MBits/Sec. The subscriber will be able to choose the data transfer rate that they require by paying more for the higher speed links.

Frame relay is another networking technology that is being introduced within the next couple of years. Like SMDS it will operate at 1.5 MBits/Sec. The main difference between Frame Relay and SMDS is that with Frame Relay a connection needs to be set up before the packets can be sent, whereas with SMDS each packet is sent individually.

New Wide Area Network Technical Details

New Wide Area Networks are emerging that allow LAN interconnect and direct connect capabilities between multiple sites. Of interest to HP AIMS are SMDS, Frame Relay and ISDN.

From the diagram below we can see that SMDS has many of the advantages of local area networks. It is connectionless, packet switched and potentially it will be able to achieve data transmission rates comparable to local area networks.

Frame relay on the other hand sets up a connection and then transfers the information. The future bandwidths are not fully defined at present.

ISDN breaks down to Narrowband ISDN and broad band ISDN. Narrowband ISDN will be able to run at speeds of up to 2 MBits/Sec and broad band ISDN will be able to run at up to 150 MBits/Sec. ISDN also requires a connection between the two machines. The connection overhead for Narrowband and Broadband ISDN could be quite different. It is likely to be much lower with broadband ISDN.

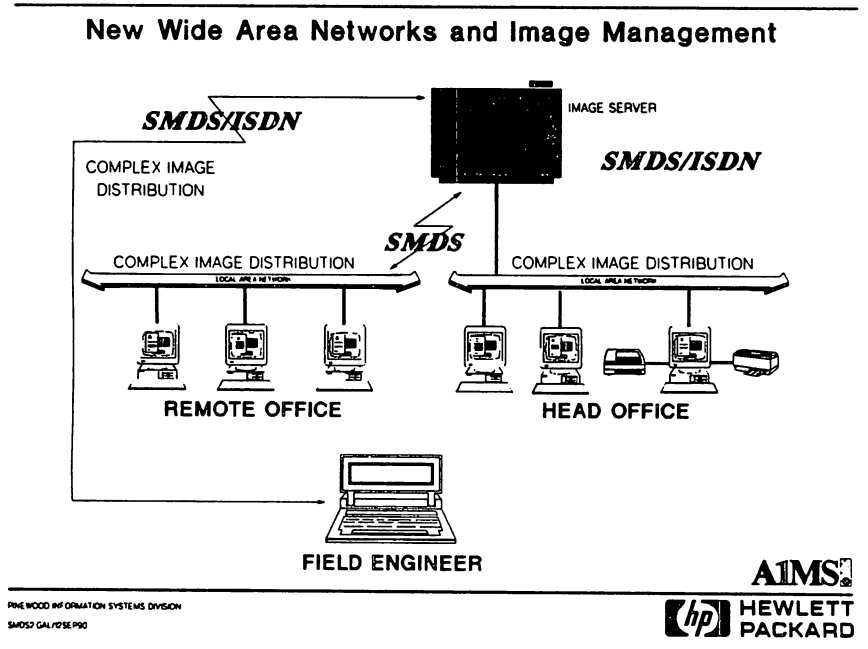
New Wide Area Networks

	Availability	Transmission Media (current)	Transmission Rate (future)	Connection Type	Network Technology
SMDS	Q2 1992	1.5/2 MBits/Sec	45/34 MBits/Sec	Connectionless	Packet Switched
Frame Relay	Limited	1.5 MBits/Sec	Research	Connection Orientated	Virtual Circuit
N-ISDN	Limited	56/64 KBits/Sec	1.5/2 MBits/Sec	Connection Orientated	Circuit Switched
B-ISDN	Very Limited	N/A	150 MBits/Sec	Lightweight Connection Orientated	Fast Packet Switched

As with LAN's, WAN's have different cost performance trade-offs. They also have different operational characteristics.

New Wide Area Networks and Image Management

The diagram below shows a possible way that organizations could use the new communications networks to enhance their existing image management system to include remote users and field people.



SMDS could be used instead of leased lines between sites. Being scalable, up to 34 MBits/Sec, in 1993 and even up to 150 MBits/Sec in the future, SMDS will give the performance required for people who need to access large amounts of information, especially images, from around the world. This will enable remote offices to access complex image data in the same ways as at the head office. Similarly, mobile engineers could use SMDS or ISDN from fixed locations to access complex image data.

This will not mean the demise of current wide area networking technology. For companies who communicate a great deal with remote offices, a leased line may still be a cheaper alternative. Companies will have to balance the cost of 'renting' the line, when they use it and paying for it all the time.

Advantages

improved communications for remote offices
access to centrally stored multimedia data for professionals in the field
easy synchronization of data within distributed organizations
other remote offices can be added easily
possibility of public access to the information if required
network managed by supplier, thus lower maintenance costs for user
user only pays for data transferred, so can be used for sparodic or continuous access

Disadvantages

Not available today
Due to the public nature of the network extra security measures may be required

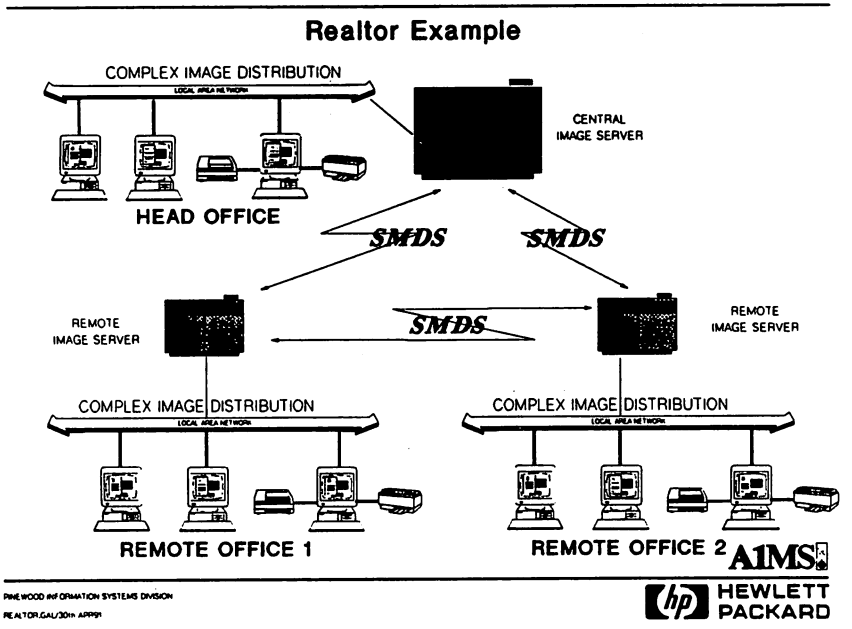
Potential Customers

Imagine how these capabilities could benefit these kinds of industries:

* Property information and photographs for remote realtor offices
* Branch insurance offices and mobile representatives
* Safety regulations in pharmaceutical and chemical companies
* Drug dependency/interaction data for doctors on call
* Clinics who need access to centrally stored patient records
* Hospitals who need the assistance of remote consultants
* Technical data for mobile repair professionals, such as:
  - > Telecommunications engineers
  - > Auto repair engineers
  - > Domestic appliance engineers
  - > Utilities (gas, water, electricity etc)
* Salesmen who need to access information on company products
* Fast reconciliation of information from remote retail outlets
* Access to criminal records data for police and immigration authorities
* Access to technical data in factories and workshops, such as:
  - > Auto industry
  - > Aerospace industry
  - > Defense
  - > Utilities (power stations etc)
  - > Petrochemical industry

Distributed Realty Service

Imagine the Realty company, who have invested in an image management system, to store photographs of the properties that they have available.



This realty has a head office, with an image server which is used for storing other information as well as images of the properties in the area. Each remote office has a smaller image server which is used to store images of the properties in the area.

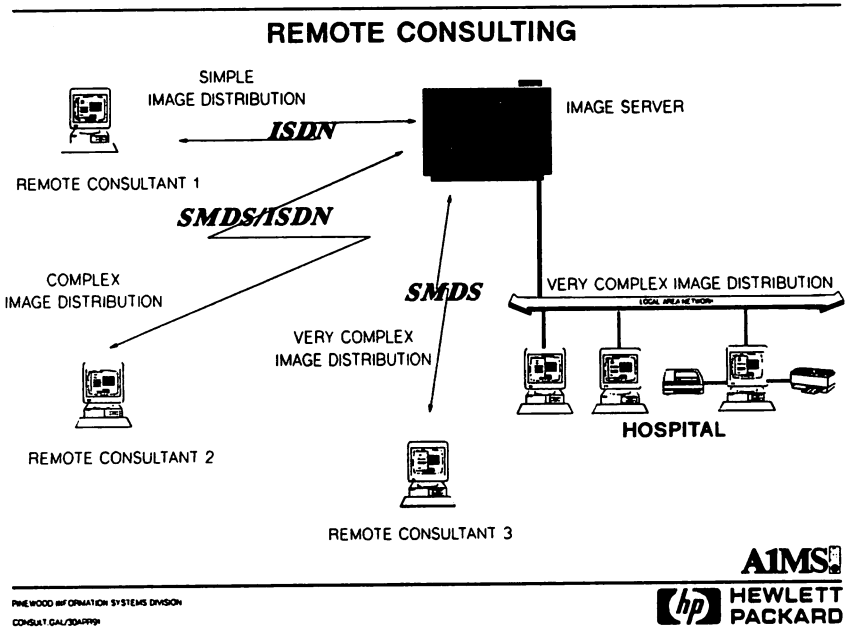
When a client wants to view the houses available in a specific price range with a certain number of bedrooms a search can be made on the database and the images viewed on the screen. The details and image of the properties that interest the client can then be printed out for the client.

If a client is planning to move out of state then his local office will be able to connect to the remote office via SMDS and search their database for properties that would interest the client. Again these properties could be viewed by the client and printed as required. It is not necessary for a request to be sent to the other office and then copies of the property details to be sent through the mail. The customer can pick up copies of the property details he is interested in right there in his local office.

The benefit to the realty company is that they can satisfy the customer request, they can connect to any remote office and they only pay for the transfer of the information from one remote office to another. There is no need for all the offices to be connected on a single network.

Remote Consulting

A hospital that stores its patient records in an image database, could allow consultants to dial in to assist with patient diagnosis.



Today it is possible for consultants to view very low quality representations of medical information such as x-rays. This allows the consultant to diagnose the patients illness effectively enough to keep the patient alive until he can get to the hospital. This could be done using current wide area networks or Narrowband ISDN. This is represented by the first consultant on the diagram.

The new networking technologies will allow much better representations of this medical information to be viewed by remote consultants. These consultants could have a specialized knowledge, which is unavailable locally and would be too costly for the consultant to visit the hospital.

The second consultant in the diagram has the capability to view structured information, images and reasonably high quality representation of medical images to enable him to make his diagnosis. This would probably be based on the first generation of SMDS or an improved version of ISDN. The third consultant has the capability to view structured information, images and very good representation of complex images, such as X-rays and mammograms to enable him to make his diagnosis. This will be made possible by the high speed SMDS

networks that will become available.

The benefits to the patients are a more accurate diagnosis of illnesses, plus a reduction in wasted time while qualified personnel are not available. The benefits to the hospital are that they can supply a better service at a cost effective price, calling on the expertise of consultants around the world. Many consultants have multiple patients in their care. Thus remote consulting also helps the consultant by limiting the amount of travel required. Thus remote consulting improves patient care at all stages from diagnosis trough treatment to recovery.

Conclusion

The new wide area networking technologies that are discussed in this paper will allow companies to use the capabilities of image management more effectively. As most companies are distributed the new high speed wide area networks will allow them to communicate cheaply and efficiently with remote offices.

Due to the public nature of the networks remote collaborators will be able to view information held on image servers. This will allow companies to call on specialized help or work closely with other companies on joint ventures. New company or intercompany structures that were not previously possible before will now become possible.

Companies benefit from the easy network set up and the capability to easily extend the network to include extra sites. The cost of the network is also controlled because the network is administrated by the supplier and the customer only pays for the network when he is using it.

There are multiple wide area network technologies being developed which will allow the customer to choose the network that best suits his needs. Whether a network is connectionless or connection orientated has a large impact on the design of a distributed system.

These new network technologies will allow companies to use image management systems to provide worldwide solutions to their paper problems.

TITLE: Enterprise-Wide Messaging in Open Systems

AUTHOR: Andy Watts

Hewlett-Packard

Nine Mile Ride

Wokingham, Berkshire ENGLAND

011-44-344-763-410

FINAL PAPER NOT AVAILABLE, HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO. 2015

Hewlett-Packard Company
Interface Technology Operation
1000 NE Circle Boulevard
Corvallis, OR 97330
503-750-2799 fax 503-750-4980

HEWLETT PACKARD

Intelligence in Graphical User Interfaces

by

Charlie Fernandez

Graphical user interfaces for computers have been around since Xerox PARC. In spite of the Xerox Star and Apple Computer's early efforts, GUIs remained mostly a home computer phenomenon until Microsoft Windows and IBM's Presentation Manager gave them official credibility in the PC world. Now, GUIs are *de rigueur* on PCs and even UNIX computers, long notorious for their user hostility, are adopting graphical interfaces and espousing the virtues of user friendliness.

What It Is

You may have heard the term graphical user interface or GUI before, but may not be exactly sure what it means. A good working definition would be something like the following: a method of communicating with your computer through the manipulation of graphically represented controls. More specific to our purposes here, a GUI provides access through direct manipulation of graphical controls to system level functionality. This definition draws a distinction between graphical user interfaces and what might be called graphical front-ends to application programs.

The alternative to the graphical user interface, what GUIs are currently replacing on desktops around the world, is the character-based user interface typified by the dreaded command-line prompt. To understand this budding popularity all we need do is think about what using each interface requires.

Why It Seems to Work

A character-based interface requires us to carry a lot of information in our head. Typically, learning to use a character-based interface is a long and painful process because we must memorize what a command does, what parameters it takes, and how to invoke it, including the correct spelling. Make no mistake, character-based interfaces are dependent on both our spelling and typing abilities.

In contrast, graphical user interfaces don't require a lot of typing skills to operate them successfully. Neither do they typically require a long and painful training process. The reason is that, by using graphics to represent real-life controls, GUIs rely on our previously learned skill set. And it's surprising to think of how minimal this skill set and the concepts behind it are: selecting a desired item from a menu, pushing a button to start a desired action or process in motion, sliding a scale to a desired degree, dragging an object to a desired location, pulling or pushing an object to a desired size and shape.

Because most of us already have these skills, GUIs have gotten a reputation of being easy to learn and easy to use. To the average non-technical, slightly computer-phobic user, the GUI offers an invitingly simple way to enjoy the gains of computing without the pains.

A Handful of Graphical Controls

All graphical user interfaces, whether to applications, some other specific computer operation, or the computing environment in general, are built using just a handful of graphical elements: 4 types of menus, 3 types of buttons, 2 types of scrollbars are really the heart of the graphical controls for every graphical user interface.

More important, though, than the number of controls, is the fact that each control behaves in a consistent manner. Namely, each time we press a push button, for example, the button visually appears to push in, then come out, and shortly thereafter an action starts. Also important is that, when operated, each control provides noticeable visual feedback, like the pushbutton pushing in then out. Without the benefit of visual feedback, the fragile metaphor of "direct manipulation" of graphical controls breaks and we are left, with a rising sense of inadequacy, wondering

whether it is the computer or ourselves who are in charge.

The Average GUI

Since a lot of what we do with computers outside of working in an application is manage information, and information is contained in files, most GUIs are built around file management. As a matter of fact, some GUIs are little more than file managers, graphical front ends to the computer's file and directory structure, with a few smarts added in.

If we try to form a composite picture of "the average" GUI, it would look like this: a file manager with a menu bar containing several pulldown menus, files and directories represented as icons, and several dialog boxes with which to control various settings. Functionally, our average GUI provides us with a way to move, copy, print, and delete a file through direct manipulation, by dragging the file icon to a specific location and dropping it. We can also choose to view our files in any one of several ways, search for and display files that match certain criteria that we have set with graphical controls, and change file attributes or properties also with graphical controls.

The good news is that using even an average GUI we can exercise a fair amount of control over the computer -- or at least the file system -- without learning a whole lot of computer mechanics. The bad news is that if we try to do anything beyond the basics of file management, for example, change the colors in our workspace or add our buddy's remote computer to our access list, we quickly fall through the graphical veneer of our average GUI to stand face to face with that ogre of interface hostility that is UNIX.

Signs of Intelligence

Fortunately for us, there exist GUIs with intelligence above and beyond that of the average GUI. What do these smart GUIs look like? Well, it's not so much their look as their behavior that determines their intelligence. A smart GUI is more robust in the graphical environment it provides and won't drop you into the underlying character-based interface the minute you go beyond file management. HP VUE in the UNIX world and Windows 3.0 in the DOS world are good examples of this. Both provide graphical services beyond simple file management including window management functions, graphical-based utilities like clocks and bitmap editors, help systems, session management, and environment customization.

Session management is a subtle but sure sign of intelligence. Session management means that the GUI is smart enough to remember the current state of your computing environment and to bring it back on command. Session management is smart because it is one of those things that adapts the computer to us instead of us to the computer. With session management, we don't always have to start each computer session back at some ground zero default state. GUIs like HP VUE and Windows 3.0 can bring back our previous computing session so we can begin where we left off.

If we're looking for GUIs with smarts, we should be looking for GUIs that help us focus on our work not on the mechanics of operating the computer. One bit of smarts in this direction is the association of data with applications. Several GUIs, among them Windows 3.0, Looking Glass, HP VUE, and X.desktop provide this capability. What it means to us is that when we want to do some work we simply point to the icon we want to work with and double-click the mouse button. Our data appears in a window on the workspace and we can get to work. We don't have to think about starting an application. We don't have to wonder whether our PATH variable is correct. We don't have to worry about what parameters we need. That's a whole lot of computer stuff we don't have to know about anymore, so we can go back to our real jobs munging data.

HP's NewWave carries the association of data and applications even further with hot links. A hot link enables us to take a spreadsheet, for example, and put it in the report we are doing in our desktop publishing system. Should we update the numbers in our spreadsheet, the report in our desktop publishing system changes to reflect the update. NewWave enables to further remove ourselves from the mechanics of computers; we don't have to think in terms of the format compatibility of data from this application with data from that application. Instead, we can think and deal with a single compound document.

Along these same lines but in a more general context is the whole idea of how we integrate our favorite

applications into our new GUI. We can expect our GUI to provide us with an icon to double-click, but smarter GUIs provide a number of alternatives from which we are free to choose the one that best fits our work style. Take HP VUE for example. HP VUE gives us the alternative of integrating frequently used applications into its front panel where they are easy to get to. HP VUE's multiple workspaces give us the alternative of creating task-oriented areas in which we run perhaps one application, perhaps a group of related applications.

The very concept of multiple workspaces is a good sign of intelligence because it allows us another degree of freedom to organize our work according to our work habits. We are no longer confronted with the computer's one-screen one-workspace dictum passed along however unwittingly by GUIs of lesser intelligence. HP VUE and Looking Glass are good examples. HP VUE gives us a workspace switch on its front panel; Looking Glass lets us create and save multiple session layouts for later recall.

Another sign of intelligence that we have mentioned but not discussed is context-sensitive help. We expect anything that claims to be user friendly to provide us with some level of help;but while all GUIs claim to be user friendly, there is a big difference between the online documentation that some GUIs call help and genuine context-sensitive help. The difference between the two can best be illustrated by the difference between our work partner telling us the phone number we have forgotten instead of throwing the phone book on our desk. Clearly if we are working away on our computer and come to a situation for which we require help, the most efficient answer to our question, the answer that will be the least disruptive and allow us to get back to work is the one that's immediate and context-sensitive. Online documentation, or simple online help without context-sensitivity, is not efficient enough -- regardless of any whizz-bang access method -- to answer our immediate question and allow us to get back to productive work quickly enough to avoid shifting our focus into the depths of the electro-reams of information contained in our online documentation system.

A Look at the Not-to-distant Future

Right now much of the focus of knowledge workers is directed toward our tools, computers, as well as our work efforts. This is true whenever a new technology appears on the scene. But basic desktop computing has been around for almost two decades now, so the emphasis is starting to change, and rightly so, away from the technology of computing, the toolness, and back to the work we use the tools to accomplish. What is happening to the technology? It's still there and improving every day, it's just that enough of the shine has worn off so we can get back to work.

Some of the technology we can expect to see in the future includes a more pervasive use of both object management and agents. Both of these technologies come to us from NewWave in the DOS world, but the future holds the promise of their migration to UNIX. Object management, with its association of data and applications into data objects, its hot links and compound documents, we have discussed before. The future may very well bring network-transparent location of objects and cooperatively shared objects. In a typical scenario, two or more of us, at widely separate locations can create a compound document together in real time from scratch, each modifying the other's data into a final satisfying form.

Agents can be likened to a type of direct manipulation, automated batch file. We can create an agent who scrupulously records our every move until we're finished with the series of tasks we want the agent to remember. Later, when we want to repeat the task, we simply tell the agent to do it. HP's 1995 video makes extensive use of agents in a dramatic scenario. If you haven't yet seen the video, it's worth finding a copy to view. 1995 isn't that far away.

As computers continue to make more information available to us knowledge workers, computers will also continue to be our main method of accessing that information. While online documentation is not a good substitute for context-sensitive help, online information will play an increasingly more important role in our future. From the GUI standpoint, the future will hold some interesting advances in our ability to access online information. We can readily imagine the use of common metaphors like the library to ease the access learning curve and provide us with a conceptual framework with which to envision the electro-reams of data available to us out there in cyberspace.

Another, related area of technology that is already beginning to receive attention is the area of multimedia. You can

tell it's receiving increased attention because everyone is wondering what exactly it means. For our computing purposes, lets just call it the ability to use multiple types of presentation media (text, graphics, animation, audio, video, video conferencing, and the like) at the same time on the same computer display. While multimedia may seem rather far fetched, most of the required technologies are already in existence. What's missing are things like industry standard protocols for the different types of media, the interoperability between media, indexing tools to access multimedia data, and the ability to compress data, particularly the video data, highly enough for efficient storage.

As a general summation of what the future holds for us knowledge workers, I think we can safely say that we will be able to feel as comfortable with our computers as we currently do with our cars. There's a lot of technology hidden under hood of each, but we don't have to know what it is, that may be an interesting hobby we're so inclined, but it's not our job. Just as the car is a transportation tool with a standard user interface -- we can get into any car in the parking lot and identify all the major controls (steering wheel, brake, accelerator, and perhaps clutch) sufficient enough to drive -- so too the computer is an information tool whose user interface is becoming standardized on particular graphical controls. In a very short time, we will be able to sit down at any computer in the office (PC, workstation, X terminal) and identify the major controls sufficient enough to use it productively. That's the goal out there in the future; that's what intelligent GUIs are leading us toward.

Paper Number 2017

Core Dump Analysis

Paper Written and Presented by:
Mark DiPasquale

19447 Pruneridge Avenue
Cupertino, California 95014
(408) 447-0911

Introduction and Objectives

There you are, crunching numbers, doing database transactions, or writing a paper, when all of a sudden, your system panics or has a High Priority Machine Check (HPMC). In other words, the system crashes! Although this is a very rare occurrence with HP systems, system crashes can happen to even the best of systems from time to time due to hardware or software problems.

In the event of a system crash, the software is set up to generate and save something called a *"core dump"*. Somewhere buried in this core dump is the answer as to why the system crashed. Some questions you may ask yourself at a time like this are: How good is my support network? How long will I have to wait to get this problem solved? Just what is a core dump anyway?

By now you may have called your response center to report the system crash. The response center will want to obtain a copy of the core dump files from your system. When these files are received, analysis tools are executed on the them to determine the cause of the problem. Sometimes a hardware fault is detected and a new part must be installed on your system to remedy the problem. Other times, a software bug is found that will need to be fixed by an engineering team. However, in some cases, your system may not be the first to experience a particular software problem, and there may already be a patch available to fix your system.

In this chain of events, there is an opportunity for us to shorten problem resolution time by supplying the *vital statistics* of a core dump to the response center as soon as possible. This paper has been written to teach customers to do just that. Herein, I will teach some of the basic skills required for core dump analysis. Armed with this information, you may be able to decrease problem resolution time.

More specifically, you will learn:

1. How to Obtain Core Dump Files
    i. Setting Up Your System to Save a Core Dump
    ii. Manually Picking Up a Core File from the Swap Partition
    iii. Problems with Saving/Obtaining a Core Dump
    iv. Accomplishing a Transfer of Control (TOC) In Case of a System Hang
2. Cursory Checks of Core Dump Files
    i. Core File Size Requirements
    ii. Symbol Information
    iii. Matching Core and Object Files
3. Core Dump Analysis Tools and Methodologies
    i. Knowing the Tools of the Trade
    ii. Core Dump Analysis
4. Summary / Recommended Action Plan

Core Dump Analysis Tools Discussed Here

The primary tools used for core dump analysis are adb and analyze. Adb is a supported utility that is available on every HP-UX system. Analyze is a contributed tool which also provides useful core dump analysis information.

QuickLook (ql) is another contributed tool. Ql is a script program that acts as a front-end to run adb and analyze automatically. To understand what ql does, you may want to read parts of the adb and analyze manual pages. However, to be successful in using ql, this is not necessary.

The contributed tools are available from the Interex contributed software library.

References

i. adb on-line manual page
ii. analyze on-line manual page
iii. savecore on-line manual page
iv. System Administration Tasks Manual, part number B2437-90006

Note
The manual references listed in this paper are current as of HP-UX Release 8.0. Titles of manuals, part numbers, and chapter locations may change from release to release.

Guide to this Paper
Unless otherwise stated, this paper presents information that applies to all HP-UX systems.

Use of fonts:

- Normal print: The text for this paper will be in normal print unless other emphasis is required.
- **Bold** print: This denotes important emphasis on a subject, or word, or the name of a command.
- *Italics* print: This denotes minor emphasis.
- `Constant width` print: This denotes output from the system.

Credits
Some of the material contained herein has been derived from, "Debugging an HP-UX Kernel", by James O. Hays.

How to Obtain Core Dump Files

What is going on behind the scenes when a system "crashes"? It could be that a software problem within the kernel has caused a system panic. Or, perhaps a hardware problem has caused a HPMC to occur. In either case, the system will attempt to leave a snapshot of all physical memory and register information on the primary swap device before it stops running. This snapshot can assist engineers in determining the cause of the malfunction because it holds a record of what the system was doing before it crashed. The correct name for this snapshot is, *core dump*.

A core dump is composed of two files, a core file and an object file. The core file is an "image" of the system's physical memory and register information at the time of a crash. The object file is the kernel file, better known as, */hp-ux*.

To retrieve a core dump, the program /etc/savecore must be executed. Savecore will retrieve the core file from the swap device, along with a copy of the system's kernel file, and save both in a specified directory. The core file and the kernel file make up the *core dump pair* (i.e., hp-core.N and hp-ux.N -where N is a number that associates a core dump pair).

The analysis tools, adb, analyze, and ql, require that both members of a core dump pair be present. In addition, for the analysis tools to be effective, it is very important that these members match. This is because the kernel (hp-ux.N) file contains symbol table and SOM (System Object Module (object file)) information; both of which are used by the analysis tools as road maps into a specific core (hp-core.N) file.

Let us begin by looking at the two methods used for saving a core dump.

Setting Up Your System To Save a Core Dump

The best way to handle a problem is to be prepared for it in advance. By default, most of the preparation for a system crash has already been done for you. When Hewlett-Packard ships an operating system, the /etc/rc script is set up to save core dumps automatically, should they occur. However, we should check the following:

1. The save_core() function in /etc/rc should be enabled (i.e., not commented out or removed). Make sure that the call to /etc/savecore is in this function. Also, make sure that the directory specified for saving the core dump meets the following criteria: It must exist, have proper permission settings (to allow writing), and the associated file system partition must have enough room to dump a core image the size of physical memory. This directory (usually called, /tmp/syscore) must be created by the system administrator. (See chapter 5 of the System Administration Tasks Manual for help with managing the file system. See the savecore(1m) manual page)
2. Make sure that the primary swap device has enough room to receive a core image the size of physical memory. (See chapter 6 and appendix B of the System Administration Tasks Manual for help with managing swap space)

Manually Picking Up a Core File from the Swap Partition

If you have elected to shut off the save_core() function in /etc/rc, you can still run savecore(1m) manually. Let us assume that a system paniced or had an HPMC, and is now in the halted state. Thus far, we have stopped the autoboot process. Here is an example of manually obtaining a core dump from a Series 800 machine:

```
Interact with ISL? Y
ISL> hpux -is (address) /* to boot single user mode after a crash */
# /etc/fsck -p          /* to fix the file system */
# /etc/mount -a         /* to mount all disks (maybe "-a -t hfs") */
# /bin/bdf              /* find enough space for the dump */
# mkdir /tmp/syscore    /* assuming /tmp has enough space */
# cd /tmp/syscore       /* go to the dump directory */
# /etc/savecore .       /* savecore in the current directory */
```

(See chapter 3 of the System Administration Tasks Manual for more information about ISL and boot-up.)

Savecore begins by reporting the date and time of the crash. Next, it looks in the specified directory for a file named, *bounds*. The bounds file contains the next sequence number (*N*) which savecore will use to create a unique core file and kernel file. Savecore will copy the core image from the primary swap device to a file named, hp-core.N. Last, it copies /hp-ux to a file named hp-ux.N to complete the core dump pair.

Problems encountered in Saving/Obtaining a Core Dump

If a core dump pair is incomplete (or not saved at all) after a panic, we can look to the savecore(1m) manual page for help. In general, there are three problems that can occur when attempting to save a core dump. Below I will state these three problems and offer reasons for why they might occur:

1. *Savecore did not run:*
   - There is no core file on the swap device.
   - The savecore command may have been commented out of (or removed from) the /etc/rc script. Hence, savecore did not automatically execute during boot-up to multiuser mode.
   - Savecore has already saved a core image. When a crash occurs, a kernel-specific pattern is written to the swap device along with the core image. Savecore checks for the presence of this pattern when it is executed. Once the core image is recovered, savecore deletes the kernel-specific pattern. This prevents savecore from running

again when the system is rebooted and no new crash has occurred.

- The kernel which has been rebooted does not match the kernel that was running at the time of the crash. Savecore can be *fooled* into thinking that no core file exists. This goes back to that kernel-specific pattern mentioned above. Savecore looks at the running kernel and the amount of physical memory installed to determine the location of the core file on the primary swap device. With that information, savecore looks for a kernel-specific pattern at a particular address. If a different kernel is running, savecore may look at the wrong address and think that no panic occurred. However, if savecore could find the image on the swap device, but the running kernel's version string does not match that of the core image, savecore warns the user.

  To prevent this problem, you can specify an object file argument other than /hp-ux, which is the default. See the *System* option in the manual page for savecore.

2. *Savecore ran, but did not save a complete core image:*

   - Savecore directory trouble: The directory (e.g., /tmp/savecore) may be on a file system partition that does not have enough room for a core dump pair. In this case, a partial, perhaps unusable, core image would be saved.
   - There is not enough room on the swap device that savecore is told to use. In this case, it is only possible to save a partial, perhaps unusable, core image.

3. *Savecore ran, but did not save a core image at all:*

   - Savecore directory trouble: The directory (e.g., /tmp/syscore) may not exist or may not have write permission.
   - Savecore is executed after the system has been running in multi-user mode for a while. Once the system starts back up, it is free to start swapping over the swap device. This could corrupt any core image written there.

Remember, it is very important to get a complete core dump in order to get the maximum value out of the analysis tools. This is because many of the dynamically allocated system structures (u-areas, kernel stacks, mbufs, etc.) are allocated from the global memory pool, which could be located in high physical memory address space.

(See the savecore(1m) manual page for more information)

Accomplishing a Transfer of Control (TOC) In Case of a System Hang

A *system hang* is a situation where the system seems to be up, but does not respond to external user control. Should this happen to your system, the response center will want to obtain a core dump so that the cause for the hang can be analyzed. The easiest and best method for obtaining a core dump of a kernel in this state is to use the Transfer of Control (TOC) mechanism. The TOC mechanism causes the machine to vector through an special address which will cause the machine to dump core. Most machines have the capability to do some sort of TOC; however, the methods for performing this task are machine-dependent:

- Series 800, Models 840, 850, 855, 86x, and 870: If you have an Access Port connected to your machine, then you must enable it through your front panel. Following that, you may type a "Control b" on the console. This will put your console under the supervision of the Access Port. You will get a "CM>" prompt. At this prompt you may type "TC".

  If you have an Series 800, Models 840 that does not have an Access Port, you may generate a TOC through the diagnostic DIP switch on the system monitor board. This is the left-most board when looking into the front of the machine. Setting the third switch from the top to the right will cause a TOC to occur when the reset button is pressed.

Once the TOC cycle has completed (might take a minute or so) and the system has rebooted, be sure to reset this switch back to its normal position. Leaving this switch in the wrong position will cause the machine to fail its self-test on the next power up.

- The following systems have a key-operated TOC mechanism: Series 800 Models 6xx, 834/5, 845, 8x2, and 8x7. To execute a TOC, turn the key all the way to the right (clockwise).
- Finally, the Series 800, Model 808 and 815 have a button-operated TOC mechanism. From the rear of the machine, look for this button on the lower right-hand side (it will be marked TOC). You will need an object, like a pen, to actuate the TOC button.

HPMC Related Hang: When a system has an HPMC it will dump core, if possible, and then reboot. If a second HPMC is encountered while it is handling the first, the machine will lock up. Machines that have a hex front panel display may indicate the type of HPMC that occurred. However, the normal case is that only one HPMC occurs, and a core image is left on the swap device.

Cursory Checks of Core Dump Files

Let us start this section with the assumption that we have a core dump to analyze. Now, before we use our analysis tools or send this dump to the response center, we should make sure that we have a good core dump pair to work with. It is good to know this information up front as it will save everyone's time.

Core File Size Requirements

Make sure that the size of the hp-core.N file is equal to that of the machine's physical memory. Since the core file is an image memory at the time of a crash, its size must be equal to the machine's physical memory size. Consequently, the hp-core.N file should be an even multiple of a megabyte.

Use the command "ls -l /hp-core.N" to check the core file size:

```
# ls -l hp-core.0
 -rw-rw-rw-  1 root  other33554432 Nov 23 07:25 hp-core.0
 # bc                /* get our trusty calculator */
 33554432/1048576    /* divide the core size by 1 megabyte */
 32                  /* to get the physical memory size in megabytes */
```

Symbol Information

Make sure that the hp-ux.N file has not been stripped. It may seem silly, but we have had object (kernel) files given to us, without symbols, because someone thought they would save time shipping it across the network, or save room on a tape! However, the analysis tools will not work without symbol information.

Use the command "file /hp-ux" to confirm that the symbols have not been stripped:

```
# file hp-ux.0
 hp-ux.0        s800 executable        -not stripped
```

Matching Core and Object Files

Finally, you need to be quite certain that the hp-ux.N file is from the same system that generated the hp-core.N file at the time of the crash. They **must** match exactly; close does not count. An hp-core.N/hp-ux.N pair should have matching "N"umbers (as created by savecore -not a user). If you have mismatched core and object files, problems will occur when running the analysis tools. Errors of various types may be printed to the screen or one may see a message that explicitly says that there is a mismatch.

Core Dump Analysis Tools and Methodologies

In this section we will take a closer look at the analysis tools. Then we will move into troubleshooting methodologies by describing what to look for in a core dump, and how to use the analysis tools to find this information.

Knowing the Tools of the Trade

The primary kernel debugging tools are **adb** and **analyze**. Ql can be used as a front-end to the primary tools to provide certain ease-of-use advantages. In this section I will provide a brief description of each of these tools.

- **Adb** is powerful in that it can be used to do detailed work, such as reading individual words and disassembling. However, adb is a bit unfriendly in it's operation; the first thing you notice when you use adb is that there is no prompt! This tool also has many commands (see the adb(1m) manual page for more information), however, we will only discuss one of them in this paper. We will use the "s" command to print the characters contained in the message buffer. (The message buffer can be used to store a panic message, among other things.) Here is an example of starting up adb:

  ```
  # adb -k hp-ux.0 hp-core.0
  ```

  The "-k" option to adb is undocumented and is used for Series 800 machines only. It simply means "treat the core and object files as kernel files". In the next section I will show an example of using the "s" command. One last thing, to exit adb, type "control d".

- **Analyze** provides the *big* picture of what was going on in the system at the time of a crash. This tool reports the state of major kernel data structures, register information, HPMC information, and more. Here are a couple of important things to know about analyze:

  i. Do a cursory check. You must obtain the version of analyze that corresponds to the kernel you are analyzing. The kernel version that an analyze has been built to support (e.g., sys.A.B...) is written out when analyze is invoked. Running the wrong version of analyze on a kernel can produce a lot of meaningless error messages. This is because each version of analyze is compiled with a set of templates that match, exactly, the major data structures in a particular kernel version. If the analyze templates do not match the data structures of a kernel, you could be wasting your time by using this tool.

  ii. Analyze has two modes of operation, *batch* and *interactive*. When executed in batch mode, analyze will format system register and data structure information and dump it to stdout (or into a user-named file). Options can be used to determine the types of information to be dumped. Here is an example of running analyze in batch mode to create a batch file named "out":

```
# analyze - hp-core.0 hp-ux.0 > out
or
# analyze - *.0 > out
```

It is necessary to use the dash ("-") for analyze to run in batch mode. Options to analyze may follow the dash as we will see in the next section.

In interactive mode, analyze provides commands that will allow a user to review the state of the kernel's subsystems individually. As with batch mode, any requested information is formatted and dumped to stdout. Here is an example of starting up analyze in interactive mode:

```
# analyze hp-core.0 hp-ux.0
or
# analyze *.0
```

To quit analyze, just type "q" (or "quit") and press the Return key.

Note
Notice that the hp-core and hp-ux arguments for analyze are entered in reverse order from adb. (This will be true of ql (below) also.) Entering the core and object files in the wrong order is a common mistake.

- **QuickLook (ql)** is an easy-to-use utility that runs adb and analyze automatically in order to print out the vital statistics of a core dump. (The "vital statistics" of a core dump are defined in the next section.) Here are some important things to know about ql:

  i. Do a cursory check. Since ql uses analyze to do some of its work, we must make sure we have the proper version of analyze before we can proceed. If more than one version of analyze exists on a system, ql will allow a user to select the proper version.

  ii. Ql has two modes of operation, *default* and *menu*. In default mode, ql will indicate its use of adb, then analyze, and then print out a summary of information from the core dump. To invoke ql in default mode, you would type:

```
# ql hp-core.0 hp-ux.0
-or-
# ql *.0
```

To enter menu mode, just type "ql" from the command line. The menu options are described below:

```
[r]un ql using the displayed core and object files
[1] set core and object file names (i.e., hp-core.N & hp-ux.N)
[2] toggle verbose output (entire message buffer, kernel params and register info)
[3] toggle the "-k" option for adb
[4] toggle whether or not to save the ql batch file
[5] toggle whether or not to save the analyze batch file
[6] set the path name (location) of the adb executable
[7] set the path name (location) of the analyze executable
[8] set the path name (location) of the ql and analyze batch files
[q]uit ql
```

Ql has no manual page, however, its options are pretty straight-forward. For more insight into ql's operation, read the adb/analyze manual pages and the ql script.

Core Dump Analysis

At the foundational level of core dump analysis, one needs to know what to look for -what are the vital statistics of a core dump? The **vital statistics** of a core dump are the basic pieces of information required to solve the problem. They are the operating system (kernel) version, the panic string, the trap, fault or violation type and address, the register information, HPMC information (if applicable), and name of the process (or routine) that was running at the time of the crash.

One of the primary objectives of this paper is to save you time in case your system panics or has an HPMC. To accomplish this goal, we first looked at setting up your system to obtain a good core dump. Then, we learned how to check a core dump pair for completeness and correctness. Now we will look at how you can obtain the vital statistics of a core dump using the analysis tools. Armed with this information, you may give your response center a head-start, and troubleshooting may go more smoothly.

As we begin this section, we will assume that we have a good core dump pair to work with (hp-core.0 and hp-ux.0).

Using ADB

We may find most of the vital statistics that we require by checking the message buffer. The message buffer is a circular queue that is used by the kernel to store its *printf* output. We will have to use adb to look at the message buffer. Remembering the example from above, you would enter adb by typing the following from the command line:

```
# adb -k hp-ux.0 hp-core.0.
```

Depending on the version you are using, adb may, or may not, print out a few lines as it starts up. But then adb does not issue a prompt at all. To dump the contents of the message buffer, type:

```
msgbuf+0x8/s
```

The meaning of this command is, "beginning at address *msgbuf+0x8* in the core file (/), print out the characters (*s*) until a zero character is reached". Be ready to use ^s and ^q as the message buffer usually holds multiple screens of information. You will be looking for a panic message, which should be toward the end of the msgbuf, and will look something like this:

```
PANIC:  please wait for core dump to complete.
@(#)9245XA HP-UX (sys.A.B7.00.3L/S800) #1: Mon Oct 30 17:59:05 PST 1989
panic: (display==0xb000, flags==0x0) pagein
```

Now, what can we glean from this information? First (line 1), we know that a *panic* occurred, as opposed to an HPMC or some other system fault. Second (line 2), we can determine the release and cycle (or the version) of the kernel. Third (line 3), we know the the name of the routine (and in this case, the process) that paniced. All translated, this panic string indicates that a Release 7.0 system paniced while the *pagein* process was running.

Good information so far, but don't stop there! Look for other key phrases like `Privilege Violation`, `Data Segmentation Fault`, or `Instruction Segmentation Fault`. Finally, should you see any *trap* information such as:

```
trap type 6 pcsq.pcoq = 0.0 isr.ior = 4.78
```

write this down. The response center may have seen the same type of panic before.

When you are ready to leave adb, type "control d".

Note
The message buffer may not yield a panic message if your core dump was created by a Transfer of Control or by a High Priority Machine Check.

Using ANALYZE

As stated in the last section, analyze has two modes of operation, batch and interactive. For the purposes of our discussion, we will only look at the batch mode. Most first-level troubleshooting can be accomplished by reviewing an analyze batch file (and the message buffer, using adb).

In every batch file produced by analyze, certain information about the kernel is always dumped. Other information that analyze can produce is dumped only by request. You should consult the analyze manual page to determine which analyze options to select. Normally, you will only need to use the A and U options for analyze. The A option dumps register information and HPMC information (if any). The U option dumps user area information. To create an analyze batch file, you would type the following from the command line:

```
# analyze -AU hp-core.0 hp-ux.0 > out
```

It will take a minute or so to create the analyze batch file.

Your next step will be to open this file so that you can look for clues. (Usually, an analyze batch file is quite large; thus, it may take a few seconds for your editor to open the file.) The following diagram shows a high-level layout of a typical analyze batch file:

Data Structure Addresses and Kernel Parameter Values
Register Information
HPMC Information
Process Information
Other Information

In the top portion of the batch file, you will see some data structure addresses and kernel parameter values. Following this is the system register information. This data may be helpful if the response center has some specific questions; however, understanding their values is beyond the scope of this paper.

The HPMC information is next in the batch file. Any indication of an HPMC will be logged in this section. After that, the process structures are displayed (with u-area information if you used the -U option to analyze). The `CURRENT PROCESS` (i.e., the process that was running at the time of the crash) is listed in this section. And finally, depending on the analyze options selected, other information may be included.

There are two major pieces of information that you should look for in an analyze batch file, an HPMC indication and the `CURRENT PROCESS`:

- To look for an HPMC indication, do a search for the acronym, "HPMC". Hopefully, the first or second occurrence of this string will be in an easy to understand sentence that says

something like this:

```
HPMC occurred
or
No HPMC occurred
```

If an HPMC did occur, then you will want to look for another string that says, `Check type word`. If you see the key words `Cache`, `TLB`, or `Processor`, after the `Check type word` string, your system probably has a malfunctioning CPU board. If you see the words `Bus` or `Assist` in the string, a non-CPU hardware error may have occurred. In either case, I recommend that you call your Hewlett-Packard response center or your Customer Engineer for help in tracking down the problem.

HPMCs are usually hardware problems. Sometimes these problems do not re-occur. Other times, the system can be shut down and, by reseating the cards in the card cage (processor, memory, I/O), the problem will go away. Most times, however, an HPMC indicates that a failing board (card) must be replaced.

- If this was a panic, not an HPMC, then it is important for us to know what process was running at the time of the crash. To find that process, search for the string, `CURRENT`, in your batch file. Your cursor will end up on the line that has the name of the process that was running at the time of the crash (last word on the right labeled `U-COMM`).

  If you are really bold, you can look at the process' stack trace, which will be a few lines below the `CURRENT PROCESS` line. The stack trace will tell you which routine was running at the time of the crash. Know that the order of the calling of the routines in the stack trace is from bottom to top. Therefore, the routine that was running at the time of the panic will be the one at the top of the stack trace.

- Similar to the panic situation, if you had to do a Transfer of Control, you will want to find and write down the name of the running process. The stack trace information will be useful too.

Note
In some cases, a core dump may not contain the string: `CURRENT PROCESS` This can occur if the system crashes during (not necessarily because of) a context switch.

Using QUICKLOOK

As stated earlier, ql is designed to be a user-friendly front-end to adb and analyze. Ql executes adb and analyze on a core dump pair, dumping their output into files. Then ql uses the fgrep(1m) command to find the vital statistics of the crash.

If you have not run ql before, the first thing you should do is start it up in menu mode. To enter menu mode, type this from the command line:

```
# ql
```

This will give you an opportunity to look at ql's default settings. You may change any of the default settings while you are in the ql menu and then run ql with those settings. Note however that these are only temporary changes. When you leave the menu, all settings return

to their default values. To make permanent changes to ql's default settings, it is necessary to edit the ql script.

If the default settings for ql are satisfactory, you may elect to run ql from the command line. One can override ql's default core and object file options from the command line. To run ql in default mode, simply type:

```
# ql hp-core.0 hp-ux.0
```

Here is an example of a panic analysis using ql:

```
QuickLook [Version 2.0] Analysis created on Mon May  6 17:32:09 PDT 1991

       Core file: hp-core.0
     Kernel file: hp-ux.0

[    ADB Output: /usr/bin/adb -k hp-ux.0 hp-core.0]
panic: (display==0xb000, flags==0x0) Interrupt
interrupt type 15, pcsq.pcoq = 0.c9584, isr.ior = 0.4000000
Data page fault on interrupt stack

[Analyze Output: /usr/contrib/bin/analyze -AU hp-core.0 hp-ux.0]
*  HPMC Registers                                        *
No HPMC occurred !!

Name of Running Process:
 uvaddr  :0x01aaf1f0  CURRENT PROCESS  u_pidptr 0x1aafc10  U_COMM dbsp

Important Register Information:
cr8     =0x00000270   cr9     =0x0000008c   cr10    =0x00000080
cr11    =0x00000000   cr12    =0x000002ac   cr13    =0x00000000
sr3     =0x00000000   sr4     =0x00000000   sr5     =0x0000c329
sr6     =0x0000ce29   sr7     =0x00000000   IIA space   =0x00000000
```

As you can see, ql executes the analysis tools to provide all of the vital statistics of a core dump:

i. The kernel version string (only in "verbose" mode)

ii. Panic string information (if any)

iii. Data/Instruction fault or privilege violation information

iv. Trap information (type and address)

v. Important register information

vi. HPMC information (if any)

vii. The name of process that was running at the time of the crash

Summary / Recommended Action Plan

Summary

First of all, congratulations and thank you for taking the time to become better equipped to maintain your system. You have learned how to obtain a core dump, both manually and automatically. You have also learned how to check a core dump for completeness and correctness. Finally, you have learned how to use the analysis tools to glean the vital statistics of a core dump. With these foundational skills, you will be a successful troubleshooter and/or information provider. Subsequently you may have a shorter turn-around time in case your system panics, hangs, or has an HPMC.

Recommended Action Plan

1. Although a system crash is very rare, it is a good idea to be prepared. Therefore, I recommend that you set up your system to properly save core dumps:
   i. Check the savecore entry in /etc/rc.
   ii. Make sure the savecore directory is on a file system partition that has enough space for a core image the size of physical memory. Also, this directory must allow write permission.
   iii. Finally, make sure that the primary swap partition has enough space for a core image the size of physical memory.
2. If ever a panic (or HPMC) does occur:
   i. Make sure that the *core dump pair* is complete and correct. The core image should be an even multiple of a megabyte and it should be the size of physical memory.
   ii. Do not strip the symbol information from your object (kernel) file.
   iii. Lastly, make sure that your core an object files match (e.g., hp-core.0 with hp-ux.0).
3. If you like, before you call your response center, run the analysis tools to determine the "*vital statistics*" of your core dump. You may use adb and analyze directly, or use ql to run the analysis tools automatically.

Interex North American Conference
August 5th-8th - San Diego, CA

Paper #2018

Network and System Management Effectiveness: The Graphical Edge

Reid Shay
Colorado Networks Division
Hewlett-Packard Company
3404 East Harmony Road
Fort Collins, CO 80525
303 229-2805

Data systems, and the networking that connects them, are critical to the operation of today's businesses. These systems and networks must be managed.

Basic System/Network Management Needs

Management Needs

* Need to Manage Critical Resources
* Need to Manage All from One Platform
* Need Flexible Management Solution
* Need to be Managing Quickly & Easily

Today's businesses need to manage their critical resources, and systems and networks are no exception. In fact, systems and networks are not only critical, they are very complex. The average person using a system or network is not easily able to handle problems when they occur. Administrators that are experienced in solving system and network problems are an expensive and valuable resource. All of this leads to the fact that system and network management solutions are one of the fastest growing markets today.

Management Steps

Management Steps
* Know Problem Exists
* Isolate Cause of Problem
* Identify Solution
* Implement Solution

What is meant by "managing" systems and networks? There are four fundamental steps to managing. First, the problem must be recognized to have happened. Second, the problem needs to be isolated to the faulty part or parts. Third, a solution has to be determined. Fourth, the solution needs to be implemented. It is important that each of these four steps takes place as quickly as possible, because of the critical nature of the systems and networks. It is also important that the steps be as easy to implement as possible, due to the expensive and valuable nature of the administrators.

A management solution is one that helps to find a problem, isolates the problem, suggests a solution to the problem, and helps to implement the solution to the problem. Another type of solution is one that involves the anticipation of a problem before it occurs, or one that takes prior corrective action. An example of this is a maintenance programs that tracks usage and schedules regular preventive maintenance. Another type of solution is one that expects that a problem may occur at some point, and prepares to fix the problem quickly and easily when the inevitable day comes. Backup programs are an example of this type of solution.

Manage All from One Platform

Manage All from One Platform

* One Management Solution
* Quality Custom Management
* Best Solutions come from Device Vendors
* Generic Solutions are Improving

System and network managers are torn between two conflicting desires. On the one hand they want all of their system and network devices to be managed from the same platform. On the other hand, they want to have quality, useful management solutions. Traditionally the only quality solutions have come from the device vendor. This is because only the vendor completely understands the device. These vendors do not generally coordinate their management solutions in order to provide their customers with one common platform. This is particularly true when the other devices are competitive products.

A few, newer management solutions manage generic devices. These solutions are much superior to general purpose solutions in the past. However, they still have a difficult task in tracking the devices from different vendors as the devices change over time.

Flexible Management

Flexible Management

* Management Solution should Fit the Environment
* Multiple Management Locations
* Diverse System and Network Types

Businesses want their management solutions to fit with their existing system and networking environment. If they generally use UNIX machines, they want a UNIX based solution. If the environment is DOS, they want DOS. If their network is TCP/IP and ethernet, they need a solution that will manage TCP/IP and ethernet.

While managers want all of their devices to be managed from one type of platform, they also may need to have multiple locations from which to manage. These multiple locations may be on three desks in the same department. When a call comes in with a problem, an administrator may not want to have to go to a central console to diagnose and solve the problem. In other instances, the multiple management locations may be at remote sites. If a small site has only one system/network manager and that manager is not in, the nearest backup person in another office may be called on to help. Some problems, such as those found in wide area networking (WAN), may only be managed from a central site.

Up and Managing Quickly and Easily

Up and Managing Quickly and Easily
* Comfort with Solution is important
* Reduce Learning Curve
* Automatic Device Entry

Businesses want system and network management solutions with which they can be comfortable. As mentioned before, the solution should fit into the business' environment. This is particularly true of the user interface. Learning time needs to be brief because of the critical nature of the systems and networks and the expense and value of the management people. If managers are use to using a Motif interface, a management solution based on Motif will be the easiest one to get up and running. If Microsoft Windows 3.0 is the familiar standard, that will be preferred.

The method in which each device is managed will likely be somewhat different, due to the inherent differences between the devices. It is important to keep as many things similar to save learning time and make regular operation efficient. The menu structure, the meaning of certain alarms, the uses of specific colors are all important. The more these things are the same across management applications, the more efficient the solution and the administrator.

Finally, one of the best ways in which a management solution can improve efficiency is through automatically entering all appropriate devices into the

management solution. This will save a great deal of an administrator's time. It will also provide an added benefit; it will help insure that the devices being managed are up-to-date. Imagine how difficult it is to track down a problem caused by a new device that the network administrator did not know had been attached to the network.

The Graphical Edge

System and network management solutions have a number of characteristics that are important. None, however, are more important than having a graphical user interface (GUI) and a system/network map. Many of the basic system and network management needs are met by having these two features. In particular, a GUI and a map are critical to efficiently executing the four management steps.

A Standard Graphical User Interface (GUI)

Standard Graphical User Interface

* A GUI, based on a standard such as Microsoft Windows 3.0 or OSF Motif, reduces the initial learning time and facilitates similarity between management applications.

* The "point-and-click" interaction found in GUIs allows for very rapid and intuitive isolation of problems.

* A GUI allows for multiple views of the status of a system or network. (Map, Table, Graph, etc.)

* In the limited world of DOS, a GUI based on Windows 3.0 allows for memory beyond 640K and multiple management applications.

The Benefits of a Standard GUI

Graphical user interfaces, such as Microsoft Windows, OSF Motif and HP's NewWave, are taking the computing world by storm. Many application

vendors have found that their products are a difficult sale if the application is not compatible with the appropriate GUI. Is all of this interest due to the superior nature of GUIs? Yes, but that is not the whole story. In addition to being a better interface than a character user interface (CUI), GUIs are addicting. Once a person has learned to use a GUI, it is very difficult to go back to using a CUI. This is particularly true if some of the person's applications use the GUI and some use the CUI.

How great are the benefits of a GUI? The only well known study available was conducted by Temple, Barker & Sloane, Inc. and was commissioned by Microsoft Corporation[1]. While Microsoft is not exactly a neutral observer in the GUI world, the study is comprehensive and appears to have been well conducted. The results consistently, and substantially, show that a GUI is superior to a CUI in a wide variety of computer tasks.

GUI Benefits,
from Microsoft Study

* FASTER - GUI users completed 35% more tasks than CUI users.

* BETTER - 91% of GUI users completed their tasks correctly, versus 74% for CUI users.

* HAPPIER - New users reported a much lower level of frustration (2.7 out of 10) using a GUI than CUI users (5.3 out of 10).

* SELF-TAUGHT - GUI users were able to teach themselves 23% more tasks than CUI users.

In system and network management, the benefits of a GUI parallel those measured by the study. Specifically, a system or network administrator can learn the management application quicker. This is due to the faster learning nature of GUIs. It is also because the administrator has probably already learned the GUI for other reasons. The administrator can isolate problems more quickly. This is due to the "point-and-click" capability of GUIs. This

capability, when combined with a consolidated network/system map, is extremely beneficial in tracking down the specific source of a problem in a complex system or network. A GUI also allows a system or network administrator to view evidence of a problem in different ways. The changing of a map symbol's color from green to red may be the first indication of network trouble. A graph of network conflicts could indicate the type of problem. A table of recent network "sends", listed by node, would further zero in on the source of the problem. A map, a graph and a table could each be the most helpful tool in different circumstance.

A Consolidated System/Network Map

Consolidated System/Network Map

* A map quickly and easily displays the complex relationships between devices.

* A map permits a high-level view of the health of the whole system/network at once.

* A map, along with "point-and-click," facilitates easy switching between different parts of the management solution.

* A map can display different levels of map scale, depending on the management requirements.

The Benefits of a Consolidated System/Network Map

A map is typically a tool to help a person find their way. A system or network map is useful in many other ways as well. Today, systems and networks are very complex, and becoming more so all the time. Knowing that five devices are not responding to a network probe is not usually enough to identify the cause of a problem. Even knowing which five devices are not responding may not be sufficient. Once the administrator can see that four of the devices connect through the fifth flagged device, they can guess that the fifth device is at fault. Chances are all five devices did not fail at the same instant.

In addition to increasing in complexity, today's systems and networks are getting larger and larger. More significantly, today's administrator is expected to manage larger and larger systems and networks. An administrator must "watch" the whole system/network for problems. They must quickly isolate the cause of the problem. They will then need to study details about the suspect device in order to determine the solution. Only the super-human need apply for such jobs. A consolidated map can permit views of the system/network from different "elevations." The "high-level" view allows the whole to be viewed at once. A symbol turning to yellow or red will clearly show a problem in that "area." By using "point-and-click," the map's view can quickly be zoomed-in to find the problem device or devices. The smallest scale view is in many cases a "picture" of the front panel of the device, complete with lights and switches. This ability to display the whole, down to the smallest part, gives the consolidated map great power to the administrator of large and/or complex systems and networks.

Conclusion

As systems and networks become larger and more complex, administrators need all the management help they can get. By using management applications to manage their devices, the administrator can become more effective. If these applications are combined and share one GUI and one map, the administrator is vastly more effective. A single GUI/Map based platform will create a management solution that is greater than the sum of its application parts. **A GUI and a map should be the minimum requirements for all system and network management applications today. A single consolidated GUI and map should be the goal for tomorrow. Tomorrow shouldn't be very far away.**

1 - *The Benefits of the Graphical User Interface, A Report on New Primary Research*, Prepared by Temple, Barker & Sloane, Inc., Copyright, Microsoft Corporation, 1990.

Paper # 2019

Fortran 90: The New Fortran Standard

Maureen B. Smith Hoffert
Alan C. Meyer

Colorado Language Laboratory
3404 Harmony Road, MS: 96
Fort Collins, CO 80525
303-229-2774

1. Introduction

FORTRAN was one of the earliest high level programming languages. It was developed in the 1950's and was first standardized by the American National Standard Association now called the American National Standard Institute (ANSI) in 1966. The second revision of this standard, informally called FORTRAN 77 [1], appeared in 1978, and it has been a key strategic language for numerical, technical, engineering and scientific applications. Today there is a new International Fortran Standard, commonly called Fortran 90 and previously called Fortran 8X, which has a number of new features and concepts. Most of these have been present in Fortran compilers as proprietary extensions from given vendors. Other new features are found in languages such as C, Modula-2, Pascal or Ada.

In this paper we discuss the current status of the standardization process for Fortran 90. We then describe in general terms the key new language constructs. Our intent is not to describe all features in detail but to give highlights and examples of key features. A complete description of the language is formally available in the standard document [2]. A more informal but thorough description of the language can be found in *Fortran 90 Explained* by Metcalf and Reid [3].

Finally, we will compare the *de-facto* industry standard extensions found in current Hewlett-Packard (HP) compilers with Fortran 90 features, and raise some issues that are related to HP's Fortran 90 strategy.

2. Current Status of the Fortran 90 Standard

Fortran 90 is the informal name for the new **International Fortran Standard: ISO/IEC 1539:1991.** The final document was submitted to ISO (International Standard Organization) before May 1, 1991, and the document should be published by the end of the summer of 1991. The ANSI (American National Standards Institute) technical committee on Fortran, X3J3, voted this April to recommend to ANSI that it adopt this identical standard document as the new Fortran standard for the United States as X3.198-199X. There have been three public reviews of the document and no further revisions should be required.

In the international arena, Fortran 90 is a revision and hence a replacement of the current FORTRAN 77 International Standard, ISO 1539:1980(E). In the United States, ANSI has taken another course for Fortran. FORTRAN 77 is considered an "archival standard", and is still a current standard. Fortran 90, also called Fortran Extended, is a new Fortran standard and hence will not replace FORTRAN 77 but will coexist with it for at least the next three years. Currently the International and the United States standard documents are identical.

One of the design concepts for Fortran 90 was for it to be a superset of FORTRAN 77. Another goal was to modernize Fortran by introducing more modern programming concepts in the form of modules and interface blocks, as well as structured data types and control constructs. One fairly controversial issue was an attempt to provide for future evolution of Fortran by identifying features that were considered as no longer necessary because there are better ways to attain the intent of the old semantics; an example is the *assigned GO TO* statement.

There has been a great deal of debate about the upwards compatibility of FORTRAN 77 with Fortran 90. There are four issues that have caused concern where FORTRAN 77 codes may not work as expected with Fortran 90 compilers. All of these issues are cases in FORTRAN 77 that were declared as **processor dependent** and hence were not portable nor reliable in implementations. Because of this lack of portability, Fortran 90 chose to specify the behavior in such cases to facilitate future code portability.

These areas that were **processor dependent** in FORTRAN 77 that are now specified in Fortran 90 are:

- In a DATA statement, specified precision of a real constant must be preserved when used to initialize DOUBLE PRECISION variables, in FORTRAN 77 the processor could supply more precision.
- A named variable initialized in a DATA statement only had the SAVE attribute if it was specified, otherwise it was processor dependent in FORTRAN 77.
- If there are not enough values in the record required by the input list during formatted input, then the input record will be logically padded with blanks unless the PAD="NO" is specified. This padding was not required in FORTRAN 77.
- There are a number of new intrinsic functions defined in Fortran 90 whose names could conflict with user defined procedure names in FORTRAN 77 programs unless those procedures were specified with an EXTERNAL statement which was recommended practice.

Because the Fortran community has become quite diverse and because FORTRAN has a long history of use, there has been a good deal of debate about the requirements for the new standard. Some did not want to change Fortran at all, wanting it to die out. Others wanted to only standardize existing practice in current Fortran compilers. Still others wanted to provide a basis for future Fortran language evolution while preserving investments in old codes and programmer expertise.

Trying to come to consensus on these issues has taken a good deal of work and time. The end result is not entirely satisfactory to any one group. The success of the effort will be measured by the user, vendor and computer communities in their acceptance and in the speed of their adoption of the new Fortran standard.

3. New Concepts and Features in Fortran 90

The new Fortran 90 concepts and features can be divided into seven categories:

- language evolution;
- compiler detection requirements;
- data types, data objects, and type declarations including dynamic storage allocation;
- modules and procedures including many new intrinsics;
- control constructs;
- source form changes; and
- input/output features.

Each of these areas will be described in more detail below.

3.1. Language Evolution

A number of features have been labeled as *obsolescent*. These features are still a part of the language but their use is no longer recommended practice. They are considered redundant because other features provide a similar capability. Further, in order for the language to be able to evolve in the future, these features possibly could be removed from the standard definition of the language if the Fortran community as a whole considered that they were no longer required.

The obsolescent features are:

- arithmetic-IF;
- ASSIGN and assigned GO TO statement;
- alternate returns for procedures;
- real and double precision DO loop variables, bounds and/or steps;
- jumps into IF constructs;
- nested DO loops which share termination statement; DO loops whose termination statement is not a CONTINUE statement or an END DO statement;
- format statement with assigned specifiers; and
- PAUSE statement.

In the past, many users have rejected giving up these features. Whether or not this opinion will be maintained in the next decade remains to be seen.

3.2. Compiler Detection Requirements

Compilers are now required to detect a variety of conditions in order that a programmer knows about potential problems. The concept of *constraints* has been introduced for which a compiler is required to have the capacity of issuing a compile-time warning. Examples of features that the compiler must be able to warn for are: obsolescent features as defined in the previous section; syntactic or semantic extensions (that is, syntax and semantics that are not currently defined by the standard) and leniency in interpreting constraints defined in the standard; the use of intrinsic procedures that are not defined by Section 13 of the standard; the use of source form or characters not supported by the standard; inconsistent application of the Section 14 scoping rules for names; and the use of precision specification (kind type specification) in a type statement whose specified value is not supported by the processor.

3.3. Data Types, Data Objects, and Type Declarations

Fortran 90 extends the data concepts available by providing extensions for arrays, adding pointers and structures, and specifying forms of type declarations.

3.3.1. Arrays

One of the most important and perhaps the most requested new feature of Fortran 90 is what has been called the **array language**. Arrays have been given a first-class status in the language by expanding array usage and also array specifications. Whole arrays and parts of arrays, or array sections, are treated like traditional Fortran scalar variable objects. Arrays can appear as primaries in expressions and they now can be constants as well as variables.

The array language is considered a more precise and succinct means of expressing mathematical expressions for vectors. This could simplify programming to avoid writing DO loops around arrays. This can also reduce the number of errors caused by incorrectly modifying DO loop indices during execution of a DO loop. The use of arrays as primaries in an expression is seen in the following example using the declaration:

```
REAL, DIMENSION (300,200) :: X, Y, Z
```

FORTRAN 77 would have the following nested DO loops:

```
DO J = 1, 300
    DO K = 1, 200
        X (J,K) = X (J,K) + Y (J,K) * Z (J,K)
    END DO
END DO
```

In Fortran 90 this could be expressed with the new array notation:

```
X = X + Y * Z
```

Functions can now return arrays in that programmers can define array-valued functions as shown in the following example:

```
FUNCTION MV_Mult(matrix, vector)              ! matrix-vector multiply
    REAL, DIMENSION(:, :) :: matrix           ! assumed-shape array
    REAL, DIMENSION(SIZE(matrix, 2)) :: vector
    REAL, DIMENSION(SIZE(matrix, 1)) :: MV_Mult
    nrows = SIZE(matrix, 1);
    DO i = 1, nrows
        MV_Mult(i) = matrix(i, :) * vector
    END DO
END FUNCTION MV_Mult
```

In the above example, a function *MV_Mult* is defined with two arguments. The argument *matrix* is an assumed-shape array that will take its size and shape from the actual argument passed into *MV_Mult* at run-time. During the execution of the DO loop, element *i* of the result array *MV_mult* is defined by the vector product of *vector* and row *i* of *matrix*.

To enhance the use of arrays there are some new classes of arrays similar to FORTRAN 77 adjustable arrays. There are **dynamic** or **automatic** arrays that are data objects local to a procedure but whose dimension and size are determined at run-time. The following is an example of such an **automatic** array:

```
SUBROUTINE What (arr_arg, in_dim)
    REAL arr_arg (in_dim)       ! adjustable array
    REAL arr_local (in_dim)     ! automatic array
```

There are **allocatable arrays** in Fortran 90 which are specified as **ALLOCATABLE** in a specification statement providing a means of establishing dynamically allocated arrays whose size will depend on run-time characteristics of the code. Their bounds and hence shape, are allocated by the execution of an **ALLOCATE** statement. Its initial status is *not currently allocated* before the execution of the **ALLOCATE** statement. In the following example, storage for allocatable array *a* is allocated at run-time, where the amount of storage depends on the input values of variables *m, n,* and *p*. The DEALLOCATE statement is used to free the storage previously allocated for the array. Note that the user must explicitly deallocate the storage.

```
PROGRAM Main
    REAL, ALLOCATABLE :: a(:, :, :)
    INTEGER m, n, p
    READ(5, *) m, n, p
    ALLOCATE(a(m, n, p))        ! Allocate "global" working array
    ...
    DEALLOCATE(a)               ! No longer needed - deallocate
    ...
END PROGRAM Main
```

Whole arrays can be passed as arguments to procedures. The use of these arguments will depend on the specification of the procedure. **Assumed-shape** arrays, for example, will have the number of dimensions specified, but the size of each dimension will vary at run-time by the shape of the argument in the procedure invocation. In an earlier example in the function *MV_mult*, the argument *matrix* is an assumed-shape array. A routine might call *MV_mult* with declarations such as:

```
REAL, DIMENSION (0:100, 1:50) :: matrx
REAL, DIMENSION (1:50) :: vect
REAL, DIMENSION (0:100) :: result
result = MV_mult (matrx, vect)
```

Another example of an assumed shape declaration would be the following:

```
actual argument declaration:
REAL, DIMENSION (0:5, 3:7) :: act_arg

dummy argument declaration:
REAL, DIMENSION (:,:) :: dum_arg
```

Array constructors is another new concept in Fortran 90. An array constructor is an array data object that can be either a variable or a constant. For programming convenience, an array can be denoted as a list of its elements inside the symbols: *(/* and */)*.

The following examples are of an array constructor used as a constant initializer of another array and an array constructor used as a primary in an expression:

```
REAL u (3), x (3)
u = (/ 1,3,2 /)
x = (/ 1,1,1 /) + u
```

Arrays can be referenced as a whole array or as part of an array called an **array section.** Array sections are also considered arrays. There are two syntax notations for array sections: **triplet notation** and **vector-valued subscript notation.** The following is an example of triplet notation:

```
REAL, DIMENSION (300, 200) :: x, z

x (2:298:2, 2:198:3) = x (2:298:2, 2:198:3) + &
   &  z (2:298:2, 2:198:3)
```

Each triplet in the above example specifies an initial (or lower bound), a final index (or upper bound) and a stride for the intervals used to step through the array. Note that in this example the array section does not have to be contiguous.

An example of the syntax for vector-valued subscripts would be:

```
REAL z (5,7); INTEGER u (3), v (4)
u = (/ 1,3,2 /)
v = (/ 2,1,1,3 /)
```

The data object *z (3, v)* will have the values of: *z (3,2), z (3,1), z (3,1), z (3,3).* Note that the *v* is the vector-valued subscript. The value *(/ 1,3,2/)* is an array constructor that is initializing the variable *u*. The variable *v* has a similar form of initialization.

Array sections can be arguments and they can be passed to assumed-shape dummy arguments. However, there must be an explicit interface provided for routines defined with array sections as arguments in order for a compiler to determine the appropriate passing convention.

Array changes to procedures include array-valued functions, mentioned earlier, and array arguments to intrinsics. The Fortran 90 standard refers to three new classes of array intrinsics: **elemental, transformational** and **inquiry.** Many of the FORTRAN 77 intrinsics are called elemental and can now be given array valued arguments which in turn will generate an array valued result for that intrinsic. Intrinsics such as *sin, cos, tan,* and *sqrt* are considered to be elemental. For example:

```
REAL :: z (20), y (20), x (20)
x = COS (z) + SQRT ( SIN (y) )
```

Other new array valued intrinsics are defined such as *DOTPRODUCT, MATMUL,* and *PACK.* These are called transformational functions since they take an array and change its characteristics. New intrinsics called inquiry functions will return characteristics or specifications of an array, such as *LBOUND, UBOUND, SIZE,* and *SHAPE.*

3.3.2. Derived Types

Derived Types are user defined data structures similar to C structs or Pascal records. These structures are composed of fields of intrinsic types that can be either scalars or arrays. Derived types can utilize the implicit typing statement. Intrinsic operations can be extended to be used for derived types by the means of interface blocks which will be discussed in detail later. New specifications of **SEQUENCE** and **accessibility attributes** of **PRIVATE** and **PUBLIC** apply to derived types. Sequence association sets expectations for the storage association of the fields or structure components. Accessibility attributes of **PRIVATE** and **PUBLIC** provides a *data hiding* mechanism which allows the type structure and/or its components to be available to procedures outside the host. The accessibility specification for type declarations can only appear within a module. The component selector is a " % " for field access specification.

The syntax for definition of a user defined derived type will be:

```
TYPE, PRIVATE :: reference
     integer              :: volume, year
     character*10         :: title
END TYPE reference
```

The syntax for declaration of a variable of that type will be:

```
TYPE (reference) :: biblio
```

The syntax for component selection of a field of that type is:

```
biblio%title = "New Title"
```

An example of Implicit Typing declaration for derived types is:

```
IMPLICIT TYPE (reference) (a-c)
```

3.3.3. Pointers and Targets

POINTER and **TARGET** are new specifications that can qualify data objects. A pointer can be thought of as a data object **and** a descriptor which holds information about a specific data object. A pointer is **not** a type **nor** an address. This means that arithmetic operations on addresses are not allowed. The initial status of a pointer is undefined. A pointer becomes associated to a data object in two ways: the execution of the ALLOCATE statement or the execution of the pointer assignment statement.

In this example, a pointer to a one dimensional array is declared by the following statement:

```
REAL, POINTER, DIMENSION (:) :: ptr_node
REAL, TARGET :: target_nod (100)
```

The following ALLOCATE statement allocates storage and associates the pointer with that space.

```
ALLOCATE (ptr_node (-3:3))
```

A pointer can be associated with a target through the execution of a pointer assignment statement.

```
ptr_node => target_nod
ptr_node (5) = 99      ! sets target_nod (5) to 99
```

A pointer is treated as an object and will be automatically dereferenced when it appears in an expression context:

```
REAL, TARGET :: targ
REAL, POINTER :: ptr_targ, ptr2
targ = 6
ptr2 => targ
ptr_targ => targ
ptr_targ = ptr_targ + ptr2
```

The two pointer assignment statements associate both *ptr2* and *ptr_targ* with the object *targ*. In the last assignment statement where pointers are appearing on the right hand side of the equal sign, *ptr_targ* and *ptr2* are automatically dereferenced with each having the value 6. Thus *targ* will be assigned the value 12 through *ptr_targ*.

A pointer is like a descriptor when the pointer appears in the following circumstances:

- in an ALLOCATE, DEALLOCATE, or NULLIFY statement;
- on the left-hand side of a pointer assignment;
- is passed to a pointer dummy argument; or
- is an argument to the ASSOCIATED intrinsic.

Pointers may appear in modules, may appear in common, and may be components of derived types. Pointers can be actual or dummy arguments, function results, appear in input/output lists and can be internal files.

3.3.4. Type Declarations

Type declaration statements can have a new syntax. To specify different **KIND's** of data types, such as *REAL*4* and *REAL*8*, there is now a KIND selector, also called a **kind type parameter**. This mechanism is viewed as aiding the portability and parameterization capabilities for type declarations.

Further, other specifications can appear in a type declaration statement. Attribute specifications such as PARAMETER, DIMENSION, PUBLIC, PRIVATE, ALLOCATABLE, EXTERNAL, INTRINSIC, OPTIONAL, INTENT, POINTER, TARGET and SAVE can all appear in a type declaration statement as in the following example:

```
REAL (4) a, b, c
REAL (KIND = 4) x, y, z
CHARACTER (LEN = 20, KIND= kanji) kanji_ch
REAL (KIND = 16), SAVE :: e (+2:+4), f, g
```

Data can be initialized within a type declaration statement as well:

```
! Note that w is an array constant
REAL, PARAMETER :: w (5)  = (/ 1,2,3,4,5 /)
```

3.4. Source Form and Syntactical Issues

Fortran 90 introduces a new optional source form called *free source form* which is an alternative to the old column oriented *fixed source form*. Probably the most important aspect of free form is that blanks are significant. Other aspects of free form source are: the length of line has been extended from 72 characters to 132 characters; labels can appear after columns 1 to 6; the symbol & indicates that the current statement will be continued on the next line. There are a few specific rules for continuation for comments and tokens.

Other changes apply to both source forms: multiple statements can appear on a line separated by a semi-colon; an exclamation mark can indicate a comment anywhere on a line; symbolic names can be 31 characters rather than 6; and both a single and a double quotation mark can delimit a string.

Intrinsic relational operators often termed **dot operators** such as .LE. have a new alternate syntax. For example, **.LE.** can be < and **.GE.** can be >=.

Numeric and character constants can be given a kind type parameter, for example a *SHORT* or *integer*2* constant of 3 might be designated as:

```
3_short or 3_2
```

A French or Chinese character string on a processor supporting a French or Chinese character set might be designated as: french_" *merci.*"

3.5. Modules and Procedures

3.5.1. Modules

Modules are considered one of the most significant programming constructs added to Fortran 90. The module concept is one of the key constructs that is currently **not** implemented in existing Fortran compilers. However, they are a common programming construct found in Modula-2 and HP Pascal, for example. Modules provide a means of encapsulating data and/or procedures into logical functional units or objects. They might be considered as providing the beginnings of an object orientation to Fortran. Modules can be used to hide certain data from other program units by means of the accessibility attribute. They can be used to modularize and structure a larger software system.

The syntax of a module begins with a **MODULE** statement and ends with a matching **END MODULE** statement. A module can have a specification part with data object definition and it can have a procedural part for **module procedure** specification. Module procedures are separated from the module specification part by a **CONTAINS** statement.

Data entities of a module can be accessed by a **USE** statement from outside a module and also by **host association** from within the module.

Entities local to a program unit which have the same name as a module data entity can be renamed on the USE statement. Further data objects can be excluded from being used in a local scope by means of the **ONLY** clause on the USE statement. Examples of such declarations are:

```
USE math_lib                          ! use the entire module math_lib
USE stats_lib, SPROD => PROD  ! use the stats_lib module but rename
                              ! the module entity SPROD to the local
                              ! name of PROD
USE my_common, ONLY: yprod    ! use the module my_common, but only
                              ! the entity yprod from it
```

In the next example, the module *work_arrays* sets up global definitions of three allocatable arrays *a*, *b*, and *c*. In the subroutine *configure_arrays* the module *work_arrays* is accessed with the USE statement. At run-time the arrays are dynamically allocated with the execution of the ALLOCATE statement.

```
MODULE work_arrays
    INTEGER n
    REAL, ALLOCATABLE, SAVE :: a(:), b (:,:), c(:,:,:)
END MODULE work_arrays

SUBROUTINE configure_arrays
    USE work_arrays
    READ (*,*) n
    ALLOCATE (a (n), b (n,n), c (n,2 * n,n)
END SUBROUTINE configure_arrays
```

3.5.2. Procedures

Procedures, both functions and subroutines, have a number of new features. A key aspect of procedures is that they can be nested one level. These nested procedures are called **internal procedures**, providing somewhat of a macro facility. Arguments to procedures can be declared as optional and can be given keywords for referencing. An example from the standard is:

```
SUBROUTINE Solve (solution, method, print)
    REAL solution
    INTEGER, OPTIONAL :: method, print
CALL Solve (print = 6, solution = 2.5)
```

Another new modern programming concept added to Fortran 90 is that of **interface blocks.** Interface blocks specify a procedure's characteristics explicitly so consistency between references and definitions can be checked and information accessed during compilation for such uses as more efficient code generation or resolution of generic references. Interface blocks can be used to extend intrinsic operators, intrinsic assignment, or to define new operations on user defined derived types.

Interface blocks can also be used to extend the **generic** intrinsic procedure concept in FORTRAN 77 to user defined type procedures. In this way a single name can be used to reference different procedures depending on the different characteristics of those procedures. An example of extending the intrinsic operator " * " is:

```
INTERFACE OPERATOR (*)
  FUNCTION RationalMult(r1, r2)
    TYPE (Rational) r1, r2, RationalMult
  END FUNCTION RationalMult

  FUNCTION MatrixMult(m1, m2)
    TYPE (Matrix) m1, m2, MatrixMult
  END FUNCTION MatrixMult

  MODULE PROCEDURE AbstractTypeMult   ! Note the module procedure reference
END INTERFACE

TYPE (Rational) rat1, rat2, rat3
  ...
rat1 = rat2 * rat3
rat1 = RationalMult(rat2, rat3)
```

In the first assignment statement the use of the extended operator will invoke the function *RationalMult* because the two operands match the type *Rational* of the function arguments. *RationalMult, MatrixMult,* and *AbstractTypeMult* also can be invoked directly as a function reference.

Other miscellaneous features for procedures have been added to Fortran 90. For example, recursion is now standardized, and functions can have result variables that are different than the function name for returning the result value of the function.

Another key area where FORTRAN has been extended is in the number of new intrinsic procedures available. New categories of intrinsics were mentioned earlier, such as elemental and inquiry functions. Further, many new intrinsic functions have been defined such as *MVBITS, MAXVAL, MINVAL, MATMUL,* and *RANDOM_NUMBER.*

3.6. Control Constructs

Fortran 90 has adopted a number of control constructs common in other languages such as C. In particular, the **CASE** statement, similar to C's *switch* statement, and the **CYCLE** and **EXIT** statements as shown in the following example of a bubble sort:

```
OUTER: DO i = 1, n-1
  swap = .FALSE.
  INNER: DO j = n, i+1, -1
    IF (a(j) >= a(j-1)) CYCLE INNER
    swap = .TRUE.
    atmp = a(j) ;  a(j) = a(j-1) ; a(j-1) = atmp
  END DO INNER

  IF (.NOT. swap) EXIT OUTER
END DO OUTER
```

Note the use of the **named construct** concept where control constructs have named labels at the beginning and end of a control block. The CYCLE statement will begin the next iteration of any enclosing loop. Similarly, the EXIT statement will exit from any enclosing loop.

There are some new forms of the **DO** statement such as the **DO WHILE** and the **DO** *forever* statement in addition to the obsolescent forms mentioned earlier.

The **WHERE** statement and construct is another feature associated with arrays that is a combination of a **masked** if statement and an assignment statement. Arrays in a WHERE statement must be

conformable and WHERE statements cannot be nested.

```
REAL pressure(n,n), temp (n,n), raining (n,n)

WHERE (pressure <= 0.0)
      pressure = pressure + inc_pressure
      temp = temp - 5.0
ELSEWHERE
      raining = .true.
END WHERE
```

In this example, the WHERE clause is executed for each positive element of *pressure,* and the ELSEWHERE clause is executed for each negative element.

3.7. Input/Output Features

FORTRAN has traditionally had a rich set of input/output (I/O) features. This area of the language has not been changed to the same extent as others. Features such as: **non-advancing, namelist,** new **input/output statement specifiers,** and some new **edit descriptors** are the key additions.

Namelist I/O has been present in Fortran compilers for a number of years with a variety of implementations. In Fortran 90 it is extended to user defined derived types and has a slightly different syntax than some existing implementations.

Non-advancing I/O provides the capability for READs and WRITEs to access a given record and to not advance to the end of it or the beginning of the next record with a given access. This could be useful with interactive terminals potentially. Non-advancing I/O is not for unformatted, direct access files, nor for internal, namelist, or list-directed file access.

The **ES** (Scientific notation) and the **EN** (Engineering notation) edit descriptors extend the current E edit descriptor.

The input value of -.5 with an EN12.3 edit descriptor would output -5000.000E-03.

The input value -.5 with an ES12.3 edit descriptor would output -5.000E-01.

4. HP Extensions and Fortran 90 Features

Most Fortran 90 features appear in existing FORTRAN 77 compilers as vendor extensions. HP compilers have supported some of these extensions. Because Fortran 90 supersets FORTRAN 77, using a Fortran 90 compiler on FORTRAN 77 codes should require no change.

Some extensions Hewlett-Packard has supported are standardized in Fortran 90. In the source form category, supported features include: upper and lower case characters; double and single quotation marks; and the "!" comment indicator. Hewlett-Packard compilers support *de-facto* industry standard structures and records, which differ syntactically from Fortran 90 derived types, but conceptually are quite similar. HP supports recursion in procedures. The *unlabelled* DO statement is supported. HP supports extended types such as BYTE, INTEGER*2, and DOUBLE COMPLEX which reflect the Fortran 90 parameterization of types with a KIND value. The Mil-Std 1753 features have been included in Fortran 90 including DO WHILE, the INCLUDE statement, and the BIT intrinsics which are also supported by HP. HP and other vendors have implementations of NAMELIST. However, each implementation will change slightly in order to conform to the new standard, but the feature should now be more portable across architectures.

One important issue is that while Fortran 90 appears to have many new features, some of them, in fact most, are already present in existing Fortran compilers. Perhaps Fortran 90's key importance is that it provides a standard specification for the many non-standard and hence non-portable features of the past. Hopefully, its design and intent will be sufficient to create a future for Fortran.

5. Conclusions

There are a number of unanswered questions which HP and other vendors are currently trying to answer that pertain to Fortran 90. How much new code is being, or will be written in Fortran? What is the customer's or potential customer's commitment to Fortran? How large a role will Fortran play in the future as a key strategic language or will it be replaced by other languages such as Ada and C++? Will they require Fortran 90 or are they only interested in using FORTRAN 77? Will customers require Fortran 90 simply because it is a new standard or will they require only a certain selection of the features for a particular functionality? What are customers' timing requirements for Fortran 90? When will customers want it, request it, use it, and/or require it? What compatibility requirements for Fortran 90 do the customers have? Do they require complete compatibility with all old extensions, or with only a subset of these extensions? Do they require Fortran 90 to be a "pure" compiler with no extensions supported? How will Fortran 90 features and development coexist with FORTRAN 77 in customer's code; for example will programmers mix old and new features in the same program, or only across files or only in linked object modules? Compiler vendors are currently actively seeking to understand these issues and customers' responses to these and similar questions.

Migration to Fortran 90 could be painless. Investment in old and current Fortran codes will be preserved. Fortran codes will continue to work while programmers adapt new, more modern programming practices. Investment in training of Fortran programmers should be preserved. Migration should not require learning a new language, only new features; and then, only those that a programmer considers useful. Whether or not Fortran 90 becomes successful will depend on its acceptance by the Fortran community and on the quality of Fortran compiler implementations. It will be interesting to watch the development of this next chapter in FORTRAN's history.

6. References

[1] *American National Standard Programming Language FORTRAN, ANSI X3.9-1978 (ISO 1539-1980 (E).*

[2] *International Standard Fortran* **ISO/IEC 1539 : 1991 (E).**

[3] Michael Metcalf and John Reid, *Fortran 90 Explained,* Oxford University Press, 1990.

PAPER #2020

MULTIVENDOR TERMINAL CONNECTIVITY WITH HP's FAMILY OF DTCs

Jean-Luc MEYER
Product Marketing Manager
HEWLETT-PACKARD FRANCE
5, avenue R. Chanas,
EYBENS
38053 GRENOBLE CEDEX 9
FRANCE
Tel 33-76625692

I. Introduction

Current trends in information technology show an increase in complexity with systems from different vendors and networked devices. As more businesses implement local and wide area networks to tie together their information resources, the demand for end-user terminal access via a network connection has increased. In response to this demand, the industry is experiencing significant increase in the use of LAN-based terminal server technology. HP systems (HP3000, HP9000 and HP1000) are more and more frequently integrated into network environments where new business needs drive HP's new offering with the Datacommunication and Terminal Controller (DTC).

II. Problem Statement definition

With the increasing need for companies to communicate on a global scale, the need to exchange information is growing rapidly. Computer networks are increasingly being interconnected to respond to business needs. Because of the tremendous growth in wide area connectivity devices, the requirement for end-user connectivity over these backbones is growing.

As networks grow, they tend to incorporate equipment and systems from different vendors. Companies are consequently looking for ways of rationalizing end-user access to different systems in a multivendor environment. There is therefore a tremendous growth in equipment such as terminal servers which can provide end users with a single access point to multiple systems from multiple vendors.

As a consequence of the evolution of end-user devices (i.e., the migration from asynchronous devices to LAN-attached devices such as PCs, X-terminals, and networked printers), companies are also reviewing their wiring strategy to accommodate emerging technology.

III. Market Requirements in Terminal Connect

As a consequence of this new terminal connect challenge, the main new business requirements can be described as follows:

a) Multivendor end-user connectivity based on standards: As customers have a variety of computer vendor's systems which speak different protocols, the communication server solutions must speak those protocols. HP's way to ensure multivendor connectivity is based on standards.

b) Terminal connectivity over LANs, interconnected LANs and WANs: More and more individual LAN segments are connected together using bridges (level-2 filtering) and Routers (level-3 filtering). Using such equipment improves efficiency and brings new benefits to local area networks. Also, there is a complementary market requirement for wide-area-network connectivity (PADs) as a simple, low-cost solution for remote end-users. The final choice will be based on parameters such as cost and availability of connection, bandwidth, etc. A complete offering must be able to address these different requirements.

c) Terminal connectivity (migrating from asynchronous to LAN-attached devices) integrated into customer wiring solutions: local area networks have been designed to offer connectivity to a wide range of products including systems, PCs, X-terminals, special devices, etc... Terminal servers must accommodate the particular structure and constraints of each network environment. Some environments tend to centralize equipment in a few places and connect them via backbones, others tend to distribute the equipment across the various company sites.

From these high-level requirements, and after surveying many customers, we determined the specific requirements faced by HP systems users for end-user connectivity :

- Consistent access to HP3000 and Telnet/TCP/IP systems for all network end-users (including DTC end-users)

- Full industry-standard protocol implementation for high-level interoperability

- Competitive cost of ownership

- Access to HP3000 block-mode applications using the Telnet protocol without sacrificing performance

- Integrated network management functionality, based on a scalable offering

IV. HP's terminal connect strategy

IV.1 Value Proposition

HP's strategy is based on a "Value Proposition" which is basically what HP wants to provide to it's customers:

VALUE PROPOSITION

"Significantly help HP Multiuser System Customers improve operational efficiency and customer service through fast, reliable and transparent end-user access to information in a multivendor environment at a competitive cost of ownership"

Companies must be able to focus on meeting their business requirements. In terms of end-user connectivity, final customer service can be significantly improved if end-users can access the information they need when they need it, with the minimum effort. This will be achieved by focusing on key elements such as performance, reliability, location-independence, and multivendor connectivity while ensuring competitive cost of ownership.

IV.2 HP's Terminal Connect Strategy -- Basic Elements

The combination of the three following elements will deliver this value proposition and introduce the concept of "communication server":

a) A competitive, scalable, hardware architecture of multiplexers, PADs and communication servers.

Two factors determine the best solution to meet specific or global connectivity requirements:

-The physical location of end-users relative to the computer room.

-The number of connections to a specific system and the number of computers that need to be accessed by a given end-user.

For small and medium standalone HP9000 systems, HP's strategy is to offer system-integrated solutions.

For multi-system environments, with medium to large computers connected by LANs, where end-users requireaccess to multiple systems, HP's strategy is to provide a modular communication server that can accommodate a small or large number of users, and potentially mix local and remote access cards. This modular communication server can be either located centrally or in distributed locations depending on the site environment.

b) A standards-based, multivendor end-user connectivity solution based on the following powerful concepts :

-Supporting multiple protocols in the same communication server.

-Providing location-independent end-user access. Local and remote end users can access any applications available anywhere on the company network.

-Combining the benefits of standard and optimized protocols. Each protocol has been designed for a specific environment or "world." Optimized protocols were created to provide efficiency and performance in the OLTP world; standard protocols were created to provide uniformity of access in the multivendor world. HP is committed to industry standards while simultaneously providing the best OLTP price/performance ratio. That is why combining standards and optimized protocols gives the best of both worlds to all end-users.

c) A scalable device management solution:

For standalone systems with or without remote access, HP's strategy is to offer host-based networkmanagement. System and network management is then integrated under the control of the system manager.

For multi-system sites, whether HP-only or multivendor, with diverse equipment such as communication servers, PADs, hubs, bridges, routers, and switches, HP's OpenView Network Management strategy allows for the integration of all management elements under a consistent windows-based user interface, controlled by a single operator.

V. Major steps to achieve the vision, from strategy to products

The multiple key elements, based on the DataCommunication and Terminal Controller (DTC), which demonstrates the today achievement are the following:

V.I The DTC family

The DTC solution consists of a scalable family of multi-vendor communication servers:

-DTC16 : provides up to 16 asynchronous connections and one wide area network link. This DTC can be used in distributed environments where small groups of end-users needs access to network applications, or integrated into low-end system cabinets.

-DTC48 : provides up to 48 asynchronous connections and up to 3 wide area network links. This DTC can be used in large centralized environments (EDP rooms for instance) where many connections are required, or decentralized at departmental level.

Both platforms are 100% functionality compatible, providing the same end-user service.

V.II Multivendor Connectivity:

The DTC, originally developed as a communications server for HP systems (DTC/3000 and DTC/9000), has evolved into a true multivendor terminal server implementing both the HP3000 optimized protocol and industry-standard TCP/IP.

The DTC runs multiple protocols in the same server to access HP3000, HP9000, and non-HP systems (using Telnet/TCP/IP), as well to non-standard computers (using back-to-back configurations).*

V.III Location-independent end-user connectivity:

The DTC offers access to these two protocols from local devices (connected directly to the DTC) or from remote devices (connected to X.25 PADs or modems) offering location-independent end-user connectivity. DTC remote end-users now have access to ARPA systems, as well as to non TCP/IP computers connected in back-to-back configurations.

Also using Telnet/TCP/IP, end-users can connect through networks interconnected with bridges or routers. Network managers can minimize the number of protocols on their backbone, making it easier to manage.

V.IV Combination of standards and HP3000 optimized protocols:

The DTC offers a protocol conversion capability between Telnet/TCP/IP and the HP3000 Series 900 optimized protocols, to provide the most powerful Telnet solution on the market. Implementing Telnet in the DTC relieves the HP3000 Series 900 of CPU-intensive telnet protocol processing.

This is possible with the addition of a protocol translator card which requires one slot in the DTC48 or with a dedicated server for customers who do not require terminal connection in the same server.

---

* DTC48s with date code prior to 3110 need the DTC48 Upgrade Kit, HP2348A to support multivendor functionality. The DTC16, and DT48s dated 3110 or later, need no upgrade kit.

V.V Scalable network management integrated under HP OpenView

Not all customer environments require all the features discussed above; a scalable network management solution allows you to chose the appropriate solution for your environment.

In standalone system environments (HP3000 or HP9000), DTC management functions are performed directly on the host. In such environments, a single operator manages both the system and the associated network. The communication server is therefore dedicated to this system.

In multi-system environments including HP and non-HP equipment, an HP OpenView PC-based workstation is the central point of management for all network elements (such as HP OpenView DTC Manager, Switch/Pad manager) under the control of one operator. It offers a broader set of features for DTC management including dynamic configuration and an easy-to-use friendly graphical interface integrated under HP OpenView.

During 1991, all the HP OpenView Management applications based on DOS will run and coexist on the same PC workstation, helping you to optimize the cost of managing equipment and the associated cost of operations, and improving your control of your network. Networkable management protocols allow one workstation to manage other OpenView workstations via the network.

V.VI Service and quality of service

Support services are based on a comprehensive test strategy centered on HP Field expertise and the execution of a specially designed set of test procedures. These procedures are built on our own stringent internal Quality Assurance tests. They benefit from extensive "real life" studies based on customer configurations.

These services combined with existing HP NETASSURE support services will give you the best possible insurance that your specific configuration will provide you with lasting and uninterrupted service.

VI. Conclusion: How the DTC meets new market requirements

HP has a very clear strategy in terms of end-user connectivity. The building blocks are in place for a very competitive offering which will help ensure the business success of you and your end-users. The DTC has evolved from an HP3000 Series 900 terminal server to a multivendor communication server optimized for the HP environment.

It provides :

- Consistent access to all HP systems

- The benefits of both multivendor standards and OLTP optimized protocols.

- Location-independent access

- Scalable network management integrated under HP OpenView

- Configuration testing based on a comprehensive test strategy

With the HP "Service and Quality of Service" concept, you have the assurance that HP's competitive communication server products and features can be used and managed in a cost effective manner. The DTC is a multivendor communication solution which is your best choice for end-user connectivity.

Interex North American Conference
August 5-8, 1991
San Diego, CA

Paper Number 2022

SNMP, OPEN SYSTEMS, AND OPEN NETWORKS: THE STATE OF THE UNION

Joseph E. Grim
joeg@hpcndm.cnd.hp.com
Hewlett-Packard
3404 East Harmony Road
Fort Collins, Colorado 80525
303-229-3910

ABSTRACT

The Simple Network Management Protocol (SNMP) is proving to be the protocol of choice in the management of open networks and open systems. But MIS directors and network managers are receiving confusing signals regarding network management standards. Both the Common Management Information Protocol (CMIP) and CMIP over TCP/IP (CMOT) are looming on the horizon. This paper attempts to cut through the confusion by discussing some SNMP implementations for managing real networks, and why SNMP will be a critical component in the Internetworking Decade. It also covers SNMP topics of interest to customers using HP 9000's, HP 3000's and HP Personal Computers.

INTRODUCTION

The changes in networking over the course of the 1980's were so vast that if some data communications Rip Van Winkle had gone to sleep in 1981 and awakened in 1990 he would hardly have recognized the world where he awoke. At the beginning of the decade, local area networks (LAN's) were a brand-new, cutting-edge concept. Networks at that time generally consisted of proprietary systems from a single vendor. If those systems communicated at all, it was usually via protocols that had been invented by that vendor. Operating systems were geared toward running specific types of applications, and the

networking tended to be specific to the operating system.

By the beginning of the 1990's, the directions for networking and computer systems were clear. Proprietary was out and Open was in for good. A number of factors went into the sweeping change toward open systems and networks:

- Customers were tired of having to entrust their business success to a single computer vendor;
- Unix, because of its power, scalability from PC to supercomputers and superb development environment, became the open system standard;
- The cost of hardware that could run Unix effectively became so low that almost any enterprise could afford it;
- By virtue of the Berkeley Software Distribution, Unix became a pervasive networked operating system with TCP/IP as the preferred communication method;
- Because of the efficiency of the Request for Comment (RFC) process, TCP/IP very quickly became a well-defined de facto standard.

During the second half of the 1980's, and particularly during the last three years of the decade, Unix and TCP/IP took off. Corporate Internetworking became a clear direction for the future, as many companies discovered the virtues of using TCP/IP as a key business tool. The movement toward standardized systems and networks was first seen in universities, government and in engineering companies. These were the first users of engineering workstations and minicomputers, which generally had TCP/IP built in. Then many commercial industries followed, particularly financial services, which needed the distributed compute power that workstations provided.

Also during the late 1980's, many users of personal computers climbed on the open networks bandwagon and implemented TCP/IP networking for file sharing and print sharing. IPX (Novell) is still the dominant protocol in the PC networking world, and users of IPX have recognized the need for connectivity to, and management by, the TCP/IP world.

Amidst the excitement of open systems and open networks, an unpleasant fact has emerged. These networks somehow have to be managed and kept under control. In most real-life cases, thoughts of managing networks come after they are installed, not during the design process. Most network managers have devised ad hoc tools based around

standard TCP/IP utilities, but these tools tend to be useful in relatively small networks and less effective in larger ones.

Thus we have the world of network management standards, with its two core contenders, the Simple Network Management Protocol (SNMP), and Common Management Information Protocol (CMIP). SNMP arose from the pragmatic Internet community, whose users have problems very much in need of solutions and who are more interested in imperfect solutions that work than world-beating ones that are slow in coming. CMIP arose from the international Open System Interconnect (OSI) community, and is architected on OSI protocols.

Because of CMIP's close ties with OSI protocols, its fate is closely linked to the pervasiveness of those protocols. For the duration of 1991, TCP/IP will continue to be the world's most commonly installed open networking standard. Since SNMP is built on TCP/IP, it is enjoying a great deal of success. It has become the network management standard of choice for Internets, and has inspired a number of useful applications. We will explore some different areas where SNMP is being used successfully today.

SNMP Today

According to the DataPro Research Group [2], as of mid-1990, somewhere between 500 and 800 commercially available, turnkey SNMP management packages had been sold, not including any packages that were given away or came from the public domain. This number is up considerably from 1989, when a small fraction of that number were sold. In 1991, the 1990 numbers could double or triple. Why have SNMP management applications experienced such success in such a short time?

The reasons are many. The primary one is that network device vendors, such as manufacturers of routers, bridges, hubs, terminal servers and multiplexers, including HP, have happily adopted the SNMP standard and put agents on their devices. SNMP is relatively simple to implement, and because of its simplicity, it doesn't use much precious ROM space in these devices. In fact, in a short time span, manageability via SNMP has gone from a novelty to a critical competitive issue for almost all network device manufacturers. In addition to network devices, SNMP agents have also been created for most major computer systems, laying the groundwork for both network and system management via SNMP.

Another reason for SNMP's success is that a number of vendors have produced SNMP management applications. These vendors include HP, Sun, cisco, Wellfleet, NCR, Advanced Computer Communications, DEC, and a number of others. At Interop '89, there was a small handful of vendors demonstrating SNMP management applications. The focus at that time in the SNMP Interoperability Showcase was on agents. At Interop '90,

there were approximately 40 SNMP manager applications being shown. Customers looking for network management solutions have a number of choices today.

The SNMP Environment

If you are a network manger, telecommunications manager, or MIS director, what are you to make of all the confusing talk about the different network management protocols? What are your best choices to make today?

If you have to manage a TCP/IP network, particularly one that consists of interconnected LAN's of PC's, Unix workstations, multiuser systems, and/or mainframes, SNMP is the obvious choice today. At the very least, SNMP will help you to monitor and control your network infrastructure, such as your routers, bridges and hubs. If there are SNMP agents available for your computer systems, then your management environment will be much more complete because you will be able to extract information about each system, such as system type, operating system release number, interface addresses, contact names and system locations. SNMP agents are available for most Unix implementations and DOS and OS/2 PC's either from the system manufacturer or from third parties. There are also agent sources available in the public domain.

In the case of HP systems, there are SNMP agents available now for the HP 9000 Series HP-UX systems. With the HP-UX 8.0 release, HP, like a number of other Unix system vendors, started bundling the agents with the operating system. HP has put a number of special features into the HP 9000 agent, such as the ability to query disk space and the ability to monitor CPU load. In the near future, there will be an SNMP agent available for the MPE XL HP 3000 systems, making 3000's integrate nicely into the SNMP management environment.

Your SNMP management environment will consist of two basic elements: one or more managers, and agents on as many devices as practical. The SNMP manager could be an application such as HP OpenView Network Node Manager that will typically run on a Unix workstation and have a graphical user interface with a map of the network.

Many of the SNMP management applications provide some sort of dynamic discovery feature that automatically finds the nodes on your TCP/IP network. This is an extremely useful capability that can save the network manager hours of effort in drawing the network map. Maps typically use color to indicate the status of systems or network devices (e.g. green for up, red for down). The manager application receives SNMP events, called *traps*, from the different devices on the network that inform the user of alarm conditions.

In order to present the status of each of the network devices and systems, the manager application polls the devices at some user-designated interval. This approach does have the disadvantage of causing some additional network traffic, but even in networks of hundreds of nodes, with polling intervals of five to ten minutes, the amount of traffic generated is not objectionable. SNMP manager applications take advantage of both TCP/IP and SNMP to provide fault management. For example, many SNMP managers use the simple TCP/IP utility ping to do ongoing status checks, but use SNMP in more complex situations, such as finding out where routing has broken down between two nodes on the network. SNMP can be used for diagnosis in numerous different fault conditions, such as routing problems or duplicate IP addresses.

SNMP agent capabilities vary with the type of device on which the agent is running. The basic set of Management Information Base (MIB) objects is defined in Internet RFC 1066, and allows the network manager to identify the type of device and gather information on the device configuration and network traffic statistics. In addition to this base set of objects, most devices also have MIB objects that are specific to both the manufacturer and the type of device. For example, cisco, Wellfleet, Proteon, and HP routers all have MIB objects specifically geared toward routers, and each has a different set of objects that were defined and developed by its vendor. As mentioned previously, HP developed a number of special objects for the HP 9000 and HP 3000 agents.

Virtually all vendors who have developed their own collections of MIB objects make descriptions of the objects available to customers in the form of ASCII text files in ASN.1 format. Many of them are downloadable from the Internet. It is the job of the manager application to access and use these vendor-specific MIB objects, and most of the commercial applications have this capability today. The application has to have some way to load the vendor-specific MIB objects, which involves interpreting or compiling the text file containing ASN.1 descriptions of the objects.

The use of vendor- and device-specific MIB objects opens up many new possibilities for managing all levels of the 7-layer OSI network model. For example, there are a number of LAN monitoring instruments such as Novell's LANtern and shortly HP's LANProbe that support SNMP agents. These agents collect statistics on parameters such as Ethernet performance and utilization, and save them in SNMP MIB variables. All a management application has to do is query the appropriate MIB objects in the Ethernet instrument and the network management station turns into an instrument with many of the capabilities of a dedicated LAN analyzer. In fact, there is a working group within the Internet Engineering Task Force that is presently defining a standard set of MIB objects for distributed link-level monitoring.

What are the downsides to SNMP? Since SNMP is a simple protocol, it is geared toward moving data in small chunks rather than in large masses. This makes it ideal for gathering

things like statistics and routing tables. But when there are larger quantities of data involved, such as in some sophisticated system management applications, SNMP can become slow because of its one-item-at-a-time type of query. In addition to that, an SNMP manager is only as effective as the agents from which it is gathering data, and in some instances agents may not provide all the information a network administrator needs. Also, since SNMP is designed such that the management application does most of the work and agents are strictly lightweight processes, polling is the method used to gather information rather than reporting. The polling creates additional network traffic, which can be a problem on large networks or those that involve wide-area components with per-packet charges.

Nevertheless, when applied properly, the benefits of SNMP greatly outweigh the problems mentioned above. SNMP's success in the market is a strong testament to that.

The Future

Future directions for open network management protocols are not totally clear, but there are several different scenarios. One, which is highly unlikely, is that SNMP will be quickly replaced by CMIP Over TCP/IP (CMOT), and then ultimately by CMIP as OSI becomes more common. This is unlikely because of the installed base of SNMP and the considerable benefits that network managers are deriving from it. CMOT has not gained anywhere near the wide acceptance that SNMP has among device and system vendors. For many device vendors, it is not presently practical to implement either a CMOT or a CMIP agent because of memory and cost limitations. And, while there are many SNMP managers implemented now, CMIP managers are still mostly experimental as of this writing.

A more likely scenario is that both SNMP and CMIP will each find their respective niches because they are good at different things. SNMP is strong at managing campus-level networks of interconnected LAN's, which have routers, bridges, multiport repeaters, terminal servers, and computer systems. SNMP is good at fault, configuration, and performance management in these types of environments because the load on systems doing real (i.e., non-network management) work is minimal, and the polling method that SNMP uses to get information and perform control is quite adequate.

CMIP will likely find its home in enterprise-level network management. Since it is geared toward getting information in bulk [1], according to Jeff Case, one of the inventors of SNMP, it lends itself to managing larger devices and systems. In fact, one of the primary groups of CMIP advocates is the Post Telephone and Telegraph companies in Europe, who are inspired by the European Economic Community's 1992 deregulation to unite their wide varieties of network devices under the CMIP/ OSI banner. In addition, CMIP will make a good platform for an enterprise-level manager that counts on lower-level site

managers to report conditions across the network.

Since CMIP is an international standard, and even today, most networks have devices with proprietary protocols, CMIP is a logical choice both for management systems to pass information between themselves, and as an umbrella protocol with proxies to proprietary protocols.

In the meantime, there is a flurry of activity on the SNMP front. Vendors of different kinds of devices are working at standardizing the MIB objects for those devices, such as bridge vendors, router vendors and link-level monitor vendors. These standardization activities will be beneficial to end-user network managers, as the combined knowledge of the different device vendors will result in a more complete set of objects in all the devices. With better standardization across devices, network management applications will be able to make more optimal use of the data they collect from the devices.

The reality of network management today is that there are protocols other than TCP/IP and OSI on networks. IPX (Novell) dominates at the PC workgroup level and SNA dominates at the enterprise level. As a result, in the near future, there will be solutions available that allow PC workgroups running IPX to be managed by SNMP applications. The solutions will likely take the form of proxy agents that instrument key aspects of IPX with SNMP, and report on them to the SNMP manager application. Looking up toward the enterprise level, SNMP managers often need to pass alarms to IBM Netview so they will be registered at a central network operations center. It is unclear whether this will more often involve Netview accepting SNMP events or SNMP managers packaging events for Netview, but solutions will be available.

Summary

Standards-based distributed computing is now a fact of life all over the world. Network management is critical to getting the most out of these computing environments, both for keeping them running and optimizing performance. SNMP has emerged as the most widely used protocol today for managing TCP/IP networks, and promises to continue to be the most popular for at least the next several years. By virtue of having broad support among vendors, it will continue to be enhanced and to provide solutions in many different computing environments. CMIP, while not very common now, will become more popular both as OSI networks are more widely implemented, and as it becomes the protocol of choice for communicating between management stations and for uniting different devices that formerly used proprietary protocols.

Bibliography

1. Sharon Fisher, *Dueling Protocols*, BYTE Magazine, March 1991, pp. 183-190.

2. DataPro Inc., *DataPro SNMP Product Guide*

Business Intelligence; A Key Component of Third-Generation Office Information Systems

By Garry Orsolini

Hewlett Packard
Software Technology Center
Roseville, California
916-785-4624
National INTEREX Conference: August, 1991

Strategic Information Systems; No Longer an Option!

Global competition is everywhere!. Companies are dispersed geographically and increased customer expectations are making time to action and time to market more important than ever.

Winners in competitive markets have successfully harnessed technology to meet business objectives. And today, aggressive organizations are looking for strategic information systems that will allow them to gather, understand, communicate, and act on critical information faster and better than ever before.

According to Jan P. Herring, Vice President - Business Planning and Strategy for The Futures Group, "Today's business environment is the most complex and competitive in history... to achieve a competitive advantage and to counter aggressive domestic and foreign competition, U.S. companies must begin to develop corporate intelligence systems". (Reference 1).

Evolution of Office Information Systems - Business Intelligence

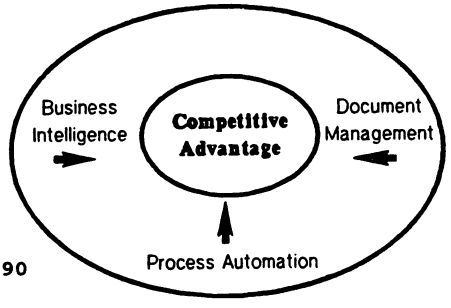
During the 1970s, many successful companies implemented first-generation Office Information Systems (OIS), which provided host-based word processing for clerical working. In the 1980s, practically every large organization installed second-generation OIS that offered personal and departmental productivity tools from PCs and host-based systems. Today there are an estimated 10 million second-generation OIS users.

But the 1990s will find the emergence of third-generation OIS- an enterprise-wide information system. According to the Gartner Group, "Third generation OIS expand the role of OIS across the enterprise, delivering - to potentially all knowledge workers - a window into the information resources of their enterprise and

a suite of applications that enables effective use of that information in each specific job function." (Reference 2).

Key components of Third-Generation OIS

A broad spectrum of customers and experts agree that there are three primary, compelling application categories that will stimulate the implementation of third-generation OIS:



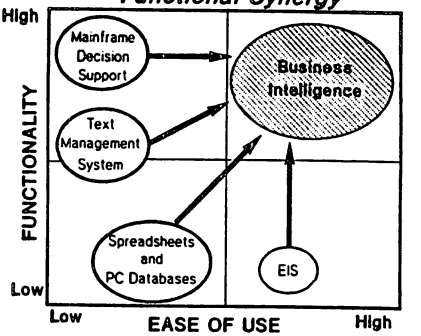
Source:
GG OIS 1990

Document Management is the ability to capture, organize, and retrieve all types of documents/images within an organization. Document management includes both electronic scanned-image management and object-based document management. A network Document Management solution should keep and find any type object: text, graphics, spreadsheet, voice, video, image, or compound. It should also provide archiving, backup, revision control, security, and administrative functions.

Process Automation enables the automation of individual and workgroup tasks and business processes. Such applications improve personal and organizational productivity by automating repetitive tasks, such as preparing and routing a monthly report. Process automation combines intelligent agent facilities with event-based triggers for initiating processes, such as alerting buyers or sending electronic purchase orders when inventory drops below an optimal level.

Business Intelligence (BI) is an emerging concept that is finding its way into the commercial business environment. Just as strategic planning migrated from the military to business after WWII, Business Intelligence can trace it's roots to military intelligence-gathering. Gartner Group describes BI as the desktop integration of decision support, text management, spreadsheet, PC database, and EIS functionality.

Functional Synergy

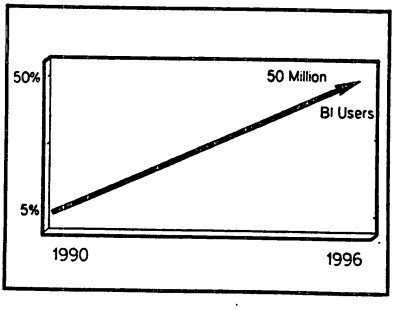


Source:
GG OIS 1990

According to Herbert E. Meyers, author and vice chairman on the U.S. National Intelligence Council, Business Intelligence "is the way a company harnesses information that will help it achieve success in a global environment". He goes on to describe BI as, "the corporate equivalent to radar, which is constantly scanning the external environment through which the company is moving". Thus Business Intelligence can be viewed as a mechanism or process used to identify what the company needs to know, a way to monitor and analyze information for what it means, and a way to make it available at the right time to support strategic decision-making activities. (Reference 3).

Business Intelligence is intended to provide companies a broad range of tools for gathering, analyzing, and disseminating both internal and external information relevant to their business. Gartner Group believes that almost all OIS users will have a need for BI, thus they conservatively estimate that 50 percent will acquire BI by 1996. They assume that as the OIS population grows to 100 million, the BI user community will grow from 500 thousand in 1990 to 50 million Business Intelligence users by 1996.

Business Intelligence Growth

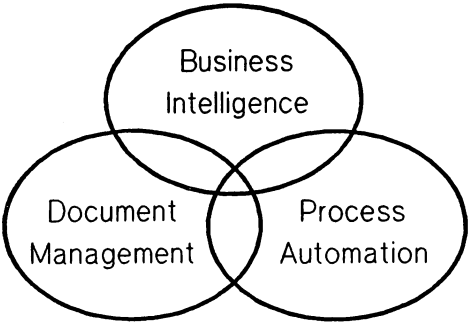


Source:
GG OIS 1990

% Of OIS Users With BI

Key Applications Are Functionally Inter-dependent

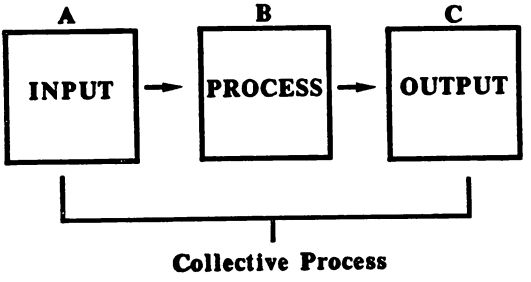
While each of the three major OIS applications can be considered a separate solution, their functionality often overlaps.



Business Intelligence applications quite often demand both document management and process automation functions. Document management applications can require process automation. Because these applications must work together effectively, they must be built within a solid third-generation OIS environment.

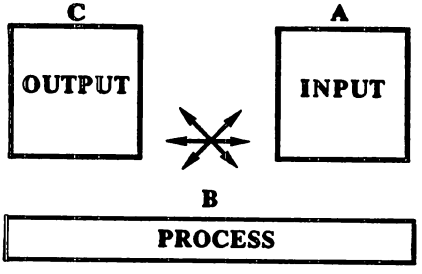
Process is KEY to BI Solutions

Fundamental to a Business Intelligence solution is the concept of PROCESS. Most students learned in Computer Science 101 the generic formula for building information systems:



These steps taken collectively represent a process that in turn becomes a component of other processes. However, in constructing BI systems, it is apparent that the linear flow from box A to B

to C should be revisited. In fact, BI solutions mandate that the architect start with box C - output; where the critical success factors, business goals, and objectives of the company are uncovered. Next, box A - input is examined to determine the condition and availability of company data and external data/information sources. Lastly, box B; to determine the mechanics and logistics of how to deliver the solution required in box C when starting from box A. This re-engineering of the macro process reveals that it is non-linear and very iterative.



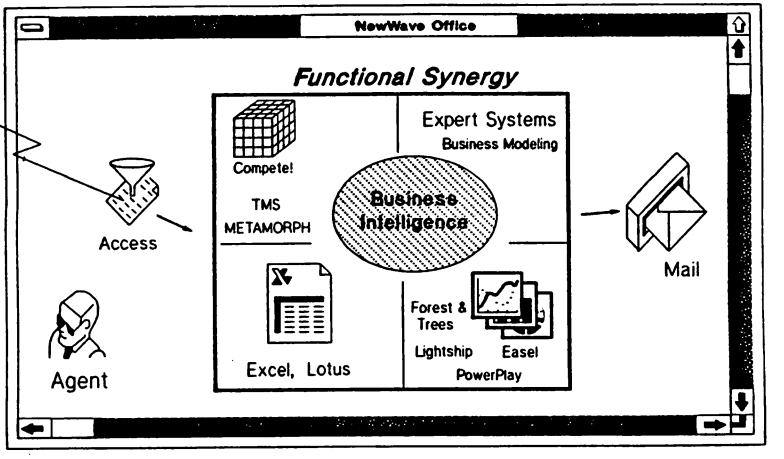
Herbert Meyers outlines his own three-step strategy for developing a Business Intelligence system. His first step is to develop a corporate intelligence profile. Next an intelligence audit is performed to determine what information the corporation currently reviews. And the last step is to create a system that analyzes the incoming information and distributes it to the appropriate decision-makers. (Reference 4).

BI Systems Must Be Built, Not Bought!

If an organization had all the right data, in the right place, in the right format, available upon request, then a Business Intelligence solution could be bought "off-the-shelf" and implemented. But alas, few if any organizations find their data and information infrastructure nicely mapped to a comprehensive corporate data/business model. Thus the problem continues to be one of building new solutions, while dealing with a quagmire of multi-dimensionally dissimilar hardware, software, and data. Most vendors selling BI component software will, if pressed, (or cornered) admit that BI solutions must be built - no magic or free lunch here!. Thus application software that functions optimally as both a stand-alone product as well as a component of an integrated solution, has tremendous value. Component applications offer greater flexibility when tailoring a BI solution to a company's intelligence processes. However, the downside to a component architecture may be a steeper learning curve and incomplete component product integration.

Open, Standards-Based BI Environment

The remainder of this paper will discuss how the NewWave environment and NewWave Office provide an open, standards-oriented environment for integrating all the BI components identified by Gartner as well as other key functionality not referenced in their definition. The following graphic shows a variety of third-party and HP applications that constitute a "suite of tools" that can be used to construct Business Intelligence solutions.



Surrounding the "Functional Synergy" box above (which contains the component applications), is NewWave Office. NewWave Office provides key functionality beyond MicroSoft Windows 3.0 and the NewWave Environment, upon which NewWave Office runs. NewWave Office provides the three key, core services needed in building BI solutions. The first is access to data, the second is organizational communications (or - E Mail), and the third is process integration and automation.

Delivering Data to the Desktop

NewWave Access provides automated data delivery from multiple data sources to popular desktop applications. Refined database information can be displayed directly by NewWave Access -- top 10 lists, exception reports, time series data -- or it can be shared with spreadsheets, graphics, documents or management reporting applications (PowerPlay, Forest & Trees, Lightship, Easel).

NewWave Access simplifies the process of data retrieval and integration. Using a graphical interface and iconic representations of data, analysts and business professionals can directly access and manipulate data without needing to know SQL syntax.

Delivering Data, Information, and Intelligence Throughout The Network

A key service needed to deploy BI solutions is the ability to get the right "Intelligence" to the right person(s) at the right time. For the dissemination of Business Intelligence media, more than simple E-Mail is needed. The service must provide complete organizational communication capabilities, accommodating all types of media including: text, graphics, spreadsheets, video, audio, animation, and others being developed. NewWave Office Information Distribution Services enable users to exchange these types of media with other users on both private and public mailing systems.

According to the I/S Analyzer in an article entitled Using Information Systems For Business And Competitive Intelligence, "the most important application is electronic mail, and it is critical to gathering and disseminating business intelligence". (Reference 5).

Process Integration And Automation

Key to deploying a Business Intelligence solution is the ability to integrate multiple applications into an automated procedure that can be repetitively executed on a scheduled or event-triggered bases. Within the NewWave Object-oriented environment, applications can be integrated and automated using the agent task language. Most activities can be recorded when developed or executed, and thus re-executed or "played-back" in any of four ways. First, an agent task can be executed by dragging and dropping the task onto the agent tool. An agent task can execute uninterrupted, or if desired, a dialogue box(es) can prompt the user for input, or simply pause, and display a message. The second method is to "hide" the agent task under a Task menu label within an application. This simplifies the initiation of the task for a novice user. The user simply selects the task name from the pull-down task menu, and the task is executed. Common examples of this would be: "Empty Waste Basket", "Create and Mail Memo", or "Display Today's Appointments". The third way to execute an agent task is to drop the task on the repetitive scheduler. This allows for great flexibility by tying the agent to the system clock. Events can be executed regularly, or a custom pattern can be specified. This feature is commonly used to initiate PC backup utilities on a regular basis in the wee hours of the morning. The forth and final way to execute an agent

task uses the "Agent Trigger" which provides for true client-server event-triggered processing. The Agent-Trigger resides on the PC within NewWave but it looks for a file (of your choosing) to be time-stamped on the network. Each Agent-Trigger task is independent and can key off the same or different network file(s). Agent-Trigger tasks can be chained together by using network file sharing to post, or time stamp the next trigger file in the sequence. This technique enables true value-added processing to occur on both the server and the PC client.

Core Services Provide Foundation For BI Solutions

With the key, core services now in place: data delivery to the desktop, organizational communication, and application integration and automation, specific Business Intelligence "building-block" applications can now be deployed to fill particular information needs. The following will discuss some of the available applications.

Cognos PowerPlay – Providing Multiple Perspectives and Clarity to Business Information

* **Multidimensional Views** PowerPlay takes large volumes of detail transaction data and transforms it into meaningful summarized business graphics or table "views". Managers can explore several layers of their business model to uncover results or trends in their critical success factors.

* **Slide Show EIS** With the PowerPlay Carousel, managers can interactively browse through a slide show of summary graphs and reports, with the option of "drilling down" into the slide to see more detail.

Channel Computing Forest & Trees – An Electronic Dashboard for Business Information

* **Management Warning System** Forest & Trees highlights changes and exceptions in key performance measures (KPMs) and "vital signs" through the use of alarms and color coding. Managers are able to monitor these metrics, while analysts can follow up using the built-in report generator.

* **Data Access Tool** Advanced users can collect and combine data from multiple data sources using an interactive SQL interface. Resulting data is then displayed in the dashboard as one of many "views" -- a number, list, or graphic. These same users can build summary reports or browse tables.

Pilot Executive Software Lightship - A Development and End-User EIS

* **A Powerful EIS Tool** Easy to use and understand, point and click development tool that can integrate ASCII files, reports, spreadsheets, popular data formats (BMP, PCX), or DDE links into virtually any graphical display - complete with charts, hot buttons, drill down, and navigation. Variable passing and application initiation make Lightship a powerful development tool.

* **End-User EIS** With a simple toggle (F4) the developer can switch to run-time mode and all development visuals are hidden. For the EIS user, complexity is determined solely by the developer. Most forms of EIS systems can be easily constructed including "dashboard type" color-coded good news/bad news reports.

ManageWare Compete! - Multi-dimensional Spreadsheet With Decision Support and Reporting Capabilities

* **Powerful Decision Support** Iterative goal seeking, sensitivity analysis, and forecasting are easily achieved across multiple dimensions (12 dimensions, current release).

* **Data Import and DDE Integration** Data can be imported from a variety of sources and Compete! will attempt to build a multi-dimensional model using default assignment to literals and variables. Users can easily add, delete, or modify any of the dimensions or parameters of the model. DDE is used to pass data between Compete! and MicroSoft Excel.

Thunderstone Metamorph - Text Retrieval and Correlation

* **A Powerful Interpretive Analysis Tool** that uses a process based on performing morphological analysis of text. This technique does not rely on either file inversion or tokenization (which require the data to be modified so that it can be searched in an orderly fashion). Metamorph uses a natural language query syntax that will reference a 250,000 word thesaurus for word or phrase equivalences. It has been benchmarked at a speed of 4MB of data per second and because it can process any "RAW" (non indexed) data or ASCII file, it becomes a powerful intelligence gathering tool.

  At Lincoln National Information Services, Inc., a wholly

owned subsidiary of Lincoln National Corporation (23 billion insurance and financial services), Business Intelligence is defined as, "finding the right text-based information and getting it into the network for use by the right people at different levels of the corporation". At Lincoln they discovered that "most executive needs were not for information in the corporate database, or perhaps any conventional database, anywhere", but rather for a "business intelligence-gather 'frontend' used to accumulate vast amounts of text and non quantitative information that people could search and manipulate in a variety of ways". (Reference 5).

Business Intelligence - The Process and the Challenge

The real challenge in building a Business Intelligence solution is not in the selection of the tools to use, but in understanding the key processes within the organization by which strategic information is disseminated and critical business decisions are made. Products that support the BI endeavor, will come and go, and the best will evolve to incorporate more of the PROCESS, inherent within their functionality.

BI Deployment; The Result Of Intelligent Analysis And Planning

Typically with the design and deployment of a Business Intelligence solution, the following consulting services are provided through either, a users group, the internal MIS department, or an external consultant to the organization:

Benefits Analysis - Prioritize Strategic Opportunities and Associated Benefits. This usually takes the form of a benefits analysis where the consulting team helps the customer identify and prioritize areas where technology can significantly help their organization reach their business goals.

Information Needs Assessment - Define Functional Requirements. Using an enhanced Requirements Analysis process, the consulting team conducts an information needs assessment. The key deliverable is a written report which documents the customer's current processes, recommends areas of processing improvement, and defines the functional specifications required to support the customer's operations. This report can serve as a basis for a request for proposal (RFP).

Solution Design and Pilot - Select Solution and Build Pilot The next step is the design of a solution that links to the customer's business goals and objectives. The consulting

team also develops a plan which defines the scope, strategy, timeframe, and resources required to implement the selected solution. This phase typically includes managing a pilot implementation to test the proposed solution in the customer's environment.

Benefits Of the Above Process

* Business goals can be achieved through a cost effective use of the companies Information Technology (IT) investments.
* IT investments can be protected.
* Risk is limited when you think BIG, but start small.
* Internal processes can be enhanced without incurring long-term personnel expenses.

Business Intelligence; Magic or Method?

Strategic decisions will continue to be made whether via a crystal ball, an elaborate computer-based "BIG Brother" intelligence, or more realistically, some method between the two. In any case, as global business competition heats up in the 90's, timely, and relevant information will increase in value. Failing to plan now for the gathering and disseminating of critical business intelligence, is no doubt, a plan to fail.

REFERENCES:

1. Building a Business Intelligence System
   Jan P. Herring
   The Journal of Business Strategy; May/June 1988

2. Gartner Group OIS; 1990
   Ninth Annual; Office Information Systems Conference
   Next (Third) Generation Office Systems: Implementation

3. Using Information Systems For Business And Competitor Intelligence
   I/S Analyzer; United Communications Group
   May 1990; Vol. 28, No. 5, Pg. 4.

4. Ibid., Pg. 5.

5. Ibid., Pg. 2.

Using an RDBMS to represent Engineering Designs

Philip Walden

Hewlett-Packard
3155 Porter Dr., Palo Alto, CA 94304

1. Overview

Hewlett Packard has used an RDBMS as a neutral definition of product design data such that functional areas from R&D, Manufacturing and Business systems can effectively operate from a common, consistent and accurate data source. This paper provides an overview of the database design and supporting application modules. It will also discuss how it supports HP's strategy of global engineering and manufacturing.

2. Introduction to the Product Information Center (PIC)

The Product Information Center (PIC) was developed to provide a neutral definition of product design data as a common medium for exchange of information between functional areas of HP. Rather than start with a "Dooms-Day" approach and include every datum known to man-kind, it was started relatively small and focused on the interchange between electrical engineering and product assembly during the process of introducing new products and new revisions of products. Even though the PIC handles a large scope of engineering-assembly data, specific PIC applications have focused even more tightly on circuit board level designs and board assembly.

2.1 Why HP built a PIC

HP has been "globalizing" its R&D and Manufacturing capabilities for several years. This process includes both "internationalization" for the global market and "centralization" to achieve the economies of scale required for "world class" manufacturing capabilities. This process has transitioned HP from a set of rather autonomous divisions with self-contained R&D and manufacturing capabilities to many R&D-marketing business operations and fewer, but larger centralized manufacturing facilities. With this change, barriers of distance, communications and coordination have arisen. The PIC and its surrounding applications were developed to overcome these hurdles.

A second factor has been the increased use of computer aided engineering (CAE) systems needed to maintains HP's competitive edge and reduction in the time to market new products. R&D has chosen CAE systems to meet their business and market needs and because HP designs to a diverse variety of markets the result has been a diverse variety of CAE systems as sources of product design data. Combined with this dilemma, is the current lack of sufficient exchange standards not only between CAE systems, but also between CAE and manufacturing systems. The PIC was developed to act as the neutral exchange format and repository of this information.

2.2 The PIC does not stand alone

The PIC exists within an environment of other information sources and in fact is coupled to some of them. Two information centers of note are the Component Information Center (CIC) and the Product Data Management (PDM) systems.

The CIC was developed as a central repository for all HP purchased component information. This system is maintained centrally at HP corporate in Palo Alto and a global set of satellite CICs are kept synchronized by daily updates distributed by HP's world-wide computer network. The PIC accesses the CIC for information pertaining to components used within product designs.

The PDM is a relatively new information center and is just now being implemented within HP. Its purpose is to store and manage product documentation electronically. Anything that can be stored as a file can be managed by PDM. Examples are CAE design archives, assembly drawings, gerber files, NC recipes and more. PDM's charter is to manage product data in its native, or intended-use form. To use data from PDM, one must have the appropriate application to interpret the data. With in the

PIC, data is kept in a neutral form with a documented relational schema and dictionary. Thus PIC data is free for use in a variety of current applications and future applications. Together, the PIC and PDM contain a bulk of the data required to define a product design.

3. A Tour of the PIC

The PIC is implemented using a commercial relational data base management systems (RDBMS) on HP9000 series 800 mainframe computers. Since the PIC is considered a site resource, it is usually centralized on one big machine. Multiple users access the PIC via a variety of mechanisms ranging from networked virtual terminals to RDBMS network software.

The entities or objects maintained by the PIC are held in a series of relational tables. The PIC handles four major entities:

- material lists
- printed circuit boards (PCBs)
- subpanels or SMT PCB carriers
- loaded subpanels

Figure 1 shows a simplified Entity-Relationship diagram for the PIC.

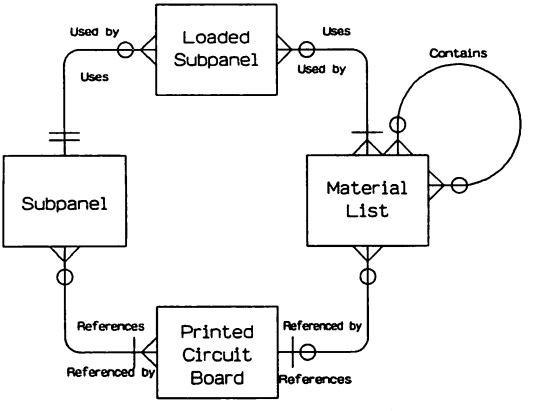


Figure 1. Entity-Relationship Model for the PIC

3.1 Material lists

Material lists (MLs) are stored in a series of 4 tables. The keys to these tables are the HP assembly number and revision. The tables handle:

- overall assembly information
- assembly revisions and production change dates
- hierarchical ML structures
- and temporary part numbers.

As illustrated in the major ML table, the pic_matl_list table (figure 2), each row represents a component in the ML. Hierarchical MLs are achieved if the Component_Number and Component_Rev number are used as an assembly number and revision. Thus a ML can contain the ML for a sub-assembly.

PCBs used to during assembly are simple components in the material list. The Component_Number and Component_Rev are used as the key to the PCB tables. All information regarding a purchased

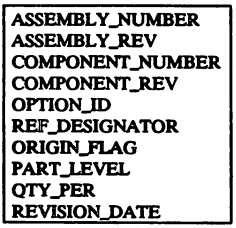


Figure 2. The PIC_MATL_LIST Table

component is available in the CIC and thus no description or parametric information for a component is stored in the PIC. Users viewing the ML table typically see views of the table with description fields out of the CIC.

The Ref_Designator attribute is used to match the ML component to the corresponding component location in the PCB tables.

The Origin_Flag attribute is used to identify the source of the component row entry in the ML. Sources of ML information can come from many places. The Origin_Flag is used to identify the source and apply precedence rules to prevent its update by sources of lower precedence. For example, a ML may come from an engineer's schematic design initially, but fine tuned by a materials engineer using the ML editor. If the ML is updated again from the schematic, the material engineer's editor entry takes precedence over the entry from the schematic.

3.2 Printed Circuit Boards

PCBs are stored in a series of 10 tables all keyed to the PCB number and artwork revision. These tables represent sub-entities within PCBs. They are:

- overall board information
- the board outline
- component locations on the board
- graphic and assembly features on the board
- component outlines
- component pin or lead locations
- shapes of pads at pin or lead locations
- the net-list
- test pads
- and via locations and net names

Although many other entities can be identified in a PCB, for example trace information, the above entities are sufficient to support almost all aspect of PCB assembly. Support of processes such as automated insertion/placement, hand-load documentation, board test, automated vision inspection and others were anticipated.

Figure 3 illustrates the major PCB table, Pic_PC_Placement. Each row represents a component location by reference designator. Reference Designators are commonly used in CAD systems to uniquely identify component locations in the PCB layout. Thus the Ref_Designator attribute is also a key to this table. Other attributes in this table identify the component's standard x-y location , rotation, which side the component is placed, and the keys of the footprint and outline used by the component. The PIC implements its own standards for component locations. Thus the interpretation of the component location is always the same regardless of which CAE system the PCB data originates.

BOARD_NUMBER
ARTWORK_REV
REF_DESIGNATOR
X_COORDINATE
Y_COORDINATE
ROTATION
FOOTPRINT_NAME
SIDE_PLACED
OUTLINE_NAME
REVISION_DATE

Figure 3. The PIC_PC_PLACEMENT Table

BOARD_NUMBER
ARTWORK_REV
OUTLINE_NAME
GEOM_SHAPE
SHAPE_GROUP
X_COORD1
Y_COORD1
X_COORD2
Y_COORD2
X_COORD3
Y_COORD3
REVISION_DATE
SEQUENCE_NO

Figure 4. The PIC_COMP_OUTLINE Table

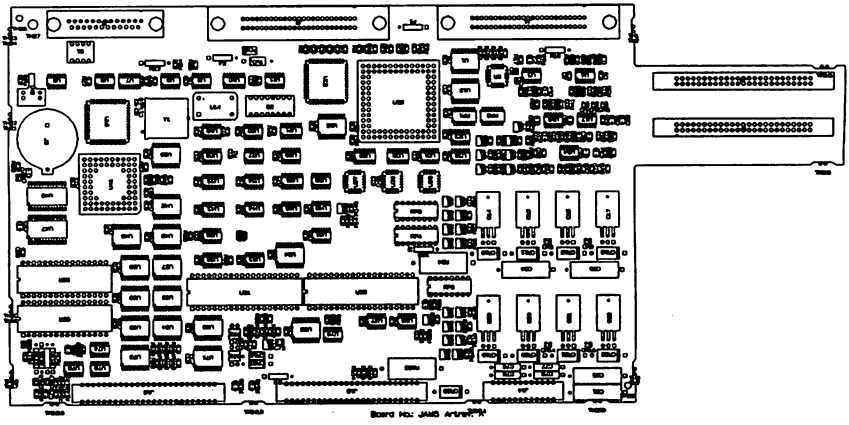


Figure 5. PCB Graphics from the PIC

Figure 4 illustrates the Pic_Comp_Outline table which is one of several PIC tables that stores graphics information. Most other PIC tables which store graphics use the same method. Each row represents a elementary graphics shape. The Geom_Shape attribute identifies the shape as either a line, arc, circle or rectangle. The three x-y pair attributes are interpreted according to the Geom_Shape attribute. For example an arc, is described as start point, mid-point and end-point. The Sequence_No attribute is used to maintain explicit ordering of shapes, such that enclosed areas can

be defined by a series of shapes. The Shape_Group attribute identifies the series of enclosed areas can be combined into a planar polygon with holes or "cut-outs" in them.

Graphics information is stored in table form in anticipation of generating human interpretable graphics independent of the CAD source. It will also be used to automate processes such as edge routers, vision inspection and design-for-manufacturability (DFM) tools. Figure 5 illustrates the detail to which graphics can be stored in the PIC.

3.3 Subpanels

Subpanels[1] are used within the HP surface mount technology assembly process. In general, they are analogous to a board carrier used in most assembly processes. The difference being that a PCB blanks are not fully cut out of the PCB panel used by the PCB fabricator. Instead they remain fixed in the panel while the panel is run through the assembly process. Only at the end of the assembly process are the now loaded boards cut out of the panel. Thus no inventory of custom board carriers is needed. The term subpanel comes from the fact that the fabricator's large full panel is typically split into two or more subpanels prior to assembly (look ahead to figure 9 for an illustration of subpanel).

Subpanels are implement in a series of three tables which represent:

- overall subpanel information
- the locations of PCB blanks in the subpanel
- and graphic and assembly features on the board (i.e. tooling holes, etc.)

The subpanel tables are keyed by the subpanel_number and revision. The table locating the PCB blanks is very similar to the Pic_PC_Placement table for the PCB. The Pic_Subpnl_feature table, in figure 6, shows how features are handled. As similar table is used to handle PCB features.

SUBPANEL_NUMBER
SUBPANEL_ART_REV
FEATURE_DESIGNATOR
FEATURE_TYPE
PLATED_FLAG
SIDE
BM_LINE_WIDTH
TAB_ROTATION
X_DIMENSION
Y_DIMENSION
X_COORDINATE
Y_COORDINATE
REVISION_DATE

Figure 6. The PIC_SUBPNL_FEATURE Table

Each row in the table represents a feature in a subpanel. Within the subpanel, each feature has a unique designator as defined by the Feature_Designator attribute. The Feature_Type attribute identifies the type of feature. This attribute can have a wide range of values from tooling holes to barcode labels. The location of the feature is defined by the x-y coordinates, x-y dimensions and side attributes. A few special features use the additional attributes such a Tab_Rotation.

3.4 Pulling it together: Loaded Subpanels

The loaded subpanel entity ties the ML, PCB and subpanel entities together. The job of the loaded subpanel is to identify which subpanel is used and which assemblies are to be built on specific PCB blanks within the subpanel. The keys used are the loaded subpanel number and the loaded subpanel revision. The loaded subpanel is implemented in two tables which specify:

- overall loaded subpanel information and subpanel used
- and the link between a specific subpanel blank and an assembly ML

LD_SUBPANEL_NUMBER
LD_SUBPANEL_REV
IMAGE_DESIGNATOR
ASSEMBLY_NUMBER
ASSEMBLY_REV

Figure 7. The PIC_LD_SUBP_ASSY Table

Figure 7 illustrates the link between the PCB blank and ML. The Image_Designator identifies the unique PCB blank image within the subpanel. The Assembly_Number and Assembly_Rev identify the ML to be used on that image or blank. This table allows multiple different assemblies to be built on one subpanel.

4. A Day in the Life of the PIC

The PIC is just an empty shell without the surrounding applications that enter data, manipulate it and use it. This section describes some of the dataflows into an out of the PIC. Figure 8 illustrates the main applications and the overall dataflows.

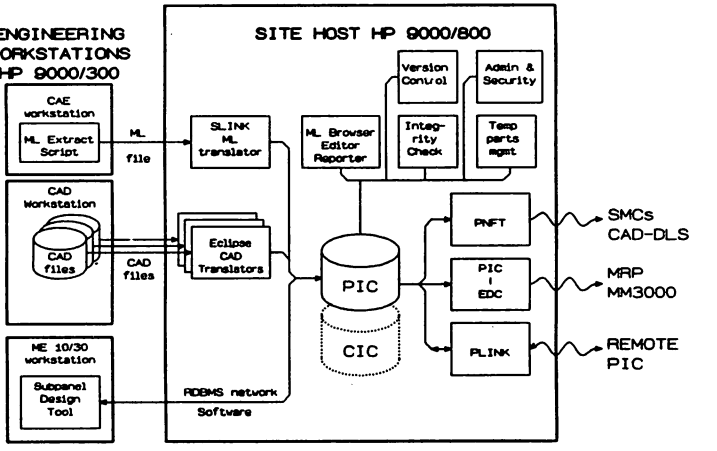


Figure 8. PIC Applications

4.1 Putting data in

Since the PIC is a neutral definition of product data, then the obvious sources of product data typically come from R&D systems. However in some cases, such as with the subpanel, specialized applications must be developed to enter the data.

4.1.1 Material lists from schematics

The bulk of material list information is extracted from either CAE schematic capture systems or in some cases from CAD PCB layout systems. This data is captured in material list files typically available from CAE systems and translated into the ML tables via the SLINK module.

4.1.2 Board data from PCB layout

PCB information is universally extracted from PCB CAD systems. A series of translators, each one tuned to a specific CAD system, is used to translate the PCB information directly into the PCB tables. These translators include much processing to covert all PCB information into the PIC

standards. Thus all PCB information is logically consistent regardless of the CAD systems source. This up front processing drastically simplifies the processing and error detection in down stream applications.

4.1.3 Laying out the subpanel

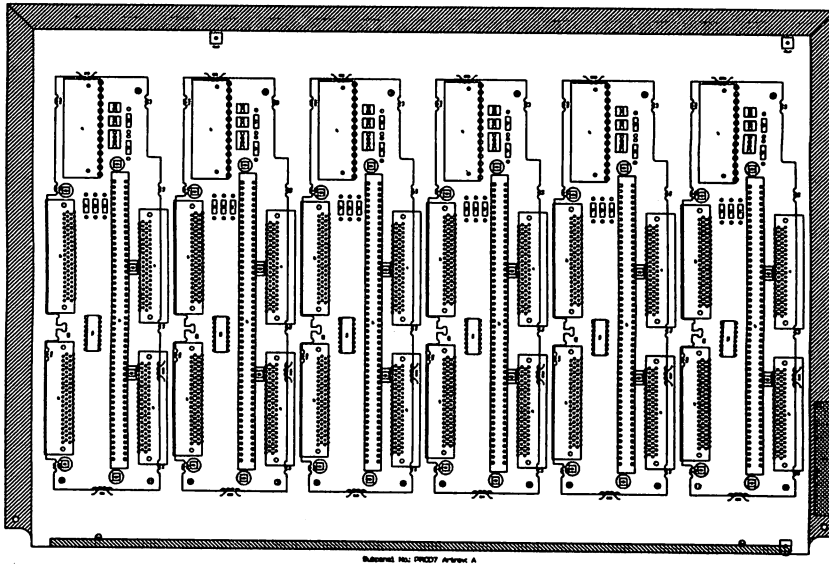


Figure 9. A Subpanel

Since the subpanel is a object somewhat specific to the HP SMT assembly process (see figure 9), a custom specialized tool was developed to provide subpanel information for the PIC. The Subpanel Design Tool (SPDT) was built leveraging the HP ME10/30 product adding custom macros and a design rule checker (DRC) module to allow product designers to layout the subpanel using interactive graphics. Since this application resides on a separate workstation from the PIC, the commercial RDBMS networking software is used to link the SPDT to the PIC. The SPDT reads in the PCB board blanks, allows the designer to step and repeat the blank images on the subpanel, and the writes the subpanel information back into the PIC.

4.1.4 Tying it together

The last step in data entry process is to define the loaded subpanel. A non-graphical application is provided for users to simply attach assemblies to images on a subpanel.

4.2 Using the data

Now that the PIC has been loaded with product data, it is ready for use.

4.2.1 Driving MRP systems

MRP systems within HP are almost universally using MM3000. A customized link was developed to supply MLs to the EDC module of MM3000. This link also handles the movement of the ML via the network to the HP3000 and running a batch job to load it into the MM3000 system. In many sites the PIC is used as the master for MLs during the new product introduction process. After

production release, the MRP system takes a more active role in the management of the ML.

4.2.2 Driving assembly operations

Surface mount assembly centers (SMCs) within HP currently use a system called CAD-DLS[2] (CAD Data Link System) to run their assembly machinery. CAD-DLS uses a set of "Neutral Files" as a common point of exchange. Thus to drive assembly, an application was developed to generate the "Neutral Files" from a PIC product entry. Since the PIC and CIC contain a superset of the required information, no further data entry is required to produce the neutral files to drive the SMC.

In the near future an additional information center and application set is planned to replace CAD-DLS. This "Manufacturing Information Center" (MIC) will couple tightly with the PIC and CIC and SMC equipment will be driven directly from PIC.

4.2.3 Controlling change

Once the data is within the PIC, a series of applications are used to control and manage the data. An administration and security module allows for the identification of users and assignment of various degrees of authority over the products and entities. It also controls data ownership and protection to prevent user collisions over the same data objects. Additional PIC tables are used to store and maintain the security data.

A version control module tracks an assembly through its life cycle by assigning and changing its status. An assembly passes through several states from a general "working" status which still under change by R&D or materials engineering to production "released" and frozen to the final "obsolete" status. Version control also handles production change orders, by creating new revisions of assemblies and identifying their production starting and ending dates.

4.2.4 Product transfers

In order to support the global product development process at HP, the many PICs will be distributed around the company. They will be linked by the HP world-wide network using the PLINK module. PLINK will unload, transfer and reload any PIC entity or series of entities from one PIC to another.

A typical scenario is that R&D sites will be responsible for the entry of data into the PIC and ensuring its integrity. Then via PLINK, the product data is transferred to the manufacturing facility which is at some other remote site for use in the assembly process. In another scenario, a manufacturing site may copy product data to a sister site for load balancing or multiple product sourcing purposes.

5. Future Directions

The increasing use of the PIC, CIC, PDM and the near future MIC have promised a future of enhanced productivity through the elimination of redundant data entry, reduced data errors, and information open for functional areas to use. Future work will focus on two areas.

First, the development of these information centers has brought to light the many holes and variations in HP product development processes that informal and manual processes have "hidden". With these automated systems now available, it is now the time for HP to adapt its product development processes to take advantage of these systems. No doubt in the future, these systems will have to evolve with the newer processes in an ever continuing improvement process.

Secondly, even though the information centers appear to be highly integrated, they were developed at different times and directed at different functional areas. Thus there is some data duplication between them and many information holes. Also looming on the horizon is the future of network computing and client-server architectures. HP plans to migrate the separate information centers to an integrated "reference data server". The surrounding applications will be partitioned into application clients invoking transactions on the reference data server. In this migration, the duplication will disappear and hopefully the holes filled.

REFERENCES

1. Walden, P. and Shain, J.: **Verifying PC Assembly Data using Computer Graphics**
Technical proceedings of the 1990 Conference at Boston.
Interex, 585 Maude Court, Sunnyvale, CA 94088-3439. Aug. 1990, paper 1059

2. Lujack, J. and Safai, F.: **CAD Data Link System,**
Proceedings of the Technical Program NEPCON East '89 at Boston,
Cahners Expo Group, 1350 E. Touhy Ave., Des Plaines, IL 60017, June 1989, p. 15.

Paper # 2027

Making Data Integration Easy

John A. Hall
Hewlett-Packard Company
PO Box 37000
Raleigh, NC 27627
(919) 467-6600

Objectives

This demonstration was developed with three objectives: to integrate and share data between the data acquisition devices and the analysis systems easily, to effectively demonstrate how applications can easily share common data areas, and to easily integrate different technologies and applications into existing environments.

The Dilemma

Data acquisition devices are often dedicated to specific tasks such as controlling, testing, and troubleshooting. These characteristics make integration into mainstream networking difficult, if not impossible. Many of the test sets in use today are 'homegrown' - that is, designed for the explicit purposes of testing and fault isolation, with little if any consideration given to the integration into a data collection environment.

Application integration can also be hampered by the use of proprietary interfaces and data structures. Through the use of such structures, application designers prevent the propagation of their development efforts and insure security within the application. However, by this action, integration of the application into an open cooperative computing environment can be quite difficult. Often the only commonality which applications can share is the ASCII data format or unstructured records, commonly known as flat files.

Application developers, especially in those specialized applications which require additional hardware (such as solids modeling applications), often will have special requirements for the operating system (such as kernel reconfiguration). These changes to the operating system can make integration of applications difficult because of the 'special' needs of the kernel or operating system.

Applications Involved

HP LAN Manager/X lets UNIX-based machines operate as file and resource servers to MS-DOS and OS/2 PC workstations. With Lan Manager/X, users gain access to a wealth of PC applications plus the power, resources and security of UNIX. HP LAN Manager/X supports industry standard network transports and links, and its open architecture supports multivendor environments. A wide variety of PC network links from HP or third parties that conform to the Network Driver Interface Specification (NDIS) are supported. In addition, Application Programming Interfaces (API) let software developers create distributed applications which take full advantage of networks and computing resources.

The SAS System offers 4GL capabilities including a programming language and pre-written, integrated procedures. Applications include data manipulation, information storage and retrieval, statistical analysis, and report writing. SAS/QC, the statistical quality control element of the SAS System, provides a variety of specialized tools for statistical quality control applications. Included are procedures for generating Schewhart, cumulative sum, and moving average control charts.

RS/1 is a fully integrated system for the analysis of technical data. RS/QCA II, the Quality Control Analysis module of RS/1, provides a full range of powerful, easy-to-use statistical quality control and manufacturing functions. This includes control charts, inspection sampling plans, cusum charts, process capability studies and trend analysis.

Labtech Notebook for the PC is a Menu Driven Data Acquisition software package for the laboratory environment. It is set up by completing various screens focusing on channel definition, measurement definition, measurement sequences, and data formatting. Once started, the system gives top priority to the I/O functions, second priority to the screen display and lowest priority to data storage. Labtech can store the data files in many different formats including numeric, ASCII 9Text (text) and LOTUS. Labtech software is supported on HP Vectra PC and 9000/S300 HP-UX systems.

X Window terminals have been designed to address a range of applications where users may require the full graphics capabilities of the X Window system on their desk, but do not require the additional functionality that is provided with a workstation. Systems configured with X Terminals provide the benefits of low cost per seat, and network access to multiple applications through the standard X Window System graphical user interface. In addition to low cost per seat advantages, many users find X Terminals appealing because of the ability to interactively and simultaneously display multiple windows, possibly across multiple applications shared across multiple hosts across the network.

The Solution

With each of the individual applications having its own proprietary data format, it was decided that the easiest way to integrate data would be through the use of columnar, unformatted ACSII files. Lab Notebook offers the ability to log data to a chosen output file. This file for our demonstration was a column oriented file which contained the 13 variable values for each observation. In addition, the use of an ASCII data format resolved all of the dependencies that our applications had on data formats, since all of the applications could readily accept columnar data.

Lan Manager/X provided the capability of physically locating the log data file from Lab Notebook on a HP-UX host system. In this case, the host system, labeled SERVER in Figure A, provided effortless access to the log data file for all other applications.

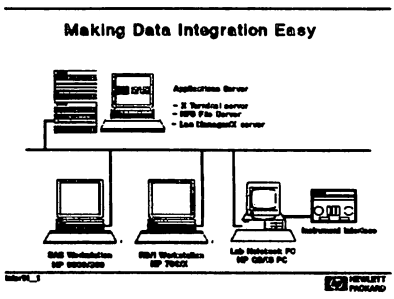


Figure A: Systems used in Demonstration

Through the use of the Network File System (NFS), the log data file could be accessed across the network transparently. This greatly simplified one of the most common problems of sharing data - that of data replication. By all applications which need access to the log data file accessing a single file, the risk associated with retrieving the most recent data is eliminated.

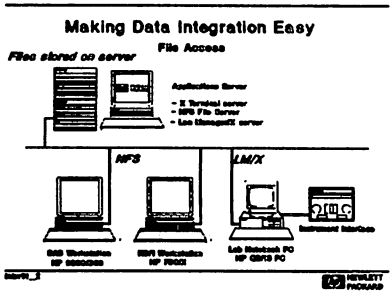


Figure B: Integration of Log Data File

Installation of the application software was straightforward, with no problems encountered. None of the applications software used for this demonstration involved operating system alterations or kernel reconfiguration, so the workstations were used without modifying the operating systems that are shipped with HP-UX. The only subsystems used were ARPA services and NFS for connectivity.

Integration of the data file into the SAS System was accomplished with a minimum of effort. The SAS System is shipped with an application called SAS/ASSIST, which is a menu-driven interface to the most commonly used features of SAS. Through this menu system, the variables were identified and the location of the log data file was specified (Figure C). The data having been identified, the data could be analyzed by using other features of the SAS/ASSIST application. A custom program was integrated into SAS/ASSIST for this demonstration, making analysis of the data quite simple.

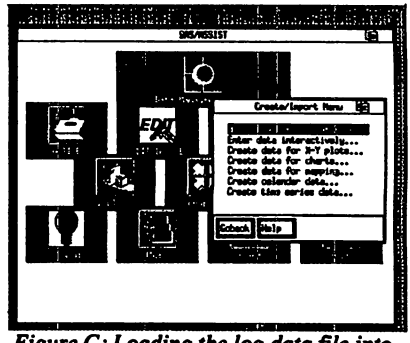


Figure C: Loading the log data file into SAS

RS/1 was equally simple to integrate. Again, identification of the data file location and variable naming produced data ready for analysis by RS/1 (figure D). A keyword-driven interface made the inclusion of data and subsequent analysis rather effortless. Here again, a custom

program was integrated into the RS/1 application, which allowed the demonstrator to illustrate the capabilities of RS/1

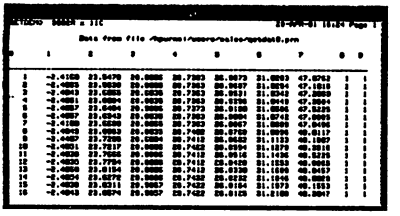


Figure D: Loading the log data file into RS/1

One peculiar problem which was encountered was the opening and closing of the log data file from Lab Notebook via LM/X. When opening a file through LM/X, the data was not available until the data acquisition activity had ended. Although this was not a significant factor for this demonstration, a real-time system could have significant problems.

The PC was configured with a special demonstration box which allowed user interaction to the data acquisition process. The box, provided by HP, contained a heater and fan which could be easily manipulated to vary the rotation speed, temperature, and environmental characteristics of the data acquisition process. Application interfacing to the Lab Notebook software was straightforward, with the documentation provided with the Lab Notebook application being quite sufficient to set up the demonstration.

Possible Alternatives

The Object Management Group (OMG) recently submitted a request for technology for a distributed Object Management Facility (OMF). The distributed OMF makes it easy for an object on one machine on the network to access and execute another object on another machine. Objects are system resources (usually data) associated with software that manipulates the data in well defined ways. For example, a spreadsheet object manipulates data in cell, row, column, or matrix operations. This encourages users and developers to concentrate their efforts on what needs to be done rather than the mechanics of manipulating the data itself. This concept allows complex tasks to be separated into clearly defined pieces.

The Andrew File System (AFS) < 8 Transarc Corporation provides a single view of a networked file system. The acceptance of AFS version 4.0 by the Open Software Foundation provides an opportunity for acceptance of AFS throughout the computer industry. Through the use of AFS, a user can address files with the same pathname from anywhere in the network, regardless of which computer they are using. AFS also provides high availability of all accessible data resources, using replicated filesets of file systems available for read access.

Conclusion

Data integration has always been the cornerstone of combining multiple applications to build a cooperative computing environment. Through the use of standards-based application enablers such as LAN Manager/X, the task of simplifying the integration of PCs and workstations can be accomplished today. With application developers increasingly using standards-based enablers into their applications, the data integration issues should decrease in the future.

Trademarks

Introduction

This is a test page for document processing. It contains multiple paragraphs of text to verify that the system can handle standard prose content.

The quick brown fox jumps over the lazy dog. This sentence is commonly used for testing purposes because it contains every letter of the alphabet.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.

Technical Specifications

Parameter	Value	Unit
Temperature	25.4	°C
Pressure	1013.25	hPa
Humidity	65	%
Wind Speed	12.3	km/h

Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum.

Subsection

- First item in the list
- Second item with more detail
- Third item

Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore veritatis et quasi architecto beatae vitae dicta sunt explicabo.

Paper Number 2028

HP Backup Strategy for HP-UX Systems

Reiner Lomb

Hewlett-Packard

Herrenbergerstraße, 7031 Böblingen, Germany

Phone: (49) (7031) 14-3869

HP Backup Strategy for HP-UX Systems

This document describes HP's backup solutions for HP-UX and Apollo Domain systems, and provides guidelines for choosing the appropriate tool. It also describes each solution's benefits and value, and compares HP's offering to the competition's.

Management Overview

Up-to-date information is one of the most important assets of a company. Therefore, availability of data, and protection against data loss with a backup and recovery strategy are critical. This is becoming increasingly important and challenging as more data is spread across heterogeneous networks.

On UNIX systems, backup and recovery solutions have historically been dominated by tools developed for systems with a small amount of on-line data and low requirements for high availability, and designed for technical, experienced users. These tools do not meet the critical needs of today's business environments.

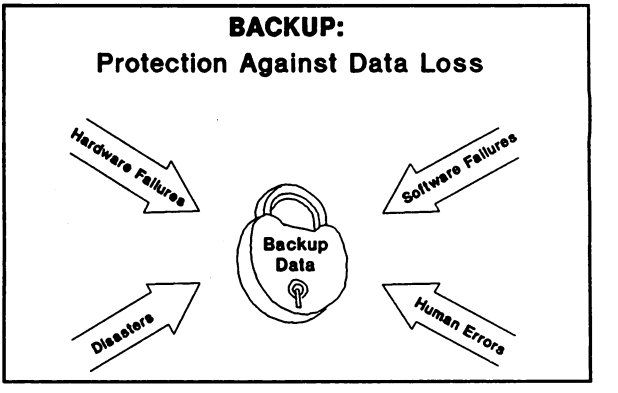
HP addresses today's needs with outstanding backup and recovery solutions with the following features:

- backup for servers, workstations, and PCs over a network
- high-performance backup
- unattended operation

Why Backup Is Necessary

Backup is necessary to protect company data. There are many reasons for data loss, most of which are included in the following categories:

- hardware failures, such as disk- or system-crashes
- software failures
- human errors, such as unintentionally deleting files or data
- disasters, such as fires and earthquakes



General Backup Needs

No matter which backup solution a company chooses, they generally want the following from their backup system:

- data protection and central backup of heterogeneous networks
- minimum system downtime for backup and recovery
- cost-effective backup

Data Protection and Central Backup of Heterogeneous Networks

Because data will be distributed in the future more and more across several systems, the backup system must guarantee the integrity of the data in a heterogeneous environment. A heterogeneous environment means that different hardware platforms (PCs, workstations, multiuser systems, and servers), different software environments (operating systems, databases, tools, and applications), and different network services (LANs, WANs, NCS, NFS, and TCP/IP) work together.

In addition, businesses require that these networks are also backed up together. This is called central backup. This means that data from different systems is backed up to one backup location.

Central backup improves the data protection of distributed systems because it offloads individuals, like personal-workstation users, from doing regular backups and allows control of a complete distributed-environment from one central point.

Minimum System Downtime for Backup and Recovery

System downtime for backup and recovery could be minimized by doing on-line backup and by increasing backup and recovery performance. Minimum system downtime is becoming increasingly important as more data is being stored.

Cost-Effective Backup

There are many solutions that provide data protection, but if it is not cost-effective, it is not a realistic solution. The cost-effectiveness of a backup solution can be divided into two major areas:

- human resources
- hardware

Human Resources

The most important cost associated with backup is the human resource cost. This kind of cost could be reduced by:

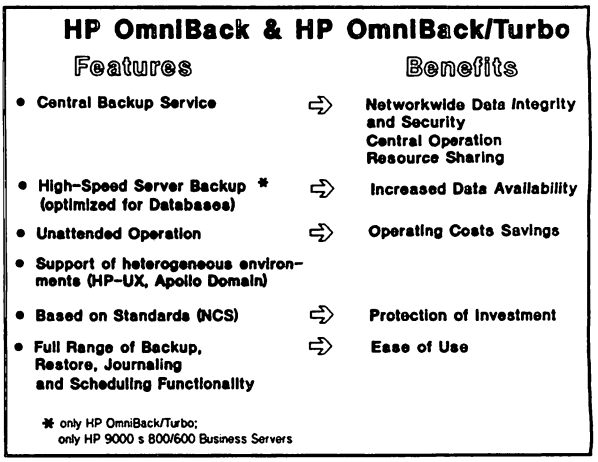
- Unattended backup and recovery.
- Increased usability of backup products. If the backup tool is very usable, end users could perform certain tasks, for example, recovering a deleted electronic-mail file, without requiring support from the operating staff.
- Consistent user-interface across platforms. The same user-interface for different computer systems (such as workstations and multiuser systems) reduces the training effort required to retrain the operating staff for each different computer.
- High-performance backup and recovery. Optimal backup speed reduces the operating time required to backup data.

Hardware

A backup solution must also be hardware cost-effective. Hardware costs include system downtime due to backup, backup devices like magnetic tape drives, and used media like magnetic tapes. These costs can be minimized by:

- Increasing the system availability by minimum downtime for backups.
- Sharing backup devices among several computer systems by backing up over a network.
- Using new types of peripherals such as digital audio tapes and rewritable optical disks. For example, a digital audio tape costs one-tenth of the classic 1/2-inch magnetic-tape.

The following section describes the HP-UX products that are available to fulfill these needs.



Central Backup. HP-UX and Apollo Domain systems can be centrally backed up over a local area network to one backup server. This guarantees data integrity and security in a corporate computing environment. With central backup, resources like magnetic tape drives can be shared among several systems.

High-Performance Backup and Recovery. The high-performance component dramatically decreases the system or application downtime during backup or recovery.

Unattended Operation. HP OmniBack and HP Omniback/Turbo support multiple devices, such as DAT/DDS drives and rewritable optical library systems, so data can be backed up without manual intervention. This reduces operating costs and protects against operator errors such as mounting or unmounting the wrong medium.

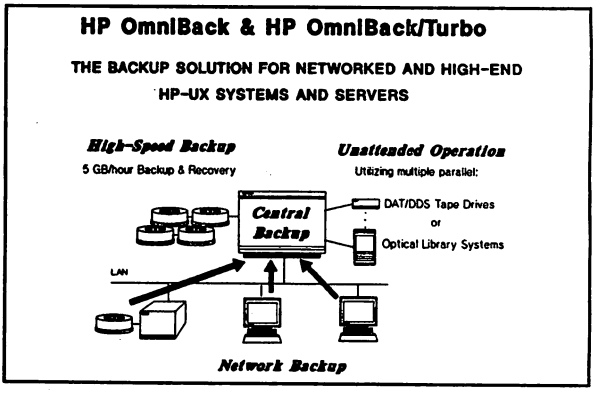
Based on Standards. HP OmniBack and HP Omniback/Turbo's network component is based on HP's Network Computing System (NCS). NCS, a de facto industry standard which has been accepted by OSF (Open Software Foundation) as part of the DCE (Distibuted Computing Environment), lets users tie together machines in heterogeneous networks to create a single pool of computing resources.

Full Range of Backup, Restore, Scheduling, and Journaling Functionality. This outstanding functionality enables effective backup and recovery, which reduces the operating and administrative efforts and protects against data loss caused by operator errors.

HP OmniBack and HP Omniback/Turbo

HP OmniBack and HP Omniback/Turbo are the leading backup solution from HP for HP-UX and Apollo Domain systems to address environments with the following characteristics:

- networked system
- large amount of on-line data
- high system-availability



HP OmniBack was first developed on the Apollo Domain platform and is available now as a new product with HP-UX 7.0 and 8.0 on all HP 9000 systems. HP Omniback/Turbo, an enhanced version of HP OmniBack, is available on the HP 9000 Series 600 and 800 business servers. The major enhancement is a high-performance component.

High-Performance Component

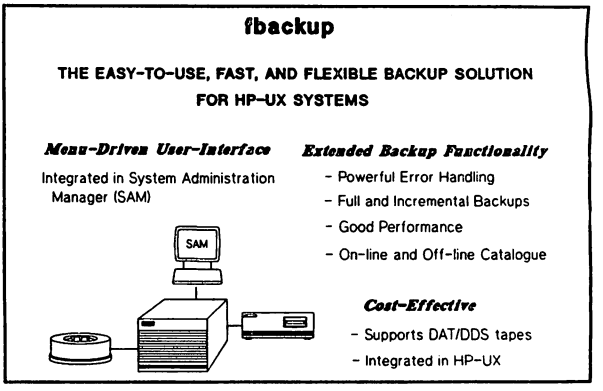
The high-performance component has a speed of more than 5 Gbytes/hour for backup and recovery. This performance is achieved for local system backup (all backup peripherals connected to the same system) and for backup of raw devices (raw disk backups of complete logical or physical disks). The backup performance also depends on the physical limitations of peripherals such as the backup devices or disk drives.

The high-performance component is for:

- database backup and recovery
- application backup, with the whole application residing on one or more volume sets
- full system backup
- disaster recovery

fbackup Fbackup is a flexible file-system backup and recovery tool developed by HP to address environments with the following characteristics:

- small- to medium-sized system
- small amount of on-line storage (less than 1 Gbyte)
- no requirements for network backup or central control of the backup and recovery process



With HP-UX 8.0, fbackup can be accessed from HP-UX's System Administration Manager (SAM), a menu-driven, operator user-interface. This means that minimum training is required to use fbackup, even for inexperienced users. This is especially important for smaller companies that cannot afford a trained operating staff.

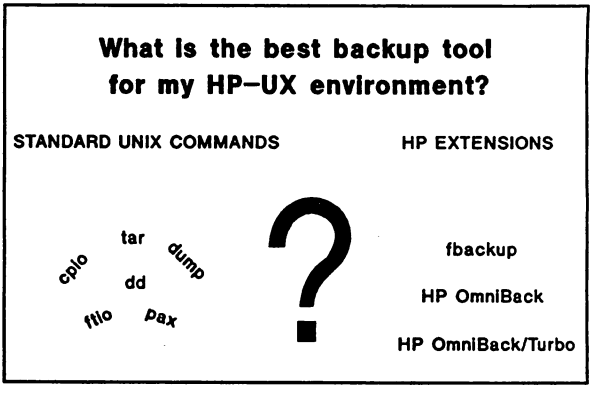
For more technical information on fbackup, see the HP-UX manual pages.

HP-UX Backup Tools

The UNIX environment developed from an academic background in which most users were technical, experienced users like engineers. Therefore, most of the backup and recovery tools, such as cpio, tar, dd, and dump, have been developed for those users and their requirements. (For more information on the standard UNIX commands, see appendix B.)

As UNIX is used more in commercial environments, end-user needs are becoming increasingly important. To address these needs, HP developed three additional tools: fbackup, HP OmniBack, and HP Omniback/Turbo.

Given all the tools and products available on HP-UX, the question in the following figure is frequently asked.



Typical User Environments

The following is a discussion of four typical user-environments. Most customer backup needs are reflected in one or more of the segments. The following figures present the four environments and the backup requirements.

TYPICAL USER ENVIRONMENTS FOR BACKUP

Low–End Stand–Alone	Regional Distributed Systems
- primary storage: < 1 GB - 2 to 15 active users - no dedicated operators - typical systems: entry level multi-user systems or PCs	- branch offices - primary storage: < 2 GB - 5 to 50 active users - no local system management - connected to "central system" - typical systems: entry level/ midrange multi-user systems
Local Distributed Systems	**Mainframe–Class Systems**
- primary storage: > 5 GB - distributed data - large databases - integrated into LAN network - typical systems: PCs, workstations, servers, and multi-user systems	- primary storage: > 10 GB - more than 200 active users - large databases - data center operating - typical systems: high-end mainframes

USER REQUIREMENTS FOR BACKUP

Low–End Stand–Alone	Regional Distributed Systems
Easy-to-use Unattended backup/recovery Cost-effective solution	Easy-to-use Unattended backup/recovery Central control of remote systems
Local Distributed Systems	**Mainframe–Class Systems**
Central backup service for all systems on network (servers, PCs, workstations) Unattended backup Data integrity/security in a heterogeneous environment High-performance backup for server	High-performance backup/recovery Unattended backup Support of all database products On-line backup Fast disaster-recovery

Common to all:

Data Integrity	Incremental/Partial Backup	Selective Recovery	Consistent User-Interface
Data Security	Automatic Scheduling	Media Management	Standard Compliance

The following sections describe each environment, define the backup requirements, and then provide a solution.

Low-End Stand-Alone

Description

A low-end stand-alone environment has no more than 16 parallel active users who manage less than 1 Gbyte of on-line storage. There is usually not a dedicated operating staff.

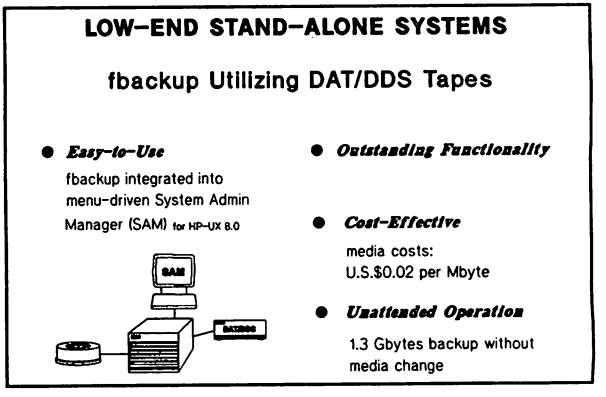
Requirements

Backup cost is the key here. Backup costs could be reduced by using relatively inexpensive hardware and backup media, and by minimizing the human resource cost (daily operating and training) associated with backup and recovery.

Solution

Fbackup is designed to fulfill all the requirements of a low-end stand-alone environment. It provides the full range of backup and recovery functionality and is easy-to-use, especially because it is integrated into the System Administration Manager (SAM) with HP-UX 8.0. Fbackup is part of the HP-UX command set, which means that it is free of charge.

Backing up with DAT/DDS tapes allows unattended backup. A user can start the backup in the evening (through the menu-driven SAM interface) and let the system back up a maximum of 1.3 Gbytes of on-line storage.



Regional Distributed

Description

A regional distributed environment has branch offices that are connected to a central system by a wide area network.

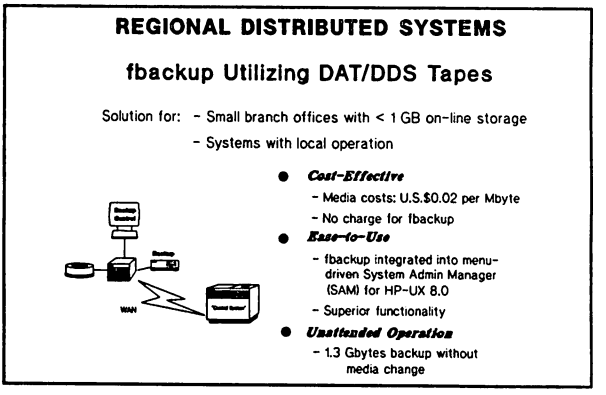
Requirements

The main requirement is the ability to centrally control the remote sites. This means centrally backing up the remote sites to the central system, or at least administering the remote backup process from a central location.

Because of the network communication cost and performance of today's wide area network, backup to a central site is a very costly solution. Therefore, backup control at the central site, but done locally, is the preferred approach for today's regional distributed environment.

Solution

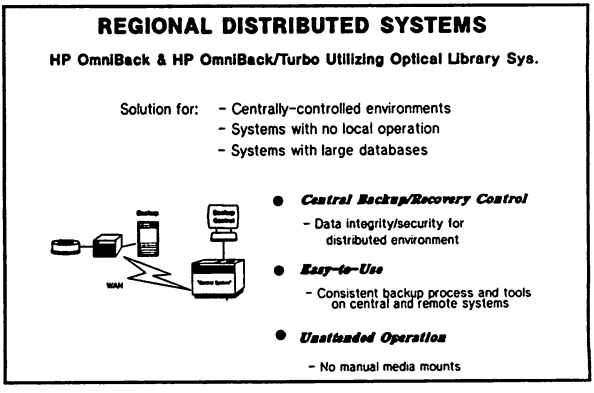
Similar to the low-end stand-alone environment, fbackup could be a perfect solution for a regional distributed environment if minimal backup operations (tape mounting and unmounting, labeling, and archiving) could be done at the remote sites. If the data volume is less than 1.3 Gbytes, a full backup fits on one DAT/DDS cassette. This means that the backup could run unattended during off-peak office hours.



HP OmniBack or HP Omniback/Turbo could be a solution for a regional distributed environment if the customer requires one of the following:

- high performance for database backups

- support of rewritable, optical-disk, library systems (allowing full, unattended backup and recovery at the remote site)
- the same tools used on the remote and central sites



Local Distributed

Description

A local distributed environment is the typical EDP environment of the '90s for large companies. Several heterogeneous systems are connected at the same site and interconnected with fast datacom lines (a LAN). The larger systems usually have 2 to 10 Gbytes of on-line storage. The applications and data might be distributed over the network. An operating and administrative staff is usually necessary to control and maintain the environment.

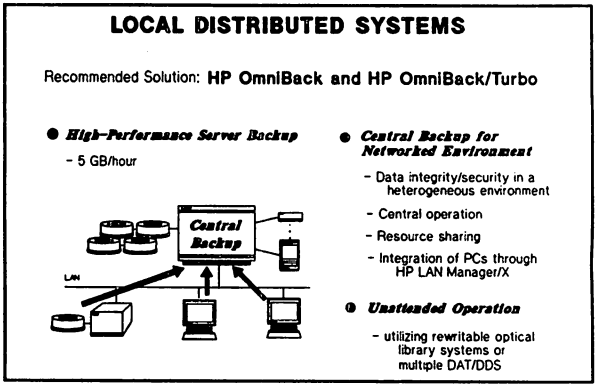
Requirements

The requirements are central backup for all systems on the local area network, unattended backup, and minimum downtime for server-type systems.

Solution

HP OmniBack and HP Omniback/Turbo are perfect solutions for local distributed environments because they offer network backup, unattended backup, and high performance.

To decrease downtime, HP OmniBack/Turbo's high-performance feature (over 5 Gbytes/hour) could be used. The operating costs can be reduced by using optical library systems for unattended backup and recovery, and by sharing backup devices among all networked systems through central backup.



PC Integration

An essential part of controlling a local distributed environment is protecting data on personal computers by integrating them into a centrally-controlled backup process.

Data on a PC can be classified in two categories:

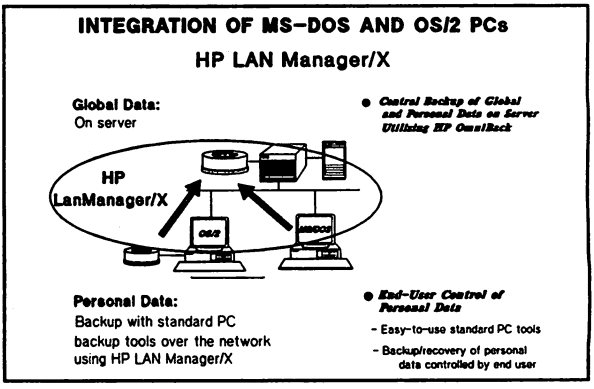
- global data, which is data critical for the company
- personal data, which is data only for individual value

Protecting Global Data. Global data should be located on a server system. This server could then be controlled by the EDP department to ensure integrity and security by using backup and recovery tools such as HP OmniBack or HP Omniback/Turbo. Also, the data could be shared among many PC users. Access to global data could be achieved by using HP LAN Manager/X through a virtual disk from the MS-DOS® or OS/2 system.

If global data must be kept locally on a PC, it should be backed up to the server system. An example of global data that must be kept on the local disk for performance reasons are large Lotus 1-2-3 files. Using HP LAN Manager/X, almost any standard PC backup tool can be used to back up the local data from the PC to a virtual disk on the server system. The data on the virtual disk should then be integrated into the server backup solution.

Protecting Personal Data. Personal data kept on PCs can be backed up in two ways. The recommended way is to back up the data to the server systems using a standard PC backup tool. Because MS-DOS is a single-task system, the backup must first be initiated by the end user. The tool copies the data to a virtual MS-DOS disk (physically located on the server) through HP LAN Manager/X. Then, the PC backup data on the server is saved with server backup tools like HP

OmniBack or HP Omniback/Turbo. The other alternative is to use local backup devices like floppy disks or HP EasyTapes.



Choosing a PC Backup Tool. See a PC software catalog to select the appropriate PC backup tool. Criteria for the selection should be:

- ability to be used with virtual disks (HP LAN Manager/X)
- good user-interface, especially for recovery
- ability to write a "batch" backup procedure that could be integrated into PAM or another application
- backup performance

Popular tools on the PC market today are Fastback Plus from Fifth Generation Systems, Inc.; Intelligent Backup from Sterling Software; and PC Tools Deluxe from Central Point Software.

Mainframe Class

Description

A mainframe-class environment is characterized by up to 24-hour system availability, large on-line storage (10 to 50 Gbytes), and 100% data protection. Typical applications are based on databases, with on-line transaction processing.

Requirements

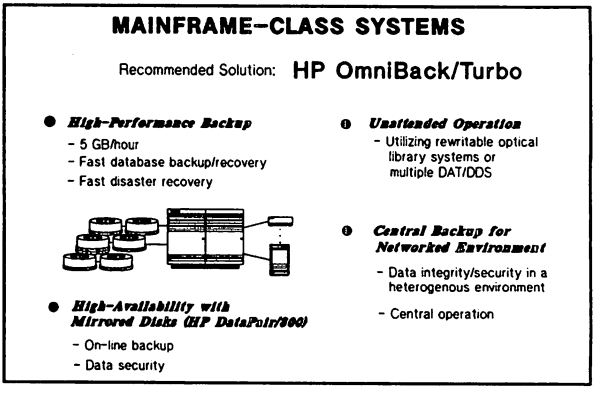
The availability of data and of the system is essential for a mainframe-class environment. The backup process should have minimal impact on system availability. In addition, unattended backup is required to minimize operation costs.

Solution

System availability could be increased by using HP OmniBack/Turbo. This tool provides outstanding backup and recovery performance (more than 5 Gbytes/hour) for database backup and recovery or disaster recovery (full system or full disk recovery).

In addition, HP OmniBack/Turbo offloads the operating activities by offering unattended operation. By using multiple parallel backup devices (up to four in the first release), automatic operation of up to 5.2 Gbytes if DAT/DDS tapes are used, or more than 40 Gbytes if rewritable optical library systems are used, can be achieved.

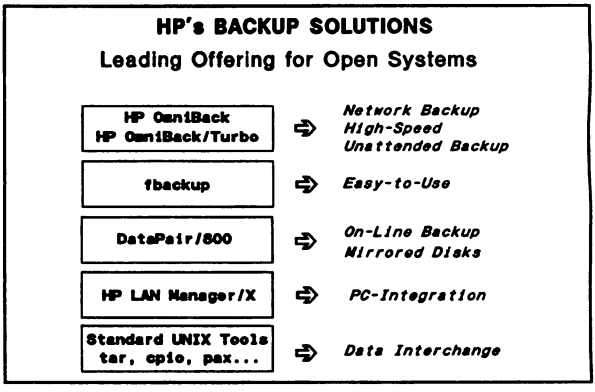
If 24-hour system-availability is required, companies should use HP DataPair/800, a mirrored-disk product, in addition to HP OmniBack/Turbo. HP DataPair/800 allows on-line backup, and there is virtually no downtime from disk failures.



HP's Backup Solution Summary

HP is the only vendor in the UNIX environment that provides complete backup solutions for all kinds of user environments. HP OmniBack, HP Omniback/Turbo and fbackup offer outstanding functionality. Products such as HP DataPair/800 for high system-availability, and HP LAN Manager/X for PC integration, complement the offering.

HP also offers the standard UNIX commands. UNIX commands should be used if data interchange between different UNIX systems or other systems is required. The pax command is designed especially for this (POSIX 1003.2 compliant).



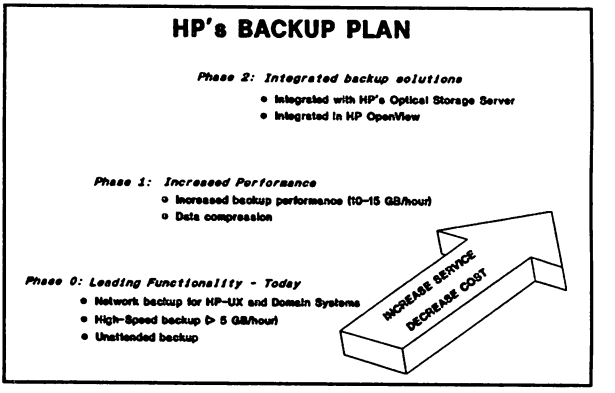
HP's Backup Plan

HP sees backup as part of a complete storage-management solution. Today, HP offers HP Omniback and HP Omniback/Turbo, outstanding backup solutions combining network capabilities, high performance, and unattended backup.

In the next phase, HP plans to increase backup performance by a factor of between two and three, and to provide data compression.

In the following phase, HP plans to combine HP's Optical Storage Server and HP Omniback/Turbo to a comprehensive Storage Management solution.

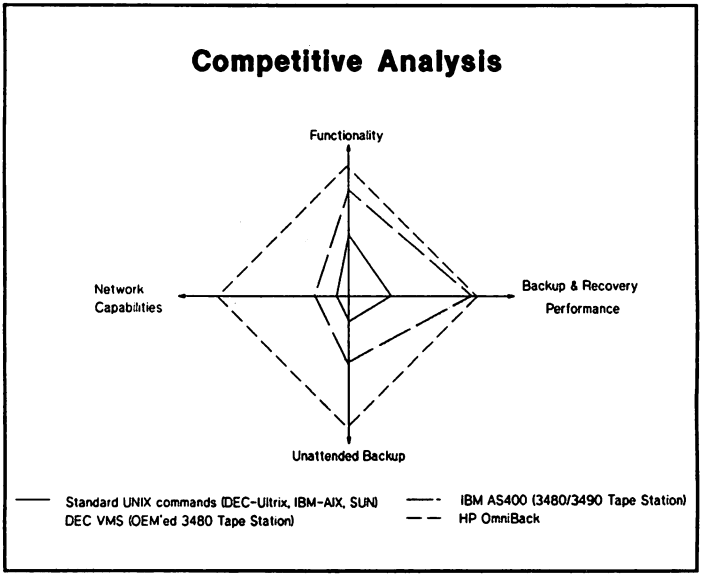
An integration of HP Omniback and HP Omniback/Turbo into HP OpenView is a further important step for HP, to provide customers with a comprehensive System Management solution.



Competitive Analysis

Competitive analysis shows that HP has the leading offering in backup solutions for open systems as well as proprietary systems. Compared with what is offered on other UNIX platforms, such as Ultrix from DEC, AIX from IBM, or SUN OS from SUN, they are far behind in offering similar functionality and commercial usability.

If the complete HP OmniBack product is compared with the solutions offered from IBM or DEC on their proprietary operating systems (OS 400 or VMS), HP Omniback still offers significantly more benefits, such as unattended backup and network capabilities. Also, the performance on IBM or DEC systems can only be achieved by using the extremely expensive IBM 3480/90 tape systems (which are OEM'ed from DEC as well).



Standard UNIX Commands

The following is a brief description of the standard UNIX commands and tools that are available for basic backup and recovery tasks. The description of the commands and tools is based on HP-UX release 7.0 (with the exception of pax, which will be released on 8.0).

There are many options available for the following commands and tools; however, in this appendix, only the most important options are covered. For more information, see the manual pages of the appropriate command.

cpio

Cpio is a tool for copying files in different directions, and is a popular method for data exchange between different machines. There are three modes for cpio:

cpio -o [options]	Reads the standard input for a list of path names and copies those files to the standard output, with path name and status information.
cpio -i [options]	Reads the standard input, which is assumed to be the product of a previous cpio -o and recreates the directory tree indicated by the input.
cpio -p [options] directory	Reads the standard input for a list of path names and copies those files into the specified directory. (This has no direct use for backup purposes, but is good for copying entire directory structures.)

The first two modes can be used for backup by doing an output redirection to a tape drive with the cpio -o command and an input redirection to a tape drive with the cpio -i command.

There is no table of contents on a tape written with cpio. Searching for a particular file means scanning the whole tape (or tape set) for that file. The files are restored in the current directory tree. This means that files can be restored to a directory other than the original.

A very simple method of performing backup with cpio is to create a list of the files to be backed up and to pipe that list to a cpio -o command. For example,

```
ls | cpio -o >/dev/rmt/0m
```

copies the content of the current directory onto a tape with a density of 1600 BPI. With this command, only the files in the directory are copied to the tape; no files in underlying directories are matched.

A popular method of backup with cpio is using the find command. For example:

```
find /users | cpio -o >/dev/rmt/0m
```

The find command produces a list of all files under the /users directory. This list is taken as input for the cpio command which writes on a 1600-BPI tape.

The disadvantages of cpio are that there is no built-in logging facility and no explicit multimedia support, and there is no provision for backing up active files. Also, using cpio to write directly to a cartridge tape unit can severely damage the tape drive and is therefore strongly discouraged. For a more sophisticated backup solution with cpio, see the command, backup, described later in this appendix.

tcio

Tcio is a filter for optimizing the data transfer between cartridge tape units and the host processor. There are three modes for tcio:

tcio -o [options] cartdev	Reads the standard input and writes the data to the specified cartridge device.
tcio -i [options] cartdev	Reads the specified cartridge device and writes the data to standard output.
tcio -u [options] cartdev	Performs utility functions on the cartridge tape, such as unload, mark, or verify.

Tcio is typically used when data to be transferred from the computer to the cartridge tape is handled by a particular tool or application and then piped through tcio to get the data on the cartridge tape and to keep the tape streaming. An example is a backup with cpio on cartridge tape:

```
ls | cpio -o | tcio -o /dev/rct/c0d1
```

Here the output of the cpio command is passed through a pipe to tcio to write the data to the specified cartridge tape.

tar

Tar (tape archiver) saves and restores archives of files on a magnetic tape, a flexible disk, or a regular file. Tar is a very popular method for data exchange between different machines. That is, the software the customer gets with an operating system update is a tar tape.

The syntax is

```
tar [options] key file
```

The following key arguments control what tar actually does:

c - Creates a new archive. Writes from the beginning of the archive.

r - Adds files to the end of the archive.

u - Adds files to the archive if they are new or modified.

x - Extracts files from the archive.

t - Lists the name of all the files on the archive.

If the file specified in the tar command is a directory, the files and subdirectories (recursively-defined) are chosen. For example,

```
tar c /users
```

creates a new archive on the default device /dev/rmt/0m with a complete copy of all files and directories of /users. When the end of tape is reached, tar prompts the user for a new special file and continues.

Recovering files relative to another directory is possible only if the archive was made with relative path names. If absolute path names have been specified, the recovery of the files always takes place in the original directories. Directories are implicitly created.

There is no built-in logging facility and no explicit multimedia support. There is no provision for backing up active files. Also, there is no table of contents on a tape written with tar. Therefore, searching for a particular file means scanning the whole tape (or tape set) for that file.

ftio

Ftio is a tool designed specifically for copying files to 9-track, magnetic tape-drives. It is faster than cpio and tar because it uses multiple processes for reading and writing, with large amounts of shared memory between the processes. It also uses large block size for reading and writing to the tape.

Ftio is compatible with cpio in that output from cpio is always readable by ftio, and output from ftio can be done so that it is readable by cpio.

Similar to cpio, there are three modes for ftio:

ftio -o [options] tapedev pathnames	Recursively descends path names, looking for files and copies those files onto tapedev.
ftio -i [options] tapedev	Copies files from tapedev, which is assumed to be the product of a previous ftio -o operation.
ftio -g [options] tapedev	Reads the file list on tapedev.

In addition to copying the files onto the tape set, ftio generates a tape header containing the current tape number, machine node name and type, operating system name, release and version numbers, user name of the backup initiator, and time and date of the backup.

The following command

```
ftio -o /dev/rmt/0h /
```

copies the entire contents of the file system / onto a tape with a density of 6250 BPI.

The command

```
ftio -id /dev/rmt/0h
```

restores all files from the 6250 BPI tape and creates directories as needed.

dd

Dd copies the specified file to the specified output with possible conversions. Input and output block size can be specified to take advantage of raw I/O. All the arguments are specified as option=value pairs. If no input or output files are specified, the standard input and standard output are used.

The dd command can do EBCDIC to ASCII conversion, character mapping, byte swapping, and reblocking. For example, if you have a tape that originated from a foreign machine (for example, IBM), you can perform a conversion with dd to read that tape on your machine.

The use of dd for backup purposes is very limited. It is usually used for an image backup of smaller disks (< 150 Mbytes, without regard to end of tape).

backup

Backup uses find and cpio to make a cpio archive of all files that have been modified since a certain modification time on the default tape drive. Backup is a simple shell script that sets up a tiny backup application.

Several local values are used that can be customized:

backupdirs	Specifies which directories to recursively back up (usually /, which means all directories).
backuplog	A file name where start and finish times, block counts, and error messages are logged.
archive	A file name whose date is the last archive date.
outdev	Specifies the output device for the backed-up files.
fscklog	A file name where start and finish times and output is logged.

In all cases, the output from backup is a normal cpio archive file, which can be recovered by a cpio -i command.

dump and restore

dump

Dump copies all files in the specified file system that have been changed after a certain date to magnetic tape. The syntax is

```
dump [ key [ argument ... ] filesystem ]
```

The key specifies the date and other options about the dump. Some keys require additional input which is specified in the arguments.

You can also specify the dump level (0-9). Dump level 0 causes the entire file system to be dumped. All other dump levels cause an incremental dump of all files modified since the last data stored at lesser levels. With the dump level, a full or partial backup scheme is introduced. Dump uses the file /etc/checklist for the file systems and frequencies, and /etc/dumpdates for the latest dump dates.

Dump periodically reports information to the operator, including the required number of tapes, the time needed for completion, and the time remaining until tape change. No file index or list of files is made available by dump. Only one file system can be backed up per execution.

Dump requires operator intervention for the following conditions: end-of-tape, end-of-dump, tape-write error, tape-open error, and disk-read error.

The following is a simple example:

```
dump 0f /dev/rmt/0h /users
```

This command causes the entire file system /users to be dumped on /dev/rmt/0h. For restoring tapes produced with dump, you must use restore.

There is also a command called rdump which dumps a file system on a remote machine. Rdump creates a server /etc/rmt on the remote machine to access the tape drive.

restore

Restore reads tapes dumped with the dump or rdump command. The syntax is

```
restore key [ name ... ]
```

Usually the appearance of a directory name refers to the files and (recursively-defined) subdirectories of that directory.

There is also a command called rrestore which restores data from a remote machine. Rrestore creates a server /etc/rmt on the remote machine to access the tape device.

pax

Pax reads and writes archive files that conform to the Archive/Interchange File Format specified in IEEE Std. 1003.1-1988, that is, the cpio and tar (ustar) format. Pax is also defined by POSIX.2. The default archive format is tar (ustar); the cpio format must be chosen by an option. When reading an archive, the input format is automatically determined. Pax also supports traditional cpio and System V tar interfaces.

There are four modes for pax:

pax -w [options] [pathname]	Write the files and directories specified by the path name to the standard output, with the path name and status information prescribed in the archive format used. The path name refers to the files and recursively-defined subdirectories of that directory. If the path name is not specified, the standard input is read to get a list of path names to archive.
pax -r [options]	Reads an archive file from standard input and copies the files relative to the current directory.
pax -rw [options] directory	Reads the standard input for a list of path names and copies those files into the specified directory. (This has no direct use for backup purposes, but is good for copying the entire directory.)
pax [options]	Reads an archive file from standard input and lists the contents of this archive.

There are options available for specifying an input or output archive other than standard input or standard output.

There is no table of contents on a tape written with pax. Searching for a particular file means scanning the whole tape (or tape set) for that file. The files are restored in the current directory tree. This means that files can be restored to a directory other than the original.

The following command

```
pax -w . >/dev/rmt0
```

copies the contents of the current directory to tape drive 0.

The command

```
pax -r <pax.out
```

reads the archive pax.out and extracts all files relative to the current directory.

There is no built-in logging facility and no explicit multimedia support. There is no provision for backing up active files.

The major advantage of pax is that it supports the cpio as well as the tar format.

TITLE: Introducing Open Software Environment

(Replaces "Open Systems Customer Projects)

(Author: Wolfram Fischer)

AUTHOR: Raj Bharqaua

Hewlett-Packard

c/o Ella Washington

(408) 447-1053

FINAL PAPER NOT AVAILABLE, HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO. 2029

Paper 2030

Providing Cost-Effective X Windows Environments

Mark R. Teter
Hewlett-Packard Company
3404 East Harmony Road
Fort Collins, Colorado 80524
(303) 229-6207

Abstract.

The large installed base of PC hardware and the advent of the X terminal allow inexpensive solutions for providing X Window client/server technology. With the benefits of X Windows, end users enjoy a common operating environment with multi-window access across three HP platforms: MPE/MPE-XL, HP-UX, and DOS. Existing workstations can become central compute servers for X terminals while PCs can double as DOS-based X servers. Both provide distributed applications and network database access. Through the superior LAN connectivity, X terminals and PCs serve as powerful front-ends for today's business and technical applications.

Introduction.

With the development of the LAN in the computing systems environment, the need and general acceptance of the X Window System standard is gaining popularity. The X Window System[1] has evolved from an emerging technology into a *de facto* standard for windowing environments by virtually every UNIX manufacturer. The benefit lies in executing clients locally or remotely over the network where information and resources are shared and accessed through a consistent user interface.

This paper presents an understanding of implementing X Windows technology apart from using traditional standalone UNIX workstations. It focuses on the technological issues providing such environments rather than specific available products. It is intended for system integrators familiar with both PC and UNIX platforms in addition to the OSI networking model. The intention is to convey the central concepts and issues involved with integrating cost-effective X Window systems into the business network.

X Windows Computing Environment.

The X Window System provides the basis for a multilayered, decentralized environment. X defines a protocol for asynchronous, bi-directional transmissions

1. *Version 11.4*

over LAN providing applications fast access to a distributed network environment. It provides a mechanism to leverage the computing power of PCs and workstations, which often sit idle, lowering business computing costs through the use of shared software applications, improved efficiency with peripherals, and greater information access.[2]

Applications can execute separately on local hosts or across the network on remote CPUs displaying their output on one monitor or by multiple monitors. In effect, powerful CPUs become remote computing resources across the network serving as database engines, statistical/scientific computational servers, print spoolers, mass data backup machines, mail hubs, and office automation (OA) tool servers. X can provide a complete business-wide LAN server infrastructure.

The X Window system is essentially a standard way of providing network-based graphical applications. The major GUIs in the X Windows arena include OSF/Motif and Open Look. There are many other variants across the industry all developed for the same general purpose. As an industry standard, X is constantly being improved with enhancements and extensions. XIE (X Image Extension) is being proposed as an X protocol extension which deals with image compression, encoding, and manipulation for efficient processing of image data. VEX (Video Extension to X) is an extension handling enhancements for graphical finite element analysis and simulations. PEX is a 3D graphics extension to X upon the 2D functionality. It support PHIGS (Programmer's Hierarchical Interactive Graphics Standard) graphic functions as well as the current 2D functionality. At the time of this writing, industry analyst speculate that the X Consortium will deliver X11R5 by midsummer 1991. [Hart] This release, among other things, will standardize the drag-and-drop mechanism, as implemented in products like HP VUE, as well as provide support for scalable fonts.

X was designed to reduce the amount of display information which has to be sent through the network. To do this, X is based on the client/server architecture. The client, or application, refers to the software program receiving input while the server assumes the display management responsibilities. A client sends drawing or graphical requests to the X server which it needs to have displayed. The server then executes the requests freeing the client from processing graphics and allowing it to concentrate on just running the application. The server controls access to displays, transmission and reception of network communications, two-dimensional drawing, and tracking resources for all X clients. X has outstanding service of graphical needs for clients, such as icons, scroll bars, fonts, cursors, and windows, allowing graphics to be rendered as fast as the X server can generate them. X minimizes network traffic by having the server maintain these graphical resources. Graphical resources are shared among the

---

2. *In fact, the CPU load on HP3000s can actually be reduced by decreasing the amount of logon occurrences by using the X Windows environment. By allowing the existence of concurrent multiple sessions on a single display, the logon process is effectually eliminated.*

clients through communications between the server with network packets. The network packets contain allocated integer ID numbers for each client along with their appropriate graphic messages.

X Servers.

The client/server model allows network workstations to exist which have no application computing power, but just perform graphics or image processing. These systems, known as X display stations or X servers, act as parasites creating displays for remote applications.[3] They access remote computing power across the network providing a complete computing environment configuration which can be adjusted to meet the business needs.

X servers may have less workstation processing power, but they have full use of LAN connectivity. They can provide load balancing for computing networks by utilizing excess MIPS which might exist across the LAN. If network processing becomes overloaded with graphics- or CPU-intensive programs, the X environment configuration can be enhanced. The processing power on hosts and servers can be upgraded expanding the application power or additional hosts can be added to the network sharing more of the system load. As a result, more processing power, and the cost associated with the processing power, is distributed to all X nodes across the network.

X servers range from remote workstations to enhanced PCs to special purpose computers or X terminals. Many combinations of hardware and software are available to implement the X Windows protocol. To a large degree the physical environment dictates how an X Windows system is to be implemented. However, there is a broad range of price/performance trade-offs. Two current X server alternatives comprise of two different types of technology: (1) PCs and (2) X terminals. These are reliable solutions for providing additional X nodes on the business network reducing the cost per X Window System seat.

The difference between X terminals and PCs is becoming more and more obscure. At present, X terminals can execute DOS applications from their own internal hard drives while PCs can emulate X terminals under DOS. The conventional X terminal, however, essentially consists of a graphics processor for executing X11 instructions, a communication coprocessor for keyboard entry, mouse tracking and network communications, a graphics monitor, a keyboard and mouse. Typically X terminals are bundled with 16 inch monitors capable of at least 1024x768 resolution, but 19 inch displays providing resolution of 1280x1024 are common. According to NCR/Applied Digital, the X terminal display market will reach 1 million units shipped in the next year. [Compet]

---

3. *Quarterdeck's DESQview/X is an exception to this since it fully implements the client/server architecture.*

PCs capable of running the X Windows environment must include additional hardware and software than the conventional setup. Typically this includes networking support, enhanced displays, and/or graphics coprocessors. A larger screen is necessary over the standard 14 inch monitor. In fact, 16 inch monitors are probably the minimum acceptable size for efficient operation, given the display of about five windows, each 80 characters by 24 lines, and a font size approximately that of a 14 inch screen.

However, X servers cannot do the job by themselves. The host servers must have enough memory, mass hard disk storage, and communication subsystems to support the LAN environment. A typical host configuration would include 6 to 8 Mbytes of memory for X Windows workspace plus 1 to 3 Mbytes of memory per X server, depending on the type of applications. Approximately two to five times the available memory should be allocated to swap disk space.[4] The actual required host hardware configuration is adjustable which allows the computing environment to be tailored to fit current business needs.

Physical Network.

X Windows requires a transport and network layer to provide host-to-host network communications. TCP/IP protocol suite is probably the most widely implemented, vendor neutral protocol providing this service. Other transport standards include XNS (Xerox Network Systems) and protocols developed by IOS, such as UDP or NVP. TCP provides services at the transport layer while IP provides services at the network layer. TCP/IP refers to a large family of protocols and services which are robust enough to allow hosts connected on different types of physical networks to communicate with each other. Figure I shows the OSI model with the TCP/IP services.

The physical layer is defined by the IEEE 802 set of standards: (1) Ethernet (802.3), (2) Token Bus (802.4), and (3) Token Ring (802.5). Ethernet was originally created by Digital Equipment Corporation and Xerox. IEEE slightly modified their definition and legitimized it into their suite of LAN interface standards.[5] The advantages of Ethernet include support of all of the TCP/IP services along with a great deal of flexibility with the physical link connection. Thick or thin coaxial cable, fiberoptic links (FDDI), twisted-pair, and broadband cable all are acceptable. FDDI (Fiber Distributed Data Interface) has real advantages in the X Windows environment with the increase from Ethernet's 10 Mbps capacity to potentially 200 Mbps as well as increasing the network cable range to excess of 100 miles.

---

4. *HP9000/375 will support 25 700/X terminals with a total of 80 Mbytes of RAM and 200 Mbytes of swap.*

5. *Ethernet and IEEE 802.3 are used synonymously even though they are technically two different standards.*

An X server session typically has two to three times the active processes as the standard ASCII terminal session. Network traffic is always a presence since most I/O sessions in X require network access causing them to be LAN performance bound. The network design is consequently a critical element when using X servers. Although X employs a queuing algorithm to reduce the number of small network packets, interactive applications can cause high-capacity network traffic loads. NFS traffic and file transfers cause additional congestion reducing the available network capacity.

TCP / IP Services

<table>
<tr><th>ISO Level</th><th colspan="4">Services</th><th>Internet Level</th></tr>
<tr><td rowspan="2">Application (7)</td><td>NFS</td><td>NS</td><td>ARPA</td><td>Berkeley</td><td rowspan="6">Application</td></tr>
<tr><td>Network File System (NFS)

Yellow Pages (YP)</td><td>Network File Transfer (NFT)

Remote File Access (RFA)</td><td>File Transfer Protocol (FTP)

TELNET</td><td>Berkeley Internet Name Domain (BIND)

REMOTE[6]

SMTP</td></tr>
<tr><td>Presentation (6)</td><td>eXternal Data Representation (XDR)</td><td></td><td colspan="2">Simple Mail Transfer Protocol (SMTP)</td></tr>
<tr><td></td><td colspan="4">Link</td></tr>
<tr><td>Session (5)</td><td>Remote Procedure Call (RPC)</td><td>NetIPC</td><td></td><td>Berkeley Sockets</td></tr>
<tr><td>Transport (4)</td><td>User Datagram Protocol (UDP)</td><td>Transmission Control Protocol (TCP)</td><td>TCP</td><td>TCP
UDP</td><td>Transport</td></tr>
<tr><td rowspan="2">Network (3)</td><td colspan="4">Internet Protocol (IP)</td><td rowspan="2">Internet</td></tr>
<tr><td colspan="4">ARP Address Resolution Protocol
RARP Reverse ARP
ICMP Internet Control Message Protocol</td></tr>
<tr><td>Data Link (2)</td><td>ARPANET
Ethernet
Token Bus
Token Ring
X.25 PDN</td><td>IEEE 802.3</td><td colspan="2">ARPANET
Ethernet
Token Bus
Token Ring
X.25 PDN</td><td>Interface</td></tr>
<tr><td>Physical (1)</td><td colspan="4">Transmission Media</td><td>Physical</td></tr>
</table>

[6] *remsh, rcp, rlogin, rexec, rwho, ruptime*

Figure I.

The network should be organized into reasonable size sub-LANs, or segments, and should provide isolation between those segments. By being able to isolate network segments, multicast and broadcast packets are controlled which conserves the network bandwidth. Subnetting is an internet addressing scheme that allows logical networks to exist comprising of smaller physical networks. This

increases overall efficiency by reducing the congestion on each subnetwork. By properly 'bridge'ing the segments, subnetting effectively distributes LAN traffic. The proper network design can be quickly installed or modified with the advancement of Ethertwist technology. Dedicated X server LANs can easily be laid off of existing Ethernet networks yielding isolated, segmented X computing environments. An extended LAN infrastructure is illustrated in Figure II.

Extended LAN Segment for X Servers

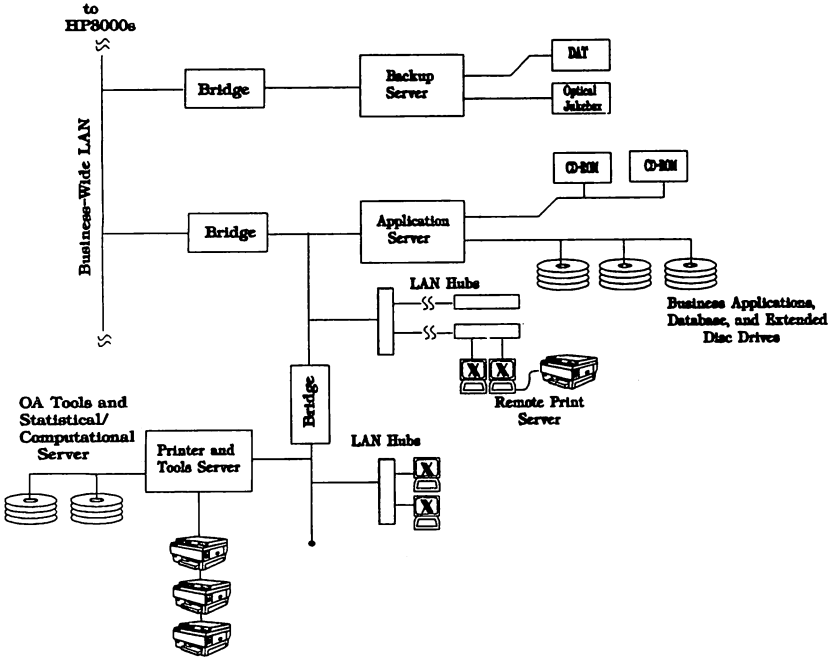


Figure II.

Network traffic should be monitored in order to determine the efficiency of the LAN topology. LAN segment traffic monitors are available via software or hardware probes to diagnose possible LAN congestion. The physical LAN design does require attention with regard to the correct positioning of bridges and servers.

Personal Computers Technology.

X Windows on the PC platform provides an ideal computing solution for mixed hardware environments. It preserves the existing hardware, software, and training investments, while taking advantage of X Windows services such as its

distributed processing power, e-mail, and print spooling. PCs can be implemented two different ways. They can either be DOS-based or UNIX-based (or both) while providing the X server functionality.

DOS-based.

Configuration

X Windows can be extended to DOS-based PC technology with multitasking and X server emulation software. This necessitates an enhanced configuration using upgraded video hardware, appropriate networking support, and additional memory (from 2 to 8 Mbytes). Essentially, the configuration comprises of the addition of a network transport layer, like TCP/IP, along with X Windows emulation. At minimum, a 12 Mhz 80286 machine with at least 2 to 4 Mbytes of RAM, and Super VGA or 8514 graphics card is required. With the addition of Texas Instrument's 60 MHz 34010/34020 graphics coprocessor board executing the X11 instructions, a 12 Mhz 286 becomes a powerful and convenient, yet inexpensive, X server.

The X server can be implemented purely in software, or it can reside in additional memory on a graphics adapter like the TI340X0. Software emulation products are available from several sources which are cited in Appendix. X emulation software usually requires memory management tools to relocate TSRs, LAN, and device drivers to either to High DOS memory or High Memory Area (HMA) to free conventional memory for the X server. If the X server emulator runs under a Windows environment, such as Microsoft Windows or Quarterdeck's DESQview, the clipboard capability allows data sharing with 'cut and paste' between X and Windows clients via toggling back and forth by a 'hot-key.' By being Windows-based, the already installed hardware investment enjoys the freedom of device independence and support for expanded and extended memory. This is a favorable advantage to effectively utilize the strengths of both DOS and UNIX.

Intel 80286 machines support X Windows environments but require a robust memory manager to handle the intricacies of memory mapping along with extra hardware for LIM/EMS support.[7] The 286 does not have the **Virtual 8086 Mode** and it is limited to segmented virtual memory.[8] The 32 bit 386/486 architecture include memory mapping capabilities, support for a flat memory model, and a paged virtual memory system allowing less expensive extended memory to behave as LIM/EMS expanded memory. These processors are more suitable for an X server platform.

---

7. *286 machines with either Chips & Technology NEAT CHIPSet or Compaq-style top memory hardware are more suitable for enabling extended memory into expanded.*

8. *When the 286 is in protected mode it can address up to 16 Mbytes of memory, using its segmented addressing scheme. However, certain operations in real mode are prohibited in protected mode, such as segmented arithmetic. DOS Extenders allow the 286 to switch to real mode so as to perform real mode operations/functions and return it back to protected mode.*

A common problem with running an X server and its clients on different machines on the LAN is occasionally poor performance due to increased LAN traffic. DOS-based X servers built upon Windows can reduce network traffic by having the Windows graphical manager become the windowing manager for the client, instead of the PC using the window manager running on the host. This is frequently referred to as using the local window management mode. If an UNIX window manager is used, X Windows on the PC is being controlled on a remote machine creating LAN traffic for mouse, keyboard, window movements.

Networking Layer

Several network hardware configuration problems have resulted with early implementations of X servers on DOS-based PCs. The difficulties arise with implementing the transport layer within the PC computing hardware environment along with PC networking software used for DOS network devices such as extended disc drives and shared peripherals. X servers require the use of a network interprocess communications mechanism, i.e. BSD sockets, to establish LAN connections. Unfortunately, early versions of PC networking software do not include such services. These capabilities have to be provided through add-on TCP/IP software packages. Networking incompatibilities may exist with implementing different ISVs solutions causing networking problems when sharing the same LANIC address.

Technically, there are several issues when implementing an X Windows configuration for DOS-based PCs. Most vendors obey the NDIS[9] (Network Driver Interface Standard) LAN card interface when implementing their networking solution. NDIS provides the capability to have multiple protocol stacks sharing the same LANIC address. For example, HP LAN Manager supports LAN Manager, NS (Network Services), and ARPA services on TCP/IP. All three can work concurrently on a single DOS PC since NDIS only sees one protocol stack: TCP/IP. Multiple protocols can run concurrently because they cooperatively use *protocol.ini*. This is a file which is read into memory to manage the protocol stack. In practice, this means if third party vendors offer protocol packages that are NDIS compliant, they are providing the necessary protocol modules and drivers which allow it to coexist with other protocols, as long as it is not implementing the same protocol already loaded in memory.[10] Problems arise when the protocol implementation used by the X server is not compatible with the DOS network software. As a result, the same LAN card cannot be used. NDIS 2.0 remedies the problem by providing dynamic binding of protocol stacks. This allows multiple protocols to be dynamically linked into NDIS. However, most NDIS products in the marketplace are still NDIS 1.0.

Workarounds include using memory management software and 2 LAN cards providing 2 LAN connections. Consequently, PCs will use one LAN interface for

---

9. *Version 1.0.*

10. *A packet driver cannot route the same protocol using a single LAN card.*

the DOS networking software and the other for the TCP software. AN additional solutions include implementing other transport layers such as XNS, NetBEUI, or Novell, instead of TCP/IP. The best solution, however, lies in using networking software which is compatible with the transport layer being implemented for the X server. These incompatibilities are presently being addressed by many of the networking vendors.

DESQview/X

One DOS-based product which deserves specific mentioning is Quarterdeck's DESQview/X. It is a DOS windowing environment which combines Quarterdeck's multitasker with X server software. It permits DOS to become part of the client/server model allowing DOS applications to be displayed on local or remote terminals across the network concurrently with X Windows clients. In effect, DESQview/X machines appear as UNIX machines across the network. This is possible due to the memory management done by DESQview along with the embedded X server code. It supports OSF/Motif and Open Look window managers as well as provides its own, DWM, which occupies only 50 Kbytes of memory. [DESQview]

The X server provided with DESQview/X controls the display screen along with its hardware type and resolution. It typically runs as a DOS Extended or Protected Mode application (enabling up to 16 Mbytes of memory on 80286 and 4 Gbytes on 80386/80486) which provides more memory workspace for X applications. If a conventional DOS application is being executed, its display is translated dynamically by translation software into X Protocol requests.[11] The requests are sent to the X server for output causing DOS programs to appear like regular X clients. This technique is theoretically possible with graphical-based applications, but it is unrealistically slow. Subsequently, DESQview/X *virtualizes* the application by remapping the video RAM to a logical window buffer maintained in unused portions of memory. DESQview/X then transfers the data to the actual video RAM clipping as necessary so it appears within its own configurable window. Virtualization is only possible on either the 386 or 486 processors. It cannot, however, perform this activity with DOS Extended programs, like Lotus 1-2-3 3.1.

UNIX-based.

SVR4

X Windows capabilities are bundled with UNIX operating PC systems. System V Release 4 Version 2 (SVR4) for 80386, 80386SX and 80486 machines with ISA or EISA busses have hooks, known as VP/ix or DOS Merge, for DOS compatibility allowing DOS binaries to run in a X Window.[12] Currently, these

---

[11] *A conventional DOS program is an application which calls DOS and BIOS routines to perform their output. Graphical based applications typically write directly into the video area.*

[12] *SRV4 is written by AT&T's UNIX System Laboratories. The difference between Version 1 and Version 2 is mainly bug fixes.*

options do not support GUIs, such as Microsoft Windows since AT&T did not include any type of DOS emulator within their SVR4 tapes. SVR4 combines the abilities of Xenix, BSD, SunOS and System V Release 3.2 (SVR3) into a unified system capable of running X Windows. The code supports ESDI, SCCI, and MFM/RLL disk controllers as well as most networking and graphic hardware cards. However, with the variety of the subtly different 386/486 machines each with potentially a hundred optional add-on boards, SVR4 can be a challenging task to implement.

Two available options are buying both software and hardware solutions from the same vendor which guarantees compatibility or just implementing a software solution which is designed to be compatible on a variety of machines. The benefits of buying bundled hardware and software can be the use of more streamlined and enhanced code as well as the addition of more UNIX software like editors, revision control systems, debuggers, and mailers. A concern when buying a software solution for existing PC hardware is guaranteeing each card in the bus has a unique hardware interrupt vector (IRQ), private address space for I/O ports, and typically 100 extra Mbytes of hard disk space.

SVR4 uses the Virtual File System (VFS) abstraction developed from SunOS. The VFS replaces the traditional UNIX file representation of an inode with a new "virtual" inode construct called a vnode. A file represented by a vnode can point to a UNIX inode, NFS mounted file, or a DOS file handle. Additionally, UNIX-based systems have the networking built-in the OS. Networking support exists for all of the popular protocols.

Currently SVR4 has a limitation to the number of disk cylinders it can support. The AT&T disk drivers do not recognize more than 1,024 cylinders per disk. Since UNIX occupies such large amounts of disk space, bigger hard drives are necessary. While neither DOS or most BIOSs allow more than 1,024 cylinders, IDE drives can mask their physical geometry expanding the potential disk space storage.[13] The firmware on an IDE controller can be altered to write larger sectors reducing the number of cylinders within an acceptable range or drives can be remapped into smaller physical geometries representing one large logical drive.[14] Most hard drives use RLL (Run-Length Limited Coding) and ARLL (Advanced RLL) which provide higher densities and consequently fewer number of cylinders, but disk drives over 300 MBytes will usually exceed the 1,024 cylinder limit.

---

13. *Both in terms of transfer rate and the potential for more storage capacity, IDE offers significant advantages over ESDI.*

14. $$\frac{\text{disk}}{\text{capacity}} = \frac{\text{sectors}}{\text{track}} \times \frac{\text{tracks}}{\text{side}} \left(\begin{matrix} \text{or \# of} \\ \text{cylinders} \end{matrix} \right) \times \text{sides} \times \frac{\text{bytes}}{\text{sector}}$$

SVR4 releases include device independent X11 graphical windowing systems. API toolkits allow developers to create applications with the same "look and feel" of a X11 GUI. Character-based windowing systems are available which require less memory and can generally execute faster on slower processors. The X11 GUI, however, requires a full graphical display management system capable of displaying bit mapped displays, icons, scroll bars, and cursors. Ironically, AT&T GUI code does not include X11R4, but implementations exist which deliver X11R4, Motif 1.1, and Open Look.

X Terminals Technology.

Configuration

The X terminal is designed to support the X Window server functions, managing the display, keyboard, and mouse input devices for its X clients running over the network. Essentially X terminals are similar to workstations, except they lack backplanes for support of internal peripherals. There is a single CPU board, containing logic to handle video, I/O, and graphics functions, a power supply, and video board. Thus, the machines have nearly zero-footprint and execute much more quietly than the typical workstation or PC. This is a significant advantage for some physical computing environments. In addition, X terminals offer less system administration over PC X servers. In fact, X terminals are ideal for Information Systems groups to regain control over the computing environment by centrally locating the servers, expensive peripherals, and mass disk storage.

The X terminal CPU chips vary among vendors ranging from the Motorola 680X0-based processors, TI's 340X0 graphics processor, RISC processors, multiprocessor technology, to Intel-based microprocessors. RISC chips, like the Intel i960CA, tend to deliver a level of performance that exceeds current complex instruction set chips, like the 680X0. Vendors using the Intel i486 chip architecture with the ISA/EISA bus can run both DOS and UNIX software.

All terminals have I/O RAM for I/O buffering, system RAM for the X server and font storage, and video RAM (VRAM) for frame buffer storage. Most of the systems also have a small amount of PROM for storing the setup programs and support for communication protocols. Subsequently, X terminals will only support the communication protocols which are burned into their ROM. NVRAM (Non-Volatile RAM) is used for storing the setup preferences.

Hardware specifications will list only system RAM (I/O RAM is usually listed if it is at least 512 Kbytes). Usually 3 to 5 Mbytes of RAM is sufficient for most software applications running on X terminals. As a precaution, frame buffer storage is commonly 8 bits wide allowing up to 256 colors. This is adequate unless an image exceeds 8 bits per pixel and the pixels get remapped. Most X terminals remap the image clipping out graphic information producing poor renderings.

Selecting the proper X terminal involves the consideration of what it would cost to upgrade. Memory chips are often the first component to consider when

upgrading. The price difference can be deceiving between vendors since systems that use industry standard SIMM can be upgraded for much less than proprietary memory boards. Memory is particularly important if the terminal is running color graphic-intensive software or many concurrent applications.

Pixel resolution and the vertical refresh rate of display monitors are important factors. In fact, the display monitor is the probably the single most costly component of the X terminal configuration and those which can use off-the-shelf displays have an obvious advantage. Refresh rates of under 65 Hz are not practical for sophisticated graphic applications. Text-based applications can be used more effectively with slower refresh rates and lower resolutions, but X Windows interfaces need more exacting graphical requirements with higher video clarity and faster displays.[15]

The X terminal CPU is the main server processor which executes X11 server instructions. X terminals maintain communication engines, like the Intel 80C186, for the network I/O managing the network communications, keyboard, and mouse. There are significant price issues regarding the type of processor being used. Slower processors offer better price/performance if the terminal is not executing display intensive applications. Their performance is measured with a metric known as X Stones. X Stones is a balance measure of how fast a terminal can render different graphics commands sent by a host. It does not measure terminal response, however, in terms of user input and network throughput, which can have a big impact on actual X terminal response. The X Stones metric should not be the sole basis of selecting X terminals. In fact, most networking environments have trouble supporting performances above 50,000 X Stones. [Vaughan]

Operation

X terminals communicate over a variety of links: Ethernet, IBM Token Ring, DECnet, or serial RS232C. RS232C can be surprisingly helpful. RS232C communications use SLIP (Serial Line Interface Protocol) which can be used for terminal cluster controllers, remote X servers accessed over 9600 bps modems, and remote printer hosts. Compression algorithms are available to reduce the overhead with the serial link to the terminal. When serial lines are necessary, the X server may reside on the host computing system. As a result, the overhead of the X and TCP/IP protocols are eliminated since only display primitives are sent over the network cable. This unique alternative gives the terminal direct access to large amounts of RAM and virtual memory residing on the host server. Other alternatives include having the window manager run on the X terminal itself. This improves the display response and limits the load on the network and server.

X terminals can implement backing-store and save-under techniques improving graphics performance by using local memory for storage and retrieval of overlaid windows. Backing-store saves raster images in memory so as to quickly

---

[15] *The dpi (dots per inch) can be calculated by taking the horizontal resolution and dividing by the horizontal width of the monitor.*

redisplay them. This is sometimes required for computationally intensive applications which have difficultly recreating their output quickly. X terminal should support save-under which is a technique of saving the image on the screen under a window. Save-unders are mainly used for transitory windows, like popup dialog boxes and menus to achieve smooth video effects.

The server function implemented in X terminals varies widely among manufacturers. The X server software can be resident in PROM or loaded from internal hard drives, booting the X server locally, or downloaded from the host to RAM.[16] Booting from PROM requires firmware to be replaced when the X server is upgraded. This can be supplied, however, in a credit card fashion allowing upgrades to be a matter of inserting the new card. If the X server is downloaded from a host, it needs to know the address of the host and the name of the file containing the downloadable X server. Additionally, if the X terminal is using fonts other than the few which might be in burned into ROM, it needs to know the address of the host with the font files and the corresponding path to those files. These are the sort of chores that all X terminals must accomplish, but unfortunately for which there are no consensus standards.

Startup and Display Management

An X terminal's network address can be entered directly at setup, but it is much more convenient to have the address dynamically determined from a network host. With TCP/IP, RARP (Reverse Address Resolution Protocol) and BOOTP (Boot Protocol) are two protocols for this purpose. RARP allows nothing more than to retrieve a network address while BOOTP is a newer solution, typically implemented in System V systems, which adds BSD networking. If your X terminal does not come bundled with the more versatile BOOTP daemon, it is available from UUNET Communications for most UNIX machines.

X terminals get their fonts from a host or from their ROM. If the fonts need to be downloaded, TFTP (Trivial File Transfer Protocol) and NFS (Network File System) are currently two popular choices. With TFTP,the path name of the font directory is entered during the ROM setup. Once the file path of the server is determined using either BOOTP or RARP, the terminal uses TFTP to transfer font files as needed. Some terminals will cache fonts in local memory for faster access. If the NFS client is coded in ROM, the X terminals mount the complete font directory over the network.

X11R4 provides a X Display Manager Control Protocol (XDMCP) which initiates *getty* and *xinit* managing the login process. Since technically there is nothing for an X terminal to login to, XDMCP establishes the client/server connection. It is an X Windows-based login daemon that runs under *init*. In X11R3, this daemon was named as *xdm*. It is started by the init process during bootstrap and runs the login widget on a list of X servers which it reads from a

---

[16] *Internal fixed disks can also store images allowing images to be refreshed locally.*

configuration file on the host. This process is suitable until an X terminal cycles power. Once the terminal loses its connection, a stale xdm socket remains along with its dangling clients. Consequently, there is no login widget for the X terminal when it is powered back up.

Efforts have been made improving upon the xdm daemon into the current XDMCP standard implemented in X11R4.[17] XDMCP requires the X terminal to initiate the management service request by broadcasting a query packet to a host. After an authorization process, an X connection is established and the login process has begun. XDMCP implements an automatic mechanism for controlling the display connection and avoiding dangling orphan processes. The current implementation uses a file, commonly named *.xsession*, capable of providing a network of X terminals with a consistent X Windows startup environment. [Braca]

There are several areas which need improvement with XDMCP. One advancement is the capability of allowing the display to select from multiple managers. This would permit users to select hosts based on the present conditions of the computing environment offloading constrained network resources across the network. Security is another area for development involving authorization and authentication. Authentication is a mechanism for the display to determine if the manager is a trusted host. This mechanism can occurs in three ways: MIT Magic-Cookie, Kerberos, and Xdm-Authentication-1. The major shortfall of the Magic-Cookie method is that it sends the password directly across the network without any character modification. Kerberos is a much better choice and has been selected by OSF. It uses DES (Data Encryption Standard) with a public encryption key. Xdm-Authentication-1 also implements DES but requires the display to maintain a private key.

Authorization becomes a problem once the display has authenticated its X connection. Any client can now freely establish an X connection to the display. The *'xhost'* type of security mechanisms provide control over what hosts can establish a display connection, but unfortunately they do not control what clients from those hosts can be displayed.

Remote client shutdown is also a current limitation with xdm. Processes executing remotely can be disconnected from their parent remaining on a system as a defunct process. Dangling processes use virtual memory and occupy the process stack. There are known solutions for such problems. TCP 'keepalives' are a mechanism which acts as policemen monitoring client TCP connections closing them if there is no response. Another display management problem is the lack of a standard method for displaying 'console' output. X terminals running xterms will not receive standard output since it is written to */dev/console*, a device which xdm currently does not support.

---

[17] *Ed Basart and Dave Mackie (co-developer of XDMCP) from Network Computing Devices, Inc. in Mountain View, California offer an excellent article on XDMCP for further reference.*

Nuances

X terminals are unable to establish network links directly to other servers on the network. They must route all processing through their host. Fixed memory limitations is another concern. X terminals have no swap or virtual memory. As a result, applications running on X terminals must be handled delicately with their memory allocation or there are client/server malfunctions and system crashes can occur. A few vendors have devised solutions to access virtual memory on the host, otherwise more memory must be added to the X terminal.

X terminals can introduce a byte-ordering problem with some UNIX utilities and compiled SNF (Server Normal Format) fonts. Some terminals are LSB (Least Significant Byte) first while the remainder are MSB (Most Significant Byte) first. UNIX utilities which access direct video memory, like screen capture utilities, simply do not work on X terminals due to the different byte-ordering. Other utilities which currently are not functional include those which write directly to */dev/console* and */dev/bil*.[18]

Conclusion.

Vendors are constantly upgrading their X terminals with new embellishments. Some improvements include running a mini OS on the terminal offloading services from the host. This allows the ability of executing terminal emulators locally. However, the idea of doing such is exactly opposite of their initial conception. Presumably individual users do not need all of their resources 100% of the time, but can share them through the host server. And when all the users do need resources, there is the safety value of virtual memory on the shared host.

X terminals functionally are not as robust as PC technology since they generally lack the dual capability of being a DOS machine. They should be selected with expandability in mind. Their components, such as monitors, memory chips, as well as the complete X terminal product itself, should be upgradeable. X terminals which have SNMP (Simple Network Management Protocol) support allow them to be effectively monitored over the network exposing any hardware or network limitations. But without the ability of upgrading the X terminal, these limitations are unavoidable.

PC technology has a strong foothold in the computing environment. It is rapidly maturing into a complete computing workstation providing accessibility to thousands of DOS and UNIX applications. The history of its development is as fascinating as its success which is substantiated by its large installed base. The benefits from its highly competitive industry include ease of upgradeability and

---

[18] *This is HP's Human Interface Link which interfaces with keyboards, mice, control knobs, ID modules, button boxes, digitizers, bar code readers, and touchscreens. Other vendors have similar raw devices for reading user input. Unfortunately, programs cannot read from X terminals' 'bil' loop.*

expansion providing a very flexible solution for the business and technical computing environment.

The issues with X servers tend to be their upgradeability, network traffic, display resolutions, ergonomics, and security. X servers technology is in an expanding field yielding cost-effective X Windows environments. Its direction is clear, however, it is not well defined. X Windows is an industry standard providing the basis for complete distributed environments, but whether X terminals become workstations of the future or PCs are blended into the UNIX platform is a matter and question of time. But with the capability of each application being able to run on separate, dedicated hosts over 10 Mbps Ethernet connections, the performance is fast enough to rival standalone workstations.

# # #

References

Hart, Denis, UNIX Today!, "Prospects Promising For ISV Ports To UNIX," January 7, 1991, p. 25.

Braca, Mike, UnixWorld, "X Display Management," April 1991, pp. 107-112.

Compet Action, November 30, 1990, p. 64.

DESQview/X, "A Technical Perspective," Quarterdeck Office Systems, 1990.

Hayes, Frank, UnixWorld, "X Terminals vs. Diskless Workstations," October 1990, pp.83-86.

Vaughan, Jack, EDN. "HP shows RISC-based X terminals at NCGA." April 18, 1991. p. 1.

Kochan, Stephen G. and Patrick H. Wood, UNIX Networking, Pipeline Associates, Inc., Hayden Books, Indianapolis, 1989.

Appendix

DOS-based X Servers

AGE
(619) 565- 7373

PC-XView / PC-XView/16
SpectraGraphics/Graphics Software Systems
(503) 641-2200

PC XSight
Locus Computing
(213) 670-6500

XVision
Visionware Ltd.
(612) 377-3627

HCL-eXceed / HCL-eXceed Plus
Hummingbird Communications Ltd.
(416) 470-1203

X11/AT
Integrated Inference Machines
(714) 978-6201

Xnth
Nth Graphics, Ltd.
(800) 624-7552

XoftWare TIGA
AGE
(619) 565-7373

DESQview/X
Quarterdeck Office Systems
(213) 392-9851

System V Release 4 X Servers

AT&T
(800) 346-7111

Dell Computer
(800) 284-3355

Interactive Systems Corporation (ISC)
(800) 346-7111

Santa Cruz Operation, Inc. (SCO)
(800) 726-8649

UHC
(713) 782-2700

Appendix (continue)

X Terminals Vendors

Datacube, Inc
(508) 535-4624

Digital Equipment Corporation
(800) 344-4825

GraphOn
(800) 472-7466

Human Designed Systems (HDS)
(800) 437-1551

Hewlett-Packard
(800) 752-0900

IBM
(800) 426-3333

Micronics
(415) 651-2300

Network Computing Devices (NCD)
(415) 694-0650

NCR
(513) 405-5000

Princeton Graphics Systems
(404) 664-1010

Samsung
(508) 685-7200

Spectragraphics Inc.
(619) 450-0611

Tektronix
(800) 225-5434

Visual Technology
(508) 836-4400

Paper # 2031 - Developing Client-Server Applications in HP System Environments using Industry Standard API's

Scott Safe

Hewlett-Packard - Colorado Networks Division

o EXECUTIVE SUMMARY

Client-Server computing is being widely adapted as the next wave in end user computing, with the PC as the primary interface into the wide network of available information. The wave is being driven by the end-user, as their need to get access to information stored on a wide variety of server platforms is required in order to conduct the business intelligence and analysis of information for better decision making, better cost effectiveness, and maintaining or leaping ahead of their competition.

The tools are in place for end users and developers to adopt this new wave in computing on HP system platforms. Graphical user interfaces, data base engines, application compilers, and especially the network API's are all available for end users and developers. All that is needed is the imagination and creativity (not to mention time) required to begin implementations of client-server solutions

o CLIENT-SERVER BACKGROUND

Today's trade press is expounding that 'client-server' computing has come of age, and that everybody is throwing out all their existing software and systems and putting in client-server solutions. Well, almost everybody...

So what is this thing called 'client-server' computing? It seems as if every different trade publication has their own description or usage of the term. Let me throw out another one. Client-Server computing is any variety of methods to distribute the user interface, the data, and the application. This description says more about what client-server doesn't do, then what it actually is. In more simpler terms, client-server computing is what your competition is doing, and you're not!

Why would your competition invest in developing applications which semmingly breaks apart what they already have in place today? Because from their perspective, they have recognized three key benefits:

1) Improved productivity and user satisfaction with a graphical user interface. This is what made the Macintosh so user friendly, and why Microsoft's stock is priced so high because of the phenomenal success of Windows 3.0.
2) Lower cost hardware implementation since the underutilized MIPS on the PC are taken advantage of.
3) Most importantly, access to information is better facilitated and communicated throughout the organization.

Client-Server computing certainly wasn't conceived by IBM, DEC or HP to sell more mainframes and minicomputers. Client-Server computing was developed by the user, at their PC, just trying to get their work done, and unable to easily get to the information located throughout the organization that they needed in order to better identify, manage and solve business problems in order to be more cost effective and competitive in the market.

From the users perspective, integrating PCs into a heterogeneous network environment really means getting easy access to services, data, database servers, technologies (voice, video) that he or she needs in order to get their job done, all from a single vantage point, the desktop.

Networking is implicit here, it's the critical highway or link from the desktop to this breadth of information stored on a multitude of minicomputers and mainframes, many of which are even HP systems. To the user, it seems simple, just plug the resource into the network, and all my problems should be solved. To the vendor It's a complex maze of protocols, links, topologies, services, API's, file structures, application layers each completely different, not designed to work together. To the end user, it's the computers worst nightmare. Access to information is all they want, why do they have to put up with this?

Fortunately most of the computer manufacturers and software writers are working together behind the scenes to simplify the users ability to get their job done, by making it easy to access the information required to be competitive in today's marketplace. This is the real definition of client-server computing.

But in order to solve the users problem of getting access to information that is needed, applications have to be put into place that can tie the desktop to the

resource or data that resides on server platforms. These distributed applications, now affectionately called "Client-Server" applications, are becoming a market reality. I'd like to explore how some of these applications are a reality, given that this maze of networking protocols exist, and hopefully give you a simple understanding of some of the alternatives open to developers.

A distributed application is essentially two different computer systems that communicate with each other via a software program. There are TWO Essential Types of Distributed Applications in use today:

Peer-to-Peer - Each computer node or resource is treated equally or another way to think about it is two workers exchanging information in a daily work setting, in a simple straightforward manner.

Client-Server - is somewhat more hierarchical. A Client requests information from a server with the appropriate resource, and the server respectively responds back with the information, given security concerns, etc. This is much like a worker asking a manager (who has the $ resources) for a raise, and the manager appropriately responds back with the information, which in most cases in not only NO, but don't ask again!

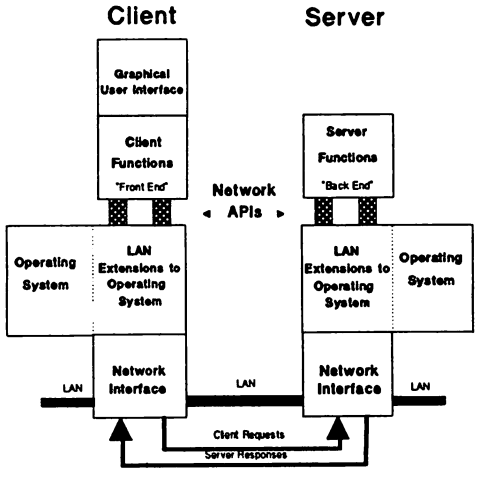
The common element here is the Application Programming Interface, or API. API's are essentially an organized routine called in software which allows for computer systems in either a peer-to-peer or client-server environment to exchange the information.

This paper and presentation will focus in on four API's that are available from HP and other networking vendors, and compare/contrast what functions they perform, and how applications can best utilize their capabilities. But before we get to the good part, a more fundamental understanding of the client-server architecture is required in order to comprehend what value the API is providing.

o NETWORKING ARCHITECTURE FOR CLIENT-SERVER

The following diagram illustrates what client-server architecture essentially looks like:

Client/Server Computing Architecture



Client-Server architecture has two main components, the client and the server. The client, usually the requestor of information, has the graphical user interface, such as Windows 3.0 for MS-DOS, the Apple Macintosh, or X.11 MOTIF for Unix. The client application displays the information in graphical format using this interface. The client application speaks to the server through the network API, which is the same or similar API on the server. It is up to the operating system or network operating system to handle the API and ensure that the data is transported over the network or LAN reliably and efficiently. The server received the data or request, and is processed back through the network operating system to the application running on the server, which processes the request, and sends back through a 'response' to the client with the requested information.

In many cases, this is no different than a terminal based application, with the exception that the client has more sophisticated processing power, and can do more sophisticated processing with the results of the information from the server, not just necessarily display the information. Many developers actually start with client-server computing by taking their terminal based application and making it work with the network, and HP's product called VPlus/Windows is an excellent first step to quickly migrate applications.

Client-server computing starts at the desktop. Access to the server resources are critical: Terminal Emulation is a must, in the absence of using tools like V/Plus Windows. HP and many third parties (WRQ, FutureSoft, GSS, VisionWare and many others) have terminal emulators that use specific API's that allow access to HP systems to run applications based on the server. Most of these applications work in a graphical or windowed environment.

Data base engines are also available that utilize client-server topology. Oracle, Sybase, Informix, Ingres and Allbase are a few that work in a client-server environment on HP systems. These vendors have adopted sockets and sockets to NetIPC API's to facilitate this processing. More API support is planned for future releases. By having these capabilities available, the server can easily manage the database, and allow a multitude of PCs to get access to this information in a client-server topology. This allows the many powerful front-end or client applications to easily manipulate and display the data residing on the database in an easy to use fashion.

Application compilers are more a function of being included in dynamic link libraries to include the calls to the network API's. This is especially important in the DOS side of the equation.

Now let's look at these specific network API's in detail to get a better understanding of what exactly they accomplish, and which ones would potentially fit your application.

Four API's will be evaluated, mainly from the standpoint that they are probably the four most widely implemented API's that are used today. These API's are:

NetBIOS

ARPA Berkeley Sockets

Novell NetWare API's

Named Pipes/Mailslots

NetBIOS

Description

NetBIOS originated as an interface to IBM's first PC LAN interface card from Sytek. NetBIOS was quickly picked up by other PC networking vendors as a de-facto standard and is widely used today for local area PC to PC networking.
The term NetBIOS is sometimes used to refer to the API that the developer uses to write network applications. NetBIOS is also used to refer to the protocols used to implement the NetBIOS interface. For the purpose of this discussion, NetBIOS refers to the API and not the underlying protocol.

NetBIOS is implemented on MS-DOS and OS/2 using several different protocol stacks. Two communicating nodes must speak the same NetBIOS protocols.

RFC NetBIOS over TCP/IP on MS-DOS and OS/2 is provided with the HP LAN Manager product, and NetBIOS is provided as part of the Novell NetWare product family.

NetBIOS provides peer-to-peer IPC, dynamic naming, session support, directed datagrams, and broadcast datagrams.

The naming service registers local names. Names may be individual or group names. An individual name must be unique and identifies an access point on a particular node. A group name is shared by multiple nodes. A message may be sent to all members of a group by sending a datagram to the group name.

Session support provides logical connections between any two names on the LAN and provides reliable transmission of messages.

Datagram support allows messages to be sent to and received from individual names and group names, and allows broadcast messages to be sent to everyone on the LAN.

NetBIOS has shortcomings in a PC-to-mini environment. Internetting, the ability to operate to remote computers via a router or gateway, is difficult to support with NetBIOS. Minicomputer networks have provided internetting for some time. However, the NetBIOS dynamic naming, group names, and broadcast datagram communication require the broadcast capability. Broadcast is easy to implement on a local network (LAN), but is difficult in an internet environment (WAN). This has

limited the usage of NetBIOS in PC-to-mini integration because of the requirement for internet support.

The Internet Activity Board (IAB) specifies RFC NetBIOS (RFC 1001 and RFC 1002) as the standard way of implementing NetBIOS over a TCP/IP protocol stack. The RFC NetBIOS specification does define a method for NetBIOS internetting, however it is still an unproven concept and is complex to implement.

HP has implemented RFC NetBIOS with TCP/IP on MS-DOS, OS/2, and UNIX to allow LAN Manager integration on all three platforms. The RFC NetBIOS interface on UNIX is only used by LAN Manger. There are currently no plans for exposing the RFC NetBIOS interface on UNIX. Novell NetWare supplies NetBIOS as an add-on module for the DOS client and 286/386 server products.

In addition to the work on RFC NetBIOS in the internet community, the TOP NetBIOS Special Interest Group is currently attempting to standardize NetBIOS API implementations on top of the OSI transport layer. The latest developments with this implementors group indicate that the OSI NetBIOS will be similar to RFC NetBIOS. A new protocol is being invented to support NetBIOS over OSI.

There are numerous shrink-wrapped applications available at your nearest dealer which are NetBIOS. It is somewhat of a misnomer though, in that alot of applications are labelled as NetBIOS compatible, but what they really mean is that the application works on a NetBIOS network, even though it really does not make any NetBIOS calls.

As an IBM standard for PC-to-PC communication, NetBIOS has achieved wide acceptance in the PC marketplace. It is important to note, however, that IBM is abandoning NetBIOS as its PC communication standard and is pushing to replace it with APPC (LU 6.2) to provide connectivity into IBM minis and hosts.

Recommended Use

The NetBIOS API should only be used when the application must be backward compatible with other NetBIOS applications or is strictly a PC-to-PC application

Advantages

- Wide acceptance in PC marketplace
- Peer-to-peer IPC model

- Sessions (message mode connections)
- Directed and broadcast datagrams
- Transport independent
- Dynamic naming with group names
- NetBIOS interface being developed over OSI
- shrink-wrapped PC applications

Disadvantages

- No PC-to-mini integration (only available on PC platforms)
- Implementations vary from vendor to vendor--different implementations of the written standard
- IBM is replacing NetBIOS with APPC (LU6.2)
- Extra layer of protocol overhead above transport

Description

Berkeley sockets, defined at the University of California at Berkeley, provides an interface to the TCP/IP transport in the Berkeley Software Distribution (BSD) of UNIX. It has grown to a de-facto standard network API under UNIX. Nearly every vendor that provides TCP/IP networking under UNIX uses sockets, including Sun, Apollo, Wollingong, Excellan, and Hewlett-Packard. A notable exception is AT&T(R), which uses TLI (Transport Layer Interface). However, a sockets application can interoperate with an application written to the AT&T TLI or X/OPEN XLI using TCP/IP.

HP provides sockets on MS-DOS as part of the HP MS-DOS ARPA 2.1 and ARPA Services for NetWare products. An MS-DOS socket developers kit is available for programmers to develop sockets applications.

The sockets API runs over the TCP/IP and UDP/IP protocol stack providing stream mode connection IPC, datagram IPC, and WAN access (internetting). While sockets does not currently provide an interface to OSI protocol stacks, it is being developed in the BSD 4.4 release. Therefore, BSD Sockets may provide a migration path to OSI.

Stream (TCP) sockets provide bidirectional, reliable, sequenced, and unduplicated data with no record boundaries. Datagram (UDP) sockets provide bidirectional data flow with record boundaries preserved, but the data is not guaranteed to be reliable, sequenced, or unduplicated.

Sockets follow the file system paradigm based on the UNIX System which allows any connection socket descriptor to be used as though it were a UNIX System file descriptor (i.e. read, write and close commands work). Sockets is a multivendor de facto IPC standard for systems based on the UNIX System.

There are calls in the sockets API that are specific to TCP and UDP. If these calls are used, then that part of the application is dependent on TCP.
Sockets allows for many connections (called "sockets") to be bound to a single transport address. This allows multiple clients to communicate with a single server process.

(R)AT&T is a registered trademark of American Telephone and Telegraph Corporation.

Sockets is a peer-to-peer API. Therefore, any process using sockets can initiate communication or accept incoming requests for communication.

Recommended Use

Client-Server database architecture really got it's start from sockets. All of the major database vendors support many versions or brands of the sockets interface. Sockets is ideally suited for applications needing the following items:

- Peer-to-peer distributed application model
- PC-to-mini integration
- Internet access
- TCP/IP protocol stack

Advantages

- Multivendor de facto standard on UNIX
- Internetting capability
- Widespread acceptance in technical market
- Many technical applications and services use sockets interface
- PC-to-mini integration
- Peer-to-peer IPC model
- Uses the file system paradigm based on the UNIX System (can use read, write, and close routines)
- No additional protocol or overhead above TCP/IP
- Potential migration path to OSI protocol stacks
- Provides local static naming or DNS access.
- Can be used for MS-DOS and OS/2 application to communicate with HP3000 NetIPC applications.

Disadvantages

- Dependent on transport
- Limited to TCP and UDP transports
- Naming is static
- No MS-DOS binary interface standard, i.e. application must be recompiled with vendor specific socket libraries if portability is desired.

Novell NetWare SPX/IPX and Portable C Interface

Description

Portable NetWare, a source code product from Novell Corporation which has been ported to run on many different computer systems, is available from HP and third parties for both the HP 3000 MPE/XL and HP 9000 HP-UX system platforms.

Portable NetWare has a wide variety of API's available for client-server application development. These API's are:

IPX - An Internetworking datagram protocol. This interface is based on the AT&T TLI (Transport Layer Interface) streams interface and allows an application to send a datagram to a process on another system running Portable NetWare. This is similar in functionality to a NetBIOS datagram, or Mailslot in LAN Manager, but is specific to the NetWare protocol, IPX. In laymen terms, IPX allows an application to send a message to another system, and IPX will attempt to deliver the message, but there is no guarantee that it will be received.

SPX - An transport level connection-oriented protocol. This interface is based on the AT&T TLI (Transport Layer Interface) streams interface and allows an application to establish a connection and to send/receive data to/from a process on another system running NetWare via SPX. SPX allows the two processes to establish a connection, and then send or receive data between these two processes. This is very similar to a sockets or named pipes interface, but is specific to the NetWare SPX protocol.

Portable C Interface allows an application to access most of the services provided by a NetWare server including: accounting, bindery, connection, file, path, queue, synchronization, and transaction tracking services. This is similar to Mailslots in LAN Manager where it allows applications to get access to administration services about what is going on in the system.

Recommended Use

Applications needing the following should consider the IPX, SPX or Portable C Interface API:

- Peer-to-peer distributed application in a NetWare specific environment.
- Access to NetWare Services (Portable C API)

Advantages

- o IPX/SPX interfaces are a binary interface on MS-DOS clients and do not have to be relinked.
- o Low overhead - low memory utilization
- o IPX/SPX protocol is efficient handling small packet sizes
- o Allows client-server applications to be developed on minicomputers that communicate with native NetWare clients.

Disadvantages

- o Very low level protocols.
- o No concept of names - uses network and link addresses.
- o Applications must deal with data integrity
- o Applications must use other services to find name or address of peer.

LAN Manager Named Pipes/Mail slots

Description

Named Pipes/Mailslots is a high level, transport independent API defined by Microsoft as part of LAN Manager. Named pipes/mailslots are an emerging standard for PC workgroups. HP, 3Com, IBM, Ungerman Bass, Excelan, and Novell have stated their intentions to support named pipes.

Named pipes/mailslots has the advantages of transport independence, multivendor support, and the ability to build "shrink-wrapped" network applications for MS-DOS and OS/2. Shrink-wrapped named pipes/mailslots applications will run on multiple vendor's platforms with many underlying protocol stacks without modification.

Named pipes is similar to pipes in the UNIX operating system. Named pipes are integrated with the file system under MS-DOS and OS/2. Application developers access the APIs in a manner similar to file system. Once a pipe is established between two processes, data is transferred by writing to the pipe. Data is received by the remote process by reading the data from the pipe. The idea is that developers do not need to understand network dependencies to write distributed applications using named pipes.

Mailslots provides a datagram message facility between processes.

Because named pipes/mailslots are integrated with LAN Manager, they transparently take advantage of LAN Manager features such as security, login control, error handling, and auditing. Communication is multiplexed over any existing LAN Manager disk and printer sharing connections.

HP LAN Manager, 3Com 3+Open, IBM OS/2 LAN Server, Ungerman-Bass NET/ONE, and a future release of Novell Netware 386 support named pipes between MS-DOS and OS/2 PCs. By implementing named pipes on LM/X, HP has extended named pipes into the UNIX environment, and is also supported on the HP 3000 MPE/XL systems.

Hewlett-Packard's LAN Manager/X Server is an industry standard implementation of LAN Manager on UNIX developed by Hewlett-Packard and Microsoft that runs over TCP/IP and RFC NetBIOS. Hence, named pipes/mailslots provides connectivity from DOS or OS/2 clients to LAN Manager/X servers.

Named pipes is a client/server IPC. The application process that accepts an incoming pipe request must be located on a LAN Manager server machine. HP currently provides LAN Manager servers on OS/2 and HP-UX (Series 300/400 and 700/800).

The application process that initiates a pipe establishment request (outbound named pipe) must be located on a LAN Manager client machine. HP currently provides LAN Manager clients on MS-DOS and OS/2. HP plans to support outbound named pipes on UNIX in a future release.

The named pipes interface under UNIX does not integrate with the file system. However, the calls are identical to the OS/2 interface. The UNIX named pipes API is a subset of the OS/2 capabilities. Only the calls necessary to implement a distributed application server (inbound pipes) is provided in the first release of LAN Manager/X. HP plans to support the named pipe calls for distributed application clients (outbound pipes) in a future release of LAN Manager/X.

Because LAN Manager is implemented as a layer above NetBIOS, named pipes/mailslots requires an additional layer of protocol when compared to NetBIOS or to BSD Sockets.

Recommended Use

Applications needing the following should consider the named pipes/mailslots API:

- Transport independence
- Allows shrink-wrapped MS-DOS applications
- PC to LM/X or OS/2 LAN Manager integration
- Client/server IPC model with clients on MS-DOS and OS/2 and servers on OS/2 and UNIX (MPE in the future)
- Ease of development and maintenance

Advantages

- Transport independent
- Allows shrink wrapped MS-DOS applications
- Follows OS/2 file system paradigm
- Uses LAN Manager security, login control, and auditing
- Applications are easy to develop and maintain
- Potential migration path to future protocol stacks such as OSI or IPX/SPX.

Disadvantages

- o Peer-to-peer IPC only between OS/2 nodes (today)
- o Inefficient for small transaction sizes (overhead is high)

o WHEN TO USE WHICH API

When looking at how developers have used the different API's, it often boils down to five simple calls: Open a session, read a session, write a session, close a session or inquire as to the status of a session. Many developers that I talk to recommend to keep it simple, and not get tied to in-depthly to a single API, as it allows the portability to move that application to another network and API much easier.

The recommendations on when to use which API is not simply a matter of which API is most technically suited for the application. The most important point to consider in selecting the API is based on the business objective: Are you going to resell the application, or is it designed for in-house use only?

The answer to this question should drive which API or API's the application should be developed to. If the product is going to be resold, into which markets will the product sell into? Developing applications for the Berkeley Sockets interface would not make alot of sense if there is no UNIX installations in that business where you would potentially sell into.

If the application is used in-house, and you are predominantly a Novell NetWare installation, then it would be wise to invest in developing applications which support the NetWare API's. If the environment is mixed, then evaluate sockets today, and perhaps Named Pipes in the future.

Unfortunately, as this industry matures, even this decision point can get cloudy. It is expected that Novell will be supporting all of the mentioned API's within the next two years, and Microsoft could certainly be considered to do the same, with maybe the exception of the Portable NetWare API's, as they have the equivalent functions with LAN Manager Named Pipes/Mailslots.

As an example, some of the client-server database vendors mentioned earlier have focused on portability, and have actually written a layer or API above the network that developers use. Then they apply the network piece or module and plug it in to talk to the specific flavor of network that the user has. While this offers tremendous portability, they do not take advantage of some of the higher performance characteristics of specific API's.

o CREDITS

Many individuals have contributed work toward outlining client-server development strategy and usage of API's at Hewlett-Packard. The author by no means takes credit for most of this paper. The following individuals have either written papers or prepared presentations on this subject matter, and credit is duly acknowledged and appreciated:

GARY STEARNS, Hewlett-Packard Company, Colorado Networks Division

TERENCE LISTER, Hewlett-Packard Company, Colorado Networks Division

ARLENE YUSNUKIS-GIBBS Hewlett-Packard Company, Colorado Networks Division

GEORGE FERGUSON, Hewlett-Packard Company, General Systems Division

Paper #2032 - Connecting NetWare Clients to HP Systems

Dan Williams

Hewlett Packard - Colorado Networks Divisions

o Executive Summary

In the past, there were limited options available when connecting PCs on Novell NetWare LANs to HP 3000's or HP 9000's. Serial connections were used and continue to be used for terminal emulation, but provide limited functionality other than basic terminal emulation and file transfer. Gateways were developed to provide high speed terminal emulation connections and to better utilize the existing LAN hardware already in place.

Products will be to provide LAN-based connectivity from NetWare clients to HP 3000 and HP 9000 computers, but without the need for gateways. These products not only include the networking for terminal emulation and file transfer, but also provide capabilities for the future with client/server functionality and integration into HP NewWave Office. HP system users can solve business problems by implementing these client/server computing products.

o Networking Basics

First, let's take a look at why NetWare clients could not talk to HP systems. In the early '80's. HP chose industry standard TCP/IP as its common transport for LAN based communication between HP systems, and between PCs and HP systems.

HP developed Network Services (NS), which runs on TCP/IP based LANs, to provide connectivity in the HP 3000 environment. NS provides Virtual Terminal for terminal access, DSCOPY for file transfer, and NetIPC for program-to-program communication. These services allowed HP 3000s to communicate with each other and were later extended to the HP 9000. NS was extended to the PC in 1984 as part of HP OfficeShare follwed by HP Network Services/MS-DOS for the LAN Manager environment in 1989.

In the Unix environment of the HP 9000, industry standard ARPA services provide the same type of functionality, with Telnet for terminal access, FTP for file transfer, and Sockets for program-to-program communication. Again, these services operated between HP 9000s and from the PC to the HP 9000 with HP ARPA Services/MS-DOS..

Novell developed IPX as its standard transport for NetWare client to NetWare server communication. IPX is a low memory, high performance transport which served this particular function well. However, IPX does not provide a good enterprise wide networking solution, so the system vendors such as HP, DEC and IBM did not implement the IPX transport on their systems. Instead, they chose TCP/IP, SNA and DecNet, respectively, to provide both system to system and client to system connectivity.

As a result, NetWare PC clients were unable to have direct LAN connectivity to these host systems. Due to memory constraints, the PC was able to only have one transport loaded at a time, and so a choice was required - either access to the host system or access to the NetWare server. But connectivity to the hosts for terminal emulation and file transfer was still a need. So users went with serial connections, or communicated with their host system via one of the many gateways which soon became available for NetWare.

Asynchronous connections

Asynchronous, or serial, connections have long been a staple of PC to host connectivity. They provide a reliable, well understood means for connecting terminals and PCs to hosts. In an HP system environment, there are basically three methods used for serial connectivity.

1) Direct Connect to DTC

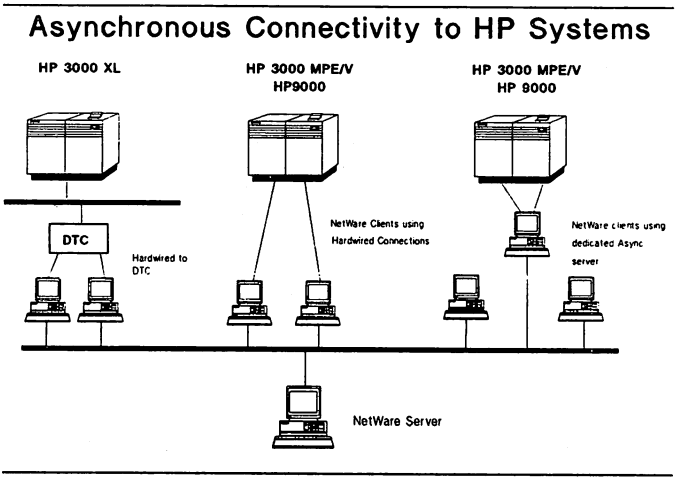
With HP 3000 MPE/XL systems, PCs using serial connections attach through a Distributed Terminal Controller.

2) Direct Connect to host

With HP 3000 MPE/V or HP 9000 systems, direct connections can be set up to terminal ports in the systems.

3) Asynchronous Server

Less common is the use of an asynchronous server. The Novell Asynchronous Communications Server (NACS) has is frequently used in these environments. It allows NetWare clients to use a dedicated PC on the NetWare LAN to share up to 16 RS-232 ports to provide connections to an HP system.



Limitations of Asynchronous connectivity

Despite their versatility in connecting PCs to host, serial connections have their limitations:

o The biggest limitation of serial connections is speed. Typical baud rates in this environment are 9600 baud or less.

o Serial connections have a distance limitation of only several hundred feet and are not very flexible for a dynamic environment when people or computes are moved.

o Serial connections also limited the PC's ability to access multiple host systems. Very often, PCs with serial connections were tied to one system. LAN based connectivity allowed clients to use the LAN to access many different systems.

HP has offered client/server products which operate over serial connections, such as HP AdvanceMail and HP Information Access. These products allow the PC and the HP 3000 to distribute the computing load for electronic mail and access to HP 3000 databases.

But its in a client/server environment where serial connections have their biggest limitations. Just the term "server" implies a LAN environment. Taking advantage of client/server capabilities requires the high speed, flexible connectivity which only a LAN can offer.

o Steps Forward - LAN to LAN Gateways

Just the speed limitation of serial connectivity for terminal access and file transfer was enough to encourage the development of better connectivity between NetWare clients and HP systems. But in addition, software was being developed using the TCP/IP transport which took advantage of ARPA or Network Services on the PC. Terminal emulators offered Telnet access to the HP 9000 for higher performance. Client/server applications for the HP 3000 were developed which used the NetIPC program-to-program interface.

But because IPX and TCP/IP could not live on the same PC, there was no way to use these applications on a NetWare client. So over the last few years, various third parties have offered gateways which provide ARPA services to HP 9000s or other Unix systems, and HP offered the NS LAN Gateway product for HP 3000 access.

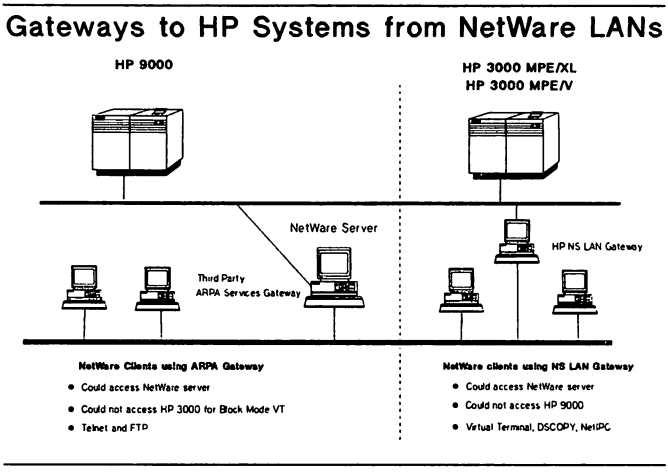
1) ARPA Gateways.

One solution which met the Telnet and FTP requirements in a NetWare environment was an ARPA services gateway. Typically, these gateways resided in the NetWare server and provided ARPA services for a fixed number of NetWare clients. A dedicated gateway card with an Ethernet port joined the NetWare LAN to the TCP/IP LAN on which the HP 9000 or other ARPA services-based hosts resided. FTP or Telnet software on the NetWare server was accessible to the NetWare clients, usually with a maximum of 32 clients.

2) NS LAN Gateway

In 1989, HP released the NS LAN Gateway for connectivity between the HP 3000 and NetWare clients. This product used a dedicated PC as the gateway, with two LAN interface cards - one for the NetWare LAN and the second for the HP 3000 LAN. An interface on the NetWare client provided the functionality of HP OfficeShare, consisting of Virtual Terminal, file transfer with DSCOPY, NetIPC application support, and the ability to use the HP 3000 as a server with Resource Sharing.

The primary use for this product was to provide the same type of connectivity available to a native OfficeShare client to the NetWare client - basically a LAN to LAN connection instead of a serial connection. There was no NetWare software involved on the HP 3000 side. Standard Network Services and Resource Sharing were used on the HP 3000.



Advantages of Gateways

Each of these gateways performed its specific functions well. The gateways eliminated the wiring limitations of serial connections. They were able to bridge different media - for instance, NetWare clients on a Token Ring network could use a gateways to access HP systems which operate on Ethernet LANs. Gateways provided the advantage of access to multiple hosts at a higher speed than typical serial connections. However, they required dedicated hardware, usually a PC, to provide the connectivity.

o The Future - Direct LAN connectivity

Although gateways were a big step forward compared to serial connections, direct LAN connections provide additional benefits. All clients using a gateway are required to converge at that one point -the gateway. Also, gateways typically have a resoruce constraint (e.g. NetBIOS) sessions) that limits the number of active sessions. In comparison, direct LAN connections are able to go straight to the host systems of choice without vying for the resources of a gateway with other clients.

Also, client/server computing requires the client be able to access the host system for program to program communication, either via NetIPC or ARPA Sockets in an HP system environment. This capability was available with HP Network Services 2.1/MS-DOS or HP ARPA Services 2.1/MS-DOS for a client in an HP LAN Manager environment. But the NetWare client could not use these services without losing the NetWare connectivity. The need was there - somehow put the NS and ARPA functionality on the NetWare client.

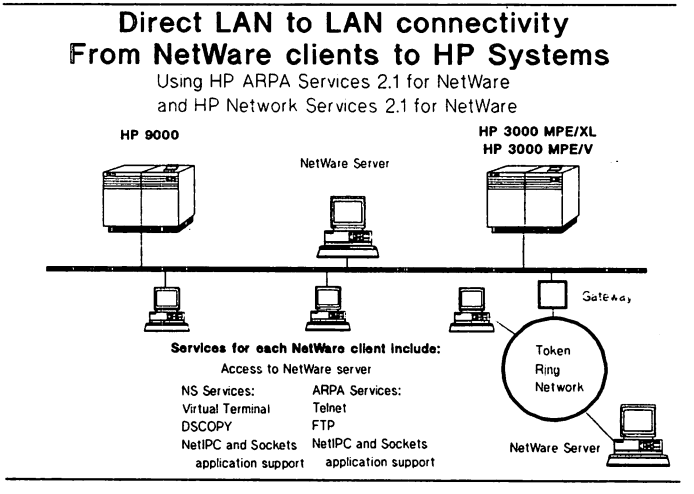
To do this requires solving the problem described in the beginning of this paper: how to put both IPX transports and TCP/IP transports on one PC. There are several key requirements for this environment.

* only one network interface card be used in the client
* access to the NetWare servers and applications based on NetWare must not be disturbed
* The terminal emulation, file transfer and NetIPC/Sockets capability must be able to run without rebooting the PC

* Both IPX and TCP/IP-based services must be able to run in a Microsoft Windows 3.0 and HP NewWave 3.0 environment

The release of HP Network Services 2.1/MS-DOS for NetWare and HP ARPA Services 2.1/MS-DOS for NetWare will solve this problem and meet these key requirements. The basic Virtual Terminal or Telnet services along with file transfer are provided. In addition, each product supports both NetIPC or Sockets, so a single NetWare client can access both HP 3000 and HP 9000 systems.

In the following environment, the NetWare clients attached to Ethernet can communicate directly with the HP hosts. The PCs on the Token Ring network will need to use either a Token Ring-Ethernet bridge or IP routing gateway to span the two networking topologies. In both examples, the PCs will simultaneously run both NetWare and ARPA Services 2.1 /MS-DOS for NetWare and/or Network Services 2.1 /MS-DOS for NetWare.



o Two End User Implementations

With these products, customers of both Novell and HP will be able to take advantage of these new capabilities to implement creative solutions to solve their business problems. The following two examples are firms who could integrate NetWare clients into an HP 3000 and HP 9000 client/server computing environment using HP Network Services 2.1 /MS-DOS for NetWare or HP ARPA Services 2.1 /MS-DOS for NetWare.

Law Firm

This large Manhattan law firm specializes in leveraged buyouts. They use an HP 9000 Series 800 for two integral parts of their business. The first application is an extensive database system to keep track of client information and client document history.

The second application involves management of the thousands of legal documents in production or maintenance. The office occupies several floors of a Manhattan high rise, and documents were spread out over many PC servers. The firm has implemented a system which consolidates all the documents on the HP 9000 and makes it simple for anyone anywhere in the office to retrieve a document.

System 1: Client Database

This group of users needs access to the client information database residing on the HP 9000. This application was developed using an Informix database, and is accessed via terminal emulation from the NetWare clients. HP ARPA Services 2.1 /MS-DOS for NetWare and HP AdvanceLink would provide Telnet terminal emulation over the LAN to this system, and would provide access to several other applications the MIS department has written on the HP 9000.

Application 2: Document management server

As with all law firms, generating legal documents is a major part of their business. They currently have over 200 PC users creating or using these documents, and the documents are stored on a number of different NetWare servers spread throughout the office. Finding a given document is a painful process of searching multiple servers or finding someone who knew its present location.

To improve this process, they have developed a system which uses an Informix database to store and keep an index of all the documents. The word processor they use, XYWrite, is able to call an application they have developed which reads and writes files to the Informix database, rather than storing the documents as DOS files on a server. This application will use the Sockets capability of HP ARPA Services 2.1 /MS-DOS for NetWare to communicate with the Informix database, moving the document into the database, and updating the index. To retrieve a document, the user would type in its name. The application would then search the index on the server and then load the document from the database directly into XYWrite.

To the end user, this process is transparent. The user will only need to know the eight character DOS filename of the document, and the standard XYWrite file retrieve command will be used. The Sockets based application would take care of the finding, retrieving and saving the document. The NetWare PC based servers will remain in place as application and print servers.

Medical Products Firm

This company manufactures medical diagnostic equipment. They use the HP 3000 extensively for production, inventory, and order systems. In addition to this terminal access requirement, Information Access is used over serial connections to provide information to many functional areas, including sales, finance, customer service and technical support. Their parent company uses HP Desk, and there are plans to install it here and tie in to the parent's system. They are also beginning to implement a sales force automation program which will enable their salespeople to get at customer and order information while in the office and remotely with laptop PCs.

The entire firm uses NetWare. To get the data to the functional areas using Information Access, one PC running a batch file nightly pulls data from the HP 3000. This data is stored in DBase format on the Novell server, which can be accessed by each functional area. The MIS department wants to improve this process, and has two goals:

1) Improve the speed and ease of access to the data by using LAN connections to each PC.

2) make it possible for users in these functional areas to access the data directly. This will reduce the support and development burden on MIS and allow users to get exactly the data they want in a choice of PC application formats (Lotus, DBase, etc.)

Many of the NetWare clients will continue to use DOS based character mode applications. But as part of the automation project, the company is interested in using NewWave and two NewWave applications, NewWave Access and NewWave Mail. To do this, they will need LAN connections from their NetWare clients to the HP 3000.

With HP Network Services 2.1 /MS-DOS for NetWare, the company could replace the current serial connections to the HP 3000 with Virtual Terminal over the LAN. This provides terminal access to their current applications and to HP Desk when it is installed. In addition, the NetIPC capability of Network Services 2.1/MS-DOS for NetWare will allow Information Access to pull data from the HP 3000 over the LAN with increased speed.

For the PCs moving to NewWave, Network Services 2.1/MS-DOS for NetWare supports both Windows 3.0 and NewWave 3.0. NewWave Mail, NewWave Access and AdvanceLink for NewWave could all be used to provide the NetWare client PCs full integration with NewWave and the HP 3000.

Finally, to provide remote access to the salespeople using laptops, the company is investigating the Novell Remote Access Server, which will allow the laptops to dial in and access NetWare servers. The PC running the Information Access batch process would remain in place and would continue to output the Dbase information to a location on the NetWare server where these laptops could access the data.

Distributed Fault Tolerance
(Paper no. 2033)

Joseph M. Eyre
General Systems Division
Hewlett Packard Company
19490 Homestead Road, MS 41AF
Cupertino, California, 95014
(408) 447-4862

ABSTRACT

In all aspects of business today, computing resources are becoming increasingly integral to productivity, competitive success, and customer satisfaction. It is crucial that these resources reliably provide application and data availability when needed. Hewlett-Packard has recognized the requirement for scalable high availability solutions and has responded with a strategy called Distributed Fault Tolerance.

This strategy encompasses both the prevention and rapid recovery from hardware, software, operational, and environmental failures in loosely-coupled, client-server environments by focusing on application availability and data integrity. Hewlett-Packard's current high availability and fault tolerant solutions and those under investigation address these issues by using standards-based transaction, system, resource management, and communications technologies. This paper describes Hewlett-Packard's Distributed Fault Tolerance strategy and direction and explains currently available high availability and fault tolerant products.

Introduction

More and more businesses today rely on converting computing technologies into competitive, revenue-generating products and services. Because of customer and time to market demands, these technologies must be quickly and reliably implemented across systems and networks. To facilitate these demands, companies are requiring flexible, standards-based hardware and software solutions.

Beyond quick implementation, customer and competitive pressures cause a subsequent concern, data and application availability. With the multitude of single-system and distributed services both operating today and planned, downtime can result in lost revenue and customer dissatisfaction. Downtime, whether it be planned because of maintenance or upgrades, or unplanned due to

some type of failure, can be categorized into four areas with regard to computing resources: 1) hardware, 2) operating system or applications, 3) operational (such as an operator error), or 4) environmental (such as power failures).

The remainder of this paper focuses on how current and emerging computing technologies can minimize and in most cases, eliminate, the first two categories of downtime, hardware and software. Operational and environmental downtime will not be covered. Operational downtime can be minimized with automatic processes, better training, and more intuitive interfaces. Environmental problems can be minimized with wise site layout and disaster contingency plans.

Traditional High Availability and Fault Tolerant Solutions

The traditional computing paradigm has evolved from the mainframe to minicomputer servers. In these situations, application processing and data is on the server only. Consequently, most of today's high availability and fault tolerant systems solve availability problems for servers. This section describes these solutions. Solutions that provide availability in a distributed environment are described in the rest of the paper.

The foundation for preventing data and application downtime is reliable hardware and software. Systems, peripherals, and networking with high MTBF (mean time between failure) are essential. The operating system must be thoroughly stress-tested and applications must be developed and tested independently and with each other.

After systems are in place, planned downtime may be required for backups, maintenance, or hardware or operating system upgrades. Planned downtime can be minimized or eliminated with on-line backup, hot replacement of hardware components (hardware, such as I/O cards, can be added or removed from the system without removing the system from service), or temporary removal of a system from the network for hardware changes. The operating system can be upgraded either on-line, if the platform supports this capability, or also by the temporary removal of the system from the network.

Unplanned downtime due to data loss has been partially solved for individual systems with disk mirroring products (see Figure 1). Disk mirroring simply uses redundant disks and disk interfaces to provide identical copies of data. Should one disk or interface fail, the system continues to operate using the second, mirrored copy. A secondary benefit is on-line backup capability. This is done by taking one of the mirrored disks off-line, backing it up, and then adding it back to the system

and reimaging it to be identical to the operating disk. A final area of concern is data integrity. Although disk mirroring provides data availability, it does not ensure integrity. Some type of logging mechanism (such as those used by databases), or data comparison algorithms (such as those inherent to most fault tolerant systems) is required.

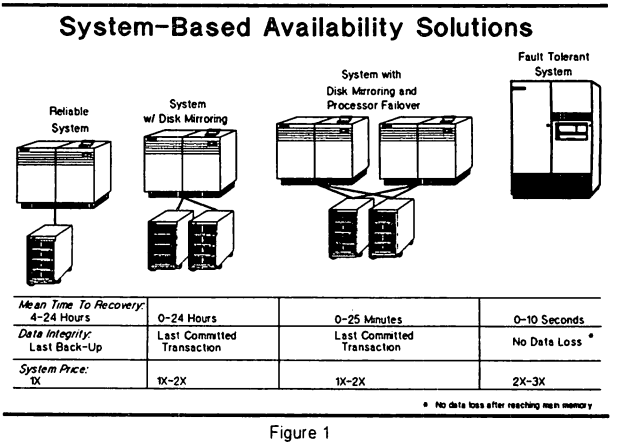


Figure 1

The next tier of availability is at the system level. In a situation where several systems are networked together, software exists to monitor the state of mission-critical systems and detect when such a system fails. Automatically, a standby system takes over for the failed, mission-critical system. Although processes are interrupted during the failover and data that is not committed to disk is lost, the second system can be available for processing within ten to thirty minutes.

The highest level of availability today is fault tolerant systems. These systems are distinctly different from high availability solutions in that data integrity is assured (of data within the system) and processing, as seen by the user, is continuous. Hardware redundancy provides both the data availability and continuous processing while data comparison algorithms provide integrity by checking the output of an operation as performed by multiple processors. Software fault detection and process retrial is also available to identify and retry unexpected results returned during application processing, thus preventing operating system panics.

The solutions that have been described in this section address the needs of server environments. Servers will always be needed for reasons such as terminal-based applications, critical applications, real time data collection, and performance. In the next sections, the paper looks at the availability needs of distributed environments as they become more prevalent in computing.

Distributed Models Today

Inherent to most businesses today is the need for distributed processing. Business support systems, for example, require information from multiple databases for customer support. Network operations must administer, survey, and test heterogeneous systems as well as provide rerouting of services because of failures. Intelligent networks require service creation, testing, and addition without disturbing other existing services. Many other solutions and services also require some type of distributed data and application replication and recovery mechanisms.

Distributed availability is provided today via several models. The simplest is possible in a read-only situation, such as directory assistance. A primary system accesses a database during normal operation. If this system or one of the communication lines fails, user traffic is rerouted to a standby system that begins to access the database. This may be possible with shared disk capability (two systems can access the same set of disks) or in cases where performance is important, applications can be modified to provide faster rerouting capability.

A second solution is the replication of data to a hot standby. Data is automatically and continuously copied from the primary to the hot standby. Should the primary fail, an operator reroutes traffic to a standby. A fault tolerant system could provide this level of redundancy; however, it would not work when disaster contingency plans or environmental constraints require physical separation of the backup processor.

These solutions potentially have several problems. First of all, data integrity is at stake when multiple systems may be writing to the same database. Secondly, unlike a reboot, application startup can become "confused" when multiple, lingering processes are still operating. Another concern is software process checkpointing, which determines the interaction of the user's state and the application state at all potential failure points. Used in hot standby situations where a backup system has identical copies of data, software process checkpointing is difficult to design and verify. Finally, data can be lost and processes interrupted during the transition from one system to another.

Emerging Technologies to Provide Distributed Fault Tolerance

Standards-based technologies are emerging that will provide greater availability of distributed data and applications. These include distributed file systems, naming services, transaction technologies, better services to manage distributed processing, and network management tools that will allow identification, isolation, and repair of transient and permanent distributed system faults (descriptions of each follow).

The architecture on which distributed availability is based consists of front-end systems or switches interfacing to a "back-end" processor network. The front-end systems may be clients hosting distributed applications, systems buffering input from large numbers of users (such as for a database forms package), or fault tolerant systems for real time data collection or critical applications.

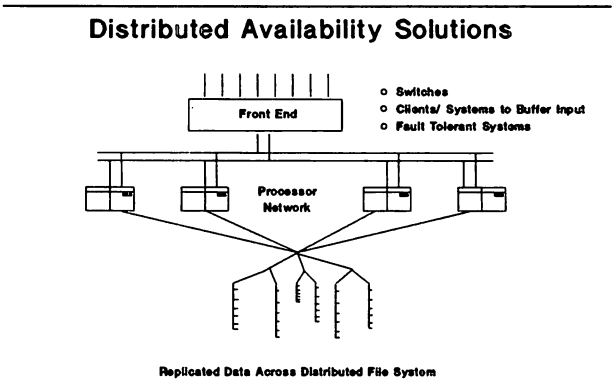


Figure 2

The processor network consists of systems operating off of a distributed file system. A distributed file system, such as the Andrew File System adopted under the Distributed Computing Environment (DCE) from the Open Software Foundation (OSF), allows users on physically separate systems to access what appears to be a single file system. This distributed file system provides the foundation for distributed data availability.

Distributed Data Availability, Integrity, and Recovery

The architecture and use of data storage and movement in a distributed environment is critical. To begin with, during normal operation the environment (such as HP's New Wave Computing Environment), must allow data interchange and sharing. Users and applications need to access and update data without knowing where it came from.

Replication, the mechanism for providing data availability, has the potential to greatly complicate data sharing, movement, and updating in distributed environments. Throughout the DCE, however, is the Naming Service, which greatly simplifies distributed data management by creating, naming, and maintaining multiple copies of data. With it, changes to names or their attributes are automatically propagated to all replicated copies. In addition, performance is increased by allowing replicated names to be near the people who use them.

Of course, system or network failures occur. When a failure happens, potentially-corrupt data needs to be isolated from other "pools" of data. Update techniques must repair this data without affecting other, non-corrupted data. Contingencies should also be made to allow applications to continue to operate with partial data availability during failure states.

Distributed data integrity and recoverability is possible through two mechanisms, both of which utilize logging. The first is the OSF Episode journaled file system. This file system keeps a journal of all unfinished actions within it. If a failure occurs, recovery is facilitated by repairing the open actions. A second, very powerful technology, is employed by transaction monitors. Transaction monitors collect transactions from the presentation layer (user level) and submit them to the underlying applications (see Figure 3).

The transaction monitor keeps track of which transactions have completed and which are in progress. It utilizes the two-phase commit mechanism, which checks the remote, receiving disk or data storage system to make sure it is ready to receive data and then afterwards to make sure it completely received and committed the data. When a failure occurs, the transaction monitor uses its logs and database logs to determine the state of the transactions and to resubmit those that were uncommitted when a failure occurs.

Of course with all distributed situations performance is a concern. Data access performance can be improved by biasing storage allocation in favor of critical applications. Storage management solutions, for example, could archive little-used or low-priority data to less-expensive media.

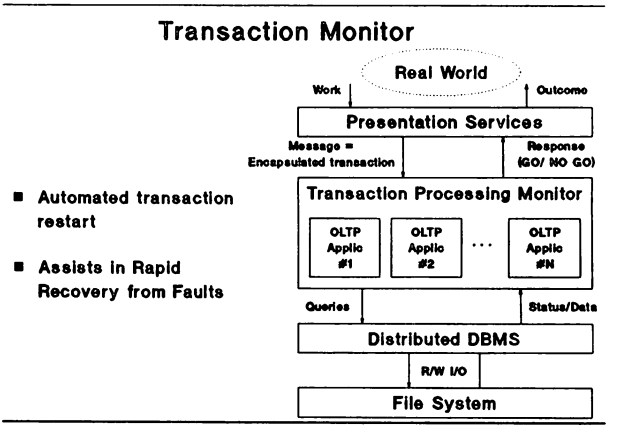


Figure 3

Distributed Process Management

Applications are distributed for many reasons, some of which include allowing access from multiple locations, making the best use of computing resources, or reducing the risk that a single system failure may disrupt many users or services. To a user, this distribution should not be apparent as he/she interacts with applications on the screen.

To minimize planned downtime, application migration must occur infrequently and be transparent to end users. Platform developers can localize operating system changes to as few nodes as possible. Application developers should not require changes but allow users/administrators to selectively use new functionality that adds value to the existing applications.

For unplanned downtime concerns, distributed applications can be written to take advantage of the standard interfaces and responses of transaction monitors. If, for example, a transaction monitor cannot commit a transaction due to a system or communications line failure (redundant TCP/IP and FDDI will greatly reduce these failures), the application can "react" to the error code returned from the transaction monitor by either initiating a follow-on transaction to another system or by pausing in a waiting state until the resource becomes available (such as during a processor failover). In these situations, the

user may see some delay in application processing, but should not experience loss of the application.

A final possibility for coping with failed resources is process migration. Processes can automatically migrate from system to system on the network for availability and, during normal operations, to provide increased performance via load balancing.

Network Management Tools

Network management is the unifying aspect of distributed environments. Core to network management is centrally-administered installation, maintenance, and network load balancing via redundant or alternate paths.

When considering distributed availability, network management tools will both prevent and react to failures. Failure prevention is done by identifying systems that are experiencing transient hardware or software errors. When these errors begin to occur above a pre-determined frequency or threshold, then the network management tools will notify the administrator so that repair or reconfiguration can be initiated.

In situations where a failure has already occurred, network management tools will isolate the failed process or node, aid in the diagnosis of the problem, and either provide repair tools or methods to reconfigure operations around the failed node.

Vendor tools that provide some of the above functionality are available today, such as HP's Openview architecture and underlying products. It is inevitable that more products and technologies will be coming, such as the OSF Distributed Management Environment (DME), when decided, will provide much of this capability.

Summary

Distributed processing is becoming increasingly necessary and prevalent in today's competitive, global marketplace. Previous system-based paradigms for availability and fault tolerance are being supplemented by distributed data and application availability solutions. These solutions will allow data integrity and recovery at a network level and will provide nearly-continuous, if not continuous, processing of applications. These solutions are based on emerging standard technologies that are interoperable on heterogeneous systems, giving companies the opportunity to provide timely, cost-effective, services to their information-oriented customers.

2034

THE STRUCTURED PROJECT MANAGEMENT FRAMEWORK

Gottfried Bertram, R&D Manager

Computer Systems Boeblingen
Boeblingen, Federal Republic of Germany
Tel. +49 7031 14-2644

ABSTRACT

A process for planning software projects is suggested. First, the SW components to be developed are identified. They are divided into categories such as user interfaces, application functions, data bases, communication mechanisms, and integration modules and interfaces. Most software products consist of components which can be assigned to these categories. Each category calls for a different design, implementation and estimation or calculation technique.

The second step of the planning process applies the software life cycle to determine the activities needed to create each of the system components. This is essentially a multiplication process: m activities times n components means that m x n work units must be completed. In step 3, each of these work units is scheduled. Step 4 involves making a "cascade" of contracts with those responsible for deliverables. Each contract defines the objectives, approach and deliverables, and the deliverables at one level define the objectives for the next lower level. (This is an application of Management by Objectives and HOSHIN planning.) Deliverables are always concrete objects such as code or documents which communicate results. Project control is achieved by using "components delivered" as the measure of progress.

INTRODUCTION

It is desirable to apply "management by objectives" to software development products. However, this requires identification of measurable objectives, because you can only manage what you can measure.

Statements like "% complete" or "life cycle phase 3 finished" are not useful for detailed management, because they don't provide any insight into individual deliverables or individual problems. This insight (and control) can be provided by defining progress in terms of actual deliverables, where deliverables are defined as documents or executable program code, implementing stated FURPS+ objectives as required to achieve defined project goals.

Projects should be decomposed into work units and tasks, each of which produces a deliverable or a component thereof. A prerequisite to this is dividing a software product under development into subsystems and components. Almost every software product consists of discrete components, such as:

- User interfaces, forms and dialogues
- Programs or procedures implementing the desired functionality
- Data bases or other data organizations (e.g., file-oriented data org.)
- Interprocess communication and exchange mechanisms
- A system base (HW/OS) providing the environment for the software to run in

A clear and detailed assignment of subsystems and components into these or similar categories allows a safer estimation of the development resources needed. (One could argue that such a separation also produces software that is more reliable and easier to maintain, but that is beyond the scope of this paper.) Once the deliverables or components thereof are identified, they serve as a basis to break down the work into units. This is achieved by applying the software life cycle's activities to the planned system, subsystems and components. Some components may pass through all phases, but some only one, for example, a reimplementation of a module followed by an integration test.

The number of work units (and their nature) is the "product" of the components "times" the activities applied to them.

Neither the software life cycle nor the functional breakdown of the system under development leads to a schedule. All that is known at this point are the dependencies: i.e., what has to be developed first, or which deliverables must be finished before others begin. The next step is estimation of the time and resources necessary to complete each work unit. Different estimation techniques are, of course, required for the different work units. An estimate for developing user interface components using rapid prototyping tools must be calculated in a different way then a data base design or the programming of an application function.

The work units can now be placed into a schedule, using the resource estimates which have been produced. The schedule can be expressed as a bar chart or PERT. It is now possible to define start and termination dates for the project, and perhaps some intermediate milestones if necessary.

Only one metric is used to measure the progress of a project: the amount of deliverables delivered. That's what management by objectives is all about:

- to ask for certain concrete objects to be delivered

- to precisely specify the criteria which these objects must fulfill

- to communicate clearly, understand and agree on the functionality, delivery criteria and resource estimates of the objects to be delivered

- o control only the deliverables, not the state, they are in or the activities or approaches taken to achieve them

The project management framework described above extends the notion of he software life cycle into a work unit identification and estimation process.

THE PROJECT MANAGEMENT PROCESS

The secret of good project management is active measurement. As Tom De Marco says, "You can't control what you can't measure." But a typical software project doesn't lend itself well to easy measurement. Typical life cycles phases like Requirements, Specification, Design and Coding aren't fine enough to allow for management control of results achieved and resources used.

Software Life Cycles do not provide project managers with estimation techniques to estimate project resource or measures to monitor a project's progress. Information like "to have finished" detailed design of a project or to have 70% code complete are not clear measures or indications of a project's state.

The solution to the problem can be stated in a few easily remembered sentences:

1. Divide a project into (groups of) objectives, whose achievement can be measured by objects delivered.

2. Check which activities of the life cycle are needed to create the deliverables.

3. Describe work units in terms of life cycle activities per deliverable, indicating objectives, approach and form of deliverable.

4. Put work units on schedule according to their interdependencies, so that you get a chain of events for deliverables.

5. Check only completed deliverables.

This project management framework provides a way to support more detailed project management and control.

In the traditional waterfall model for a software life cycle, general deliverables are identified for entire projects and are all due at certain checkpoints, e.g.

- Requirements Definitions
- External Specifications
- High Level Designs (System Design)
- Detailed Designs (Program or Module Design)
- Implementation and Coding
- Integration and Quality Assurance

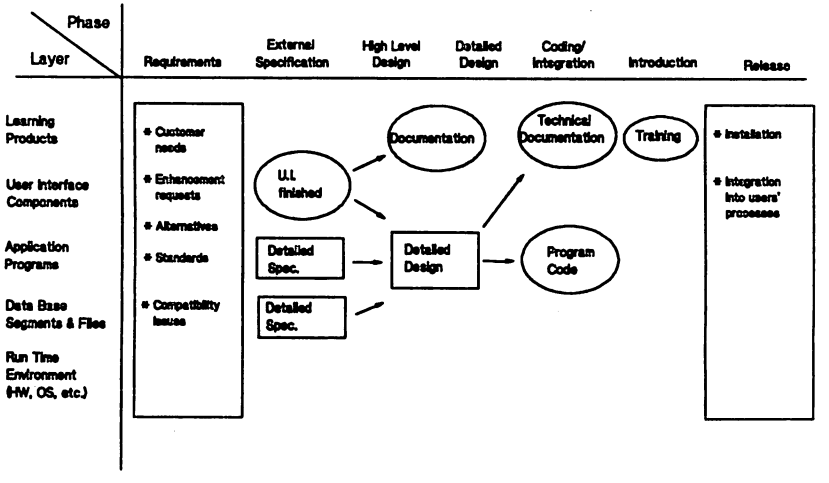
However, in reality, the specific deliverables for a single checkpoint are finished over a range of times during the life cycle phase. Naturally, the engineers don't sit on their hands while waiting for a checkpoint meeting. They usually continue with work associated with the next phase. But project schedules using the traditional waterfall model may not take this into account.

Basically life cycle phases are fine for "taking the pulse" of a whole project, but are not nearly detailed enough for actual task definition and coordination. A life cycle suggests a logical sequence of operations to apply to an object. It is not a prescription for the scheduling of project activities. A life cycle also doesn't help to calculate the manpower needed for a project's individual work units. This detail can be supplied by a more structured project management framework.

An information system can be divided conceptually into a number of layers, including:

- Learning products and documentation
- User interfaces including dialogues
- Application or system programs
- Data base segments and file definitions
- Run time and/or development environments

Layered Model vs. Life Cycle



Software systems usually consist of modules of all or some of these categories. The best designs usually result from actually (not just conceptually) separating these layers. For example using rapid prototyping or graphical user interface tools implies separating user interface software components from application software modules. This leads to cleaner design and to portable, reusable code.

The separation into layers also simplifies the definition of which (and how many) modules or objects have to be developed. The following example illustrates this:

A certain functional design consists of 10 on-line transactions and 5 batch processes. Each on-line module requires a dialogue to gather data. The batch processes use one single dialogue to start them and to display their output data. This means 6 dialogue modules have to be developed. The entire application can be based on 3 relational data base tables.

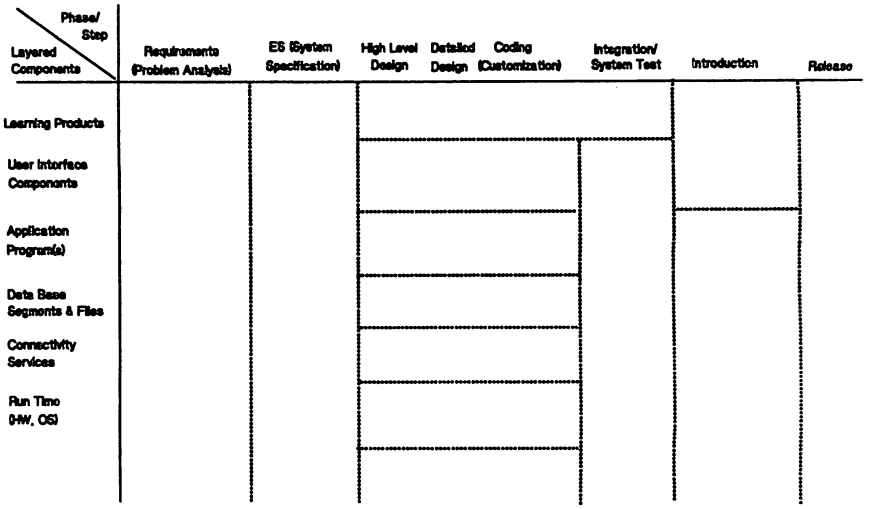
For each of the identified objects different estimation and calculation methods must be applied. A dialogue design and implementation is different from the highly complex calculations required for application functionality and is also different from data base design.

In addition, separating the product into layers allows the manager to separate the work units. Engineers with special qualifications (e.g. UI experts) can be called upon to design or implement those portions which use their expertise. It is also easier to add or subtract engineers from the project when the work units are small and well defined.

When the different layers of a system are carefully separated, they each proceed through the life cycle somewhat independently. Of course, there are a number of dependencies between layers which must be accounted for, but a fair amount of parallelism is still possible. The figure entitled "Layered Model vs. Life Cycle" shows this. The next step is to actually divide these deliverables into work units for individual engineers or teams.

We suggest doing this by taking the software life cycle as one axis of a project customization matrix, as shown in the "Layered Model vs. Life Cycle" chart. The vertical axis is made from a list of a software system's modules, layered into the different categories. Then each of the modules or deliverables, like a complete product's Requirements Specification (a document to be delivered), is run through one or more phases or activities. Each activity applied to a deliverable constitutes a work unit.

Work Unit Matrix



The figure "Work Unit Matrix" shows a prototype for "blocking out" the work of an entire project into work units.

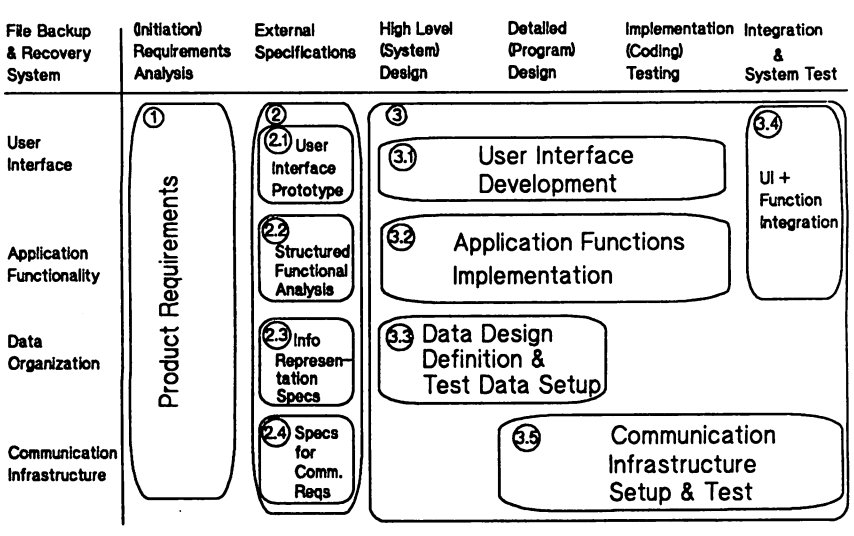
Note that the Requirements phase is seen as one single (large!) work unit. It can certainly be subdivided into component parts, but needs to be finished as a unit before too much effort is expended on the specifications. Similarly, freezing of the specifications needs to be largely completed before entering the design phase in any layer.

This figure is only a prototype. Some projects can certainly be arranged to allow more parallelism in the first two phases, and this is recommended if practical. (The algorithm for making this determination is unfortunately not simple.)

In our prototype, the High Level Design, Detailed Design and Coding phases of each layer are grouped together. They can be developed in parallel with each other, allowing maximum flexibility in scheduling resources. The Integration Phase is shown as a unit, although it is often easy to plan a phased integration in many projects.

For the introduction of an information system into a user environment, it makes sense to see the integration of the user interface into existing user organization as one unit, and the integration of the programs, database, etc. into the EDP operation as another. This may not be applicable to introduction of a software product to the field marketing organization.

Now that we have identified the basic "system level" work units, we need to descend one level and identify the individual tasks in each work unit. An example of this is shown in the figure "Customization of Work Units". The example is based on a simple file backup & recovery system.



We begin with the Product Requirements specification, which is shown as work unit #1. It must be produced as a single unit for all layers, as described previously. The requirements for the example include file backup and recovery to and from removable media, such as diskettes or tape cassettes. Three different selection functions should be offered to backup or restore files:

1) by file or path name to identify single files or groups

2) by attributes like owner names or time stamps on files to select them

3) by predefined script or schedule.

The system is required to execute on multiuser systems with terminals as well as diskless PCs or workstations in a client/server environment. In the latter case the user interface, including all dialogue, should be executed by the workstation, but the backup and recovery function should remain on the server. Log files are required to keep track of what files having what attributes are on what media. Script files written in some specification language are needed for scheduled operations.

The "External Specification" phase, which also involves activity across all layers, can be subdivided into detailed and partially independent work units. The communication infrastructure to transmit operational commands from a user interface on a workstation to a server (2.4) may be specified independently from the structured functional analysis (2.2). The latter is a prerequisite to specify user interfaces or information representations, e.g. for the scripts.

After step 2. and/or phase 2, (External Specifications), we are in position to set up more detailed lists of modules to develop. Now we can freeze the specifications and execute a detailed calculation, planning and scheduling.

Step 1. divides the project into objectives to create the modules or software objects:

3. Backup & Recovery System

3.1. User Interface

3.1.1 Final Dialog & UI Specs

3.1.2 Main Dialog

3.1.3 Dialog for "Path-name" module

3.1.4 Dialog for "Attributes" module

3.1.5 Dialog for "Scheduled" module

3.2 Application Functions

3.2.1 Path-name-directed Backup & Recovery

3.2.2 Attribute-directed Backup & Recovery

3.2.3 Script-directed Backup & Recovery

3.3 Data Design

3.3.1 Data Value Definitions

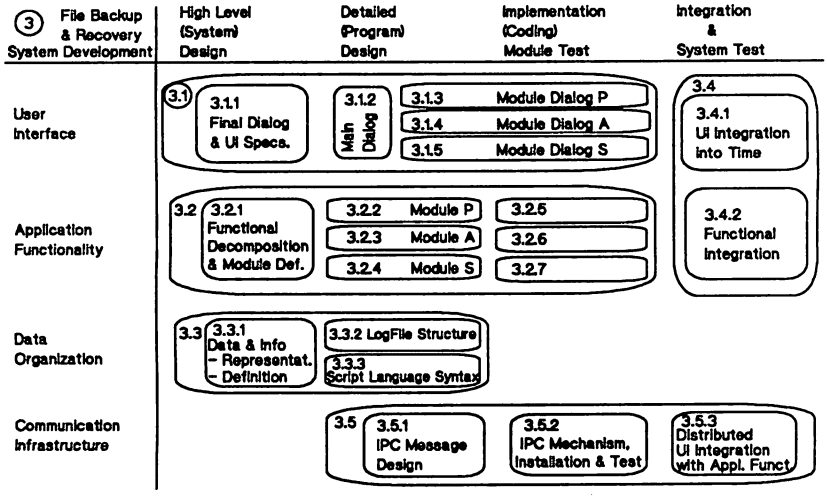
3.3.2 Log File Structure (Records)

3.3.3 Script Language Syntax

etc.

The following chart "Customization of Work Units (Expanded View)" shows an expanded view of work unit 3.

Customization of Work Units (Expanded View)



The expanded view has been produced by applying step 2 of the Project Management process. The software life cycle is applied to the list of objects that must be developed. This identifies all work units as shown in the expanded customization matrix.

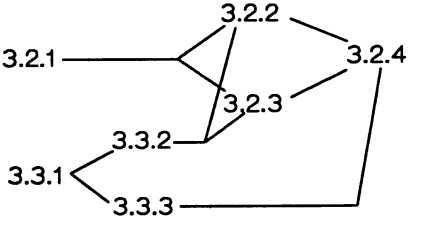
Step 3 is to write work unit descriptions in the form of HOSHIN-like cascades of assignments. These are written for the whole project, its work units and tasks. The following form could be used:

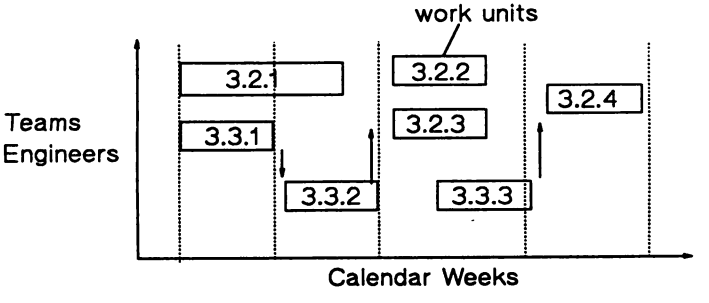
CSB - R & D PROJECT (WORK UNIT) DEFINITION SHEET		
Project:	Revision:	Date:
Project Title:	Reason for Rev.:	
Project Start Date:		
Project Deadline:		
Project		
Objectives: in terms of functionality required to achieve product goals or functionality to be realized by a purchased SW product		
Approach: in terms of technology used and design chosen to meet the goals/objectives		
Deliverables: describe objects to exchange results, such as documents, program code, media, e.g. SW system in source and object code executable on ... that implements functionality as described in "objectives"		
Manpower Allocation (man weeks):		
Other resources required:		
Prerequisite results: Work units results are prerequisite for:		

Each work unit must be described in terms of objectives, approach and deliverables. That parallels HOSHIN's goals, strategies and metrics. Indeed, only the deliverables should be used to control the project's progress. Anything else is less concrete, unsure, and leaves the project manager in shifting sands.

Step 4 depends on Step 3's definitions of what work unit results are required for other work units and on the allocation of manpower. Both control the scheduling, which is done by putting the work units from the customization table into a PERT or bar chart. Scheduling is not defined and also not controllable with a life cycle, but only by prerequisite dependencies and by manpower calculated and/or available.

Dependency Graph / Schedule





For the example of a backup and recovery system, a detailed examination of the project plan shows that:

3.2.1 Functional Decomposition and module definition must precede:

3.2.2 Module for path name operation and
3.2.3 Module for attribute-directed operation

Both 3.2.2 and 3.2.3 are prerequisites for

3.2.4 scheduled backup based either on path name or attributes.

3.3.2 comes before 3.2.2 and 3.2.3, because both functions require the definition of the log file record structure to record where backed-up elements are on media.

3.3.1 is before 3.3.2, since in 3.3.1 one designs record structures together with a syntactical description of a language to express backup scripts.

3.3.3 describes syntax for the input to function 3.2.4. Therefore it is prerequisite of 3.2.4.

All these conditions may be assembled into a dependency graph, which looks very much like a PERT chart. (See "Dependency Graph/ Schedule")

Whether to use a PERT system or a simple project calendar showing bar charts is a matter of convenience. It is often sufficient to draw a bar chart like the one above. The arrows are used to signal dependencies between work units.

The project management process outlined above leads to more precisely defined projects. The project plans profit from more degrees of freedom in scheduling. The HOSHIN-like cascade of work unit definitions allows for management by objectives, since it offers project control by deliverables. Concrete, countable and presentable deliverables are necessary for good project control and progress measurement.

(Paper no. 2035)
Improving HP-UX for OLTP

by

Roland N. Luk

Hewlett-Packard Co.
General Systems Division
19490 Homestead Road
Cupertino, CA 95014
(408) 447-1004

1. Introduction

The success of Unix in the engineering application environment is well established. Running mission-critical, on-line transaction processing applications on Unix has become more attractive because of the promise of standards-based solutions and price/performance benefits. As a result of the tremendous demand and appeal to run business applications on Unix, system vendors such as HP are involved in making Unix more robust for OLTP by tuning the operating system, and integrating key technologies and solutions which have only existed in proprietary systems in the past.

Running OLTP applications successfully on Unix requires a hard look at how the system vendor can address basic issues such as performance, high availability, data integrity, and security. In addition to these requirements, there needs to be additional technologies and tools to support application development and system management. Increasingly, there is a need to execute transactions in a client/server distributed computing model.

This paper will examine how HP is addressing the needs of this growing market, and how this market is rapidly evolving towards new and exciting technologies.

2. OLTP Market Evolution

The nature of business computing has drastically changed since the days of batch processing where data requests were collected together, converted into a machine-readable format, and then brought to a central computer for collected processing. Reports resulting from the batch processing reflect the most current information up to the last batch job that was performed. On-line transaction processing brings the end user closer to the computing environment, allowing data to be entered via terminals and processed immediately.

Large mainframe proprietary environments were the primary and only means of batch and on-line business computing during the 1960s and 70s. In the late

1980s, a complement of variables such as acceptance of RISC computing, attractive price/performance of Unix based solutions, maturity of Unix and RDBMS technologies to meet OLTP challenges, as well as the momentum towards open systems have all contributed to the rapidly growing Unix OLTP market. During the 1990s, new technologies in the area of distributed OLTP, client/server, graphical user interface-based tools, and interoperability at the software transaction level will emerge and mature. Figure 2.1 below depicts some of the key OLTP industry trends:

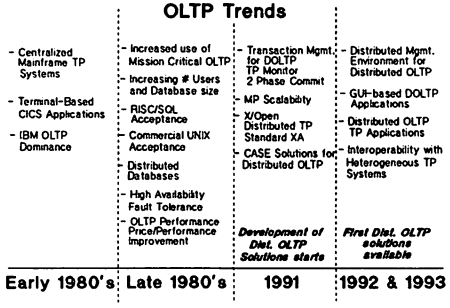


Fig. 2.1

3. Unix OLTP Market Size

One of the biggest growth opportunities for Unix applications is in OLTP. Figure 3.1 shows the overall Unix OLTP market size from 1990 to 1994. The rapid growth rate has influenced a lot of hardware and software vendors to jump on the bandwagon to deliver OLTP solutions on Unix.

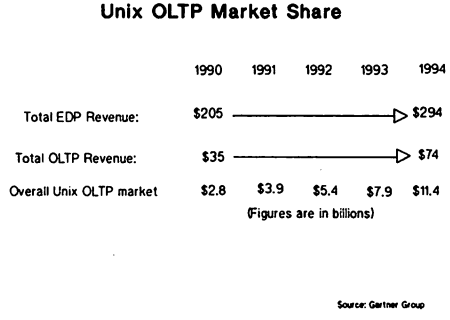


Fig. 3.1

4. OLTP Requirements

Individual customers have varying OLTP needs depending on the size and nature of the business operations. Some of the most important criteria which OLTP vendors are measured against can be shown in the next figure:

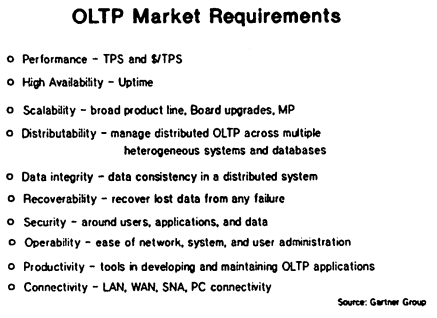


Fig. 4.1

a) Performance and Price/performance - Both H/W and RDBMS vendors are increasingly being asked to show OLTP system throughput based on "standard" benchmarks. In addition, the total cost of the system which includes a 5 year cost of ownership are being analyzed closely by MIS decision makers. The latest industry standard benchmarks for OLTP are the TPC (Transaction Processing Council) benchmarks which include the TPC-A and TPC-B benchmarks. TPC-A exercises the system components necessary to support OLTP environments characterized by multiple on-line terminal sessions and significant disk input/output. TPC-B exercises the database components in OLTP environments characterized by significant disk I/O, and it is a batch vs. an OLTP benchmark in that no terminals, networking, or think-time are included. The metrics reported by TPC-A and TPC-B include throughput expressed in transactions per second as well as the associated price($)/TPS.

b) High Availability - Mission critical applications require absolute data integrity and no downtime (both planned and unplanned). In general, customers are concerned with limiting the maximum length of time that a system is down.

c) Scalability - Customers want their investment protected and would like to see a highly flexible set of upgrades at minimum cost through board upgrades or added expansion through MP configurations.

d) Distributability - As RDBMS technology evolves and customer environments become more complex and distributed, there is a need to manage transactions across multiple distributed sites.

e) Integrity - Distributed systems over multiple sites and departments need facilities to insure that no lost transactions occur causing inconsistencies in the various databases.

f) Recoverability - As failures occur, either hardware or software, the system including the databases must be rapidly restored to the most stable point before the failure.

g) Security - Data is a valuable asset. It must be secure in today's increasingly networked system environments.

h) Productivity - OLTP systems will require sophisticated application development tools such as 4GLs and CASE to support new application development as well as helping existing applications evolve towards new technologies or models (e.g. client/server, object oriented solutions, etc.).

i) Operability - Client/server networked environments will require networked-based system management tools for OLTP environments.

j) Connectivity - Corporate mainframe and workstation/PC connectivity will be stressed such that LANs, OSI, and IBM connectivity solutions will require increasing integration.

5. Hewlett-Packard's Strategy

The weaknesses of Unix for OLTP are being addressed by various OLTP vendors. Criticisms in the past have focused on reliability, the Unix file system, data integrity, as well as process and memory management. In this section, we will discover how HP-UX has been successful in meeting these requirements. The issues of security, operability, and networking are not covered here, although HP has published separate white papers to articulate our strategies for these three important topics.

5.1 Performance and Price/Performance

Performance and price/performance are two fundamental metrics used to measure system vendors. The trend in OLTP is to move the system closer to the actual users which means a greater demand for fast on-line data access. Because the volume of these transactions are usually large, response times between transactions is also important. Buyers are also concerned about the cost of these systems. Hence, vendors (PCs to Mainframe) will usually quote a $/TPS number (the lower, the better) along with a TPS figure (the higher, the better).

HP, along with its database partners have provided significant tuning and enhancements in HP-UX and the databases to yield one of the industry's leading TPS and $/TPS. Some of the operating system changes HP has made have yielded up to 50% improvement in performance. Figure 5.1.1 below shows that traditional barriers for Unix in OLTP are increasingly being addressed:

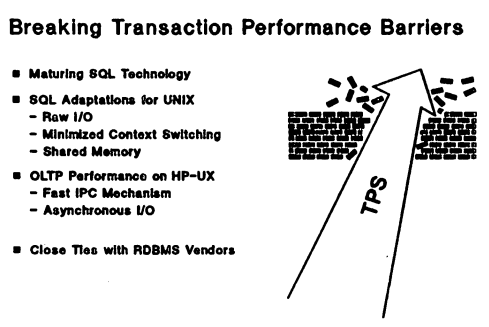


Fig. 5.1.1

One of the characteristics of OLTP is the heavy use of IPC (inter-process communication) which to the operating system means a lot of process context switching. Some of the database vendors such as Oracle and Informix use a 2 process/user architecture, and IPC calls are being made constantly between the two processes. As a result, a faster IPC mechanism which optimizes the number of context switches is available from HP to those 3rd party software vendors who can benefit from this HP-UX feature.

Another feature of the OS which is included in HP-UX is asynchronous or non-blocking I/O to the disk. This basically allows a process to continue executing while the I/O is being processed. The asynchronous I/O feature includes the ability to get notification after the I/O is completed. This feature is important for Sybase, for example, which uses a single back-end process for multiple front-end processes. The asynchronous I/O feature prevents the single back-end process from blocking the other front-end processes that are trying to gain access to it.

A criticism of Unix is the limitation of the file system for OLTP. OLTP applications need fast access to large files randomly, and the Unix file system is designed to access mostly small files. What is important here is that most of the major database vendors have bypassed the Unix file system and have essentially created their own file system through Unix's raw I/O disk feature. This allows

the flexibility for database vendors to take full control of buffer allocation schemes and disc I/O algorithms.

Database vendors are also tuning and enhancing their SQL products to include features such as multi-threaded server, disk I/O optimizations, and pre-compiled procedures to enhance CPU utilization. RDBMS vendors are doing these optimizations because they feel that it is not the relational model, but the implementation of it, that prevents them from delivering high-volume transaction performance. Multi-threaded database servers such as the one provided by Ingres allows each back-end server to process multiple front-end client processes thereby reducing the number of processes created. This is quite different from previous vendors' architectures which impose a two process model for each user.

Disk I/O optimizations used by major RDBMS vendors include things like multi-volume tables, group commit, and deferred writes. Multi-volume tables allow a relational table to be transparently partitioned or "striped" across multiple disk volumes, resulting in the increased utilization of the available disk I/O bandwidth. Group commit eliminates log file bottlenecks. A log file is used to ensure transaction data integrity. Every transaction commit will normally generate one I/O to the log file. With group commit, the log information from different transactions is grouped together and flushed in a single I/O.

Deferred writes allow a database server to commit transactions without writing the changed data to the disk immediately. The frequently changed data can remain in memory and be flushed at a later time. Data integrity is maintained with deferred writes because the transaction log is flushed during commit time.

Compiled database procedures are used by almost all database vendors. Database procedures are SQL commands that are grouped together, pre-compiled and stored in the database, thereby reducing context switching and IPC time. These optimizations are only a small sample of how database vendors are reducing operating system overhead.

As a result of the tuning and enhancements, it is becoming apparent that Unix can match the performance of higher-priced proprietary systems. HP was the first system vendor to publish TPC-A results on Unix, and the results are shown here in figures 5.1.2 and 5.1.3:

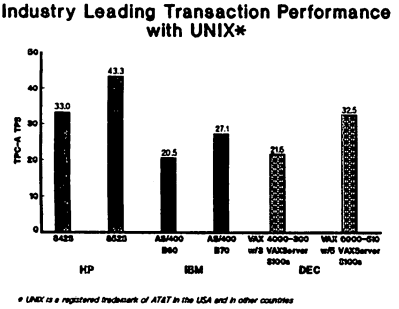


Fig. 5.1.2

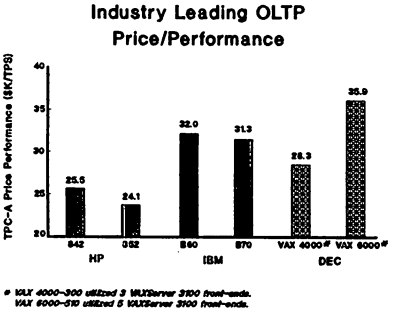


Fig. 5.1.3

5.2 High Availability

There is an increasing demand in the OLTP market for highly available systems to run mission-critical applications. This demand is particularly strong in the manufacturing and telecommunication industries. The key features under high availability, which customers need, include data integrity, no planned downtime, and minimal unplanned downtime.

Data integrity has to do with the need to maintain consistent and durable data even after failure occurs. Planned downtime is the time the computer is unavailable for doing scheduled operations such as preventive maintenance, disk backups, and hardware and software updates. Unplanned downtime is the time a system is unavailable because of a system failure.

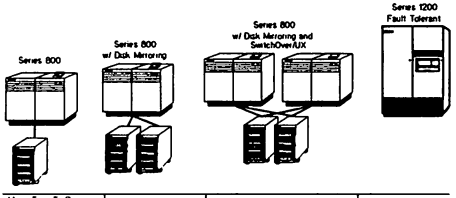
The system availability needs will vary depending on the cost associated with lost time to perform the business transactions due to the system being down. Service oriented firms such as airline or hotel reservation systems and telephone services will require 24 hours by 7 days availability. Other companies have less stringent requirements.

HP's high availability strategy offers a spectrum of solutions at different price points ranging from highly reliable standalone systems all the way to the Sequoia fault tolerant system which HP is reselling.

HP's standard systems and disk drives already achieve 99% reliability with very little or no preventive maintenance needed on the disk drives. HP has also included extra features such as power-fail recovery and battery backup as part of its standard HP-UX systems. Disc mirroring (Datapair) protects users from storage access problems by maintaining two identical copies of the data on two disks. The Datapair product can mirror any disk partitions including the root and swap devices. A key benefit of disk mirroring is the ability to do on-line backups.

HP is also adding new features such as processor fail-over and hardened file system. Processor fail-over allows a standby processor to back up one or more primary processors in a loosely-coupled network. The hardened file system allows a file system to maintain higher reliability such that the reboot time involved in file system checking is greatly reduced. Figure 5.2 below depicts the flexible set of solutions available from HP:

A Full Range of High Availability Solutions



Mean Time To Recovery: 4-24 Hours	0-24 Hours	0-25 Minutes	0-10 Seconds
Data Integrity: Last Back-Up	Last Committed Transaction	Last Committed Transaction	No Data Loss *
System Price: 1X	0X-2X	0X-2X	2X-3X

* No data loss after reaching main memory

Fig. 5.2

5.3 Scalability

OLTP customers want the flexibility of expandable systems with wide performance ranges, including the scalability provided by

high-end multi-processor systems. Customers want their hardware and software investments protected which translates to providing a scalable growth path with portability in mind.

HP has been successful in delivering a broad family of OLTP systems based on its PA-RISC architecture. HP continues to produce more powerful processors, so powerful that high-end performing processors are being built for mid-range packages. An example of this is the 842S and 852S systems which are the latest mid-range packages performing at the rate of 1.4 and 1.8 times an 855 at 50% to 75% the cost of an 855S, respectively. More aggressive $/TPS boxes are expected to come out late summer, 1991. Figure 5.3.1 shows the current S800 product line family of business servers and the TPS (TPC-A) range:

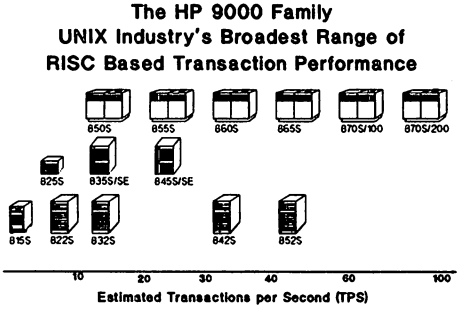


Fig. 5.3.1

5.4 Distributability, Data Integrity, Recoverability

In the 1990s, we will see networked heterogeneous systems become a reality. Users will want to perform and manage distributed transactions across these systems transparently. Data integrity and recoverability issues become more complicated in a distributed environment.

Networked systems include client/server configurations with intelligent workstations such as MS/DOS, OS/2 PCs and Unix workstations. Distributed transaction processing is the future trend in OLTP and key technologies for distributed computing are appearing on Unix.

5.4.1 Transaction Processing Monitor

One of these key technologies needed to support distributed OLTP on Unix is a transaction monitor. A TP monitor handles many of the

same tasks that an operating system does such as scheduling of resources and managing user requests. In networked environments, the TP monitor can also direct database transaction requests to an idle processor which enhances the capacity of an OLTP application.

TP monitors are more cost-effective in high-volume transaction environments with more than 100 users, and most Unix OLTP systems are currently limited to about 100-200 users on the high-end. Although Unix OLTP systems can handle 100-200 users without difficulty, putting more users beyond 200 requires additional software features and tuning. An OLTP monitor, through its ability to efficiently schedule and manage transaction requests and services, promises to deliver more users for the same hardware configuration. TP monitors which extend the client/server model across the network and off-load screen or forms processing to local workstations are expected to deliver on that promise.

Another key benefit of a TP monitor is the ability to coordinate transactions in a distributed OLTP computing environment. The two key features which enable this are transaction logging and a protocol called two-phase commit. A transaction log is used to store the status of a transaction's progress and the state of systems involved.

Two-phase commit allows a TP monitor to tell all involved systems to prepare to commit their transaction parts. If all of the systems respond affirmatively by saying that they are ready to commit, only then will the TP monitor send the global commit instruction. Otherwise, an abort message is sent to all participating sites, thereby insuring data integrity in a distributed manner.

To help enable recovery after failures, most TP monitors include a log which stores the history of a system's operation which can, in turn, be used to reconstruct a system after a failure. Another point about recovery is when a transaction fails, a TP monitor can detect the system failure and notify the application of origin. It is then up to the application to abort or retry the transaction. Besides these OLTP run-time execution features, a TP monitor usually includes software development tools to help develop these applications as well as an administrative support environment to help users install, configure, monitor, and manage these systems.

Fig. 5.4.1 shows an example of a global transaction which consists of sub-transactions that are executed in a distributed, heterogeneous environment:

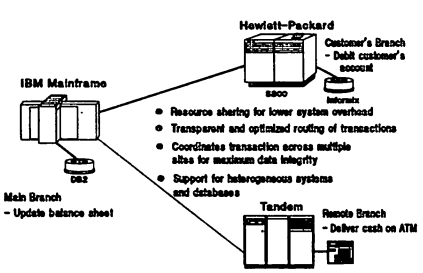


Fig. 5.4.1

HP's strategy is to extend its OLTP leadership by using Transarc Corporation's OLTP monitor technology to evolve its open, OLTP platforms to a distributed, client/server environment. Transarc Corporation is based in Pittsburgh, PA. and has been providing systems software for local and wide-area networks of distributed computers. Transarc's distributed file system, the Andrew File System has been well received by the market as well as OSF which has chosen it to be part of the Distributed Computing Environment (DCE) request for technology. The reasons why HP has chosen Transarc's technology are that it is:

a) Built on top of HP's NCS RPC technology, and is consistent with HP's strategy to use DCE as the framework.

b) Compliant with X/Open's distributed transaction model and the XA interface standard (XA is described in the next section).

c) Functionally rich and robust.

d) Architecturally modular which allows for portability and extensions.

e) Key endorsements from other major OLTP players (e.g. IBM, Informix, Sybase, etc.) makes it a pervasive defacto standard.

HP has OEMed Transarc's technology and will be integrating it with HP-UX by 1st half of 1992. HP has also recognized the need to supply AT&T's Tuxedo TP monitor especially for its telecommunications customers. As a result, HP is delivering the Tuxedo TP monitor through a third party by the name of Independence Technologies, Inc.

5.4.2 Emerging OLTP Standards

Distributing the execution of transactions across a heterogeneous set of platforms is a difficult task since the environment needs standardization and cooperation. Open OLTP will require that a set of common APIs be defined and implemented by all. X/OPEN has defined an OLTP model which consists of basically three functional components - application, transaction manager, and resource manager. Resource manager includes RDBMS, file systems, and print services. Applications and transaction managers work together to request resource managers for a certain task. A transaction manager includes most of the TP monitor features discussed earlier plus some communication services for an application to access. However, the approach X/OPEN might take on communication services is to break up the communication services to another module such that an application can deal directly with the communication services without going through the transaction manager. One of the first interfaces under definition is the XA interface which is the interface between the transaction manager and resource manager. A formal XA proposal has been distributed to various X/OPEN companies and users for review, and it is expected to be finalized by fall of 1991. Figure 5.4.2 below shows the X/OPEN OLTP model:

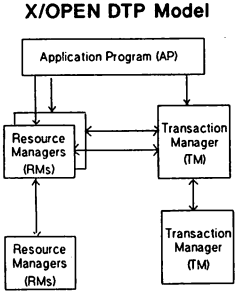


Fig. 5.4.2

Some of the key interfaces being defined are the XA interface described earlier, the ISO RDA (Remote Database Access), and the ISO DTP protocol which defines the interface between two TP monitors cooperating in a single transaction. In addition, X/OPEN expects to tackle the AP/TM interface in the near future. All these various efforts are being undertaken to deliver users of DTP systems with the following ultimate benefits:

a) Interoperability - the ability to have transaction programs which operate on several different RDBMS's which may be on different sites.

b) Application portability - the ability to move applications to different systems easily.

c) Interchangeability - the ability to substitute databases without major rewrite efforts for OLTP programs.

5.5 Productivity

OLTP systems will get more complicated as new technologies emerge and begin to be integrated with existing or new applications. One of the biggest consideration when purchasing an OLTP system is the availability of software development tools. Customers are faced today with issues such as application backlogs, poor quality software, large maintenance efforts, and migration problems as they move code from one system to another. Customers will also start facing new challenges such as development of client/server and distributed OLTP applications.

Unix has been well known as a programmer's ideal operating system because of the richness and flexibility of the tools it provides. The only problem with the tools is that not all programmers find Unix tools particularly easy to use. Unix tools tend to be point tools and they are not very well integrated. Unix has also been criticized for delivering tools for the technical environment. Commercial environments tend to have tools such as database design and generation, automated code generation, 4GLs, report writer, etc. Finally Unix has not been well known for having an integrated set of tools for the production and maintenance of large software systems with active software project management and control activities.

The CASE (Computer-Aided Software Engineering) business has been on the rise lately. CASE promises to deliver an integrated set of tools to automate tasks across the entire software development life cycle. It promises to increase development productivity, quality of software, and reduce maintenance cost. IBM gave a lot of credibility to the CASE business when they introduced their AD/Cycle CASE strategy last year. AD/Cycle's strategy is to supply a central repository (data dictionary containing information about data, relationships, etc.) to help management, database administrators, systems analyst, and programmers effectively plan, utilize, and control the company's data based on the MVS Repository Manager as well as various tools supplied by their key partners (e.g. KnowledgeWare, Index Technology, etc.) to enable the full software development life-cycle. The only glitch so far for IBM is that its repository is still immature, and analysts have stated that it will be about 2 more years before AD/Cycle can be useful for end users. In the meantime, the market is flooded with CASE solutions from third parties who are targeting not only proprietary systems but also open systems.

HP's view on software development toolset needs on Unix is that customers will choose their toolset depending on two major variables. The first consideration is the flexibility or choices of combining tools from different sources, and the other is the degree of integration needed as well as the project size. An implied factor here is that the less integrated toolset will demand a lower price in the market. Fig. 5.5.1 shows this varying set of commercial CASE solutions:

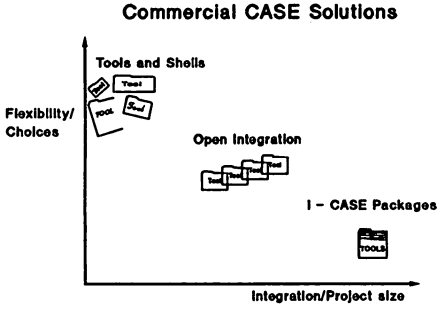


Fig. 5.5.1

There are tradeoffs to consider when choosing the tools and shells solution versus the integrated CASE (I-CASE) solution. Tools and shells are less costly but deliver less productivity because of the low degree of integration and usability. I-CASE solutions from third parties offer higher productivity at a higher cost, and they usually involve investing in a vendor's proprietary solution.

HP's strategy for software development is to offer a portfolio of solutions under its CASEdge program to help end users and VABs bring existing code and support new code development for HP-UX.

HP has its own CASE solution called HP Softbench which delivers an open integration framework such that other tools can be included. It also supports tool communication and a consistent standard user interface based on OSF Motif. In addition, HP has develop strategic relationships with some of the world's leading CASE vendors who are aggressively supplying solutions for the open systems market. The vendors announced so far include CGI, Softlab, and Texas Instruments. Other CASE offerings come from HP's premier 3rd party RDBMS vendors such as Oracle, Informix, and Ingres. Figure 5.5.2 illustrates the solution set available from HP so far:

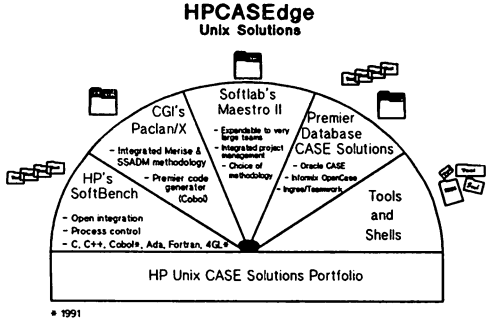


Fig. 5.5.2

HP has also initiated discussions with several CASE vendors to address the emerging client/server OLTP environment. In fact, vendors such as TI, Softlab, CGI, and PowerSoft are beginning to address this need, and HP will be working with them to deliver the solutions.

6. Conclusion

Unix OLTP will be maturing in the next 2 years as new technologies mature and are integrated. Even though vendors and users are still struggling with how to build bigger, more powerful, distributed OLTP systems, the key ingredients for success are appearing quickly.

Clearly, standards in OLTP will need final definition, acceptance, and implementation. Further optimization and integration of solutions will also need to happen. The good news is there is no shortage of support from users, vendors, and standards groups to make Unix a viable alternative OLTP solution to costly, proprietary mainframes.

Fig. 1.2

Troubleshooting FORTRAN on Multiple Platforms

Helen K. Morimoto
Hewlett-Packard Company
Computer Language Operation
19447 Pruneridge Avenue MailStop:47UB
Cupertino, California 95014
(408) 447-5424

Abstract

FORTRAN migration from non-HP computer systems to HP machines can be more difficult when there are many unknowns. Traveling the road from a land where the terrain is known to an area where things are unfamiliar can make one feel unprepared. Most of the problems encountered have been experienced by others previously migrating in the same direction and the lessons they learned can prove very helpful. Those lessons can also prevent unnecessary pitfalls in the migration process. Migration here can mean going from one series to another on the HP9000.

My job at Hewlett Packard is to receive calls from those helping with FORTRAN migrations and those helping with FORTRAN consulting. This paper reviews recommendations for planning, suggested compiler options and directives for compiling, suggested tools for ensuring expected execution, and an explanation of optimization features for fine tuning.

PREFACE

I am one of thousands of Online engineers at Hewlett-Packard who answers calls for a living. Sometimes we feel like glorified telephone operators when we dispense information by reading from a manual and provide 'directional' assistance. Other times we feel like we have saved a soul from near destruction when an SE is onsite at a critical defense department and claims to be held hostage unless we bail him out with a patch or fix for a problem to him yesterday.

The job is interrupt driven and intense at times. I've worked this beat 8 years (in FORTRAN) in Online at Hewlett Packard and have coded in FORTRAN for over 20 years.

This feeling that its an interesting job and that someone has to do it motivates us to continue. We can be called troubleshooters. What is a troubleshooter anyway? When all else fails and we are lost for words, let's go to the dictionary. The term means "a person who consciously or unconsciously causes trouble."

That's Online -- we unintentionally cause trouble by bucking the status quo and championing some esoteric features that the customer needs in every other line of his code or would save a customer from recoding most of his sources because of an extension to the standard that is not there. It's a hard life and sometimes we are not on the side of the customer but on the side of R&D because it is a Standard that would break everyone else's code just to please one customer. We like to fight for truth, justice, and the ... OOps -- the definition above was for 'troublemaker' which is next to 'troubleshooter'in the dictionary (how interesting). Webster says, " one who resolves diplomatic or political disputes, mediator of disputes that are at an impasse." Well, we don't often come to that but we get pretty close to that kind of action when we have to deal with issues that design conflicts with industry accepted practice or priorities and schedules.

DEFINITION OF CUSTOMER

The customers can be the Response Center engineers whose job is to get the calls from customers who purchase support service through their company, systems engineers who go on-site to companies, and lab engineers who work on software and hardware for Hewlett-Packard to sell. The situation can be a development project, a presales meeting, a port, or a migration. A migration and a port are different: A port is a movement from one platform to another. A migration is a translation from one language to another or from one version of the operating system to another.

CALL TYPES

When a call comes in, it is usually in one of four areas:

1. Planning
2. Compilation
3. Execution
4. Fine Tuning

The first area is essential to pre-sales situations where there are questions and problems that must be answered to make the sale of the software and/or hardware. The other areas are the logical stages of development and porting/migration.

BASIC REQUIRED INFORMATION

Regardless of the customer, application, or problem, the most important information that is required at the beginning of the communication between the Online engineer and the person on the other end of the phone is what product(s) is involved, what platform, what version number, and a description of the problem, symptom, or request.

What is said and done next depends on obtaining the "BASIC REQUIRED INFORMATION" and determining in which of the four areas the need is. A discussion on what steps to take, a review of existing errors that have lately been uncovered and fixed, or a general/specific way to trim down code to exhibit the problem in a more efficient manner are some of the approaches that can be used.

PLANNING

In planning, the most often asked questions are "What's new?", "What bugs have been reported lately?", "What kind of problems can I expect when I go from ... to ...?" and "Will I have problems with ... (feature, version, and so on)?".

I strongly recommend the following steps when planning a migration, port, or new development on an unfamiliar HP9000 system.

1. Line up your documentation resources in hardcopy, or ensure that the information is on the system. The necessary information and resources are man pages, documentation, release notes, Software Status Bulletin (SSB), Software Release Bulletin (SRB) and any pertinent white papers or portability guides.

2. Review the Operating System (O/S) changes or differences in the file system,signal handling, trap management, memory management, process management, scheduling, I/O. Simple things like looking at the logical unit that are the default for FORTRAN (some are 1 and 3 while others are 5 and 6) for Standard input and standard output.

3. Gather and understand the tools that are available for use. The tools you are most likely to use are 'flint' or 'lintfor' (the names are different but the tools are the same. 'lintfor'/ 'flint' detects many useful pieces of information about your application. The utility checks for variables never used or labels never referenced as well as mismatch numbers and types of parameters in calls. This is useful in development or in a port when some changes are made because there might be unintentional typographical errors in variable names or perhaps some parameters are missing. A major reason to use this lint-like utility is that, unlike the compiler which goes file by file, this utility will go across files.

   There may be conversion tools and migration utilities that are worth checking into for help. Port/VX and Port/RX are examples of software packages that help the users port their software. Port/VX helps that user go from DEC VAX/VMS to the HP9000 series 800 and Port/RX helps the user go from HP1000 to HP9000 series 800. Documents that define equivalent features are also worth having. "The Portability Guide" and "Programming in HP-UX" provide the comparable options among the multiple platforms and gives the user sample source code.

4. Look into tools that will help breakdown the application into logical chunks. 'fsplit' is a utility that will break up programs in large files into multiple files that contain subprograms and subroutines. "What's that useful for?", you may ask? Well, when looking at problems or seeking out what is wrong, this will segment your application so that you can home in on areas quicker than if you were looking at one big mass.

5. Plan during compilation to get informative runs that help characterize the application. Using a profiler enables the user to see where the calls are made, how often they are called, and who calls who. Using special compiler options that inform and can 'warn' help the user to prepare for any possible changes that he needs to make. Some of the compiler options are range checking, ANSI flags, and static analysis information. Cross references are very helpful and can be done upfront. Since the cross reference information is static, the earlier it's done, the better.

COMPILATION

In compilation, very rarely do I receive calls for assistance on syntax errors. Perhaps the messages are clear and concise or they are very obvious. The not-so-obvious errors are gotten best by brute force method or using the process of elimination. Brute force requires changing the area where the error occurs until it is correct. The process of elimination is to trim the code down by eliminating the areas that are not producing the syntax error. A typical call is when the specified line is at the end of the program, and something is missing or wrong with the general program. Here are some errors that point to the end of the program and give you very few clues about what is wrong:

```
"compiler error: no table entry for op REG
"Logical end of statement already encountered"
or "Expecting expression or subexpression"
```

Compile time errors and warnings are not meant to be cryptic. If anything, they should be straight forward and obvious. When the message is not obvious, use 'fsplit' to cut the file to a workable level.

There are standards and there are STANDARDS. Some industry extensions are often taken as standard but may not be ANSI STANDARD. In cases where there is a conflict, ANSI standard is always taken over the special cases of different vendor features. Hewlett-Packard strives to provide extensions to our FORTRAN where it is reasonable and industry accepted. If there is a conflict between industry standard and ANSI standard, Hewlett- Packard may provide both features but the industry accepted standard is not a general default, rather it is a special enhancement to the product. In such cases, there are directives and options that need to be turned on or explicitly used. If there is a question about what the STANDARD is, an option that warns of non-STANDARD features is provided.

At this point, there are discussions with the customers on conformance and conflict with non-conformance. Hewlett-Packard will always accept your inputs through our Software Tracking and Reporting System (STARS).

There are internal errors where no amount of work on the users' source code can help. In this case, the compiler aborts, or fatal internal errors may occur. This is something that you should tell Hewlett-Packard about because we want to fix it and it is definitely unexpected and catastrophic.

Execution

When compilation is done, you run your application. You may or may not get errors. If you don't get errors and all the results are correct, you are home free. If that is not the case, you can get error messages or you can get incorrect results. Incorrect results are errors too but they are a horse of another color. Incorrect results are obviously more difficult to troubleshoot.

There are errors and there are ERRORS, and they all can and should be looked into with a debugger (symbolic debugger, hopefully). Whether they are runtime errors or incorrect result errors, the symbolic debugger can help to focus on where the problem is occurring as well as where the correction needs to be made to fix the problem. The error message is a good starting point, unless you have incorrect behavior and unexpected incorrect (sometimes these two words are redundant) results.

At this point, you should call Hewlett-Packard. I will ask you what the version numbers are and whether some up-front characterization was done on the code -- such as getting informative options and profiling. Informative options are something that should be turned on only once for information and then removed after that one run because the option can be costly in terms of making the executable size larger and making the application run slower.

So here's a checklist of what to look at methodically for errors.

1. Use -C for range checking subscripts that are out of bounds.

2. Run the cross reference tables to look at variable names that may not have been typed correctly and which were therefore never used; also variables that are used may not be declared properly.

   If you have declared all your variables, use IMPLICIT NONE in your source code to call out undeclared variables.

3. Look at alignment issues in equivalence, common blocks, and structures.

4. Check that interlanguage calls are properly matched on parameters.

5. Use -K or save locals when migrating from a non-HP machine.

6. Make sure optimization is not turned on until the program is completely debugged.

7. Use +Q option to include directives into the source code without making changes to the source file.

The silent and deadly bugs are the worst nightmares one can have in an application. These are problems where the values are not as expected but the program does not abort or otherwise show an error. Uninitialized values are the most common culprit. The sources should not assume that memory will be initialized to zero when the program starts up. There is no guarantee of value unless one uses the special directive on some machines that specifies that the values must be set to zero. In any case, this initialization is not a default.

Fine tune

Now this is the fun part of the whole process. The application is running correctly and the next logical step is to make it run fast and well. There are many opportunities available to the user to make the code run faster or take up less space. Using utilities like 'strip' or the '-s' option in the linker (ld) will condense the size of the executable.

Looking at your system helps you understand the environment in which your application is a process. This in turn helps to eliminate some possible problems that can hinder the best performance. Here's a list of things to check when you begin fine tuning.

1. Use 'ps' to see what's running. If there are many users or another process is taking up most of the time, this will definitely slow everything down.

2. Use 'monitor' to see what the memory resources are and how much is being used during the process. Check the swap area configured to make sure it is not on the verge of being completely used. Monitor is also useful in determining if a task is behaving as expected. Make changes in swap area if you find that you close to utilizing all your swap area; consider adding memory for the same reason.

3. Run statistics on disk usage and disk arrangement of data because disk balancing can affect performance when a process has to wait on the availability of a disk area.

4. Look at 'nice' and 'plock' to modify the priority of a running process and to lock parts of the process into memory. This will give you a better handle on what your process is doing by knowing the interactions of the other processes with the system mixed in with your application.

5. Look at your application with a utility that shows whether all paths of your source are covered. There are a few utilities on the market today that provide this facility.

OPTIMIZATION

Now let's talk about optimization because it works and makes your application very efficient.

Once again, profiling will help to determine where the most time is being spent. This means that the code should be profiled and checked to see where the most likely areas of improvement through optimization are. The optimizer works best in cpu intensive applications or areas of source code.
But let's say you turn on full optimization. It runs and it flies -- your job is done. And, in most cases this is true.

Sometimes however, there will be an error when optimization is turned on. The first step is to go down a level (that is if you were at level 2, go down to level 1). If this works, great! This means you can continue. We intend for your applications to run at the highest level of optimization. Sometimes this is not possible. There is a limit to the size of a basic block of code that the optimizer will take in to work. If your code is 10000 lines of straight code with no branches or labels, chances are that full optimization won't work. Breaking up the code will enable the optimizer to optimize your source code.

Fragmenting the code and segmenting the areas where the error is occurring can help just like when you're troubleshooting execution errors.

Here's a list of what can be done to troubleshoot optimization errors:

1. Check to ensure the application adheres to 'standards' by turning on the ANSI flag through options or directives.

2. Check to see that the application does not violate compiler or optimizer assumptions. The compiler and optimizer assumptions are detailed in the reference manual(s).

3. Check to see that the application does not expect initialization from either the compiler or the system. The optimizer will warn about uninitialized variables. This should be taken as an action item and you should initialize the variable or turn on the directive that will perform initialization.

   During optimization, variables can be promoted to registers and registers are very rarely zero.

4. Check to see that there are no timing dependencies, or paging patterns.

5. Use 'volatile', save locals, or globals to protect variables that appear not to change but that really do. This is when the application uses data that can be modified asynchronously without communicating this to the compiler.

6. Change ill-structured applications with convoluted control flows and split very large procedures.

7. Run two debugger sessions side by side where one session is with optimization and one is without. Then, follow the flow of the program until you see a difference in results. That will be your first clue as to where the problem is.

There are calls that come in when compilation that normally takes 2 minutes runs for over 4 hours when optimization is turned on. Sometimes it never seems to complete and sometimes it aborts with a fatal internal error. This is probably an optimizer problem and not your problem.

Don't be afraid to do us a favor by spending time with your code to home in on a bug that may be the compiler's problem. The errors that are probably the optimizer's problem are fatal internal errors. Troubleshooting this type of problem can be fun and rewarding.

It can be fun to learn more about the assumptions that the optimizer makes for you. It may prove rewarding because it may not be the optimizer's problem, but yours. It might be a problem that you can fix. Even if it turns out to be an optimization problem, if it is distilled down for us, we can quickly look at the problem rather than spend time to investigate and characterize the problem before fixing it.

Well, I hope this trek through the troubleshooting process of what can be done in general on every FORTRAN platform when you are developing, migrating, or porting your application will be helpful to you. Plan, compile, execute, and fine tune!!

Integration and Analysis of Manufacturing Data

John M. Williams and Jerry W. Akers

Hewlett-Packard Company
Fort Collins Site
Information Technology
3404 East Harmony Road
Fort Collins, Colorado
(303) 229-3086/229-3487

Abstract

Integrated circuit manufacturing is a complex process. Like many manufacturing processes, process control and continuous process improvement are a function of feedback, which is simply the ability to react to relationships between data at various stages in the manufacturing process. Historically, data has been generated by a number of discrete systems, and a fair amount of effort (custom programming, data movement, and analysis) has been necessary to integrate and analyze data from different systems.

At the Colorado Integrated Circuits Division (CICD -- a division of Hewlett-Packard Company in Fort Collins, Colorado), a project was initiated to integrate all manufacturing process data, and provide access and analysis tools to end users. This Strategic Data Integrator (aka *SDI*) would replace a number of existing systems and be supported as part of the information infrastructure of the division.

This paper will present an overview of *SDI*'s hardware and software platform, and also discuss design and implementation issues associated with the integration and analysis of manufacturing data.

Prelude

Imagine it's 1971. As you survey the typical American manufacturing plant, you see labor intensive manufacturing that utilizes little of the advanced technology of the day. You see manufacturing equipment of varying sophistication and age, some utilizing electro-mechanical controllers or sequencers. The manufacturing process operates without statistical process control, instead relying on inspection and the innate repeatability of the manufacturing process to maintain acceptable quality (of course, the definition of acceptable quality was about to change). Computers, if evident at all, are probably mainframes that reside in the financial domain of the business.

In subsequent years, increased competition would force manufacturers to improve their manufacturing processes. Automated equipment would infiltrate the manufacturing floor and new manufacturing techniques (statistical process control, just in time, etc.) would gain popularity.

Now it's 1987. As you survey the manufacturing floor, you see islands of computer technology. PCs, workstations, mini-computers, and automated manufacturing equipment abound. Different computer systems may be used for almost every distinct function in the manufacturing plant -- from shop floor control to statistical process control to engineering analysis. Even process control or engineering analysis systems may not be homogeneous across the span of manufacturing.

The Beginning

HP's integrated circuit shop in Fort Collins was a classic example of this piecemeal automation. Separate systems in the manufacturing area included:

- HP 1000F
  - Electrical Parameter Test Database
  - Facility Environmental Monitoring System
- HP 3000 Shop Floor Control System
- HP 9000 Series 200 Shared Resource Management Systems
  - Statistical Quality Control System
  - Wafer Tracking System
- HP 9000 Series 300,500
  - Engineering data collection and analysis system
  - Parameter Testers

In addition to the aforementioned systems, there were also a number of isolated pieces of computer controlled equipment. Computers ranged from PCs to Unix workstations to HP 1000s.

The differences in hardware seemed overwhelming, and the sheer volume and variety of data was staggering. Thousands of different parameters and possibly hundreds of mega-bytes per week would have to be incorporated into *SDI* (Strategic Data Integrator).

Over the course of several meetings in late 1987, a user task force met to determine requirements for *SDI*. The requirements for *SDI* read much like the requirement list for any project: speed, reliability, user friendliness, data integrity, flexibility, accessibility, maintainability.....ad infinitum.

Research was also done to determine the 'hardware and software platform of the future'. This research indicated that Unix, C, and a relational database on HP hardware would provide the best platform for *SDI* implementation and evolution.

1. Data Structure

1.1 The Tao of Data Integration

Some underlying philosophy is necessary when trying to make sense of the chaos and vastness of the universe (in this case our manufacturing plant). The manufacturing plant builds objects, each one of which is unique and has a corresponding set of parameters that describe it. While a unique object exists, it accumulates some history based on location and time (the time aspect poses relative problems on a cosmic scale, but should be of no consequence for the scope of the manufacturing plant). Parameters associated with this unique object can be compared with similar parameters on other objects. Events in time associated with some location can also be associated with the unique object's time and/or location.

Let's try to diagram our universe.

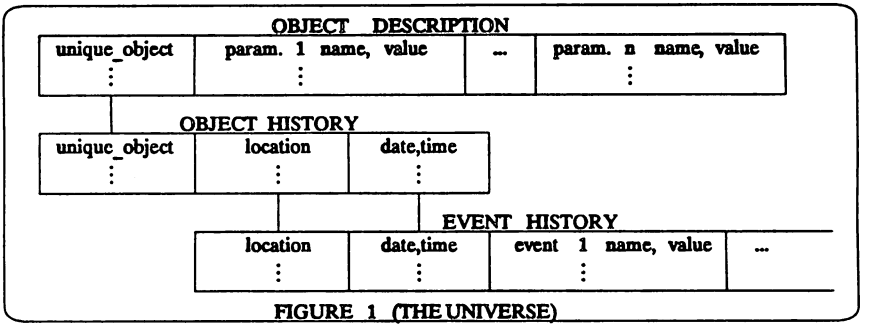


FIGURE 1 (THE UNIVERSE)

This model, although quite simple, served as the foundation for *SDI*'s data integration model. In fact, with an assigned staff of one and one half programmers and and implementation schedule of less than a year, simplicity was a key element throughout the project.

1.2 By the book

The process of designing and building a database structure is generally straight forward, but *SDI* presented some special problems. A simplified first pass at the relational database structure for the object description was as follows :

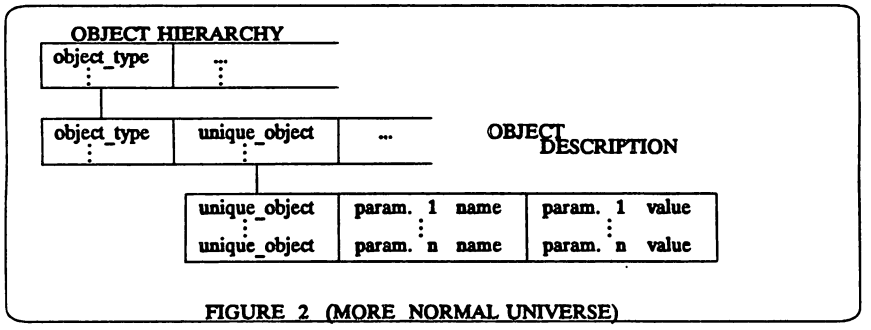


FIGURE 2 (MORE NORMAL UNIVERSE)

and indeed, other integration projects within HP had used similar structures. This structure

handles the repeating groups of parameter name and value in the textbook fashion , and the above structure facilitates easy retrieval of information about a single object or a single parameter. But from that point the issues become more complex. Considering that hundreds of parameters could be associated with a single object, how can all parameters for a group of similar objects be retrieved into a meaningful structure? The physical implication of a parameter table so constructed becomes even more intimidating -- hundreds of thousands of unique objects are produced monthly, each with possibly hundreds of parameters. This translates into tens of millions of rows per month, which in turn translates into high index overhead because of the large number of rows in the table.

1.3 Breaking the rules

An alternate, albeit unconventional, structure was conceived for handling object descriptions.

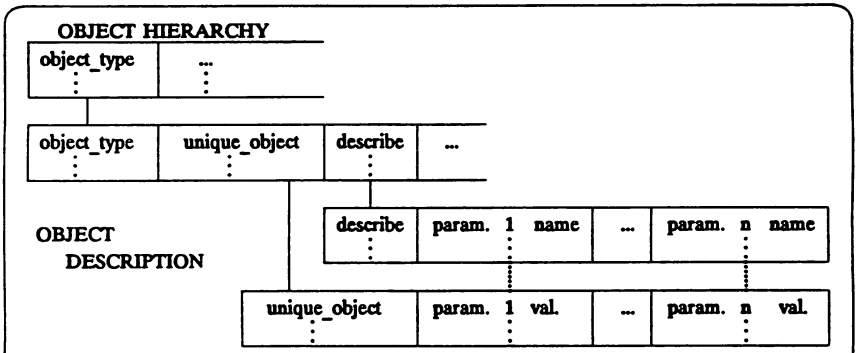


FIGURE 3 (LESS NORMAL UNIVERSE)

This data model allows the users to describe a set of parameter names, and then reference this set of names when entering parameter values for an object. Any subset of parameter values can be entered with an object, and unused parameters are represented as null fields in the parameter table.

Since a single row in the parameter table can now contain hundreds of parameters for a unique object, there are fewer rows, and consequently index overhead is orders of magnitude less than in the previous model. In addition, a reference to a parameter name does not have to be stored with every parameter value, and as a result, less table space is used. This structure also lends itself to correlations between parameters for a group of objects.

A similar data model is also used for event history.

2. DATABASE I/O

2.1 Overview

SDI must co-exist in an environment of varied testers and equipment, and hundreds of end users. *SDI* data I/O is via the LAN, and direct terminal or machine hookups are not allowed. Systems and equipment that do not support networking are interfaced to a computer that acts as a data gateway to *SDI*.

2.2 Output data format

Since SQL is closed over the set of relational operators, the result of any query is a table. Each column in this output table represents a field in a database table, and each row a particular instance of data. *SDI* represents these query results as simple flat files with the following properties:

- Each field in the flat file is a sequence of characters delimited by white space. Fields containing white space are quoted.
- The first line of the flat file contains the names of the columns
- Each row in the query result is a line in the flat file.
- Missing data (data that has been screened out or was null in the data base) is represented by 'NA' in the flat file.

The results of a simple query might be as follows:

```
UNIQUE_OBJECT_ID     PARAM_NAME_1  PARAM_NAME_2  PARAM_NAME_3
1234.5678            4.5           5.4           2.31
1234.5679            4.6           6.4           2.32
1234.5680            NA            7.4           2.33
1234.5681            4.8           NA            NA
1234.5682            4.9           9.4           2.35
```

Size of these output files varies -- a small file may only be a single instance of data for a few parameters, while a large one could encompass thousands of parameters for tens of thousands of instances.

2.3 Getting the data out

Although the core of *SDI* is a relational database, the user does not directly request data via a query language. Instead, all *SDI* data extraction is done through a set of data extraction utilities. These extraction utilities use embedded SQL to query the database and then output flat files in the format described in the previous section.

Now at this point, the more knowledgeable reader may be thinking, "Why can't the user just have access to the query language?" Part of the answer has to do with the way the database is queried -- the extraction utilities are constructed to take full advantage of the design of the relational model, while a user with general query access would be free to construct any query, no matter how inefficient. If fact, users with general query access could inadvertently consume significant system resources, thus degrading overall system performance. The extraction utilities also isolate the user from query complexity and provide a simple mechanism for building virtual utilities that reside on remote systems.

2.4 Input transactions

All input transactions in *SDI* are file oriented. Each input file (transaction) is considered a single unit of work and will succeed or fail as a whole. Each input transaction is of a specific transaction type. The transaction type targets a specific set of tables within the database.

2.5 Input data format

The main objectives in determining how to put data into the database where simplicity and accuracy. As a result, putting data into the database is simply a matter of writing a flat file (same format as an output file) and adding a small amount of header information. This additional information is as follows :

```
line 1     ....  mail address of data owner
line 2     ....  transaction type, describe
line 3     ....  column names
line 4     ....  first data line
line n     ....  last data line
line n+1   ....  END
```

If for some reason the transaction (i.e. file) cannot be loaded into the database, a mail message is sent back to the owner (line 1) with error messages on why the input failed. 'Transaction type' on line 2 targets the appropriate tables in the database, and describe (also on line 2) selects the set of parameter names that are valid column names in this file.

2.6 Data Input

To submit data for input into the database, the file is copied into a spool directory on the database machine using a network file copy utility (typically rcp). This spool directory is checked every fifteen seconds by a daemon (*input manager*), which passes any files it finds off to the *input program*. The *input program* reads the file and tries to stuff the data into the database. If the data is input successfully, entries are made in a transaction statistics table and an archive table. The input file is then compressed and moved to an archive area. If the data input is not successful, the input file is compressed and moved into a trash area, and mail is sent to the data owner (line 1 of the data file).

2.7 Aging / archiving data

Although *SDI* is large enough to contain approximately one year of manufacturing data, at times it is necessary to remove old data to make room for the new. This 'old removed data' may be needed again someday by someone for some obscure reason -- this necessitated stuffing the old data somewhere in a format that would facilitate easy retrieval. Since input transactions have already been moved into an archive area, completing the archive process is done by periodically writing a tape, deleting the archives, and making the appropriate entries in the database archive tables. Since all transactions are archived, data can be deleted as needed based on alerts mailed from the *space manager*.

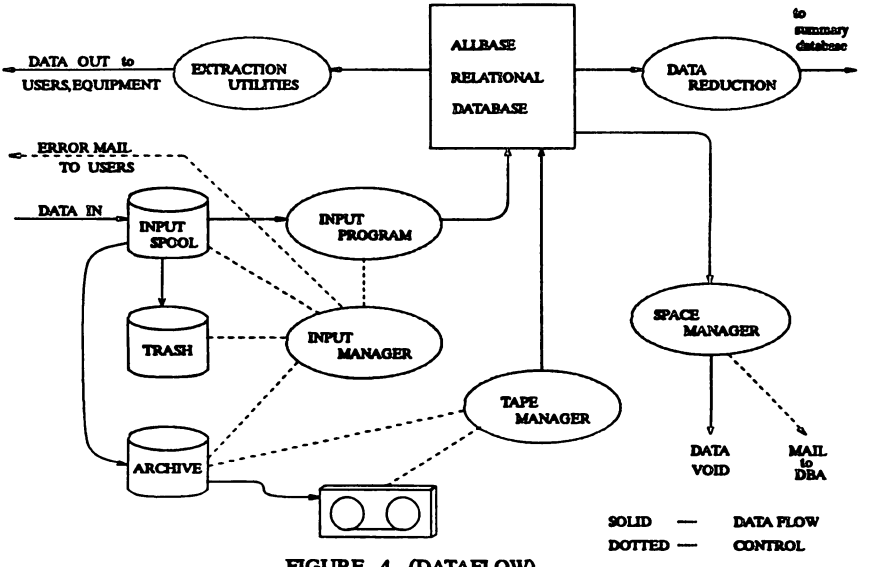


FIGURE 4 (DATAFLOW)

2.8 Constraints

Specific fields within the database may have limitations on the range of the data. The database itself forces a type constraint (i.e. numeric fields can only contain numbers), but this was not sufficient for parameter values. A parameter constraints table contains valid upper and lower bounds for parameters. Constraints are not applied as data is input to the database, but they can be optionally applied as data is extracted.

2.9 Uniqueness

Depending on the transaction type, uniqueness violations are handled in several ways by the input program:

- when data collides, increment 'unique key' field in transaction record until transaction becomes unique. The sole purpose of this unique key is to be able is to guarantee the uniqueness of a transaction. This allows all data sent, even duplicate submissions, be input into the database.
- when data collides, update database with new transaction
- when data collides, ignore the transaction

3. SYSTEM OVERVIEW

3.1 Hardware

SDI was initially built on two HP 9000 Series 825 computers using Allbase as the RDBMS. The amount of data collected grew rapidly, as did the number of users. Both 825 systems were upgraded to 835s and later to 845s. An analysis machine (another 845) was also added to handle analysis tasks. Both hpfisdi and hpfixdi are data servers, while the main focus of hpfiadi is analysis tools.

3.2 Environment

Since all of *SDI*'s data I/O is via the LAN, all *SDI* data generators and data requestors needed to somehow talk to the LAN. In order to talk to SRMs, SRM cards were installed in the DIO I slots of the HP9000 Series 300 gateway computers. SRM utilities on the 300 such as srmls, srmcp, etc., were used for data movement to and from the SRMs. The balance of the non-LAN equipment that needed access to *SDI* used RS-232 to talk to one of the gateways.

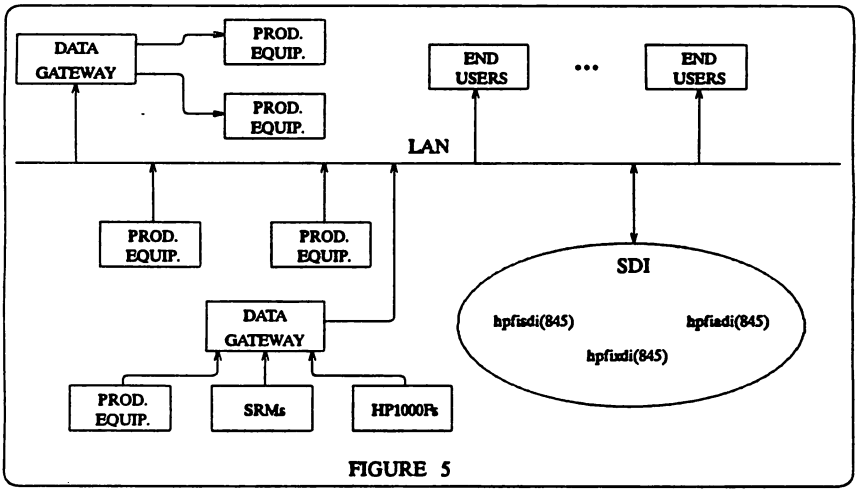


FIGURE 5

3.3 User access

Administration of the system is an ongoing concern and with literally thousands of users, some method was necessary for controlling or adding users. Again, simplicity was the key. A generic user account called 'data' was added to the database machines. All users access the database as user data (they execute a 'remsh -l data extract_utility options... ' command) and no maintenance of individual user accounts is necessary. A user is added or deleted simply by adding or deleting an entry in data's .rhosts file. All extracts done by user data are logged and the /etc/profile is modified to prevent user logon via remsh or rlogin. Since extraction utilities exist in only one place for one user (data), database permissions are simple to maintain and extraction utilities are easy to modify and support.

4.1 It's about time

Since object histories and event histories are generated asynchronously by separate systems, time presented some unique problems when analyzing data. The actual date and time of an instance in object history did not necessarily correspond exactly with any date and time in event history. When trying to do a join between event and object histories, SQL didn't think close was good enough.

To solve this problem, a time join function was written that could take the output from an event history extract and join it with the output from an object history. Four methods of joining these outputs were provided :

-nearest	finds the nearest time event, whether before or after the target event
-floor	picks time event that is the floor of the bounding time
-ceiling	picks time event that is the ceiling of the bounding time
-interpolate	computes new event values from bounding times

Let's consider an example :

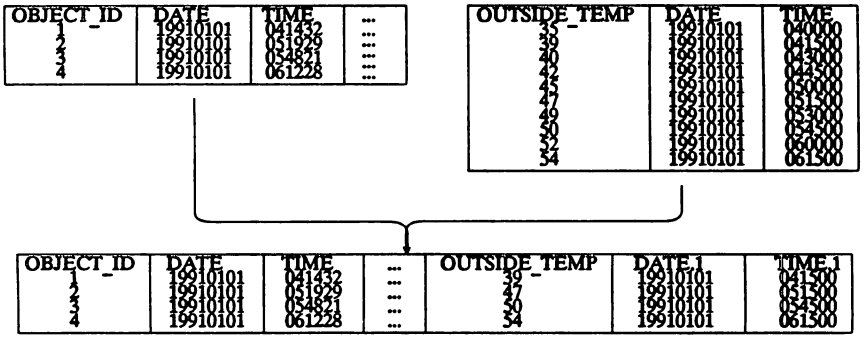


FIGURE 6 (JOIN by 'NEAREST' DATE, TIME)

The user can choose to include or exclude repeating occurrences of columns such as date and time. If repeating columns are kept in the join, a suffix is automatically appended to maintain unique column names.

4.2 User interface

There are two levels of user interface into the database. The more knowledgeable computer user may construct utilities that make use of the extraction utilities via remsh. For the computer novice, an environment was built that allows the user to extract and analyze data without prior system or network knowledge.

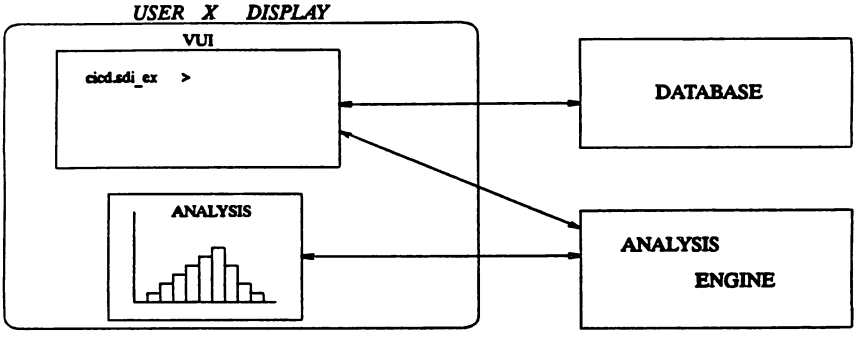


FIGURE 7 (VUI interaction)

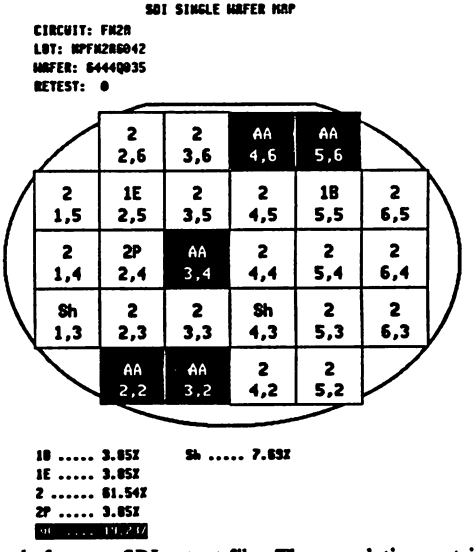
This verb driven user environment (VUI) contains on-line help and can be installed on any HP 9000 Series 300/800 computer with network access to *SDI*.

4.3 A picture is worth a million bytes

SDI gave users access to unprecedented amounts of data. It was now possible to get data in a single extract that contained information about almost every aspect of the manufacturing process, and it was clear that some new tools were required.

4.3.1 Wafer maps

Since our manufacturing shop builds wafers, a utility that could represent parameter values as they applied to the physical product was built. This tool allowed the user to view a parameter over a single wafer, a parameter over a group of wafers, or a composite map of many wafers. This utility was written in C and uses Motif widgets. It is interactive in nature and has been used to solve problems that are not easily found with more conventional types of analysis.

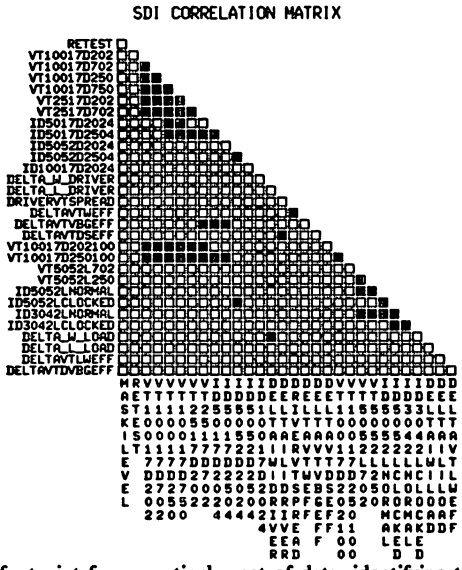


4.3.2 Correlation matrix

Another of these new tools was a correlation matrix that can be computed directly from an *SDI* output file. The correlation matrix takes every two column subset of the set of columns and computes a linear correlation coefficient for each subset. The correlation coefficient ranges from zero to one, with one indicating a high degree of correlation and zero indicating little correlation. The number of correlations

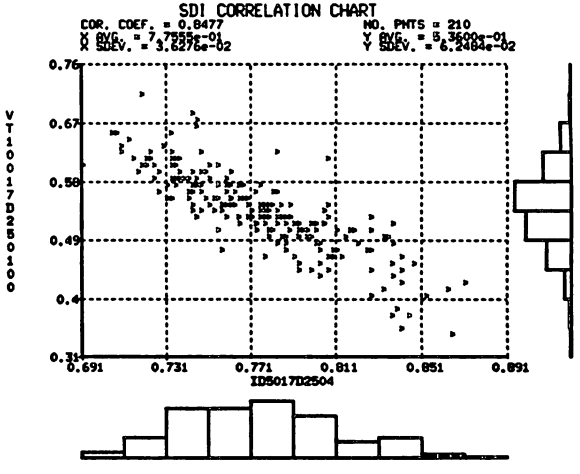
computed is a $O(n^2)$ function of the number of columns in the output file. Where c is the number of columns, the total number of correlations is c (c - 1) / 2.

For an extract file of 200 columns by 200 rows, the correlation matrix (19900 correlations) is computed in about 40 seconds.



The correlation matrix is a triangular matrix, with each rectangle representing the correlation result of a pair of columns. An unfilled rectangle indicates that the correlation coefficient for that pair of columns was below the user defined threshold (NOTE : the threshold can be passed as an argument to the correlation matrix program or can be set interactively via a pop-up menu). A filled rectangle indicates that the correlation result for that pair of columns was above the threshold. If a location in the matrix is blank, this indicates there was no data and no correlation was possible. As a result, columns of ASCII data will just show up as blank areas in the matrix. The matrix can also serve as a footprint for a particular set of data, identifying trends in correlations and dependencies between columns.

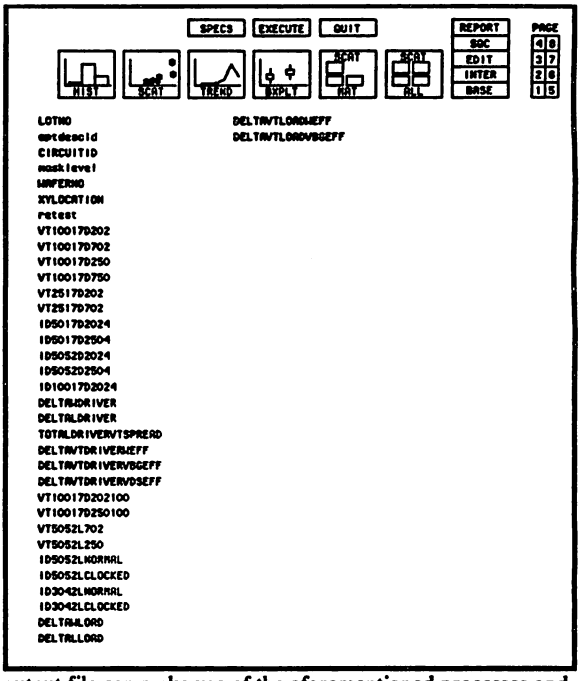
The more inquisitive user may desire to know more about a specific rectangle in the matrix, and doing so is easy -- just move the pointer to the desired rectangle and click a mouse button. Within a few seconds, a window containing a scatter plot of the desired columns will appear. Averages, standard deviations, and histograms of both columns are displayed as part of the plot. A third field (i.e. a *Z* field) may also be selected, and it can consist of either numeric or ASCII data. If the *Z* field is numeric, the plotted points are colored based on a linear scale of the Z data. If the Z data is ASCII, user specified ASCII categories within the Z data can be highlighted via color on the scatter plot.



4.3.3 S-PLUS

S-PLUS, from Statistical Sciences, Inc., is an enhancement of the NEW S language from AT&T. It is a high level language for manipulating data and generating graphics reports, and has both interactive and batch capability. Part of the analysis environment of *SDI* is S-PLUS, and tools were written that accept input from *SDI* extraction routines and can write S-PLUS data objects directly.

As the user extracts data for analysis in VUI, the extract can be piped through a process that writes S objects directly, and either a single S object or an S vector per column can be generated from the extract. Typing (char vs. real) is done automatically based on the data encountered.



Typically, a single object is written per extract, instead of a vector per column. Elements within the object are named by the column names that would have appeared in the first line of the extract file. Generic S functions have been written to do a variety of control charts, plots, and object editing. Menuing functions have been written using the S-PLUS X11 driver that provide the user point and click access to any element within the object and to many of the generic functions written for these objects. Any flat file written in the same format as an *SDI* output file can make use of the aforementioned processes and functions.

5. OBSERVATIONS

5.1 Success ?

Now its 1991. *SDI* has been running in a production environment for nearly three years and has become an integral part of the information infrastructure of the division. Information regularly flows into *SDI* from dozens of different systems, ranging from shop floor control systems to automatic parameter testers to small data acquisition systems. *SDI* also feeds data on request to manufacturing equipment, users, and batch reporting jobs. The number of daily transactions (at Fort Collins site) ranges from three to fifteen thousand, and transaction size varies from tens of bytes to megabytes. *SDI* in Fort Collins supports more than a thousand users (some of them very occasional) from a half a dozen different sites. *SDI* has been installed at two other divisions within HP and is under consideration at others.

5.2 Flexibility

SDI is anchored by gigabytes of historical manufacturing data. Any modification that changes relations within the database must ensure historical continuity of data. Although input programs, extraction utilities, and database relations can be modified relatively easily, it is maintaining historical continuity that requires the effort.

5.3 Reliability

In nearly three years of operation, there have been no hardware failures on the three Series 800s and associated discs, unless one counts that free 7914 disc early in the project.....

The input programs to the *SDI* database have handled millions of transactions, and have never lost any data. A runaway DBA disguised as super-user once blew away an active dbefile, but the system was restored with no data lost.

5.4 Perceptions about SDI

5.4.1 User view

Somewhere, out on the network, there exists this *black box* that soaks up all the data thrown at it. At any time, any combination or subset of this data can be retrieved upon request. The system does not have discernible physical or logical bounds, and anything I ask, it should do.

5.4.2 The system administrator view

They tell us that if a disc fails, we have to restore all of the dozen or so discs on the system at the same time to maintain a *consistent* copy of the database.

I hope a disc doesn't fail....

5.4.3 The management view

Somewhere, out on the network, there exists this *black box* that that soaks up all the money we allocate for computer resources. At any time, no matter how much money we throw at it, more requests will be forthcoming....

5.4.4 The classical IT view

You should have used MPE, COBOL, and a good 4GL.

5.5 Accessibility/availability

SDI is easily accessible from any machine with network access, and adding or revoking permissions for access can be done in seconds.... Although database accessibility is a function of the overall computing environment, much of the evolution toward Unix and interconnectivity was driven by the need for information and access to *SDI*.

Currently, during system backups, the database is not available for data input (i.e. files are not picked up out of spool area). This is necessary to ensure a consistent backup of the database.

5.6 Performance

Performance was an issue from the projects conception. It was never clear from any of the benchmarks or specs available, how Allbase on an HP 9000 Series 825 would meet our needs. Prior to buying into the database design, several alternative databases were built, and simple programs were written to stuff and extract data in an attempt to measure performance and system limitations. The final database design was selected based on information learned from these tests, and additional performance improvements were realized with hardware upgrades (825 to 835 to 845).

5.7 Future

Product life cycles are getting shorter, and manufacturing processes are becoming more complex. Users are demanding more information and some needs are already apparent:

- database performance and capacity must keep pace with increasing user and process demands.
- performance in the analysis environment must also increase to handle the increasing volumes of data.
- the friendly user environment isn't friendly enough for some users.

If *SDI* evolves to meet the ever expanding user needs, it will continue as the data integration solution for CICD. If not, users will meet their needs locally, and new 'islands of data' will be created.

Paper Number 2038

Publishing for Paper and Online

by
Alden Chang and **Wesley Cheng**
Hewlett-Packard Company
100 Mayfield Avenue
Mailstop 36-LR
Mountain View, CA 94043

Abstract

Traditional methods for publishing information have been for books, which involve producing camera-ready typeset copies for reproduction. Recent computer software tools have enabled writers to publish books more quickly.

With the proliferation of computers in the workplace, the demand for making information available online has been overwhelming. Unlike traditional databases, publishing voluminous books for online use has its own challenges; document interchange formats, authoring tools, and software for retrieval and publishing. Manufacturing that information for both paper and online adds another layer of complexity.

HP has been working towards the goal of producing media independent information from a single source, thereby reusing a writer's valuable work. This also preserves consistency between paper & online. The current process enables HP to manufacture paper manuals and full-text indexed online manuals distributed via CD-ROM from one structured document.

Emerging technologies such as multimedia, expert systems, OCR, imaging systems and evolving document interchange and retrieval standards promise to feed the ensuing information explosion.

Introduction

Being able to communicate ideas, concepts, and thoughts effectively has been an age old practice since the beginning of man. Over time, we have developed a variety of ways of communicating: natural language, books, pictures, music, to name a few. With computers in the workplace, a new mode of communication has arisen, which is information online.

Information must have *relevance* before it becomes *knowledge*. Therefore, accessing the right information is as important as the information itself. That is one of the reasons why man had devised so many ways to communicate, for certain forms are more appropriate in certain circumstances. Moreover, different forms are simply more comfortable to a given type of audience.

Likewise, Hewlett-Packard has many ways of communicating information to its customers: customer training, consulting, books, online, to name a few. This paper focuses on two modes of communication: books and online. We will cover the reasons behind creating both forms of information. We will focus on the challenges of creating both modes effectively and economically, and describe real-world experiences of document creation at Hewlett-Packard.

The HP Customer

Based on customer surveys, HP customers want information both in hardcopy, book form and online form. Certain customers are more comfortable with books, while others prefer online. Moreover, there are customers that are comfortable with both forms of information, but find that one form is more appropriate depending on the context of their work and tasks.

Each form, books and online, has its advantages and disadvantages. Books are printed on paper whereas online information can be distributed in a variety of media such as floppy disk, cartridge tape and compact disc read-only memory (CD-ROM).

The advantages of books are:

- *Portability.* You can carry a book anywhere. With online, even if the information is stored on a CD-ROM or floppy disk, which are very small, you need a computer to view its contents.
- *Familiarity.* We have had information on paper since the printing press was invented. Information online has been more recent, within the last two decades during the computer revolution. Moreover, each collection of online information has its own retrieval user interface that a user has to learn. Books are so familiar and simple to use that nobody needs much explanation on how to flip a page.
- *Easy to Annotate.* Related to familiarity, people find it natural to write notes on a paper document. With online, annotations require writing through a computer and its keyboard, if this is even possible with the software being used.
- *Readability* Currently, the resolution of a computer screen for online does not match that of paper publishing. With low resolution, aesthetically pleasing and sophisticated higher quality typography of printed books cannot be accurately reproduced.
- *Reliable* Books are strictly standalone, whereas online requires the computer, which is vulnerable to system failures and power outages. If your computer is down, you cannot bring up the online manual to find out how to bring it back up.
- *Easier on the Eyes.* For extended reading, paper is easier on the eyes than a cathode ray terminal. Working in front of a computer monitor has been likened to staring at a light bulb, which is not the best thing one can do for one's eyes.

The advantages of online are:

- *Full Text Search.* Every significant word is indexed for quick retrieval. Significant words are words other than conjunctions, prepositions, and the like. This enables users to form complex search expressions using boolean and wildcard operators to zero in on the information they wish to locate.
- *Space reduction.* Having information online means that the physical books do not have to reside in one's office. This could result in substantial shelf space savings, space that could be used for other valuable reference material.
- *Complete & Up-to-date.* You do not have to worry about lost books or pages within books. With all the online information at your fingertips and in one place, you are assured that a search across the online manuals will produce the most comprehensive results. In the case of books, it would take hours to pour through hundreds of thousands of pages and that would still not guarantee that every last detail has been found due to the human tendency to miss details.
- *Easy to Update.* Assuming online information is stored on CD-ROM, the technology for CD-ROM manufacturing lends itself easily to more frequent updates. Also, distribution is easier and more economical due to the CD-ROM's compact size.
- *Context Sensitive Access.* Finding the appropriate piece of information is far more direct and specific, with no extraneous references. With an book index, you are often lead to information that is only peripherally related, and can often disrupt train of thought.
- *Speed of Access.* Having the computer do the search is often faster than searching through the pages of a book. If you are searching across a large volume of data, computers can find information within seconds when it would take a human hours, or days to find that same information.

From the above lists, it is easy to see that customers may want both online and books, depending on what they are doing. If a customer is learning a concept that requires a large amount of reading, then reading a book is more natural and appropriate. However, if that same customer needs to find reference information quickly while on the computer, online would be better.

As a result, HP offers its customers information in book form, as well as online, distributed via CD-ROM's.

The Objectives

Given the competitive markets that HP is in, there are tremendous pressures for HP to shorten development time and cost. Studies have shown that, with a 20 percent growth market, a 12 percent annual price erosion, and a five year product life, a product that is 6 months late reduces the profit over its life by one-third. Delivering products on time is critical. Reducing the cost of development is also critical, given the price erosions of both the computer and electronics markets.

Cheaper and faster is not enough, unless the product is also better. For example, the recently announced RISC workstations, the HP Apollo 700 series, offers more than double the performance of its predecessors.

Similar objectives also apply to documentation. We use the term **documentation** to mean information designed to enable the customer to use or service a given product or system. Documentation is very much a part of the product development cycle and costs. Reducing the developent time and costs apply to documentation as well.

However, HP customers demand better and more ways of viewing documentation. This paper focuses on two views: paper and online. Thus, the challenge becomes: how does HP provide customers both paper and online, while at the same time reduce costs and development time in its documentation, and ultimately, in its products?

Hence, our objectives are to deliver documentation products that are:

- Higher quality.
- Cheaper to produce.
- Faster to produce.

The Solution

HP's solution is a two-pronged approach:

1. Focus on document creation: efficiently creating, interchanging, and reusing information.
2. Focus on document production: utilizing reduced inventory practices and economies of scale.

Document Creation

When an author is creating documentation, we want to improve and achieve the following:

- Create the document source only once, and have software do the work of generating for both paper and online. Reuse the writer's work for multiple outputs. See following figure.

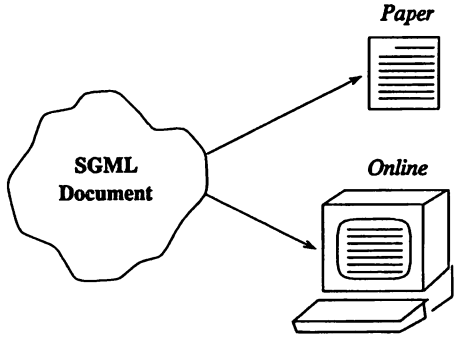


Figure 1-1.

- Create and edit the document source in some electronic format. This allows easier transportability for documents, and can take advantage of other electronic publishing technologies. Finally, an electronic form allows for a fully automatic publishing solution for both paper and online material.
- Base the document creation commands, or language, on a standard, which then can facilitate document interchange. HP has a diverse product line, ranging from medical instruments to business computers to workstations. Consequently, HP has many autonomous writing groups focused on a particular product line. Yet these writing groups are interdependent when the need to exchange information arises.
- Allow document interchange to occur across different platforms and software. If we want multiple and different types of outputs from a single source, we want the document source based on a standard which can be accepted by various publishing software and platforms.
- Keep the document creation language extensible to emerging technologies, such as hypermedia. If the creation language is based on a standard that is flexible and adoptable by other standards, emerging technologies such as hypermedia can be more easily incorporated.
- Help facilitate the translation of documents to other natural languages. HP is an international company, with customers worldwide. It is critical that the documentation be in their native language. The document creation language must be in a form which helps, and not detracts, from the translation process.

Document Production

When producing the document, and manufacturing for mass release, we want to achieve the following:

- Miminize inventory for paper publishing. Produce just enough manuals as needed, rather than produce many and store them in warehouses. The costs for storage and handling are reduced. Turnaround time to print an older manual is also substantially reduced, as it is simply a matter of retrieving the electronic master from an archive and running it through the printer. The traditional approach would require taking the time to find the physical paper master, producing the photographic masks, and setting up the printing assembly line.
- Achive economies of scale for publishing paper & online.

- Configuration flexibility. This is especially important when bundling a set of online manuals onto a CD-ROM. The database would allow an operator to select the appropriate manuals and compose the contents of the CD-ROM easily. Changing the contents when new manuals are added or when old ones are deleted is also trivial.
- Reduce time and overhead in handling material. Use of electronic masters do not have to be sent through mail which would result in transit delays and possible misdelivery. When deadlines are tight, this becomes a significant advantage over regular mail services.
- Increase reliability. An electronic master can be checked for correctness immediately when transmitted electronically. This allows for higher reliability since a master can just as easily be retransmitted if there was a problem.

The Implementation

We will describe the implementation first in terms of document creation, then in terms of an overall process for both paper and online.

Implementation: Document Creation and Editing

In our approach, we wanted to base our document creation language on a standard, and we wanted the language to be general enough to encompass both paper and online formats. As a result, HP has been actively using an International Standards Organization (ISO) standard, 8879, called Standard Generalized Markup Language (SGML).

What is SGML?

We will talk about the salient points of SGML, and what advantages it provides us. We will also explain those points in some detail. However, we will not describe what SGML is all about, for there are plenty of references to get a total picture.

SGML is a general purpose language that allows you to define your own set of commands, or "elements," for describing your document. Therefore, for each document, there is an accompanying description of what each element means. This gives you freedom to customize an element set which suits your particular needs. Naturally, software and other applications would have to be available in order to parse and process documents that are written in SGML.

Now, the following are the most important points to remember about SGML:

- When describing your document, you focus on the document's **structure**.
- You do *not* focus on format or appearance.

We introduce the notion of **structure**, because through structure, we can create information rich enough which ultimately can be translated into a particular format automatically. In other words, once you have format independence, you need only to create the information once; back-end processes generate specific formats for various media such as paper and online.

The term "structure" can best be described by some examples. When you describe your document in terms of chapters, heads, subsections, and paragraphs, you are describing in terms of structure. When you use terms such as "14 point Bold Helvetica," you are focusing on format. So, use "chapter head," rather than "14 point Helvetica." Use "level2 head" rather than "11 point Bold Helvetica." These examples illustrate the notion of "tagging" or "marking up" a document, which some formatting packages have introduced through their macro capabilities.

Structure, however, also implies hierarchy. When you compose an outline, you are in effect organizing our thoughts in a structured and hierarchical way. Indentations in our outline represent the various levels of our document. Hence, you can look at any document as a hierarchy of information. Most formatting packages view a document as "flat."

Let's look at a section of a document. Suppose a it has a level one head, followed by a paragraph and level two head. The level two head is followed by two paragraphs. We can draw a tree diagram to represent the structure of our document, as follows:

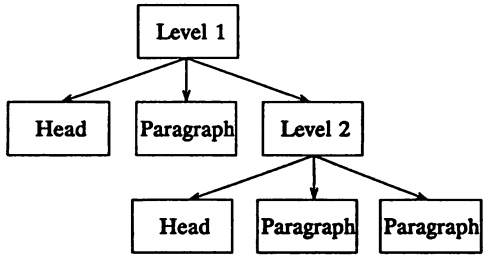


Figure 1-2. Hierarchy in a Document

Now, let's view the document as "flat:"

Figure 1-3. A Flat Document

Notice that with a flat structure, it's difficult to tell whether the last paragraph really belongs to the level one section, or to the level two section. This can be important if the paragraph font is different inside a level one than in a level two. Put in another way, the format and typography of a document really represents the outline, or hierarchical organization, of our thoughts on paper.

Another important advantage of structure is context sensitivity. A paragraph inside a level two can be distinguished from a paragraph inside a level one. The writer needs only to remember the command, or element, "paragraph." The net result: less commands to remember, with a more natural way of using the commands.

If we had a flat structure, then we need an additional command which distinguishes a "paragraph inside a level one," from a "paragraph inside a level two." If our commands are based on format, then we may or may not distinguish these two types of paragraphs; it all depends whether the resulting fonts and sizes of type would be different. Suppose we did not need to distinguish the two types of paragraphs; we then effectively locked ourselves into one type of format. What happens if, at a future time, we needed to support a new format, or media, which requires the distinction?

Finally, a structural view allows easier and faster translation. For a translator, "level one" means much more than a formatting command. "11 point Helvetica Bold" would be totally meaningless in a given local language, such as Japanese.

One SGML Source, Multiple Presentation Views

This leads us to crux of structured documentation: it is better to have more general information, in structural terms, than the more restrictive information, in formatting terms. Once we have a hierarchical structural view of the document, we can translate to multiple formats. Hence, at HP, we are able to produce both paper output, as well as CD-ROM output. What we want to achieve is a single SGML-based source, and multiple **presentation views** of paper and online.

We present an actual application of our edit forms and multiple presentation forms. Suppose our source looks like the following. Assume that it is in ASCII electronic format, and the indentations are for readability only, empasizing structure. Note that elements are in angle (<>) brackets. The (</) delimiters indicate the end of a structure. Refer to the earlier tree diagrams which describe the corresponding structure.

```
<s1> <head> Starting the X Window System </head>

     <p> Your system may be configured to start X11 as part of
     your login procedure.  If so, skip to the next module,
     "Finding the Active Window."  Otherwise you'll start X11
     after you log in.  When you see the command-line prompt,
     type the X Window System start command.  Shortly thereafter,
     your screen will change color, and the pointer and a terminal
     window will appear on the screen. </p>

     <s2> <head> Typing the X11 Start Command </head>

          <p> If you have logged in correctly, you will typically
          see a welcome and copyright message, followed by a
          <computer> command-line prompt </computer>.  A
          command-line prompt, as the name suggests, shows that the
          system is ready to accept commands.  By default, the
          command-line prompt is either $ or %. but <emph> it can
          be different </emph> depending on how your system
          administrator set up your account.  This guide uses %
          to represent the command-line prompt.  </p>

          <p> You can locate the command-line prompt by pressing
          the <keycap> Return </keycap> key several times;
          HP-UX displays the prompt every time your press
          <keycap> Return </keycap>.  </p>

     </s2>

</s1>
```

Figure 1-4.

The three elements, <computer>,<emph>, and <keycap>, represent computer text, emphasized text, and keycaps on a keyboard, respectively. Notice that the names are generic, and that they can look very different from one presentation view to another. One presentation view, on paper would look like the following:

Starting the X Window System

Your system may be configured to start X11 as part of your login procedure. If so, skip to the next module, "Finding the Active Window." Otherwise you'll start X11 after you log in. When you see the command-line prompt, type the X Window System start command. Shortly thereafter, your screen will change color, and the pointer and a terminal window will appear on the screen.

Typing the X11 Start Command

If you have logged in correctly, you will typically see a welcome and copyright message, followed by a **command-line prompt** . A command-line prompt, as the name suggests, shows that the system is ready to accept commands. By default, the command-line prompt is either $ or %, but ***it can be different*** depending on how your system administrator set up your account. This guide uses % to represent the command-line prompt.

You can locate the command-line prompt by pressing the Return key several times; HP-UX displays the prompt every time your press Return.

Figure 1-5. Paper Presentation View

On CD-ROM, the same output would look like the following:

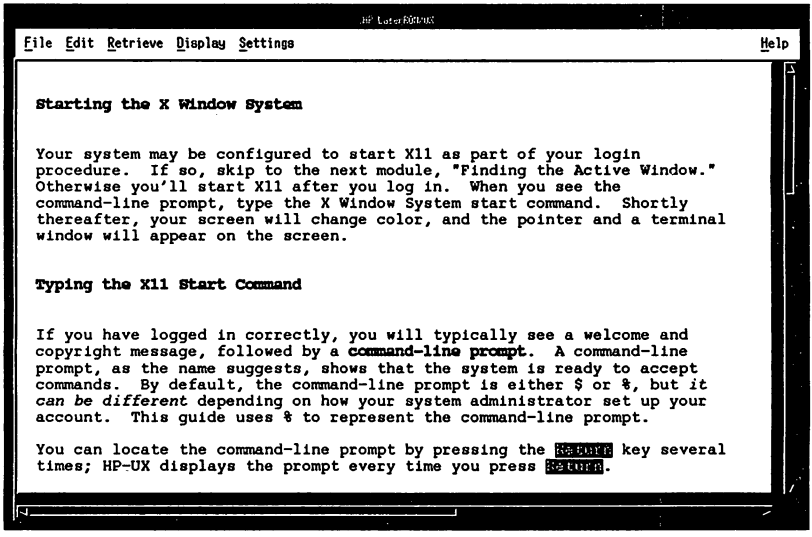


Figure 1-6. CD-ROM Presentation View

Implementation: Document Production

HP believed that SGML would address the concern of maximizing the work that an author has put into writing a manual while minimizing the costs of manufacturing that manual for paper and online. To that end, we started developing procedures and software to manufacture documentation. With a single source, we were able to produce multiple versions of the document suitable for different media. For example, a document might be processed into Postscript for printing while an online version might require it to be processed into a wordprocessor format.

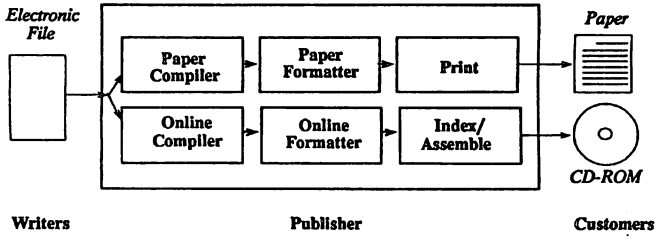


Figure 1-7. Single Source, Multiple Media

Why is CD-ROM so popular?

While there is no argument that books must be printed on paper, online information can be distributed in a variety of media. We had mentioned such as media floppy disks, cartridge tapes and CD-ROMs. It seems that CD-ROMs are everywhere these days and every vendor wants to ship their products on CD-ROM.

The most compelling reason for shipping information on CD-ROM in its online form as opposed to hardcopy books is economics. This can best be explained with an example. It takes an average $1500 to $2000 for CD-ROM premastering (produce a set of masks that can be used to stamp out CD-ROM's) and $2 to $5 to press a CD-ROM. The cost of printing a book averages about $10. In a CD-ROM, with capacities of 660MB of data, one can easily fit 100 books. As such, a CD-ROM for $5 is equivalent to 100x10 = $1000 worth of paper manuals. It would seem that it would only take a few customers to justify distributing on CD-ROM. However, these are numbers that reflect only the production costs. The actual costs from adding the overhead usually result in the breakeven point to be on the order of some hundreds of customers. It is clear that for high volume, high content distributors, CD-ROM offers substantial savings. Since a single CD-ROM can store so much more than other online media, it is still the cheapest medium to use.

In addition, the turnaround time to press CD-ROM's can also be much less than printing books, or even other online media such as cartridge tape and floppy disks, where data has to be fed serially to the media.

The reason CD-ROM had not been as popular previously is because of the high price of CD-ROM drives. With the CD audio technology bringing down the cost of making readers for CD, the CD-ROM industry got a much needed boost and is now the distribution medium of choice.

Publishing Tools

A variety of tools must be developed or purchased to support an efficient manufacturing process. These are some of the tools that are recommended for producing paper and online documents using SGML.

Authoring system

Authors must be provided tools to create and submit their work. SGML editors which check on syntax are particularly useful. Given that SGML editors are still in a relative state of infancy and have not been ported to many platforms, regular text editors such as **vi** or **emacs** on UNIX and **ed** or **write** on the PC will suffice. This gives the author the freedom to write on any platform. The document must be checked for correctness in both syntax and format and has to be run through an SGML checker and formatter.

In addition, to ensure that their document adheres to the submission standards, a packaging tool that wraps all relevant files and sends them electronically to the publisher is also needed.

Publishing system

The publishing system consists of two key components: the conversion software that checks SGML and formats for paper or online, and the production software which manages the electronic masters. The figure below illustrates the flow of information.

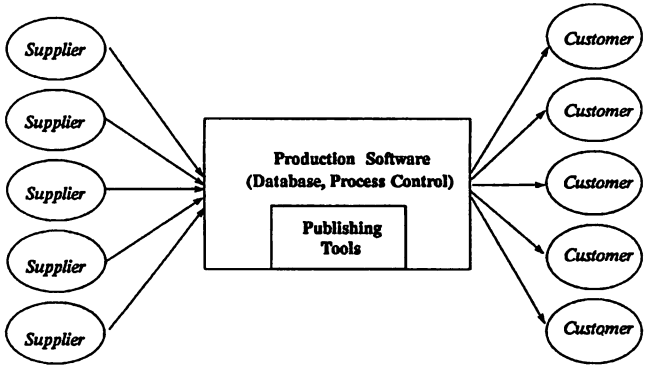


Figure 1-8. The Electronic Publishing System

Specific tools might include:

- SGML Markup compilers. These tools check the syntax of the submitted master and processes it into formatting commands to be sent to a formatter. Different flavors of the markup compiler are needed for paper and online since the formatting commands would be different. However, the same markup compiler can handle different languages such as English, European languages, and Asian languages such as Japanese.
- Formatters. One popular formatter is TeX from Donald Knuth at Stanford University. Once the SGML compiler has translated the document into TeX commands, it is run through the formatter to produce the **dvi** or device independent output file. Another popular formatter in the UNIX world is the nroff/troff family.

- Drivers. After the format stage has been completed, the output file needs to be translated into instructions understood by an output device such as a laser printer or typesetter. For instance, after TeX produces a **dvi** file, it can be run through a driver program that outputs PCL suitable for an HP LaserJet or PostScript suitable for a typesetter. The **dvi** output produced by the TeX stage for online processing can be run through a similar dvi driver which outputs a wordprocessor file.
- Printer. If a printer understands a standard format such as PostScript, then it is a simple matter of sending the file to the device for printing. Otherwise, the file produced by the dvi driver might need to be further processed into specific instructions understood by the particular hardware. Quite often, the hardware vendor will provide these device drivers which read standard formats.
- Index/Assemble. The resulting file produced by the dvi driver for online processing must undergo an extra step of fulltext indexing so that speed advantages when retrieving documents can be realized by a user. After that, the resulting files must be assembled into a directory hierarchy which is translated to a format such as ISO 9660 for CD-ROM premastering.
- Electronic warehouse management. The database component keeps track of the inventory of masters in an electronic warehouse. It must be able to accept, deliver and archive masters as well as produce periodic reports. It also orchestrates the production management software which controls the sequencing of the markup, formatting, printing, indexing and assembly tools.

What about desktop publishing software?

Traditional desktop publishing has been characterized by the WYSIWYG nature of their editors. Software such as FrameMaker, Ventura Publisher, Aldus PageMaker and Interleaf TPS are examples of some of these desktop publishing packages.

We offer some reasons why using this class of desktop publishing software without SGML output capability has some disadvantages.

- *Not media independent.* There are some constructs which cannot be translated from WYSIWYG packages into the media we wish to support. Take, for instance, the humble ASCII terminal which is still predominant in the industry. Though it is clear that graphics cannot be satisfactorily represented on an ASCII terminal, the majority of the text that comprises manuals should still be accessible to it's users. WYSIWYG software relies on matching exact fonts specified by the author.
- *Limited indexing.* Most page description languages used by WYSIWYG software limit the amount of indexing that can be done to its text. As such, the search capability is limited to pattern matching whereas an SGML document can be stripped of its tags and fulltext indexed for efficient retrieval.
- *Information cannot be shared.* WYSIWYG software is also tied to a proprietary document format which does not allow the sharing the information easily.
- *No structural information.* This is important when importing a document to different media. With only rendering information available, a document would only make sense if it was presented in the medium for which the rendering was meant.

Retrieval systems

A retrieval system is a set of components that is needed to present information to its human reader.

The retrieval system for paper is fairly universal and needs little explanation: it is the ordinary book. You only need to know how to flip a page. It is somewhat random accessible if you know that a certain piece of information is in the front or back of a book. The page numbers are usually in ascending order starting from page one.

The retrieval system for electronic documents is not as universal as paper. In fact, there are hundreds of vendors peddling different software for displaying and searching against a proprietary database of online documentation. HP LaserROM is the retrieval engine for documentation produced by HP and the X11/Motif version is illustrated below. It also runs on the PC under MS Windows and on regular ASCII terminals on UNIX. For manufacturing efficiency reasons, only one CD-ROM is pressed for all platforms so all the software must understand the same format on the CD-ROM.

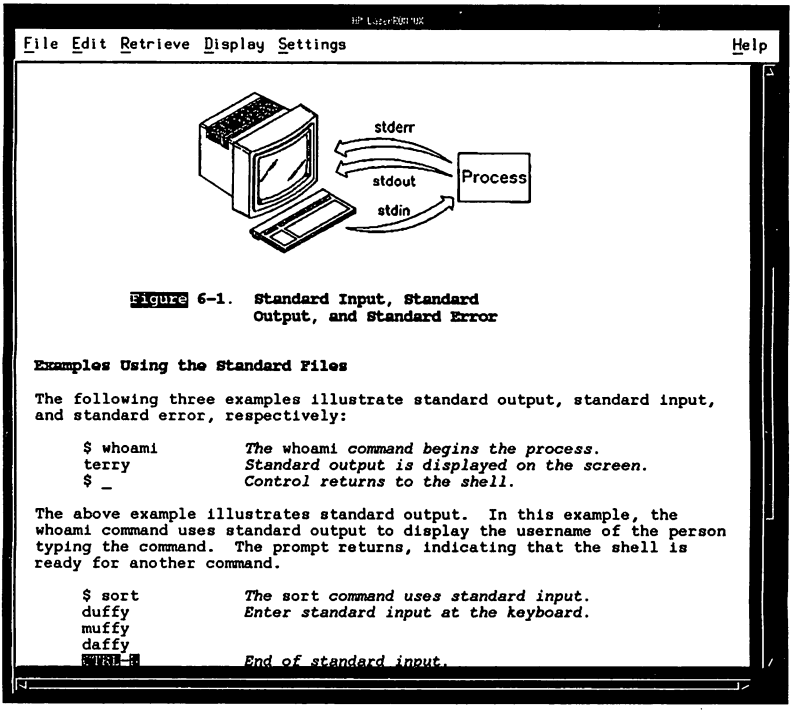


Figure 1-9. HP LaserROM Online Reference User Interface

The Challenges

Electronic publishing is not without its disadvantages, however. Listed are some of the challenges in bringing such a system to reality.

Author involvement

In addition, one of the most important components to assure the success of electronic publishing for paper and online information is to make sure that the writers are aware of the vision and benefits of such a scheme. Quite often, the initial problems faced by authors who have to work with new tools and a seemingly less productive environment than WYSIWYG tools will result in discontentment.

The classic dilemma with standards is that while they provide a uniform means of communication among disparate processes, they can limit the most innovative solutions. For example, the X11 Windows architecture allows standard communication protocols over the network across different platforms and operating systems. However, because it defines a certain set of primitives, it might not allow the most efficient implementation of a 3-D graphics program to run on an architecture that might have special hardware graphics accelerators.

Authors who have spent substantial efforts to select tools and set up an environment that is most productive to their work are not likely to want to adopt a different standard just because it is more efficient for the organization on the whole. While we believe that ultimately, standards will result in tools that benefit everybody, the *Catch-22* is that while a standard is in its infancy, nobody wants to develop tools for it, and nobody would want to adopt a standard if there are no mature tools for it.

Bringing the authors on board early and getting their feedback to help define the tools and standards is vital to the success of a standard. Constantly communicating the benefits of the standards and improving tools and processes for the authors would result in higher levels of cooperation and satisfaction.

Determining documentation standards for paper and online

It was mentioned earlier that defining or selecting standards and implementing them is a big challenge when moving to electronic publishing. Some of the specifics which require careful consideration when using SGML are tables and device independent constructs.

Tables are one of the best and most concise ways of representing information. Authors would spend hours perfecting the look of a table so that it conveys the information in the most readable and comprehensible form. However, once a table has been created for one medium, it has to be reformatted for another medium. For example, creating a table for paper using a variety of fonts and reformatting that table for the ASCII terminal using a single monospaced font would result in some of the hand crafted information infused into the careful layout to be lost. As such, the ways in which tables are created and the amount of control given to the author to specify the layout should be carefully balanced with the problem of formatting that table for different media. In some cases, a standard might allow the use of nice proportional fonts in regular text but only allow one monospaced font in a table so that it can convey the same information regardless of media.

Resolving device dependent features before a standard has been set in stone is also crucial to the success in manufacturing that information efficiently. The first step is to identify the different output media. In HP's case, it was clear that we needed to support paper, workstations using X11/Motif, PC's using MS Windows and ASCII terminals on UNIX. The next step is to determine the lowest common denominator among the media and decide how to represent information on the different media. For the case above, the lowest common denominator is the ASCII terminal. However, it would be far too limiting to only allow plain ASCII files with no graphics. At the same

time, it would be inefficient to let each medium have its own manufacturing process. A good compromise position would be to allow paper to have all the typesetting capabilities available in its process. The online standard would be to allow plain text and graphics only.

For example, complex formulas and equations must necessarily require arbitrary typesetting using different fonts to properly convey the idea. This presents a problem as online media might not allow arbitrary typesetting and use of different fonts. A simple resolution can be to convert all such formulas into a standard graphics format which can then be presented on bitmapped screens. ASCII terminals would not have the same capability but the user should still be able to print out the formula onto an output device such as a laser printer.

The final result should be a standard which should give an author sufficient freedom of expression to convey ideas but yet not include any special formatter commands to adjust layout for any particular medium. Having these special layout commands for one medium and not another would often result in loss of information. Having the author provide equivalent special commands for each medium is neither efficient nor portable. Certainly commands that enhance the value of a specific medium (such as hypertext or color for online documentation), but do not affect layout and can safely be ignored for paper, should be allowed.

Startup Costs

In order to support the manufacturing and distribution of information on CD-ROM, there are additional costs involved. These include the overhead of a staff of computer programmers and operators to manage and support the electronic warehouse. This alone can add over $100,000 to the costs. Computers are vulnerable to power outages and system failures which can prevent orders from being filled in a timely manner.

The startup time and costs to set up such an operation can also present many hurdles. In paper publishing, a good camera-ready master is all that is needed. It is a simple standard which cannot easily be violated.

In electronic publishing, there are *many* standards. For example, a document could be sent electronically through a variety of means such as ftp, mail, or some other custom transfer mechanism. Also, the submittal standards must be defined, such as which files have to accompany the master and how it should be identified and processed. The document itself can come in different formats. At HP, we standardized on an implementation of SGML called HP Tag. Other companies may choose page description languages such as PostScript or PCL, or some word processor format. Graphics formats can be TIFF, HPGL, PCL, Postscript or myriads of others, each having its own idiosyncracies.

Therefore, authoring tools and guidelines have to be developed to ensure that these standards are met. In the event that standards are violated, which happens quite frequently as a result of ill-defined standards or ignorance, manuals must be reworked by hand and reprocessed, adding time and cost to manufacture that manual.

On the whole, the advantages of electronic manufacturing far outweigh the disadvantages when volume is high and people have had time to iron out the kinks in the process.

Looking into the Crystal Ball

Some of the emerging technologies in the world of computers would make online documentation much more natural. Anticipating some of these advances would help us better prepare for them by making sure that authoring and production tools are available.

- Multimedia Documentation. Producing multimedia information can be a very time consuming, manual and costly process. Adopting standards such as SGML and implementing an efficient production system will provide the infrastructure for manufacturing multimedia documentation.
- Managing the Information Explosion. There is so much information being produced that much of it seems like meaningless data; until you really need it and cannot find the information you want amidst the gigabytes of information available. There has been some interesting work done on filtering "interesting" information from live feeds, such as making a customized newspaper from wire services. In addition, expert systems will be able to help formulate correct queries for databases to find what you are looking for when you do not know exactly how to express the query.
- Bridging the Paper-Online chasm. Transforming online information into paper is easy, with the exception of multimedia documentation. However, getting the stacks of paper documentation to an online format is not as easy. Yet there is great demand in the research field to be able to search across all the printed material regarding a particular topic without having to take many years to do so. With the maturity of optical character recognition (OCR) technology, which scans in a page and converts it into text, that possibility of having any printed or even handwritten documents stored electronically might be a reality soon.
- Concurrent engineering and documentation. Many documentation groups are far less integrated with the development lab than we might think. Consider integrating the documentation process electronically. Tables of information in a database can be updated automatically by the lab; then a converter automatically translates the information into SGML for document production. Sign-off from various groups and individuals can be done electronically, as well as querying the stages of a document's readiness for publication, and for distribution.
- "Modular" documentation. By using SGML, we can organize documentation in a modular way, and thereby reuse common structures. Legal pages, common hardware specifications, for instance, can be created once, and shared by many documents. When updating information, the only the shared structures need to be updated.

Conclusion

Publishing for paper and online is only the beginning. As new types of media emerge for accessing and viewing information, so will the need to create and deliver those media. However, with increasingly competitive markets worldwide, delivering quality documentation on any kind of media needs to be done cheaper and faster. We believe that the general approach we have outlined here, single source with multiple outputs, will be an important trend in the future of paper and online publishing.

Bibligraphy

1. International Organization for Standardization, "International Standard ISO 8879, Information processing - Text and office systems - Standard Generalized Markup Language (SGML)", First Edition (1985), Ref. No. ISO 8879-1986 (E)
2. Herwijnen, Eric van, "Practical SGML", (1990), Kluwer Academic Publishers.
3. Goldfarb, Charles F., "The SGML Handbook", (1990), Oxford University Press.
4. Horton, W. K., "Designing and Writing Online Documentation", (1990), Wiley & Sons, New York.
5. HP Data Sheet, "HP LaserROM Information Tools", 5952-0258 (1990), Hewlett-Packard Company, Palo Alto.
6. Wurman, R. S., "Information Anxiety", (1989), Doubleday, New York.

Paper 2040

OSF: Open Systems Through an Open Process

Rod Johnson
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
617-621-8700

Introduction

The computer industry is on the brink of explosive change, with respect to technology as well as with respect to who is in the driver's seat.

For 30 years, vendors have been producing a range of mutually incompatible systems that have locked in users to proprietary architectures and operating systems, making it difficult or impossible to interconnect systems from different vendors, to move applications from one platform to another, or to host the same software environment on systems of different sizes and capabilities.

Today, users have lost patience with such restrictions and are demanding "open systems": systems based on industry standards.

It is important to note at the outset that "openness" is not synonymous with the use of the Unix® operating systems. It is not a function of operating system software at all. A truly open computing environment would employ a standard set of interfaces for programming, communications, networking, system management, and user "look and feel," so that software applications would become uncoupled from the platforms on which they run.

This white paper discusses the historical evolution of the open systems movement, including the development and role of the Unix system; the advantages of open versus proprietary systems, the benefits of open computing to various classes of users, and the role of the Open Software Foundation in the open computing movement.

The Evolution Of Openness In Computing

Sophisticated computer users have known for years that the information their systems contain is a strategic resource, and that it becomes even more valuable when it is shared. With closed systems, information sharing can be difficult.

But when computer hardware prices began to tumble several years ago, many users realized it now made economic sense to digitize information and transmit it from place to place over telephone lines. Very quickly it became apparent that having standard computer interfaces would greatly facilitate such communication.

Although standardization was familiar in the communications industry, it was foreign to computer makers. At first, the idea met resistance. Gradually, however, standards bodies began to formulate protocols designed to make communication between dissimilar systems easier.

In 1984, the International Standards Organization (ISO) published a 7-layer model known as Open Systems Interconnection (OSI) to serve as a framework for structuring communication between separate end users. With that framework in place, more detailed specifications for the interface were developed.

In the United States, the National Bureau of Standards and others developed an Applications Portability Profile (APP). The IEEE spurred development of the Portable Operating System Interface (POSIX), and the Corporation for Open Systems was formed to foster open networking standards. In 1985, a Europe standards organization called X/Open was formed which has developed the Common Applications Environment specification (CAE).

The last few years have seen the development of broad-scale, world-wide networking, in which hundreds of host systems, file servers, workstations, and PCs, often from different vendors, may be linked together. Some of the world's largest manufacturers have forged enterprise-wide networks that interconnect not only their own internal organizations but their suppliers as well.

In an ideal world, such networks would operate as giant parallel processors, transmitting information from place to place with complete transparency. In fact, of course, the structure and management of large heterogeneous networks is extraordinarily difficult and complex.

The result is that in today's information processing industry, the need for openness based on standards accepted world-wide, across the industry, is acute. Standards break down barriers: barriers that keep different vendors' systems from communicating; barriers that make it difficult to port applications from one platform to another; barriers that make it difficult to move files between systems of different sizes.

Open Versus Proprietary Standards

In a world in which vendors provide mutually incompatible systems, vendors' installed bases of equipment have important ramifications for competition, because the costs to the user of switching from one vendor to another are so high. This is vendor "lock-in."

The extent of vendor lock-in goes far beyond the original investment in hardware. Over the years, users with proprietary systems write their own software; train employees in system-specific skills; and write supporting documentation. When they need new systems, even for completely separate purposes, they are likely to purchase from the same vendor, in the event that at some point they want to interconnect these stand-alone systems with others. Frequently, these purchase decisions are made in favor of the dominant vendor even when another vendor may offer functionality that the organization needs -- but cannot get from the vendor of its installed systems.

In addition, independent software vendors find it uneconomical to write applications software for vendors with small installed bases, giving the largest system vendors still another edge.

Thus installed bases have a tendency to be self-perpetuating. In general vendors that have been successful in the past continue to be successful, and their users remain captive. Problems for users arise when the current vendor no longer meets, or is responsive to, the users' needs.

For these reasons, users have rebelled against the closed system.

The Role Of The UNIX® System

The Unix system was created in 1969 in an industrial laboratory setting: AT&T's renowned Bell Labs. Since the beginning, it has been known as an "open" operating system, easily portable to a variety of hardware platforms and especially well suited to engineering and technical applications.

Because its source code was easily and inexpensively licensed, the Unix system became popular in colleges and universities, where it has been modified extensively over the years. Today, some 200 variants exist, the most widely used being those derived from work done at the University of California at Berkeley. A number of proprietary operating systems based on the Unix system also have been developed, including Xenix™ (Microsoft), ULTRIX™ (Digital Equipment Corporation), Sinix™ (Siemens) and AIX™ (IBM).

According to the International Data Corporation, the Unix system now controls 9 percent of the worldwide hardware market. It is the third most important operating system environment, and by 1992 it will vie for number 2 position. By that time, says IDC, "industry-standard computing, once the exception, will become the rule."

The greatest growth in Unix systems is occurring in small-scale systems, particularly reduced instruction set (RISC)-based workstations and network servers. This is especially significant as the traditional time-sharing model of computing, in which one host system serves many users, gives way to the client-server model, where computation and storage are shared by workstations (clients) and host machines (servers).

It's also significant that the Unix system delivers some of the best price/performance in the industry, and that it is now well accepted by Japanese and European vendors.

Many users see systems based on Unix technology as having the best potential to provide solutions for today's most pressing computing needs: better application portability, improved network support, and preservation of investment in applications and databases. So over the years many groups and bodies have been formed in an attempt to standardize Unix-based systems. But standards bodies do not deliver code.

Formation Of The Open Software Foundation

The Open Software Foundation was formed in May of 1988 specifically to develop core software technologies and supply them to the entire industry, on fair and reasonable licensing terms. Although OSF is using established Unix technology as the basis for its initial software development, its objective is not to develop the definitive version of the Unix system.

OSF's objective is to broaden the definition of openness in computing by providing users with the greatest possible portability, interoperability, and scalability of applications. The Foundation will achieve that goal by providing a standard interface between application software and any computer system that conforms to that interface, including standard networking protocols for seamless interconnection between disparate systems and standard interfaces to peripheral devices.

The ultimate goal, a truly open computing environment which would make possible truly distributed networks, could increase service to users by orders of magnitude. In a distributed networked environment, computing could occur transparently across the network; rather than dedicating computing power to one machine, or application, functionality and power would be shared seamlessly throughout the environment.

Such an environment would not mean the elimination of proprietary operating systems. It would provide a point of connection to them, so that vendor lock-in could be avoided. Because applications could be purchased from any software vendor whose products were compatible with the application interface, independent software vendors would have an enormous incentive to provide more applications software.

End users who wanted to upgrade, add standalone systems but retain portability, network, or add memory could make their purchases from whichever vendor offered the most value.

Once standard interfaces were accepted, hardware vendors would have the incentive to differentiate their products in more meaningful ways than they have in the past, as manufacturers of cars, audio equipment, VCRs and cameras have done.

Ultimately, an open environment would mean better worldwide communications capabilities, more effective use of users' investments in technology, and faster development and adoption of new technologies.

Who Would Be The Beneficiaries of Open Systems?

The Open Software Foundation believes four broad categories of users will benefit from its work: system vendors, independent software vendors, OSF members, and end users.

1. System vendors

For hardware and system vendors, the greatest benefit of the open computing environment will be a lowering of their investment in developing operating system software, resulting in faster time to market for new products.

Freed of the necessity to "tweak" their operating systems for each new revision, vendors will be able to innovate and differentiate in architecture, features and functionality. They will compete and add value on the basis of specialization, quality, solutions, and service: factors that not only will serve users but allow for adequate margins as well.

2. Independent software vendors

Independent software vendors will be clear winners in an open computing world. ISVs will develop products that adhere to a standard set of application interfaces, rather than to a set of proprietary operating system interfaces, assuring a wider market for new products.

3.OSF members

The Open Software Foundation is developing its products through the "Request For Technology" process, under which proposals for products to be developed are evaluated by the membership. The entire process is open to scrutiny, by members, by press, and by the world at large.

The chief benefit of membership in OSF is the privilege of participation in the open process, as well as advance exposure to research in progress. These advance "snapshots" enable members to develop their own key applications in parallel with the OSF effort. In addition, their corporate officers can plan when and how to use new technologies for best competitive advantage.

4. End users

For end users, the most compelling benefit of an open computing environment will be the knowledge that, for the first time, they will be able to purchase hardware without fear of backing the wrong horse, and be assured of having the same operating environment with

any system: microcomputer, workstation, minicomputer, mainframe or supercomputer. Mixed-vendor environments will be simplified, as will networking.

Software development, training and maintenance costs will be lowered. More off-the-shelf applications will be available, at more competitive prices. In fact, once user lock-in is reduced, increased competition should lead to lower prices across the industry.

Individual operating units of major corporations will be free to choose the products that best serve their purposes without compromising corporate synergy.

Finally, corporate officers will be able to make better use of new technologies for competitive advantage, without fear of risking their existing investment in software and databases.

Conclusion

International Data Corporation wrote, in a 1988 report on the Unix systems market from 1987 through 1992, "Ultimately, standards will accelerate the rate of innovation in our business and lead to a much wider array of choices for the user and strategies for the vendor. . .

"As IDC looks at the future of open systems, we see [the layers of software outside the operating system] -- the user interface, the network architecture, the programming languages, the database management system -- becoming the real determinants of system flexibility and portability. The irony may be that as these outer layers define themselves, the requirement for a standardized internal design may in fact go away. . . .Perhaps the final legacy of Unix (sic) will turn out to be that it sowed the seeds of its own destruction."

Whether that happens or not, it seems clear that users of computing systems can only benefit from greater openness across systems.

TITLE: OSF/1: The HP Perspective

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2041

TITLE: OSF: Distributed Computing Environment "DCE": The HP Perspective

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2043

TITLE: OSF: Distributed Management Environment

"DME": The HP Perspective

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2044

Paper 2045

Applications Neutral Distribution Format for Application Portability, and Open Systems

Rod Johnson
Open Software Foundation
11 Cambridge Center
Cambridge, MA 02142
617-621-8700

ANDF, Application Portability, and Open Systems

The ability to run a single version of a software application on different hardware platforms is one of the prime open systems goals. End users demand it. Software developers need a way to provide it. Hardware vendors can help achieve it. For all parties, the economic benefit of application portability is compelling motivation for working together to realize it.

The current path toward application portability is to write applications in a way that is hardware-independent, using a standard programming language and, perhaps, a widely available and accepted graphical user interface. For open systems software development, this language is most often ANSI C, and the graphical user interface is most often based on the X Window System.™ But what end users and software developers have found is that these efforts are not enough. Something is missing from the application portability equation.

Today, users must purchase and install a different version of each application they want to use for each different computer architecture they want to run it on. In some cases, a version for each machine is not available. Independent software vendors (ISVs) must, at the least, modify, recompile, repackage, and redistribute their applications for each hardware platform they can afford to support.

The result, for end users, is multi-vendor open systems hardware environments that are not as fully configurable as they could be, and do not deliver the full economic and human benefits imagined for them. For ISVs, the result is the inability to capitalize fully on the market potential of their products and the need to divert resources from product development into product porting and maintenance. For hardware vendors, the result is the limited availability of applications for innovative systems and, consequently, a higher risk associated with innovation.

SOLVING THE APPLICATION PORTABILITY EQUATION

Achieving the application portability envisioned for open systems requires the convergence of several trends in computing as well as the existence of a key technology (Figure 1).

These trends are in motion today and include

- The proliferation of standard computing environments — the application programming interfaces (APIs) for user interface, operating system, networking, and other services provided to applications by the system — on a wide variety of hardware architectures
- The use of standard programming languages (such as ANSI C)

- The use of programming techniques which do not make assumptions about hardware-specific attributes for application performance or functionality.

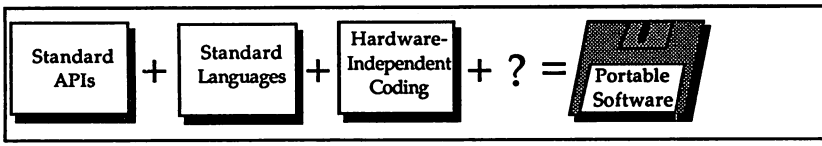


Figure 1. The application portability equation.

ANDF IS THE KEY TECHNOLOGY

Because applications are usually distributed in a compiled, "object code" form, and object code is specific to a particular hardware architecture, even an application using 100% portable code must be recompiled and repackaged for distribution to each hardware architecture. Because each hardware architecture requires its own compiler, differences in compilers can introduce unexpected changes in application behavior or can even cause an application to fail to compile. As a result, the application source code often must be changed a little each time this happens. What began as portable code is no longer portable.

The OSF Architecture-Neutral Distribution Format (ANDF) enables developers to compile and distribute their applications in a form that can be installed and run on any hardware architecture that supports ANDF. By providing this capability, ANDF enables the solution of the application portability equation (Figure 2).

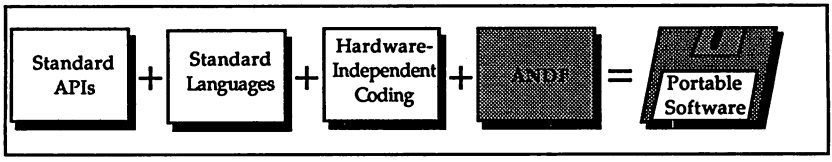


Figure 2. The application portability equation solved.

ANDF establishes a software distribution method based on a compiler intermediate language which adequately balances the need for hardware independence and source-code confidentiality. This method essentially divides the source code compilation process into two parts -- one machine-independent and the other machine-dependent. Machine-independent processing is done with a tool, the ANDF producer, which software vendors will use to produce the ANDF code for mass distribution. Another tool, the ANDF installer, performs the machine-dependent processing on the end user's system and produces executable code for that system at install time (Figure 3).

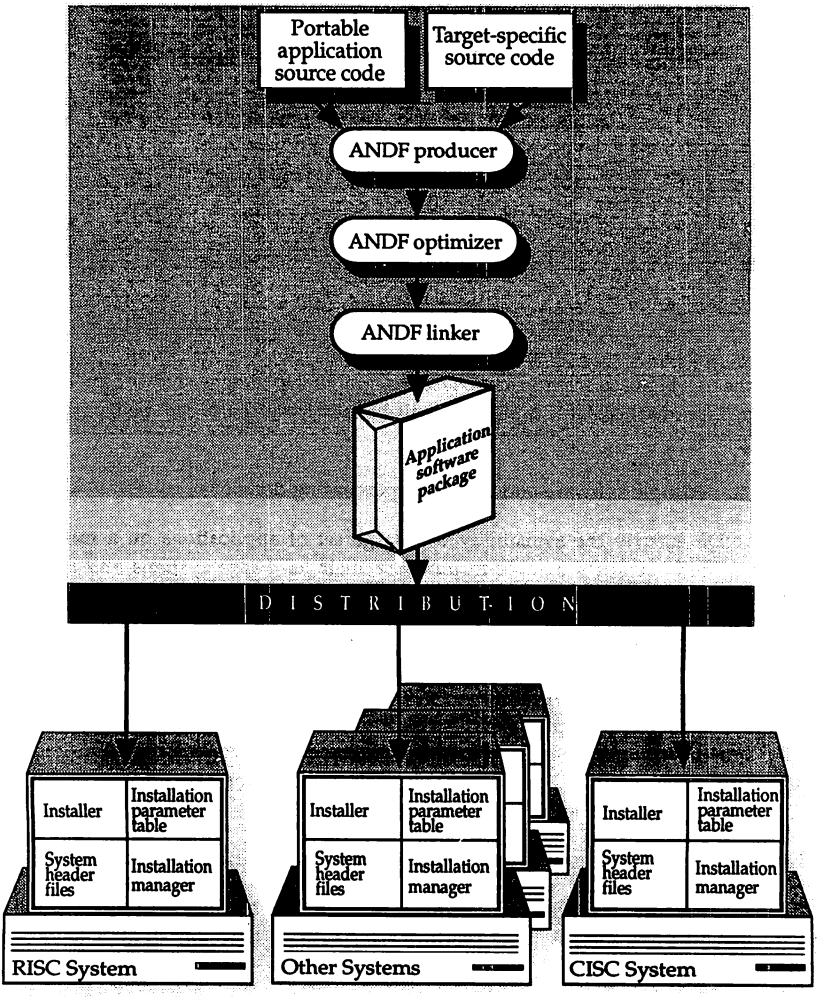


Figure 3. Developers use the ANDF producer, optimizer, and ANDF-to-ANDF linker to create the ANDF application package for distribution. On the user's system or through a network server, ANDF technology supplied by the system vendor guides the installation process.

ANDF BENEFITS

ANDF's fundamental effect will be the elimination of hardware and software interdependency, to the benefit of end users, software vendors, and system vendors.

End User Benefits

End users of open systems are the ultimate beneficiaries of ANDF technology. They will enjoy numerous benefits, including

- Increased availability of software for open systems.

  ANDF is expected to attract mass market applications — the word processors, spreadsheets, database management systems, graphics packages, and others — to the open systems marketplace. End users will benefit from the broadened usability of these systems.

- Ability to decouple software and hardware purchasing decisions.

  Since ANDF ensures the availability of a large set of applications on a number of platforms, end users will be able to purchase their hardware systems and software independently. This will simplify their purchasing process.

- Ease of distribution.

  Corporate users who buy application software in volume will find that ANDF allows them to distribute the applications throughout various departments easily without concern for the diversity of the hardware. And corporate users who develop software will be able to use ANDF to ensure that their applications can operate on all the hardware used throughout the company.

- Reduction in training costs when users move from one platform to another.

  The investment in employee training to learn an ANDF application on a particular platform will carry over to new platforms because of the consistent behavior of ANDF applications.

  With the availability of the same applications on a variety of platforms, the transition for end users from an obsolete to a new platform will be less expensive.

- Increased longevity of software investments.

  Since ANDF will increase the longevity of software by decoupling it from hardware obsolescence, end users will have better protection of their software investments.

  ANDF applications will run on new hardware that supports ANDF.

- Scalability of applications and interoperability of various machine architectures.

ANDF will facilitate the availability of a particular application on all types of machines — from microcomputers to supercomputers. This will provide better scalability and interoperability at the application level.

- Reduced need to upgrade an application for new system versions.

  End users will be able to limit software upgrading to new releases of applications offering additional functionality, as is the case in the PC market — whereas today they often must upgrade their applications merely to support new versions of an operating system. In the future, they will be able to re-install the original ANDF application by using a new version of the installer, supplied by the system vendor, that is synchronized with the new version of the operating system.

Software Vendor Benefits

Software vendors will benefit because ANDF will lead to cost reductions in software development, maintenance, testing, and distribution. ANDF will enable software vendors to reapportion their resources to create new functionality and new applications rather than devoting resources to porting and maintaining multiple versions of a single application. For software vendors, ANDF is more than just a software distribution vehicle. It enables them to support the diverse open systems market with a single version of their software product. This increases the potential sales volume of a software product by enabling vendors to target a larger market. In addition, ANDF allows a software vendor to create an application that can run on (and generate revenue from) a future architecture without alteration of the original product. This capability is unprecedented.

- Simplified software development.

  ANDF will simplify software development and increase product consistency across platforms. A vendor will ship a single version of an application produced using one ANDF producer for all platforms. This will solve the problem experienced as a result of today's practice of shipping a binary version of an application for each platform. That approach can lead to inconsistencies from platform to platform due to variations in compiler implementations.

- Reduced maintenance costs.

  The ability and the need to limit development to a single source version of an application will lead to reduced maintenance costs. Also, compiling source with a single ANDF producer for the various target platforms rather than using multiple compilers will reduce maintenance costs.

- Reduced testing costs.

ANDF will lead to a significant reduction in costs associated with testing applications for new versions of an operating system. Because ANDF installers will be synchronized with new operating system versions, many operating system changes will be transparent to the application developer and the application. The result will be a reduced need to change and test ANDF applications.

- Reduced manufacturing and distribution costs.

  ANDF enables ISVs to package one version of an application to serve many different hardware architectures. Producing only one version of each application will cut inventory and distribution costs. The money saved could go into new product development or improving existing products.

System Vendor Benefits

To stay competitive, system vendors bring to market new hardware and system innovations or enhancements of system performance and features. But for each innovation, system vendors must spend time and money helping ISVs port their applications to new hardware or operating systems. ANDF reduces the cost and the time required for these efforts by preserving the application base. This allows system vendors to focus more time and effort on improving their platforms. Specifically, the major benefits of ANDF to system vendors are

- Immediate access to an application base for new architectures.

  System vendors will be able to launch new, high-performance hardware without fear that a lack of applications will stunt acceptance. Today, when system vendors introduce a new platform, they expend considerable time and money to negotiate and obtain an agreement with software vendors to port their software to the platform. With the ANDF model, the entire set of ANDF applications is readily available to be used on a new platform as soon as an ANDF installer is implemented for it. This will lead to a substantial reduction in time-to-market for new hardware and in the cost of bringing new and existing applications to the new platform.

- Freedom to create new hardware and software while preserving the application base.

  Modifying an existing ANDF installer to conform with enhancements to a platform enables use of those enhancements — such as pipelining, multiprocessing, or 64-bit enhancements in hardware, or the implementation of shared libraries, dynamic linking, or an enriched object format in operating environments — without losing access to the platform's application base.

  The insulation of application developers from variations in the target operating system is an important advantage of ANDF. It provides a layer under which operating system implementors have room for variations in implementations.

- Reduced cost to carry a diverse product line with multiple architectures and operating systems.

  Most large system vendors support multiple architectures and operating systems. ANDF will enable vendors to demonstrate a set of applications which work across a heterogeneous product base. This will ease a customer's migration from past architectures to future architectures. Also, over time, system vendors will notice a significant reduction in the cost of compiler development and compiler-related tools, due to the streamlining and sharing of compiler technology across their product lines.

ANDF IN ACTION

End Users

For end users, ANDF application installation will proceed in a familiar way. A user will insert the application distribution medium (such as a disk or tape) in a machine and invoke the application's installation procedure. From that point on, installation is guided by the installation procedure and the system's resident installation manager. User interaction will be minimal, as it is today.

ANDF also enables end users to take advantage of alternate methods of application installation. For example, a network software server in a corporate network can generate executable binaries from ANDF for each system in the network. This takes advantage of the server's processing power and simplifies accounting for each binary version of the application distributed.

Software Vendors

ANDF provides the essential technology required by software developers to create and distribute truly portable applications for the architecturally diverse open systems marketplace. Because ANDF requires the use of standard application programming interfaces (such as POSIX and XPG3), it also provides a significant incentive for developers to increase their use of these standards. The adoption and use of these standards is essential to the success of the open systems industry.

By using ANDF, software vendors will reap the benefits of the decoupling of software from hardware while preserving the same protection of their proprietary information as provided by object code distribution.

Software vendors developing ANDF applications will continue to use many of the same tools to create, debug, and optimize their source code as they do today. But instead of using multiple compilers and linkers to create application packages for multiple platforms, developers will use a single ANDF producer and linker to create one application package for all ANDF-compliant systems. The producer is invoked in the same way as a compiler, giving it command line options and names of source files to process.

The installer on a user's machine completes the compilation process, turning the ANDF application into executable binary form and calling the native linker to combine the binary objects and native libraries.

System Vendors

OSF will provide reference producers and installers for several platforms. System vendors (and compiler vendors) can use these reference implementations to move the producer to, and to create installers for, particular platforms.

The effort to create an installer for a platform will be approximately the same as writing a code generator for that platform. This typically translates to about one person year.

Once an installer is created, tested, and shipped with a platform, its users will have immediate access to the entire range of ANDF applications.

By adopting ANDF technology, system vendors will jump start the industry's movement to standards-based computing and enable application portability to the entire range of platforms in open system environments.

CONCLUSION

Much progress has been made in recent years toward achieving the goals of the open system movement. Standards from the IEEE POSIX group, and specifications from OSF (embodied in the Application Environment Specification, or AES), from X/Open (embodied in the X/Open™ Portability Guide, or XPG3), from AT&T (in the System V Interface Definition, or SVID), and from others help to specify operating systems, high-level languages, user interfaces, graphics libraries, and networking services for open systems.

ANDF is the catalyst, when used in combination with these standards and specifications, that will enable the open systems industry to achieve the goal of application portability in diverse, multi-vendor computing environments. End users will not need to standardize on a single platform to benefit from the availability of off-the-shelf software applications. Instead, they will be able to choose the hardware that best meets their immediate needs for performance and economy while retaining the flexibility for future upgrades. Moreover, they will enjoy these advantages without compromising their software base.

By decoupling software from hardware architecture, ANDF enables the distribution of mass market software — the word processors, spreadsheets, database management systems, graphics packages, and others — for open systems. With the greater availability of these applications, users can more broadly apply their open systems to the business problems they face each day.

Finally, ANDF technology is unobtrusive. It fits easily into the way that developers and end users are used to working. ANDF establishes a new, more rational model for software development, distribution, and purchasing in the multi-architecture open systems marketplace.

WHAT IS A SYSTEM ADMINISTRATOR, ANYWAY?

Kathleen M. Sagunsky
James P. Langan
J.P. Langan & Associates, Inc.
2800 South Fish Hatchery Road
Madison, Wisconsin 53711
(608) 273-0428

The UNIX operating system, developed in the scientific/technical arena during the early seventies, has always enjoyed great popularity there. Today, because of the increasing interest in open systems, UNIX has begun to penetrate the commercial arena as well. Companies embarking on all types of system conversions are looking at software that runs on UNIX. It is no longer uncommon for a company to find a business application that fits its needs, only to discover later that the application runs on top of a specific operating system -- increasingly UNIX.

But UNIX has never been known for its user-friendliness. Its command language is cryptic, and its terminology novel, to say the least. For example, who would guess that "grep" is an acronym for "Grab Regular Expression and Print?" And what about those little "daemons," hard at work behind the system scenes without need for human supervision of any kind? It's easy to understand why new users are intimidated by this kind of jargon and why the presence of an in-house UNIX expert can go a long way toward raising the comfort-level for novice users.

The applications found running on a typical UNIX-based system are varied. They could include a fully integrated business system, along with office applications such as word processing and spreadsheets. The system might also have one or more resident databases and specialized software packages such as Computer Aided Design (CAD).

The users of these types of applications all rely on the availability of computer resources to do their jobs. And, clearly, a system like this requires a certain degree of management to ensure efficient use of resources. Enter the System Administrator or "SA", the in-house expert mentioned above.

Resource management, one of the SA's primary responsibilities, encompasses all of the items listed below. These are the responsibilities which require technical expertise. But the SA is much more than a technician. One of the greatest assets an SA can possess is the ability to work well with other people. The task of training users, teaching them how to interact with the system, recovering from their predictable, albeit inevitable mistakes with patience and grace -- all are equally important SA responsibilities.

SYSTEM ADMINISTRATOR RESPONSIBILITIES

1. System Configuration

   Managing Peripheral Hardware

2. Account Management

3. Establishing / Enforcing System Security

4. Selecting / Developing User Interfaces

5. Maintaining File System Integrity

6. System Housekeeping

   Backup and Recovery

Entire books have been written on the subject of System Administration. This paper is intended to act only as a primer of the most elementary responsibilities of an HP-UX System Administrator. Examples of some of the more common administrative commands will be demonstrated. Personal experience in the role of System Administrator supplemented by the HP-UX System Administration Tasks and Concepts Manuals provided the resources for this paper.

1. SYSTEM CONFIGURATION

An important aspect of system management is knowing the required and anticipated hardware/software computer resources necessary to fulfill the needs of its users. Whether you are purchasing your computer and applications from a vendor or you are developing your own HP-UX applications it is important for you to understand the principals of system configuration. A properly configured system will have few problems, and allow you the time necessary to perform those routine tasks demanded by your users, i.e., backups, account management, operating system revision changes etc... A poorly configured system may display one or more problems, i.e., slow response, can't log in, programs that crash, etc... and, most importantly, user complaints.

To properly configure a system the SA must understand the requirements of: 1) operating system (HP-UX), 2) user applications, 3) development software (if any exists), 4) individual users, and 5) hardware. Series 300 and Series 800 platforms have different HP-UX requirements. EXHIBITS A and B are **ps** (program status) listings from an HP/9000 model 350 and model 835 respectively. The

hardware configurations for these systems follow:

Model 350 Hardware	Model 835 Hardware
16 Mbyte memory	48 Mbyte memory
1 x 4 channel mux	1 x 6 channel mux
2 x RS232 interface	1 x 16 channel mux
1 x HPIB disk interface	4 x HPIB (disk interface)
1 x LAN (ethernet)	1 x LAN (ethernet)
1 x 6 color monitor	1 x SCSI
	1 x backplane expander

The Series 300 HP-UX/7.0 kernel requires approximately 750 Kbytes of memory in a minimum configuration. In this example system, the kernel requires 948 Kbytes. It takes approximately 120 Mbytes of disk space to install all of the features included with HP-UX/7.0. The **ps** listing for this model 350 system indicates a memory usage of 1.9 Mbytes giving a total of 2.8 Mbytes (1.9 Mbytes + 948 Kbytes). Notice that several of the programs listed have a size of 0. These are programs which are disk swapped and memory must be available for them at some future time.

Series 300 HP-UX/7.0 operating on a small system without X-Windows will perform well in 4.0 Mbytes of memory. This particular system has three X-terminals which were not logged on at the time of this PS listing. Each one of these X-terminals requires approximately 4.0 Mbytes of memory, giving an overall system requirement of 16 Mbytes.
The series 800 HP-UX/7.0 kernel has a minimum memory requirement of 1.8 Mbytes. The kernel in this example model 835 requires 2.1 Mbytes of memory. Series 800 HP-UX/7.0 requires approximately 200 Mbytes of disk space. This model 835 **ps** listing indicates that 22.3 Mbytes of memory are required for the executing program, giving a total memory requirement of 24.5 Mbytes (2.1 Mbytes + 22.3 Mbytes). This system requires 48 Mbytes of memory because it is a server and host for several users including: 1) disk-less model 345, 2) model 350, 3) four X-terminals, and 4) 10 standard terminals.

It is very difficult to specify the correct hardware and disk storage necessary for an HP-UX system because of the wide range of applications which are currently in use. System memory is much easier to specify. The following are our recommendations for memory allocation:

1) allow 2.0 Mbytes for the HP-UX kernel
2) allow 4.0 Mbytes for each X-terminal
3) allow 100 Kbytes for each terminal
4) allow 500 Kbytes for each logged on user
5) allow 1.0 Mbyte for each LAN interface
6) allow 100 Kbytes for each on-line disk
7) add special application requirements, e.g. CAD, DBMS ect...

EXHIBIT A Program Status Listing of HP/9000 Model 350

```
F S UID   PID PPID C PRI ADDR    SZ  TTY     CMD
3 S root    0    0  0 128 ff138    0  ?       swapper
1 S root    1    0  0 168 ff305   35  ?       /etc/init
3 S root    2    0  0 152 ff2fb 1072  ?       pagedaemon
3 S root    3    0  0 100 ff301    0  ?       netisr
1 S root  132    1  2 168 ff3d4   70  console  -csh[csh]
1 S root   38    1  0 168 ff429    9  ?       syncer
1 S lp     42    1  0 154 ff3fd   56  ?       /usr/lib/lpsched
1 S root  114    1  0 154 ff676   43  ?       /etc/cron
1 S root   49    1  0 154 ff4a4   48  ?       /usr/bin/netlogstart
1 S root   57    1  0 154 ff4ee   21  ?       /etc/rlbdaemon
1 S root   60    1  0 154 ff514   28  ?       /etc/sockregd
1 S root   65    1  0 154 ff53d   37  ?       /etc/syslogd
1 S root   70    1  0 154 ff57c   30  ?       /etc/portmap
1 S root   88    1  0 154 ff661   56  ?       /etc/inetd
1 S root   72    1  0 154 ff597   25  ?       /etc/nfsd 4
1 S root   74   72  0 154 ff5a5   25  ?       /etc/nfsd 4
1 S root   75   72  0 154 ff5ab   25  ?       /etc/nfsd 4
1 S root   76   72  0 154 ff5bb   25  ?       /etc/nfsd 4
0 S root   77    1  0 154 78000    0  ?       /etc/biod 4
0 S root   78    1  0 154 8a000    0  ?       /etc/biod 4
0 S root   79    1  0 154 8f000    0  ?       /etc/biod 4
0 S root   80    1  0 154 94000    0  ?       /etc/biod 4
1 S root   83    1  0 154 ff601   52  ?       /usr/etc/rpc.statd
1 S root   85    1  0 154 ff630   58  ?       /usr/etc/rpc.lockd
1 S root   99    1  0 154 ff6cb   30  ?       /etc/rwhod
1 S root  116    1  0 155 ff72d   34  ?       /etc/ptydaemon
1 S root  108    1  0 154 ff6f3   20  ?       /usr/bin/nftdaemon
1 S root  110    1  0 154 ff715   27  ?       /usr/bin/rfadaemon
1 S root  119    1  0 154 ff73b   26  ?       /etc/vtdaemon
1 R root  185  132 12 181 ff8ac   44  console  ps -adelf
```

EXHIBIT B Program Status Listing of HP/9000 Model 835

```
F S UID   PID PPID C PRI ADDR      SZ  TTY  CMD
3 S root    0    0  0 128 373800     0  ?   swapper
1 S root    1    0  0 168 3742c0   380  ?   init
3 S root    2    0  0 152 966580  8192  ?   pagedaemon
3 S root    3    0  0 128 967040     0  ?   statedaemon
3 S root    6    0  0 152 96c300     0  ?   syncdaemon
1 S root  692    1  0 168 139b080  680  ?   -csh[csh]
1 S root  173    1  0 156 a31340   220  ?   /etc/getty -h ttyd0p1...
3 S root    4    0  0 154 a3de00     0  ?   lcsp
3 S root  121    4  0 153 ac28c0     0  ?   gcsp
1 S lp     47    1  0 154 b66380   556  ?   /usr/lib/lpsched
1 S root   43    1  0 168 b1de40    80  ?   syncer
1 S root   54    1  0 154 b1e900   436  ?   /usr/bin/netlogstart -d...
1 S root   62    1  0 154 bdbbc0   184  ?   /etc/rlbdaemon
-
```

```
1 S root 129   1  0 154 e222c0  452  ?   /etc/cron
1 S root 131   1  0 155 e84d80  340  ?   /etc/ptydaemon
1 S root 142   1  0 154 eb2840  496  ?   /ect/delog -e 2
1 S root 174   1  0 156 f34300  220  ?   /etc/getty -h ttyd0p2...
1 S root 175   1  0 156 9e5dc0  236  ?   /etc/getty -h tty0p3...
-
1 S root 179   1  0 158 f0b8c0 1256  ?   /usr/bin/X11/xdm
1 R root1323 692 12 181 fd2b80  412  console ps -adelf
3 S root 273   4 0 153 1148bc0    0  ?   gcsp
1 S root 191 179 0 154 1075c00 1328  ?   /usr/bin/X11/xdm
3 S root 272   4 0 153 10756c0    0  ?   gcsp
1 S root1169 179 0 154 16f3c40 1332  ?   /usr/bin/X11/xdm
1 S root 723  93 0 154 1a74f40  476  ?   rpc.mountd
```

MANAGING PERIPHERAL HARDWARE

As a system ages and user requirements change, it is often necessary for an SA to add, reconfigure or remove HP-UX system peripherals. A knowledgeable SA must be prepared to handle these tasks with skill and expedience to avoid user discontent and system downtime. Removing system hardware may take as little effort as unplugging the device and purging its device file(s) or as much effort as rebuilding the HP-UX kernel (possibly removing device drivers). Adding and reconfiguring system hardware may require building a new HP-UX kernel or simply adding a new device driver file(s).

Because systems usually expand, it is more common to add or reconfigure peripherals than to remove one. Disks are one of the most common additions to an HP-UX system and demonstrate the full realm of SA tasks. For example purposes, let's assume you are the SA of a small HP/9000 series 300 workstation. The hardware for your system includes: 1) monitor, 2) 7958 disk (HPIB address-1), and 3) printer. The decision has been made to add a second 7958 disk.

When installing a new HP-UX peripheral it is first necessary to configure the device for proper communication. The 7958 disk drive utilizes HPIB bus interfacing via a unique bus address. This HPIB bus address is selected by setting a switch on the disk's rear panel. Because HPIB address 1 is used for the existing 7958 disk, the new 7958 disk will be assigned HPIB address 2. While changing the HPIB bus address it may be necessary to reference the disk's hardware manual.

After the disk's HPIB bus address switch is configured, two device files must be created. These device files act as the pointer from the HP-UX kernel to the physical hardware. The first device file is a character device file which is located on directory **/dev/rdsk**. This device file is used by format and diagnostic programs which access the disk in special modes. The second device file is a

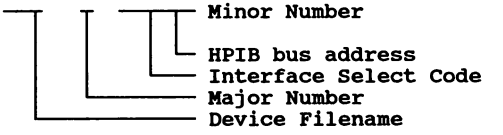
block mode device file located on directory **/dev/dsk**, and it is used by the HP-UX kernel and file system.

Device files are created using the **mknod** command. **mknod** requires four arguments: 1) device file name, 2) device file type block/character, 3) major number, and 4) minor number. The device filename must be a unique filename which will be specific to the disk being installed. The device file type specifies whether this file is a character or block device file. The major number tells the kernel the device driver used to control the hardware, the minor number tells specifics about the device hardware configuration e.g. HPIB bus address, hardware select code, port number etc...

This particular device will interface to the host HP/9000 via the internal HPIB interface (select code 7) and communicate on HPIB bus address two. Specific device file parameters and definitions are found in the **HP-UX Installing Peripherals Manual** under **Mass Storage**. The **mknod** command for adding this 7958 is:

mknod /dev/rdsk/1s0 c 4 0x070200 create character device file

mknod /dev/dsk/1s0 b 4 0x070200 create block device file



The next step is to format the disk. This is performed by executing the system utility **mediainit** (media initialize). **mediainit** accepts several arguments **-v** (verbose mode print all messages), **-r** (recertify, cartridge tape only), **-f** (select format option), **-i** (disk interleave factor), and finally character device filename.

mediainit /dev/rdsk/1s0 format the 7958 disk

Once formatted the disk must have a file system placed on it. A new file system is created on our disk via the utility **newfs** (new file system). It is necessary that this file system is created with the correct parameters for our requirements. In this example disk installation, we want a standard file system without system swap area.

The file **/etc/disktab** (disk table) contains entries on all standard disk devices and each disk's specific parameters pertaining to the file system. Many disks have multiple table entries in **/etc/disktab** to allow easy selection of swap space, sector size, block size, etc... Be certain to select the correct entry in

/etc/disktab or create a new entry to fit your requirements. The entry **hp7958_noswap** has been chosen and the file system is installed with the command:

```
newfs /dev/rdsk/1s0  hp7958_noswap      install file system
```

In order to make the newly created HP-UX disk available to users, a new and unique directory entry must be made on the HP-UX system **root** directory. Then the new disk is mounted to this new directory. The system command **mkdir** is used to create this **root** directory entry.

```
mkdir /usr3     create a new directory on root
```

The disk must finally be mounted to **/usr3** by the **mount** utility.

```
mount /dev/dsk/1s0  /usr3      mount the /usr3 file system
```

If the disk is to be used at all times it may be advantageous to add the **mount** command to the system init file. Once added to the system init file this new disk will be mounted and ready for use after each system boot.

2. ACCOUNT MANAGEMENT

One of the first tasks a new SA becomes familiar with is account management. Effective account management requires a thorough understanding of the HP-UX File System because the task involves the creation of new directories and even new file systems which, in turn, has a direct bearing on system performance.

Account management responsibilities are threefold:

1. Create new user accounts.
2. Modify existing user accounts.
3. Delete/Deactivate obsolete user accounts.

Notice that the accounts in need of management are called "user" accounts. In the HP-UX world, anyone who uses the system is said to have an account on the system. In many organizations, especially those in which Information Systems is considered a profit center, all departments who use computer resources are charged for their usage. HP-UX System Accounting provides the means to accurately track departmental usage.

Creating new user accounts is accomplished by using **vi** to edit two files: **/etc/group** and **/etc/passwd**. Because HP-UX requires that each user belong to a group, and because the group name is specified in **/etc/passwd**, the account must have an entry in both files and the **/etc/group** entry must exist first. EXHIBIT C shows the format of the **/etc/group** record and provides examples of

typical entries.

EXHIBIT C

/etc/group

group-name:password:group-id:group-members

root::0:root
bin::1:root,bin,daemon,lp
adm::2:root,adm,daemon
sys::3:root,bin,adm,sys
users::100:kats,jiml,bobs,doriss,marks,joans,pats,mikes,tome
cad::101:tome,mikes
sales::102:marks,joans,pats
actng::103:bobs,doriss
db::104:katdb,jimdb
prog::105:kats,jiml
wp::106:katwp,jimwp,joanwp,pats

Group-name describes the system usage of group-members.

The password field is not presently used by HP-UX.

Group-id is used to distinguish system accounts from user accounts. This is illustrated in EXHIBIT C by assigning 100 as the first user group-id and continuing on from there for remaining user accounts. Group-ids 0 through 99 are reserved for system accounts such as **root**, **bin**, **adm**, and **sys**. This practice lends more clarity to accounts and makes them easier to manage.

Notice the different group-names used in the examples. "Users" includes the names of all users and could well be the only user account on the system. This is fine if the system is small and if the organization does not want to track departmental usage. There are also examples of grouping users according to department (actng, sales) and application used (wp, db, cad). The choice is yours -- whatever is most meaningful to your organization.

If your organization is a large one, you may want to create a new file system for your user accounts. Because of the dynamic nature of the user file system, you will want to back it up more frequently than other, less dynamic file systems. Its independence from other file systems would allow you to do that.

EXHIBIT D shows the format of the **/etc/passwd** record and provides examples of typical entries.

EXHIBIT D

/etc/passwd

```
login-name:password:user-id:group-id:comment:pathname:
                                executable-file

kats:,..:100:100:Kathie Sagunsky:/users/prog/kats:/bin/csh
katdb:,..:101:100:Kathie Sagunsky:/users/db/kats:/bin1/databas
katwp:,..:102:100:Kathie Sagunsky:/users/wp/kats:/bin1/wp
jim1:,..:103:100:Jim Langan:/users/prog/jim1:/bin/csh
jimdb:,..:104:100:Jim Langan:/users/db/jim1:/bin1/databas
jimwp:,..:105:100:Jim Langan:/users/wp/jim1:/bin1/wp
bobs:,..:106:100:Bob Short:/users/actng/bobs:/bin1/actrec
doriss:,..:107:100:Doris Sunn:/users/actng/doriss:/bin1/actrec
```

The login-name, usually a derivative of the user's name, is found in the group-members field of the group file.

Entering ,.. in the password field will force the users to create their own passwords during their first attempt to logon to the system. This practice contributes to tighter system security because by entering his or her own password, the user is the only one who knows what it is. Normally, the SA has no need to know user passwords and should not assume the responsibility for creating them.

The user-id should be a unique number since it is used to specify file ownership to the system. These numbers can range from 0 - 60001 with 0 reserved for **root** and 1-99 reserved for the system accounts. As with the group-ids, user accounts should be separated from system accounts by using a specific range of numbers for each.

The group-id is taken from the corresponding entry in **/etc/group**.

Use of the comment field is optional, but can hold up to four separate sub-fields of user information separated by commas. The pathname of the user's home directory is specified next, followed by the program (actrec,wp) or shell (sh, rsh, csh) that the user will execute.

After creating the **/etc/group** and **/etc/passwd** entries, several other things must be done before the user can actually logon to the system.

1. Change to the directory (group-name) specified in **/etc/passwd:** **cd /users/prog**

2. Create the home directory: **mkdir kats**

3. Specify directory ownership: **chown kats kats**

4. Specify group ownership: **chgrp prog kats**

5. Specify write permission for kats only:
chmod 755 kats

Step 3 is necessary because **root** created the directory and, therefore, owns it until the ownership is changed to **kats**. Also, ownership of the **prog** group needs to be amended so that **kats** can access the group (Step 4).

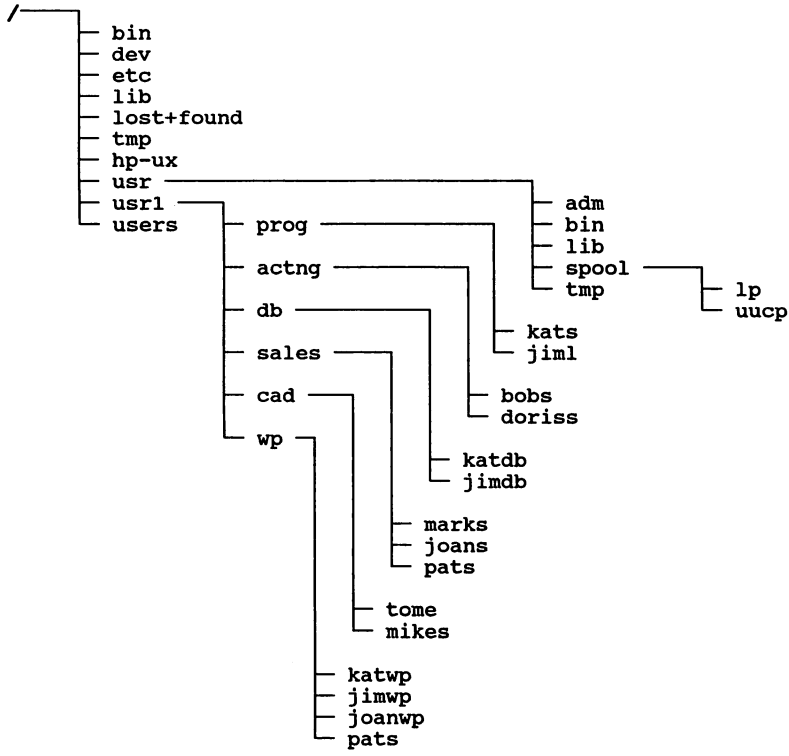
Vi can be used to modify parameters in individual entries of **/etc/group** and **/etc/passwd** or to delete entire entries. Upon deletion of an account, the corresponding directories and files should also be deleted after determining that they are no longer needed.

It is important to note that file permissions for **/etc/group** and **/etc/passwd** and other system files **never** be changed. When it is necessary to change system file contents, always access these files via **root** or **su** (switch user) accounts and then update the file by ending **vi** with **"wq!"**. Changing file permissions can destroy system security and significantly affect system performance.

An account may also be deactivated, meaning that it remains on the file system but cannot be accessed by the user. Deactivation is done by using **vi** to place an asterisk in the password field of the **/etc/passwd** file entry. To reactivate the account, use **vi** to place **,..** in the password field, thereby forcing the user to enter a password on the next attempt to logon to the system.

EXHIBIT E shows what the file system looks like after the **/usr1** directory and its sub-directories have been created.

EXHIBIT E



3. SYSTEM SECURITY

Anyone who knows anything about UNIX, probably knows two things: 1) UNIX is an open system and 2) the UNIX operating system is difficult (some say impossible) to secure. While the openness of UNIX accounts for its increasing popularity, it also accounts for its reputation as an easy mark for unauthorized access by system intruders. And once access is gained, the intruder also has access to all unprotected parts of the file system. Unfortunately, the security-related problems of UNIX have been widely publicized, as have the different resources the system makes available to thwart break-in attempts.

So, of all the SA's responsibilities, system security is truly an administrative one. By adopting and adhering to a few common-sense practices, your system can be secure from all but the most sophisticated would-be intruders. And, fortunately, the odds are against your encountering one of this breed.

You should begin developing your strategy by reviewing the security features available in HP-UX and deciding which are appropriate for use in your environment. The next step is to formulate a written policy for publication throughout your user community. Your users can either help or hinder you in enforcing your security policy, but they must first understand its importance and what behaviors it will require from them.

There are three different threats to the security of your system. They are: 1) an intruder from outside your organization, 2) an intruder from inside your organization, and 3) a good-intentioned, but HP-UX illiterate system user. The first two, although malicious in intent, are less common and therefore, less of a threat than the third. But your policy should consider all three.

The key that any intruder needs is access. Access can be gained through unattended logged-in terminals, by discovering or guessing a password, through modems, and by looking at unprotected system backups.

TERMINALS

Users need to be educated about the importance of not leaving their terminals unattended while logged-in and the SA should certainly never walk away from a terminal logged-in as **root**! This would be an invitation to disaster. Simple as it sounds, this rule is often difficult to enforce. As you tour your installation, logoff any terminals left unattended by users who should know better. Or consider use of **autologoff** to automatically logoff terminals that are idle for a specified amount of time. Sooner or later, the irresponsible users will learn the lesson.

PASSWORDS

The subject of passwords and their importance to system security was discussed briefly in the Account Management topic. Force your users to create their own passwords by entering **,..** in the password field of **/etc/passwd**. Instruct users to think of passwords which are meaningful to them so that they are not easily forgotten, yet difficult for others to guess. The best passwords are combinations of numeric, alphabetic, and upper- and lower-case characters. Passwords should never be shared.

You can also force users to change passwords periodically by using the aging parameter in **/etc/passwd**. To use the aging parameter, append a comma and one other character to the contents of the password field. The other character will specify the number of weeks until the password will expire. The characters are:

. / 0 1 2 3 4 5 6 7 8 9 A B ... Y Z a b ... y z

The first character, . signifies zero weeks; / is one week; 0 is two weeks; 1 is three weeks, etc., until the lower-case z which signifies sixty-three weeks.

More importantly than using the aging parameter for user passwords, the SA should change the **root** password frequently.

Never allow password free accounts on your system. Place an asterisk (invalid password character) in the password field of all system accounts that will not be used for login purposes. Used this way, the asterisk will prevent anyone from logging in under the account name.

MODEMS

If your environment requires absolute system security, a modem should never be included in the configuration. A system with a modem is an unsecured system.

When one is required, use of specialized modems provides a measure of security for your system. Dial-back modems require an automatic dialer and special software that compares the calling phone number with a table of authorized numbers. After a call is received, the calling number is recorded and the modem hangs up and checks the table. If the calling number is on the table, the modem will dial the number back, providing the caller with access on the dial-back.

If your system requires dialing out capability only, outgoing- only modems are available. They will not answer incoming calls. These modems, however, still invite sabotage from within the organization.

BACKUPS

Any backup media that is stored on-site should always be kept in a secure place, preferably in a locked safe or cabinet. Because these media are compatible with other systems, meaning that they can be read by other systems, you never want to let them get into the hands of anyone else.

ADDITIONAL MEASURES

For those requiring more rigorous security, HP-UX includes two files that can be used to monitor system logins.

/etc/wtmp records all successful logins; identifies the login terminal, login date and time, and login name.

/etc/btmp records all unsuccessful logins; identifies the login terminal, login date and time, and login name.

Another file called **/usr/adm/sulog** records all attempts to use the **su** or "switch user" command. **sulog** identifies the account name

that issued the command, the date it was issued, and the switched-to account name. A plus sign after the date indicates a successful switch and a minus sign indicates an unsuccessful switch. Anyone other than **root** who is allowed to use **su** should be given permission to do so by the SA.

All unauthorized attempts to use **su** should be considered as potential system break-in attempts. **btmp** can be indicative of someone trying to login by guessing a valid password. Watch the dates and times reported. If login attempts occur on weekends, holidays, late at night, or at other unusual times, they should be regarded with suspicion.

4. USER INTERFACES

Now that UNIX has successfully bridged the gap from technical to commercial applications, its user community has become more diverse. The major difference between the two groups, from the SA's perspective, is the level of computer-related expertise demonstrated by each. Generally speaking, commercial users are unfamiliar with, and consequently intimidated by, the esoteric nature of the UNIX operating system. The potential problem for the SA in a commercial environment is clear. With user satisfaction a major concern, the SA will want to shield the users from unnecessary interaction with the operating system. This can be accomplished through use of a Graphical User Interface (GUI).

These windowing applications can be developed in-house if the technical expertise is available or they can be purchased from commercial software developers. It should be noted that when companies purchase integrated business system packages, the packages usually provide a menu facility which helps the user to navigate from program to program. The menu functions as a non-graphical interface for the modules within the package. If, however, additional applications, such as word processing, spreadsheets, etc. are added to the base system, use of a GUI should be considered. An SA who is knowledgeable about GUI options will be able to guide the company in choosing the GUI most suitable for their needs.

5. FILE SYSTEM SECURITY

Your organization's most valuable resource is its information -- all stored deep within the recesses of the computer on the file systems which you have created and for which you, the SA, are responsible. The responsibility is an awesome one.

You can effectively ensure the integrity of your file system by:

1. following proper shutdown procedures

2. regular execution of **fsck** and **sync**

3. practicing good backup procedures

shutdown
The **shutdown -h** command should always be used to halt the CPU. Doing so ensures an orderly termination of system activity by executing, in reverse order, essentially the same procedures invoked during system startup. The time option also provides a means of warning system users of the impending shutdown. In the example below, users are being told that they have five minutes to terminate their processes and logoff the system before it is halted. The time value is expressed in seconds.

shutdown -h 300

Normal termination of a process causes all files accessed by the process to be closed and all associated data stored in buffers to be written to the target file(s). The CPU should never be halted before data has been properly flushed from the buffers. If an unexpected halt does occur, all processes would terminate abnormally, all buffered data would be lost, and possible file corruption could result.

To perform an immediate system shutdown without regard for system activity the command **reboot -h** may be used, reboot will sync the disks and terminate all processes without user warning. Performing a powerdown without using either **shutdown** or **reboot** can damage the file system beyond repair (see fsck) and cause costly system downtime.

fsck and sync
If the system has been improperly halted (the disk drives and CPU are just turned off, for example), the next startup attempt will issue a message notifying you of the improper shutdown. It will then proceed to execute **fsck**, the file system check command. **fsck** can, and will, correct certain file system problems without your assistance. If it is unable to correct a problem, it will ask you to run **fsck** manually so that it can ask you how you want certain error conditions to be handled. Always include **fsck** in your startup routine, since it is impossible to know for sure if a system has been shutdown correctly.

Throughout working hours, a multi-user system performs many disk read-write operations. Writing to an in-core buffer with periodic flushings to disk is much more efficient than writing directly to disk each time an update occurs. Flushing occurs
normally whenever a process terminates, when buffers are full, when a file system is unmounted, and when the **sync** command is executed.

sync should be executed frequently, either manually or automatically through use of **syncer**. If **sync** is executed every ten

minutes, for example, and an unexpected halt occurs, the only data lost would be that which entered the buffer since the last **sync** was issued.

6. BACKUP AND RESTORE

Every system, from the smallest single-user system to the largest multi-user system, needs to have its files backed up on a regular basis. A disaster may be as simple as a user accidentally deleting a file or as catastrophic as a full disk crash. Make no mistake about it, they are all disasters to the owners of the destroyed data. And you, the SA, do not want to be caught unprepared and unable to recover lost or damaged files because you neglected to do your housekeeping chores.

HP-UX provides several different backup commands and utilities which can be executed manually or through scripts. Using scripts makes the backup process easier and eliminates the potential for errors that occur when complex command lines are entered manually.

After studying the file system, the SA should devise an appropriate backup method and a schedule in which to do the backups. One commonly used routine includes daily backup of dynamic files (incremental backup); weekly backup of all files other than the operating system (partial backup); and monthly backup of the entire system (full backup).

If at all possible, the backup should be performed when the system is completely idle, to insure against the file corruption that can occur if a file is being updated at the same time that it is being backed up. Also, the success of the backup should be verified during backup by using the **-v** option, or after completion, by doing a "read-after-write" operation on the backup medium. All of the commands and utilities are listed below. Examples will be given for **cpio**.

Commands:

/bin/dd	used to back up an entire disk; the back up medium must be large enough to accommodate all of the files on the disk
/usr/bin/ftio	does a fast tape input-output; **/etc/backup** can be modified to use this command
/usr/bin/tar	is most commonly used, not for backing up, but for transferring files, directories, or file systems from one system to another

Utilities:

/etc/fbackup is used for backing up to 9-track tapes or cartridge tapes

/etc/frecover is the recovery companion to **/etc/frecover**

Scripts:

/etc/backup performs either a full or incremental backup of a file system; should be customized to fit individual needs

Piped Backup Commands:

/bin/find is used to specify individual files to be backed up

/bin/cpio copies input - output; options include
-o, out (backup);
-i, in (restore);
should be piped to **tcio** for cartridge tape

/bin/tcio tape cartridge input - output

Backup Examples Using cpio

Backup the entire **/users** file system to a cartridge tape:

find /users -print | cpio -o | tcio -o /dev/rct/dev-filename

Perform an incremental backup of all **/users** files that have been modified in the last two days. Include **-print** and the **-v** (verbose) option for screen verification of the backup process. Direct the output to a magnetic tape:

find /users -mtime +2 -print | cpio -ov /dev/rmt/dev-filename

If, as in the examples above, the complete pathname is used in the backup command, the files will be restored using the full pathname. If the same cartridge tape that was used in the first example is mounted on the tape drive when the following command is issued, the entire **/users** file system will be restored.

tcio -i /dev/rct/dev-filename | cpio -i

Use the relative pathname to backup a single user's files to a cartridge tape:

cd /users/prog/kats
find . -print | cpio -o | tcio -o /dev/rct/dev-filename

Restore all cartridge tape files, archived from **/users/prog/kats** to a different user's directory. Prior to restoring, perform a table

of contents listing on the tape by using the **-t** (table) option. Include the **-v** (verbose) option for a long listing:

tcio -i /dev/rct/dev-filename | cpio -ivt	perform tape listing only
cd /users/prog/jim1 **tcio -i /dev/rct/dev-filename | cpio -iv**	restore entire tape contents to user directory

AND FINALLY, THE REAL CHALLENGE . . .

Until now, the focus of this paper has been the technical expertise required to effectively administer a multi-user UNIX-based computer system. But as we suggested in the introductory paragraphs, technical savvy is only the tip of the SA's administrative iceberg. The vast amount of the SA's effort is directed toward the "multi-user" aspect of the system. In fact, all of the administrative tasks discussed thus far are done for one reason: to "smooth the way" for system users; to make their interaction with the system as effortless as possible.

Throughout this paper, we have tried to demonstrate that, as System Administrator, you must be willing to wear many hats: software guru, hardware technician, resource allocator, security officer, problem solver, teacher, even peacemaker in disputes among system users. Most often you will be the users' only resource in all these areas. They will expect the most you can give, and more. Are you ready to accept the challenge?

Paper Number: 2055

Title: A Novel Client/Server Network Service Developed using NetIPC HP 3000/9000

Author: Dennis Harvey
Company: Applied Biosystems Inc.
850 Lincoln Center Drive
Foster City, CA 94404
415/570-6667

Introduction

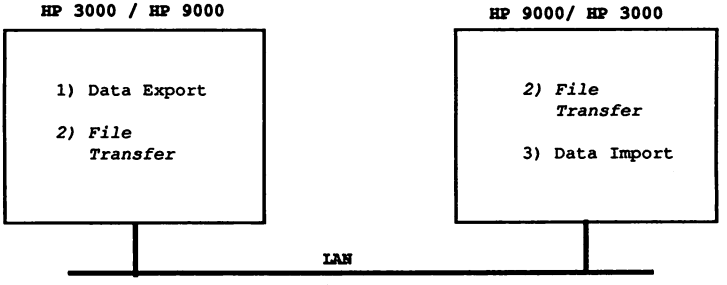
There is a "snow balling" interest in UNIX, distributed computing, and open systems throughout the information systems community. Customer demands are the primary force driving this trend, and the reasons behind this trend are clear. UNIX is the first non-proprietary, multi-user, multi-tasking operating system supported by virtually every major computer manufacturer thereby promising a considerable measure of hardware independence. The concept of open systems in accordance with the standards being developed by the relevant standards organizations (POSIX, X/OPEN, OSF ...) suggests that UNIX computers will be able to "cooperate" in a heterogeneous computing environment - something we all want to do. In addition, UNIX offers access to myriad programming productivity tools (shell scripts, awk, make, sccs, lex, yacc ...), support for the state-of-the-art RDBMS systems, graphical user interfaces and access to the broadest range of networking services and tools (ARPA services, Berkeley services, NFS, socket programming, TCP/IP, PC connectivity, MAC connectivity, SNA connectivity, E mail ...) suggesting that the application software products with increasingly impressive features will be vast.

Despite the compelling advantages of UNIX, proprietary systems will continue to exist productively in many commercial IS departments for a long time. The reasons for this are equally clear. Companies have very significant investments in proprietary systems that include investments in hardware, software, internal staff expertise, and user commitment to existing business applications. Although UNIX based applications are developing at an impressive rate, there is a significant repository of proprietary OS based applications; these will not be displaced quickly and, indeed, there may be little incentive to displace them at all. In addition, proprietary OSs (MPE/XL) are promising "open systems" features (such as POSIX compliance) - this factor will also serve to extend the tenure of proprietary systems.

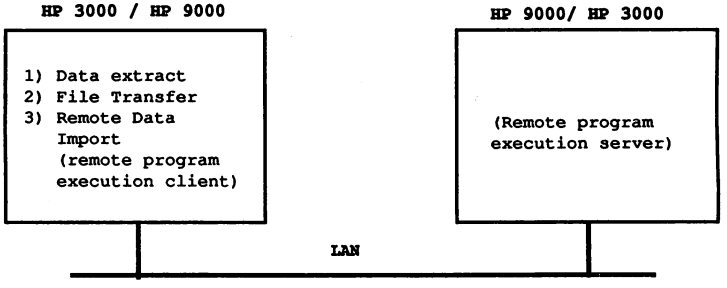
These factors indicate that the most likely scenario is that UNIX and proprietary systems are going to peacefully coexist on the same corporate network in increasing numbers. In the HP world, many IS departments will opt to migrate selected MPE/XL based applications to UNIX (HP-UX) in a phased manner, while other applications continue to execute in the MPE/XL realm for the foreseeable future. End users and MIS departments will be increasingly faced with the task of interfacing HP-UX and MPE/XL based applications. Unfortunately, there are currently some "holes" in the network services that are required to adequately interface MPE/XL and HP-UX based applications. The possible approaches and tools for reliably transferring data between

Figure 1. Three Application Interface Models

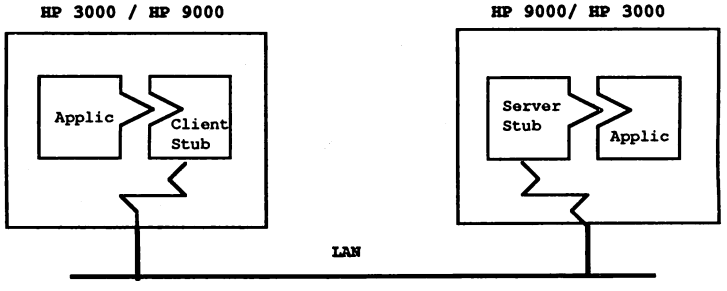
Model 1. Two Job File Transfer Interface



Model 2. Single Job File Transfer Interface



Model 3. Remote Procedure Calls (RPC)



applications over the network in this heterogeneous computing environment is the primary topic at hand.

This article examines the general problem of interfacing applications in this heterogeneous environment. The networking tools available at present are examined. Finally, a client-server network service (Remote Process Execution Protocol), developed using the HP NetIPC programmatic interface to TCP/IP, is described in detail. The result of all this, is a straight forward model for interfacing MPE/XL and HP-UX based applications and an appropriate set of network services to accomplish this task.

Application Interface Models

This discussion assumes that one of the applications being interfaced resides on an HP-UX system while the other resides on an MPE/XL system. As such, all of the interface models will involve data transfer over the network. Application interfaces can be described by three models listed in the order of increasing complexity (refer to Figure 1):

Model 1 - Two Job File Transfer Interface

Figure 2 shows a simple model for application data exchange based on file transfer. This model involves three steps:

1) export data from application A to a file
2) cross-system file transfer
3) import data into application B

In this model, two distinct jobs/programs must be separately scheduled to implement the model: job I to invoke step 1); job II to invoke step 3). Step 2) can be incorporated in either job I or II.

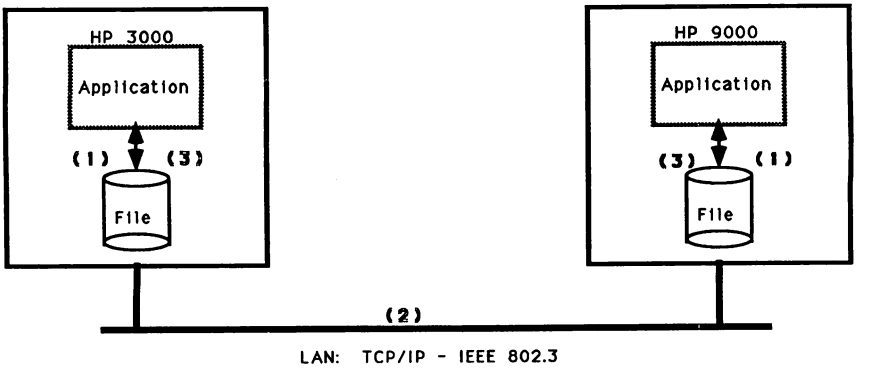
Model 2 - Single Job File Transfer Interface

Model 2 is similar to model 1 in that it involves the same three steps. The important refinement involved in Model 2 is that all three steps of the file transfer interface can be invoked from a single job. A cross-system network service that allows one node to execute a remote program on a remote node is required to implement Model 2. It is important to note that the remote program (which could be either step 1 or step 3) is a complete standalone program.

Model 3 - Remote Procedure Calls (RPC)

A third model for interfacing applications involves remote procedure calls. Basically, remote procedure calls involve a mechanism such that a local application can invoke a remote procedure (as opposed to a remote program in Model 2) on a remote system. The intent of the RPC model is to make this remote procedure call act like a normal local procedure call, however, remote procedure calls are inherently different from local procedure calls. The RPC model requires both client and server **stubs** to implement the remote nature of the procedure call. The client stub code basically bundles up the arguments to the remote procedure call into some standard protocol format; this bundled procedure call is then passed across the network. Upon receiving the bundled procedure from the client, the server stub executes a local procedure call (from the server's perspective) and

Figure 2. File Transfer Application Interface Steps



Interface Steps:

(1) Remote/Local node: export data to a file
(2) File transfer over LAN
(3) Local/Remote node: import data into application

bundles the return values or returned data from this local procedure call into the protocol standard format and sends this data back to the client. In general, implementation of the RPC model involves the development of the client and server stub code for particular remote procedure call that is to be implemented. In many environments, RPC code generation tools exist that help to generate the stub code.

In general, the RPC model is outside the primary scope of this paper. The remainder of this discussion will focus on Models 2 and 1 in detail.

Implementation of the File Transfer Interface Models

The interface between applications on the HP 9K and HP 3K systems should support both on-line data exchange as well as overnight, batch data exchange.

With reference to the three step file transfer inteface model(Figure 2), steps 1 and 3 could occur at either a remote or local node from the user's perspective and the operating system could be either HP-UX or MPE/XL at either node. There are four possible cases with reference to the three steps above:

Case 1: Local OS is MPE/XL, step 1) occurs on local node, step 3) occurs on remote HP-UX node.

Case 2: Local OS is MPE/XL, step 1) occurs on remote HP-UX node, step 3) occurs on local node.

Case 3: Local OS is HP-UX, step 1) occurs on local node, step 3) occurs on remote MPE/XL node.

Case 4: Local OS is HP-UX, step 1) occurs on remote MPE/XL node, step 3) occurs on local HP-UX node.

HP Network Services (NS) can be used to handle the required file transfers in this cross-system environment (DSCOPY) and can be used in both a push or pull fashion originating from either an MPE/XL or HP-UX node (although the syntax is a bit inconsistent and confusing). With HP NS handling the file transfers, the simplest approach to providing this type of interface is to separately schedule steps 1) and 3) on the different systems using the resident job scheduling tools (e.g. cron for HP-UX, one of many job scheduling tools for MPE/XL). To provide some synchronization, step 3) can check for the presence of a file or check the date stamp on a file to assure that step 1) and step 2) have completed. While this approach my be acceptable to support a batch, overnight interface between the systems, it will not properly support an on-line type interface. The primary shortcoming is that the three step chain of processes can not be driven to completion from just one of the systems, and that neither of systems really knows whether the whole set of processes was successful or not. In fact, a programmer or an operator would have to view both $STDLIST (MPE/XL) and perhaps a stdout and stderr redirection file (HP-UX) to determine what happened during this three step chain of events. This approach is entirely unsatisfactory to support an on-line interface. In fact, this approach may be too tedious for routine support of even the batch, overnight interface; we found this to be the case.

In order to support an on-line interface it is necessary to be able to drive all three of the steps to completion from one node or the other (either the

HP-UX side or the MPE/XL side) driven by one master process or job. The virtual terminal client, vt3k, introduced in HP-UX 7.0 and supported as part of HP NS, allows programs to be launched on a remote HP 3000 from an HP 9000. Command line options to vt3k (-a and -I options) allow HP 3000 CI commands to be read from a file resident on the HP 9000:

Figure 3. HP-UX 7.0 vt3k Example

To invoke:

```
vt3k -a ci.commands node
```

Contents of ci.commands file:

```
hello session,user.account,group
password
run myprog.group.account
bye
```

Using this approach, all three steps in the interface model can be driven to completion from the HP 9000 node. It is still problematic to provide a reliable mechanism such that the HP 9000 master process has a way of determining whether or not the processes executed on the HP 3000 node were successful. Overall, the vt3k approach can provide an adequate solution using the three step interface model to support both an on-line interface and a batch interface for cases 3 and 4 above (where the local OS is HP-UX). Cases 1 and 2, however, require a different approach.

3K9K Remote Process Execution Protocol (RPEP)

Cases 1 and 2, where the local OS is MPE/XL, requires a more creative approach. Cases 1 and 2 may be particularly important in environments where the core business systems reside on HP 3000s and data is "pushed" over to ancillary systems running under HP-UX (e.g. an MPE/XL based manufacturing MRP system with shop floor quality control or shop floor work scheduling software based on HP-UX systems), or perhaps in an environment where it would be advantageous to control all of the batch processing from the MPE/XL side using well established production scheduling tools. To address cases 1 and 2, the suitable solution requires a network service that allows a local MPE/XL system to launch a process on a remote HP-UX node. In addition, the service must provide a reliable mechanism such that the MPE/XL based master process can determine whether the remote UNIX process was successful so that it can act accordingly. Given this type of service the entire three step application interface model could be driven to completion from one MPE/XL based job or session with reliable feedback indicating whether the chain of events were entirely successful or not.

HP NS does not presently support this type of service and to date ARPA services (ftp, telnet) are not supported on the HP 3000, although an ARPA product has been announced. Even if HP 3000 telnet service was available, this would provide only a partial solution as telnet must be invoked interactively; telnet can not be invoked from a command file. While HP NS does not offer an off-the-shelf solution in this case, HP NS does provide the tools to develop this type of network service: NetIPC. NetIPC is HP's socket

interface or API to TCP/IP and NetIPC is, at present, the only network inter-process communication facility supported on the HP 3000. Figure 4 shows how NetIPC fits into the protocol stack in relation to the OSI model. Using NetIPC, a client-server network protocol (3K9K Remote Process Execution Protocol - referred to as RPEP) was developed that allows a local HP 3000 to launch a remote HP 9000 process.

The client-server model is very straight forward in concept. Network inter-process communication is the fundamental basis for the client-server model. Given that a network service is invoked by simply running a client program and that both the client process and the server process must be running to allow for inter-process communication, it follows that the server process must be running all of the time. Basically, a server process runs continuously "listening" for client requests, while a client process is invoked either interactively or in a background job and generally lives for the duration of the client request. In the HP cross-system environment, NetIPC provides the programmatic interface to provide the required network IPC to implement the client-server model.

The purpose of the Remote Process Execution Protocol (RPEP) service is to address cases 1 and 2 properly by allowing remote HP-UX processes to be launched from an MPE/XL node on the network. In addition, the service provides a reliable mechanism such that the MPE/XL based master process can determine whether the remote UNIX process was successful so that it can act accordingly. With RPEP, the entire three step application interface model could be driven to completion from one MPE/XL based job or session with reliable feedback indicating whether the chain of events were entirely successful or not.

The RPEP network service includes: the rpepd server (UNIX daemon: rpepd) that runs continuously on an HP 9000, and the rpepc client program that can be invoked from an HP 3000. The rpepd daemon is a concurrent server; it can handle multiple remote process requests in parallel. The rpepd server is conveniently invoked out of /etc/rc on an HP 9000 and executes continuously "listening" for RPEP client requests. The rpepd server is implemented in C using HP NetIPC. The rpepc client is invoked from an HP 3000 and has been implemented in Fortran (also using NetIPC).

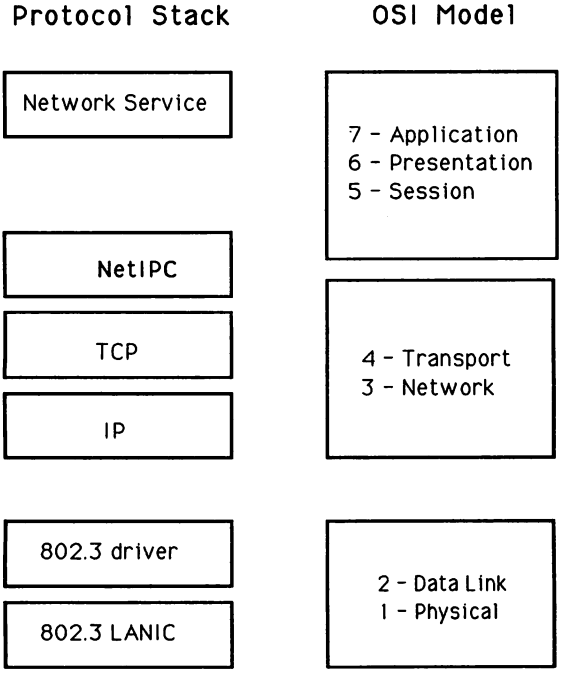
The RPEP protocol includes a series of messages passed over the network that handle:

- remote process requests
- remote process feasibility confirmation
- remote process completion status (success or failure) confirmation
- orderly network disconnection

The remote process request messages , send to the rpepd server by the rpepc client, are the most complicated and define all of the parameters (tokens) associated with an RPEP request including:

- remote process group key (explained later)
- remote node in node name notation
- remote login name
- remote password
- remote process priority
- remote process (HP-UX based executable program or shell script)

Figure 4 . Protocol Stack and OSI Model



Other Network IPC APIs:

-> Berkeley sockets (BSD)
-> TLI (Transport Layer Interface - System V.3)
-> NetBIOS (MS-DOS, OS/2)

The rpepd UNIX daemon handles the following tasks:

1) Cooperates with the rpepc client to establish a virtual circuit over the network.

2) Validates the UNIX user login and the associated password.

3) Returns a "Request Feasibility Acknowledgement" back to the rpepc client indicating whether the UNIX login and associated password are valid.

4) If the request is feasible (step 3 is successful), then the UNIX environment is created for the specified UNIX login and the requested process (executable program or shell script) is executed on behalf of the user.

5) A "Process Completion Status" message is returned to the rpepc client indicating the success or failure of the requested process. The success or failure of the process is determined based upon its exit status (exit(2)) per UNIX convention; if the exit status is equal to zero, then rpepd concludes that the process was successful; if the exit status is non-zero, then rpepd concludes that the process failed. Note, it is crucial that the requested process, whether a shell script or executable program, be written to exit with an appropriate exit status - this is just good UNIX programming.

6) Cooperates with the rpepc client to assure an orderly shutdown of the network virtual circuit after all of the remote processes specified by the rpepc client have been executed.

The rpepc client can run an arbitrary number of remote processes in a linear fashion and logically associate them with the same remote process group key. The linear chain of remote processes associated with the remote process group, are executed in a single threaded manner until all of the remote processes have been executed, or until one of the remote processes fails. In the event that a remote process fails, then the remainder of the processes in the remote process group are ignored. This approach provides a very useful production schedule group function. When the rpepc client is ready to quit, it sends a remote request message that indicates that the client is ready to quit; the rpepd server recognizes this as a client request to shutdown, and the network virtual circuit is shutdown in an orderly fashion. Figure 5 shows $STDLIST from a typical rpepc session on an HP 3000.

How does HP NetIPC Fit In?

NetIPC provides the programmatic access to the TCP/IP network. NetIPC is to the HP world, as Berkeley sockets or TLI is to the UNIX world, and as NetBIOS is to the PC world. TCP/IP provides a reliable, connection-oriented virtual circuit over the network and can be viewed just like any other form of byte stream oriented I/O. Given that a virtual circuit connection is a byte stream

Figure 5. Rpepc Client in Interactive Prompt Mode

```
Remote Process Execution Protocol
        RPEP Client
        @(#)rpeps 1.9
==================================

Enter remote process group key(rpkey)
value - test

Enter timeout in seconds:
  ( 0 to 3276; 0 = infinite ) - 0

RPEPFIL schedule file not found.

Enter remote node (<= 16 chars) -
sys3

Enter remote login (<= 12 chars) -
dennis

Enter remote passwd (<= 8 chars) -
xxx

Enter priority (<= 4
chars;1000=timeshared) - 1000

Enter remote process (<= 252 chars) -
program1 > program1.out 2>&1

    Request key         :test
    Request node        :sys3
    Request login       :dennis
    Request priority    :1000
    Request process     :
    "program1 > program1.out 2>&1"

RPEP: Connecting...

RPEP: Waiting for Request
Feasibility...
RPEP: Request Allowed....
RPEP: Waiting for Request Completion
Status...
RPEP: Process Successful.

Enter remote node (<= 16 chars) - //

RPEP: Disconnecting...
RPEP: Done.
```

Figure 6. Rpepc Client in File Mode using RPEPFIL

```
Remote Process Execution Protocol
        RPEP Client
        @(#)rpeps 1.9
==================================
Enter remote process group key(rpkey)
value - article
Enter timeout in seconds:
  ( 0 to 3276; 0 = infinite ) - 0

    Request key         :article
    Request node        :sys3
    Request login       :dennis
    Request priority    :1000
    Request process     :
    program1 > program1.out 2>&1

RPEP: Connecting...

RPEP: Waiting for Request
Feasibility...
RPEP: Request Allowed....
RPEP: Waiting for Request Completion
Status...
RPEP: Process Successful.

    Request key         :article
    Request node        :sys3
    Request login       :dennis
    Request priority    :1000
    Request process     :
    program2 > program2.out 2>&1

RPEP: Waiting for Request
Feasibility...
RPEP: Request Allowed....
RPEP: Waiting for Request Completion
Status...
RPEP: Process Successful.

    Request key         :article
    Request node        :sys3
    Request login       :dennis
    Request priority    :1000
    Request process     :
    program3 > program3.out 2>&1

RPEP: Waiting for Request
Feasibility...
RPEP: Request Allowed....
RPEP: Waiting for Request Completion
Status...
RPEP: Process Failed.....
RPEP: Disconnecting...
RPEP: Done.
```

link between two processes, there are 5 components that define the connection:[1]

(protocol, local address, local process, foreign address, foreign process)

In this case, the protocol is TCP; the local and foreign addresses are specified in domain name notation in NetIPC. This five tuple is referred to as an association. The half-associations (protocol, local address, local process) and (protocol, foreign address, foreign process) are generally referred to as sockets. The virtual circuit socket descriptors in NetIPC are half-associations and can be viewed as the endpoints of a virtual circuit connection between two processes.

The actual programming of the UNIX rpepd daemon and the associated rpepc client (MPE/XL), requires a combination of UNIX system programming, UNIX daemon considerations, MPE/XL programming and HP NetIPC programming. The intricacies of NetIPC programming are beyond the scope of this article.

Additional rpepd Server Features

The rpepd server has several important features. As described above, the rpepd server is a complete, stand-alone concurrent server. The rpepd service can be conveniently started out of /etc/rc or can be invoked from a foreground session on the HP 9000.

The rpepd server can be invoked with several useful command line options:

The -l option allows the absolute path to the rpepd log file to be specified. The rpepd log file contains extensive date and time stamped messages that indicate: 1) rpepd server status, 2) rpepc client requests, feasibilities, and completion statuses, and 3) error messages (refer to Figure 8 for an example). The rpepd log file provides a complete history of all rpepd activity on a particular HP 9000 node.

The -p option allows the TCP port for the rpepd service to be specified (Note, that HP recommends using ports 30767 to 32767 for cross-system applications).

The -R option, which enables or disables HP-UX real-time priorities is interesting. HP-UX versions 7.0 and some earlier versions include support for both real-time and time-shared process priorities by providing a preemptive kernel. If real-time priorities are disabled when the server is invoked, then the process priorities specified by the rpepc clients are ignored - they are all treated as time-shared processes.

The -S option disables password security for non-root RPEP requests (refer to Security Issues).

The -x option provides an rpepd usage message with the information about the options supported by rpepd.

The rpepd server provides a default environment for user processes modeled after the cron daemon (which executes programs at specified dates and times).

---

1 Stevens, W. Richard, UNIX Network Programming, Prentice-Hall Inc., 1990, pp 194

If a more complete environment is necessary, then the additional environment must be constructed in the actual program or script. As in the case of cron, the default profile (/etc/profile) and the users .profile is not executed for efficiency reasons; this would be too slow. In general, executable programs or scripts that are suitable for local execution by cron, are also suitable for remote execution by the RPEP - this a measure of consistency that is valuable to users. Note, the requested process is executed as the rplogin user id and using the corresponding group id specified in /etc/passwd.

Additional rpepc Client Features

The RPEP protocol requires the six tokens be passed from the rpepc client to the rpepd server: remote process group key, remote node, remote UNIX login, remote password, remote process priority, and the actual remote process. The rpepc client allows the six tokens associated with each request to be input in two modes: prompt mode, or file mode. These two modes of operation are complementary.

When rpepc is first invoked, the user is prompted for a remote process group key (rpkey). The remote process group key allows a series of remote process requests (possibly associated with different UNIX logins are different nodes) to be logically related and executed with a linear dependency; the execution of each remote process relies on the successful completion of the prior process in the same remote process group. Subsequently, rpepc tries to open the RPEP remote process group schedule file (RPEPFIL/RPEP) on the HP 3000 and searches for records with the user specified rpkey value. In file mode, a whole series of RPEP requests can be stored in the file RPEPFIL. RPEPFIL contains a sequential series of records each containing the six tokens associated with a request (refer to Figure 8). The RPEP request parser is quite flexible, and RPEP tokens are delimited either by colons, or white space (spaces or tabs). In addition, RPEP delimiters are escaped if they occur inside double quotes; the remote process token should always be enclosed in double quotes as there will generally be white space encountered within the request (due to redirection of stdout and stderr). If RPEPFIL is opened successfully, and if there are one or more records matching the user specified rpkey value, then these RPEP requests are sequentially sent over to the rpepd server for remote execution. The RPEPFIL records associated with rpkey are executed sequentially in the order encountered in the file. If one of the remote requests in the sequential chain of records matching rpkey completes unsuccessfully, then the remainder of the requests associated with rpkey are ignored and the rpepc client requests an orderly disconnection from the rpepd server. Refer to Figure 6 for an example of $STDLIST for the rpepc client used in conjunction with RPEPFIL. Figure 9 shows the source code for both the shell scripts and executable programs shown in the previous figure. Note, that program4 is not executed as specified in RPEPFIL because program3 fails (non-zero exit status).

If RPEPFIL is not opened successfully (e.g. RPEPFIL does not exist), or if the successfully opened RPEPFIL does not contain any records associated with the user specified rpkey value, then the rpepc client switches over to prompt mode, where the remote request tokens are prompted for interactively. In prompt mode, rpepc will prompt for remote node, remote login, remote password, remote process priority, and remote process in a loop until the user enters '//' to one of these prompts, or until a remote process fails at which point rpepc will request an orderly disconnection from the rpepd server. Figure 5 shows $STDLIST for the rpepc client operated in prompt mode.

Figure 8. An Example of RPEPFIL

```
# RPEPFIL:  RPEP client remote process group schedule file
# Author:   Dennis Harvey
#
# RPEPFIL/RPEP Instructions:
# - comment lines have a '#' in column 1.
# - each RPEP request must have exactly six tokens.
# - tokens can be delimited by colons, spaces, or tabs.
# - delimiters are 'escaped' if they are inside of double
#   quotes.
#
# Examples:
#rpkey:rpnode:rpuser:rppwd:rppri:"rpproc > rpproc.out 2>&1"
#rpkey rpnode rpuser rppwd rppri "rpproc > rpproc.out 2>&1"
#  where:
#  rpkey - Remote process group key value
#  rpnode - Remote HP 9000 node name notation
#           node[.domain[.organization]]
#  rpuser - Valid remote HP 9000 user login
#  rppwd  - Valid remote passwd for user rpuser
#           Exception:  If rpepd server is invoked with the -S option
#           then security is disable;  passwords are ignored for all
#           NON-root remote RPEP requests (passwords are ALWAYS checked
#           for root remote requests.
#  rppri  - Remote process priority
#           1000 = normal time shared priorities
#           0 to 127 = HP-UX real time priorities
#           Note:  Real time priorities are only allowed if the rpepd
#           server was invoked with the -R option.
#  rpproc - Remote process that rpepd server will invoke on behalf of
#           user rpuser.  Rpproc can be either an executable program
#           or a shell script.  The rpepd server will set up an
#           execution environment similar to that provided by cron.
#           As with cron, stdout and stderr should be redirected to
#           file.
#
article:sys3:dennis:xxx:1000:"program1 > program1.out 2>&1"
article:sys3:dennis:xxx:1000:"program2 > program2.out 2>&1"
article:sys3:dennis:xxx:1000:"program3 > program3.out 2>&1"
article:sys3:dennis:xxx:1000:"program4 > program4.out 2>&1"
:
```

Figure 9. Executable and Shell Script Examples

program1: sh script

```
echo "program1:  executing..."
echo "program1:  successful (exit 0)."
exit 0
```

program2: C source and executable

```
main()
{
  printf("program2:  executing...\n");
  printf("program2: successful(exit 0)\n");

  exit(0);
}

$ cc -o program2 program2.c
```

program3: sh script

```
echo "program3:  executing..."
echo "program3:  failed (exit 1)."
exit 1
```

program4: sh script

```
echo "program4:  executing..."
echo "program4:  successful (exit 0)."
exit 0
```

The rpepc file mode is most useful for batch production control where multiple remote process requests can be logically grouped in a single remote process group. Within a remote process group, the remote requests are executed in the sequential order determined by RPEPFIL. Furthermore, each remote request in the remote process group has a dependency on the prior remote request, which must execute successfully in order to continue with the remaining requests in the group.

The rpepc prompt mode is most useful for interactive remote process execution where a user needs to execute RPEP requests on demand without the tedium of generating the requisite RPEPFIL entries. Note, the choice between file mode and prompt mode is made based on 1) the presence of RPEPFIL, and 2) the presence of RPEPFIL entries matching the user specified remote process group value. These two factors can be easily manipulated by the user; if prompt mode is desired and RPEPFIL exists, then the user can either enter a remote process group key that is not represented in RPEPFIL, or the user can set a file equation for RPEPFIL that points to a non-existent file (more efficient because RPEPFIL does not have to be read by the rpepc client).

To provide for additional logic control when the RPEP service is involved in job streams or complex logic, the rpepc client sets RPEPJCW to 1 in the event of an error, otherwise RPEPJCW will be equal to 0. The value of RPEPJCW can be checked in subsequent code to take appropriate actions.

Security Issues

The features provided in RPEP provide for different approaches to security. There are basically two options: 1) running the rpepd server with security enabled (default), or 2) invoking rpepd with security disabled (-S option).

If security is enabled, the rpepd server will validate all passwords associated with remote process requests. If an invalid password is encountered, then the corresponding request will be considered as "infeasible", and it will not be executed. In cases where the rpepc client is being used in conjunction with RPEPFIL, the RPEPFIL entries will have to contain valid UNIX passwords. Obviously, this may pose a security risk if access to RPEPFIL files is excessively permissive. To provide for adequate security users should store RPEPFIL in an MPE/XL group or account that restricts read access appropriately. Additional security may be provided by controlling execute permission on the group or account where the actual rpepc native mode program resides - this can assure that only certain "friendly" users can invoke the rpepc client. The lockword expected on the RPEPFIL file provides some minor additional security.

If the rpepd server is invoked with security disabled, then the passwords associated with non-root remote process requests are ignored; requests for remote processes to be invoked as root always require valid passwords (for obvious reasons). In some environments, this second approach of disabling security may actually be considered as more secure. If execute permission to the rpepc native mode program is sufficiently restricted via the MPE/XL group or account permissions, then it may be assumed that all rpepc client users will be friendly users. The disabling of password checking eliminates the necessity of having valid UNIX logins and passwords "peppered" throughout various RPEPFILs. For good reasons, valid UNIX passwords are always required

for remote requests to be invoked as root, whether or not security is enabled. Interactive use of the rpepc client in prompt mode for root requests poses no additional security risks above those normally associated with the knowledge of passwords. The read access to RPEPFILs containing remote process requests for root must be carefully controlled.

Security is generally a "sticky" issue in the realm of network services and distributed computing. The alternatives provided in the RPEP service provide some measure of flexibility in making security decisions. As is often the case, the use of diligent control is the real ingredient assuring the desired level of security.

Summary

The rpepd and rpepc features allow the RPEP protocol to provide a powerful and flexible network service. The RPEP service enables the single job file transfer application interface to be implemented cleanly in cases 1 and 2, where the local OS is MPE/XL and complements the services provided by HP-UX vt3k. Specifically, the three steps of the file transfer application interface in a heterogeneous HP-UX|MPE/XL environment can be driven to completion from one MPE/XL job stream or session with reliable knowledge of the success or failure of the whole chain of events. In addition, the RPEP prctocol is able to provide this service with a minimal amount of network traffic.

The capability to invoke remote HP-UX processes from HP 3000s over a TCP/IP LAN, allows sophisticated and well-established HP 3000 MPE/XL job scheduling tools to be used to control all job scheduling in a heterogeneous HP-UX and MPE/XL data processing environment. In addition, the RPEP network service can be used for interactive execution of remote HP-UX processes from an MPE/XL node. The RPEP service can also be embedded within MPE/XL resident applications to implement the "one job file transfer interface" model.

Common sense dictates that it is unwise to expend time and effort re-inventing the wheel. If off-the-shelf network services, based on standards, are available that properly address your systems requirements, then it is certainly both sensible and cost effective to use them. The development of the RPEP service, however, is a good example of a network service developed to fill a void in the commercially available services.

Bibliography

Stevens, W. Richard [1990], UNIX Network Programming, Prentice-Hall Inc.

TITLE: Rapid Development of Client/Server Applications

AUTHOR: Stephan Stephansen

GenGold

P.O. Box 1209, Bedford View

Transvall, 2008

REPUBLIC OF SOUTH AFRICA

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2057

Warren R. Weber
Cheryl T. Dodd
AGS Genasys Corporation
9710 Patuxent Woods Drive
Columbia, MD 21046
(301) 596-7410

Paper 2059

MAGNETIC MEDIA CERTIFICATION SYSTEM (MMCS)

MMCS is a 9000-based system which controls a suite of test equipment used to test the magnetic capabilities of both new and used tapes. The system assists the user in configuring and setting up the test equipment, performs the tests (to certify the record and playback capabilities), compares the results to an established baseline, and graphically presents the results to the user. The results may be saved for later review by the user.

The user may test any or all of the following for each tape: signal to noise ratio, wavelength response, third harmonic distortion, print through, ease of erasure, wear, and signal dropout/amplitude uniformity. When used in a very demanding environment, MMCS has proven itself to be a valuable tool. MMCS provides the capability to assure the quality of tapes for re-use where previously only new tapes could be used. Depending on a user's volume/throughput requirements for tapes, the savings realized by MMCS can be well into the hundreds of thousands of dollars. Even more importantly, by performing quality assurance tests, a user can make sure the tapes he received meet the specifications he purchased them against. In other words -- he got what he paid for.

Introduction

The Magnetic Media Certification System (MMCS) was developed as a "switch hitter" -- it could certify new tapes for use and assure the quality of tapes ready for re-use. The quality (QUAL) function of the system allows the user to select individual tests and run them with any test parameters the user desired. QUAL tests are typically used to verify that new tapes delivered meet procurement specifications and to determine why tapes failed to meet the recertification specifications. These tests are usually run by engineers. The recertification, or quality assurance (QA), tests verify that tapes returned from end users are still able to meet operational requirements. QA tests are designed to run in a production environment by less experienced technicians and are packaged together with default parameters to make their execution simpler.

Previously, testing instrumentation tapes was a labor-intensive process that required a highly skilled technician (or technicians) and a great deal of time. The technician was required to be able to collect and interpret data from a variety of laboratory instruments, such as spectrum analyzers, frequency counters, oscilloscopes, and signal generators. The results of these individual tests were then compiled, and an overall assessment of tape quality was made. All in all a very subjective process.

With the development of the MMCS, instrumentation tape testing has become more efficient, less labor intensive and no longer requires a highly skilled technician . Several functions are now performed at the request of an operator:

- Create/maintain reference standards
- Hardware calibration and verification
- Test data collection
- Instrumentation tape certification
- Provide quality assurance for instrumentation tapes
- Archive tape test results and traceability information

The automation of these procedures has allowed a greater quantity of instrumentation tapes to be tested more quickly, accurately, and efficiently. An operator must still have a basic knowledge of tapes, but not at the level previously required. Now, the principle skill requirements involve knowing how to load a tape recorder, select a test to be performed and wait

for the test to complete. The MMCS will perform the selected test(s), access the correct baseline reference data for test comparison, programmatically control the appropriate laboratory instruments and the tape recorder(s), display the test results, and archive the test results.

Hardware Configuration

The MMCS is comprised of three hardware components: the system controller, tape recorders, and Hewlett-Packard Interface Bus (HPIB) programmable laboratory instruments. All of the system components are commercially available, and most can be replaced by similar products with only minor impact on the MMCS software (the major exception being the system controller).

Controlling the MMCS is an HP 9000/332 which runs the MMCS software, contains the test results archives, and directs the HPIB programmable laboratory instruments via HPIB buses. The computer system is configured with 32 Mb of RAM, one 300 Mb hard disk drive, a cartridge tape drive (65 Mb capacity), and five HPIB interfaces (one on the CPU board and four 98624A interface boards). The system is operated from the 35741A color console.Test results are presented on the console, and hardcopy output can be generated on the HP Thinkjet printer or the HP 7550A sheet feed graphics plotter.

The current MMCS configuration contains three Honeywell 96B instrumentation tape recorders, two Honeywell 5600 audio tape recorders and a video tape recorder. Through experience, it has been determined that an accurate assessment of tape quality can be obtained by only testing selective channels (or tracks) on a particular tape. This increases the functionality of the MMCS. For example, MMCS tests seven tracks on a Honeywell 96 recorder (out of a possible 14 or 28); and can also be configured to test four tracks on a Honeywell 5600 recorder.

Five HPIBs were designed into the system to increase system performance and reliability. The five buses contain: the printer, the plotter, the disk, the tape recorders and the test equipment. By taking advantage of the system spoolers and giving the printer and plotter their own buses, the operator is able to generate hardcopy of the test results while running the next test. Each test session generally generates 25 to 30 pages of printouts and plots, therefore, the time saved by not tying up test buses is quite noticeable. The tape recorders are on their own bus to isolate them from the rest of the system because they have proven to be temperamental.

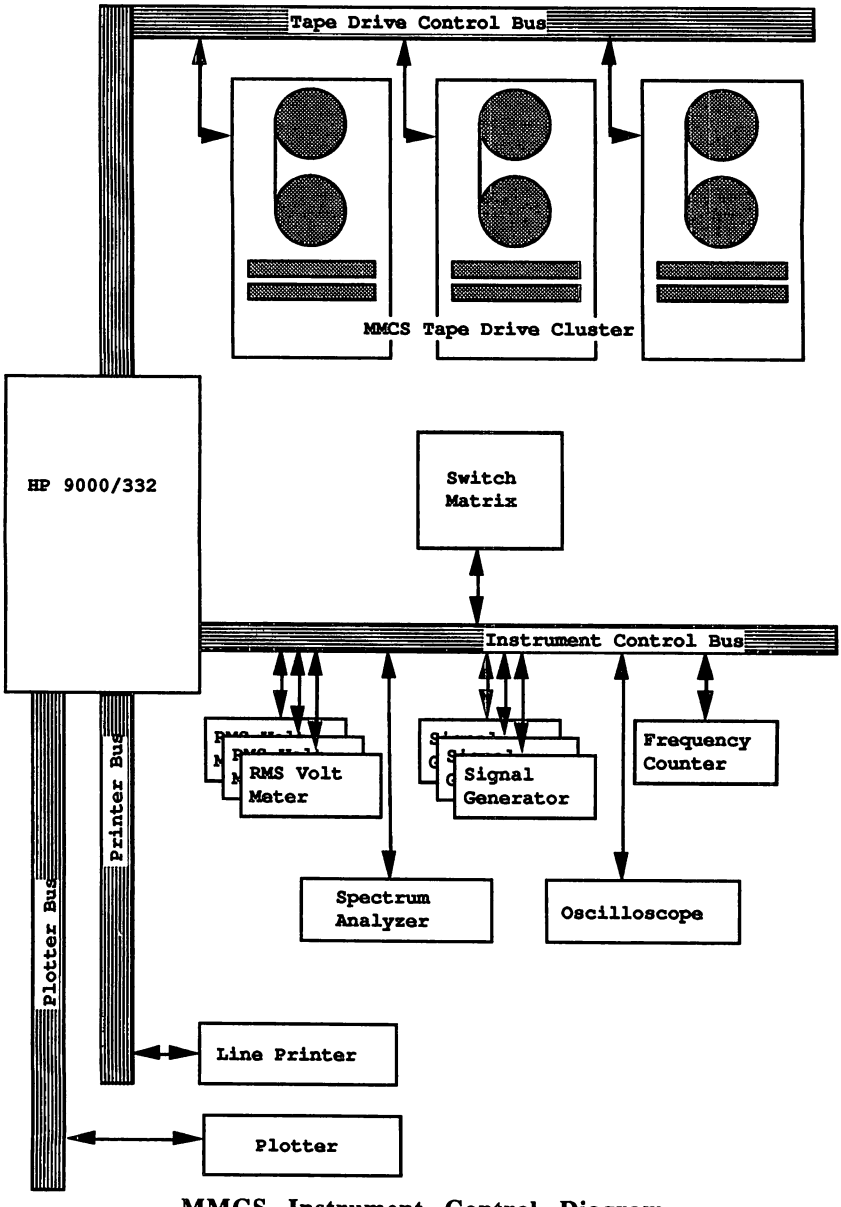
There are occasional problems with getting a particular drive to initialize correctly which has caused bus problems. The test equipment are on their own bus for reliability reasons, allowing data from the test equipment to be sent back to MMCS over a single bus.

The various HPIB programmable laboratory instruments used by the MMCS include frequency counters, signal generators, oscilloscopes, spectrum analyzers, voltmeters, and switch matrices. The system currently includes one HP 5328B frequency counters, three HP 3325B signal generators, one Tektronix 2465A oscilloscope (to allow the operator to monitor the test in progress), one HP 3585A spectrum analyzer, three Fluke 8920A RMS voltmeters (with one Fluke 1120A IEEE 488 translator) and two Racal-Dana 1250 VHF switches. The signal generators are used to supply signals to calibrate other system components (e.g., the voltmeters), and to also supply test signals to be recorded onto the tape(s) under test. The frequency counters, spectrum analyzer and voltmeters are used to collect the test data and forward it to the HP 9000. The switch matrix (with 100 inputs (of which approximately 95 are being used) and 100 outputs (about 60 used) is used to route both input and output signals between test equipment and tape recorders under the control of the HP 9000. Routing is done by the MMCS software based on the test(s) selected by the operator, thus eliminating the need for an operator to repatch signals during tests. See figure below for a graphical representation of the hardware configuration.

Software Configuration

The MMCS software is written in "C" currently running under HP-UX version 7.0. The graphics for the system are created and displayed using the Starbase graphics package.

MMCS was designed to be a modular system. The system currently uses with of the original test equipment; however, new equipment can be added (or replaced) simply by writing a new device interface module. Software modules have been designed to allow general commands to be translated into device-specific instructions at a fairly low level.



MMCS Instrument Control Diagram

An example of this modular design is the file that specifies the MMCS channel to tape recorder track assignments. The tracks being tested can be easily adjusted by simply editing the file. This file will also allow the migration from testing 14 track tapes to 28 track tapes to be a simple one. Finally, another file in the system keeps track of the tape speeds supported by the tape recorder. All of these features allow the upgrade from the eight speed Honeywell 96B to the nine speed Honeywell 96C to be a relatively easy task.

MMCS software is divided into five general areas: tests, database, operator interface, reports, and device interfaces.

The testing software is further divided into individual units -- one for each test. These tests are: signal to noise ratio, wavelength response, third harmonic distortion, print through, ease of erasure and wear. The test algorithms used in MMCS have been derived from Federal Specification WT-1553A (Federal Specification for Magnetic Oxide Coated Recording Tape). Tests may be run individually or as a group. The controlling software for test execution runs the selected tests and monitors the system for problems. Should an error occur during testing (e.g., the tape breaks), the test controller aborts the test, notifies the operator of the error, and allows the operator to either abort the session, or continue testing after the error condition has been corrected.

The MMCS database software is not a database in the truest sense of the word. Two directories exist that contain the raw data from tests and the baseline files for each tape recorder that hold "perfect" data for each tape recorder. The baseline files are used to compare tapes under test to insure that the recorder the test is being run on is the same as the recorder on which the baseline tape was created. Both sets of files are accessed through a higher level file containing the file names and access keys. The files are distinguished by unique serial numbers assigned prior to each test. These serial numbers allow the system to store each test session of a tape, while also storing information such as the test date, run number (how many times that tape has been tested), tape identification and the operator name. If an operator decides to make another test run on a tape (with an existing serial number), the system automatically appends the serial number with a new "one up" run number. Using unique serial numbers also allows the operator to retrieve the results from previous tests for re-analysis.

The operator interface software is menu driven with multiple pull-down windows. Wherever possible, the tape traceability information is

filled in by the system (current date and time, etc.) but the majority must be entered by the operator. Each field is checked for correctness as it is being entered, and error messages are displayed as soon as an error is detected. The information entered by the operator is saved from one test run to the next on the assumption that given the nature of the system, multiple runs will be made on tapes that have a great deal in common (belong to the same shipment from the manufacturer, etc.). Over the years, this has proven itself to be a time saving feature.

The reporting software takes the results files from the tests (the same ones stored by the database software) and presents the results graphically and in tabular format. These results are available to the operator both at the system console and the system printer/plotter.

A Quality Assurance (QA) Test

The hardware and software configurations described above are used in various combinations to conduct magnetic tape performance tests. One such test, wavelength response, can be selected to run from the QA Test Menu. The wavelength response test determines the difference between the amplitude of the reproduced test signal and the reproduced noise of the tape (which is caused by grounding the input). This difference is measured using a wide bandwidth RMS voltmeter. Other test equipment required to complete a wavelength response test include a tape recorder, a signal generator to produce the test signal, and a VHF switch to route the analog test signals to/from the tape recorder and the instruments.

After the operator loads the tape onto the selected tape recorder and selects the wavelength response test from the QA Test Menu, the following actions are taken by the MMCS software to conduct the test.

1. Send commands via HPIB to open all input/output channels to the tape recorder via the VHF switch.

2. Start the tape recorder in a "record forward" state using commands sent via HPIB.

3. Send commands via HPIB to ready the signal generator to produce the test signal.

4. Wait for the tape recorder to reach the speed selected by the operator.

5. For each channel selected to be tested (two channels):

a. Record the test signal onto the channel of the test tape.

b. Measure the amplitude of the reproduced test signal using the RMS voltmeter. Store this value in memory.

c. Ground the input(s) of the selected channel and record no signal
(i.e., noise) on the test tape.

d. Measure the amplitude of the reproduced signal noise using the RMS voltmeter. Store this value in memory.

6. Send commands via HPIB to stop the tape recorder and the signal generator.

7. Use the values stored in memory in step 5 to calculate the signal to noise ratio and store this value in memory. Determine compliance with specification requirements.

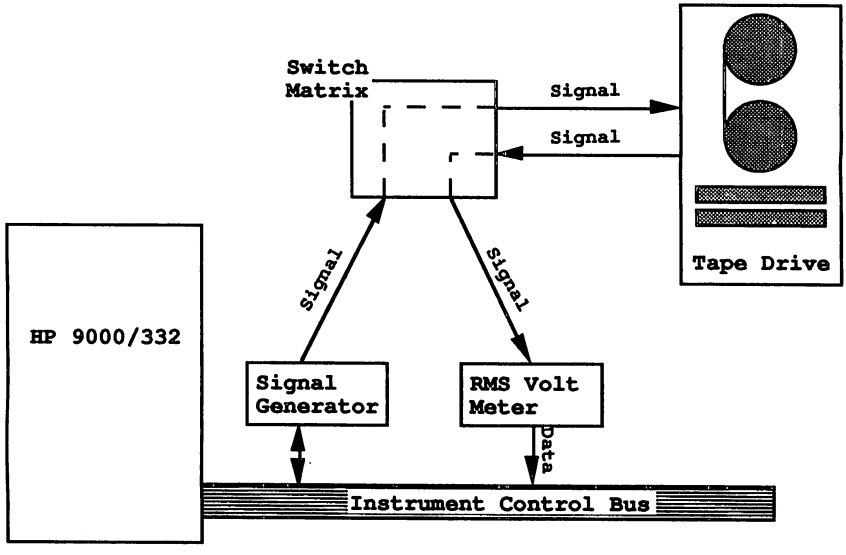
8. Display the test results in tabular format on the system console.

9. Print the test results at the operator's request.

10. Save the test results in the MMCS database at the operator's request.

11. Return the to the QA Test Menu.

The following diagram shows the signal and data flows of the wavelength response test.



MMCS Signal/Data Flow Diagram

What's Next for MMCS

MMCS was designed to be a modular system where tests and additional functions could be easily added, and test equipment could changed. There are currently a number of enhancements in the works for MMCS.

The system is currently being upgraded to replace the 14 track heads on the Honeywell 96Bs with newer 28 track heads to allow higher density tapes to be tested. This will require only the modification of the channel-to-track file. Next, the Honeywell 96Bs will be replaced with Honeywell 96Cs which are capable of running twice as fast as the Honeywell 96Bs. This requires the modification of the system file containing the tape speeds supported by the tape recorder and the default parameter files.

As this paper is being written, a specialized test system is being integrated into the MMCS to perform tests not available on other pieces of commercial test equipment. Developed by Kineticsystems Test Management Systems Group, the H•TMS performs tests using sophisticated data collection, filtering and analysis techniques to provide additional insight into the quality of the tape under test. The H•TMS is a standalone testing system with its own operating system, testing software, and archiving capabilities. However, AGS Genasys and Kineticsystems have modified it to run as a "black box" piece of test equipment that can be used like any other programmable laboratory instrument in the MMCS. In this configuration, the H•TMS will be controlled by the HP 9000 over a dedicated HPIB, allowing the operator to select the H•TMS setup parameters and run the tests directly from the MMCS console, and analyze the H•TMS data at test conclusion

Once the H•TMS is fully integrated into the MMCS, it will be put into a production environment. A new production line system will include two H•TMS units and four Honeywell 96B drives to allow more tapes to be tested in a production environment. Software will also be developed to perform the analysis of the test results, freeing the operator from that task. The increased level of system sophistication will mean more consistent results, and the skill requirements for the operators will not be as stringent as those required for the laboratory version of MMCS.

Concurrent with the development of production line systems, a Management Information System (MIS) is being developed to provide a central point for the collection of test results and operator tasking. This system will allow a production line supervisor to access the test results from any connected test system, and will provide the capability to instruct operators to perform any special tasking. For example, a special task could specify special handling requirements or specific tests to run. Additionally, the MIS would allow the supervisor to update any of the test parameters or pass/fail criteria on any of the connected test systems from one centralized location.

The last enhancement currently under development will provide much the same capability as the MIS described above for the laboratory systems. The laboratory systems will interface to an MIS that will, in turn, be connected to an inventory control system. This will allow the MMCS to notify the warehouse that the tape samples have been passed and the shipment meets specification. In this configuration, the MIS' ability to re-analyze the test results and compare them to other test results from the

same manufacturer will be very important. The MIS will also contain post-processing routines that will allow more in-depth analysis of the test data.

Summary

The Magnetic Media Certification System is an equipment control/data analysis system that can save money for user organizations that have a large requirement for a large quantity of high quality tape. While the initial cost is significant (the hardware cost is between $250,000 and $300,000), the cost savings can be quite dramatic over a long period of time. Currently instrumentation tapes on glass reels cost $200 each. The quality control function of MMCS insures that the tapes do meet the user's specifications. In other words, MMCS helps the user make sure he got what he paid for.

Additionally, the quality assurance function of MMCS can offer the user significant savings by allowing previously used tapes to be recertified for use. Assuming that the cost of recertifying previously used tapes (which includes physical inspection of the tapes and their reels, cleaning of the media and reels, electrical testing by MMCS and packaging) costs $50, a savings of $150 for each tape recertified can be realized. Using these figures, the system must recertify only 2000 tapes before the hardware costs have been recovered. For a high-use organization, MMCS can result in large savings in a very short period of time.

2062

The Integrated Workstation, A Realtime Data Acquisition, Analysis and Display System

Thomas R. Treadway III

Lawrence Livermore National Laboratory
P.O. Box 808 L-573
Livermore, CA 94550
(415) 423-4997
treadway@llnl.GOV

Abstract

The Integrated Workstation was developed at Lawrence Livermore National Laboratory to consolidate the data from many widely dispersed systems in order to provide an overall indication of the enrichment performance of the Atomic Vapor Laser Isotope Separation experiments. In order to accomplish this task a Hewlett Packard 9000/835 turboSRX was employed to acquire over 150 analog input signals. Following the data acquisition, a spreadsheet-type analysis package and interpreter was used to derive 300 additional values. These values were the results of applying physics models to the raw data. Following the calculations data were plotted and archived for post-run analysis and report generation. Both the modeling calculations, and real-time plot configurations can be dynamically reconfigured as needed. Typical sustained data acquisition and display rates of the system was 1 Hz. However rates exceeding 2.5 Hz have been obtained. This paper will discuss the instrumentation, architecture, implementation, usage, and results of this system in a set of experiments that occurred in 1989.

Introduction

Lawrence Livermore National Laboratory (LLNL) is developing the Atomic Vapor Laser Isotope Separation (AVLIS) technology. This technology uses the world's highest average power visible light lasers to enrich natural uranium for use as fuel for commercial light water nuclear reactors. Development of AVLIS will provide the United States with the most cost effective enrichment technology available. The AVLIS process involves the vaporizing of uranium with an electron beam. These vaporized atoms are illuminated by laser beams having very precisely tuned optical frequencies. The laser beams photoionize (i.e. remove an electron from) the U-235 isotope, giving it a positive electrical charge, while leaving the other isotopes neutrally charged. These ionized atoms are attracted to a negatively charged surface, while the un-ionized U-238 atoms pass through the ion extraction zone. This technique is also applicable to many other elements.

An Engineering Demonstration System using the AVLIS technology was developed and tested in 1989 at LLNL. This facility required a computer system to consolidate the diagnostic data and to provide a real-time indication of the overall enrichment and photoionization performance of the process. The system dedicated to this task was called the Integrated Workstation (IWS). Accomplishment of this task required the acquisition of over 150 analog signals, along with the calculation (derivation) of approximately 300 additional values. Analysis was performed using the above input signals along with results from intermediate calculations, as input data to physical models. Following the analysis, the data were plotted and archived. The IWS provides the physicist and experimenters access to all of this data upon a 19-inch, high resolution color graphics display, in the form of real-time trend plots.

All of the data was available for nearly any mathematical calculation desired during the experiments, and more extensive interactive post-run analysis. The **IWS**'s default configuration allows for up to 64 plots, each plot displaying up to 8 variables on the same axes simultaneously, for a total of 256 individual variables displayed simultaneously. Designed with maximum flexibility in mind, the plot configuration and modeling calculation executed during real-time could be changed dynamically.

History

Early AVLIS experiments were performed using the Hewlett Packard (HP) 1000 model A900 computer, using two 32 channel data loggers each coupled with a 64K memory module, installed on a CAMAC crate. Data scans were typically taken over a 20 to 50 second period, followed by down loading of the results from the CAMAC data logger to computer memory over HP-IB (IEEE 488). Once the data was in computer memory and a copy written to disk, another scan would be triggered, and the process repeated. While the data loggers were sampling at 20 to 50 Hz, the computer would be averaging the raw data of the previous scan. Following the averaging, the 12-bit integer data were converted to volts, or their respective engineering units. Subsequent to the data acquisition, the modeling calculations were performed by a powerful, real-time modifiable spreadsheet code, followed by the plotting. User selected data was plotted upon a graphics display, which was tightly coupled with a button box, allowing for the instant selection of any one of 8 multivariate sets of plots.

The entire sequence from the CAMAC data read to the final plot was typically done in just under 20 seconds (at least a half dozen other applications were running on this HP at the same time). Post run this same HP 1000 was used for local data analysis and generation of tapes for analysis by the Cray. Similar characteristics were needed from the Integrated Workstation in support of the AVLIS demonstration, but at least 4 times better performance was desired.

The **IWS** started as a software port of the HP 1000 system, but this time on a new computer; an HP 9000 model 825 with an SRX graphics subsystem. Later it was upgraded to a model 835 with turboSRX. Porting of the software involved modification, testing, and debugging of about 80,000 lines of FORTRAN code. The initial (unoptimized) post-migration test provided only a 2 times execution speed improvement compared to the HP 1000 version. Unfortunately, this new implementation would require that the **IWS** support at least 4 remotely located data acquisition units, and about 4 times more data.

Hardware Architecture

HP 3852A Data Acquisition / Control Units (scanners) were substituted for the CAMAC front ends, which yielded very impressive results. The HP 3852's were no more expensive than the CAMAC solution, but proved to be much more reliable and flexible.

Each scanner functioned like a standalone computer with its own real-time multi-tasking Basic programming language that supported data acquisition and control. The back of the scanner consisted of an eight slot card-cage that accepted a multitude of different modules such as: high-speed voltmeters, high-speed multiplexers (that read VDC, VAC, thermocouples, current, ohms, and strain-gages), analog output, stepper motor control, pulse counters, digital input/output, breadboard, and even a HP-IB disk or tape drive interface controller.

Instrumentation

Each scanner used a HP 44702B 13-bit High-Speed Digital Voltmeter (100,000 reading/second), which took up two slots, as the primary acquisition unit. By adding multiple HP 44711A 24-Channel High-Speed FET Multiplexers, almost 150 signals could be acquired by a single scanner before an expansion chassis was needed. One expansion chassis would increase this number to over 350 signal channels, and the maximum of seven extender chasses would allow over 1500 channels.

The flexibility of the HP 3852's facilitated the straight forward addition of special input signal measurements by the plugging in of the appropriate data acquisition module into the scanner. Two other modules were used: an HP 44713A 24-Channel High-Speed FET Multiplexer with Thermocouple Compensation for measuring temperatures and an HP 44715A 5-Channel Counter/Totalizer module for pulse counting up to 200 KHz.

The data acquisition configuration had 4 scanners trigger-synchronized to each other. One scanner was used to control the initiation of a scan, typically once every second. While sampling was at 600 Hz, an averaging was done for each channel to remove any 60 Hz line noise. The data was then read by the computer from each scanner as double-precision floating-point voltages, temperature, or counts depending on the signal. The HP 3852A's are HP-IB instruments, so in order to span the distance to the computer, HP 37204A HP-IB Extenders were used. The preprocessing of the data on the front ends, reduced the amount of work the computer needed to do, as well as reduced the amount of data transferred. With the data in computer memory, each point was converted to engineering units. All of this was done in well under a second. The **IWS** hardware configuration is shown in Figure 1.

Graphics System

Selected data signals along with results from analysis calculations were displayed on a high resolution 19 inch color graphics display monitor. A 32 button box, keyboard, and mouse, connected by HP-IL were the input devices for the user of the **IWS**.

Graphics was provided by the HP 98731A turboSRX graphics subsystem. The monitor displayed the user selected plots, upon the 24-bit graphics planes at 1280x1024 pixel resolution. Superimposed upon the graphics planes was an additional 4-bit overlay planes in which X-windows was running. When an opaque window was displayed in the overlay planes, the image in the graphics planes would be blocked (obscured). The 24-bit graphics planes of the turboSRX are composed of three 8-bit banks. One bank was used by each of the primary colors: red, green and blue. The intensity for each bit in every bank was set to maximum intensity. By accessing only 1-bit at a time out of each bank, eight 3-bit graphics planes were created. A 3-bit graphics plane will allow up to 8 colors to be defined; 3 primary, 3 secondary, black and white.

The 32 function buttons allowed the user to switch between the any one of the eight 3-bit graphics planes containing the plots, zoom in or out (thanks to the turboSRX), as well as pan across the screen. The keyboard was used for command input by the user, and the 3 button mouse used for X-windows operations. A typical experimental configuration had a terminal emulation window in the lower left hand corner of the overlay screen, through which the user entered commands to the **IWS**.

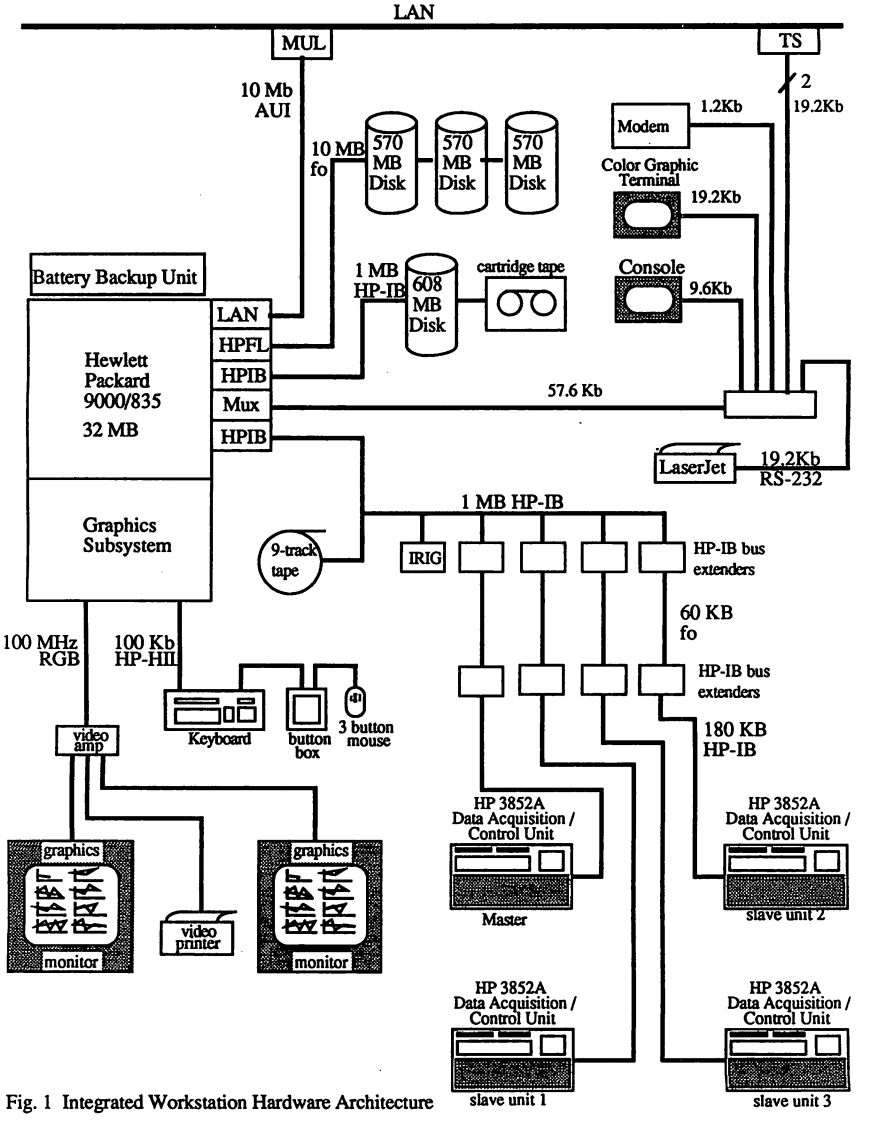


Fig. 1 Integrated Workstation Hardware Architecture

Software Architecture

The **IWS** contained a set of flexible independent software programs. Flexibility and speed were the two most important considerations driving the design of the **IWS**. In experimental environments in which the **IWS** operated, many changes must be accommodated, including varying the data signals, acquisition rates, analysis calculations performed, and the experimental objectives. Most of the internal data structure was table driven, thereby reducing the amount of code that needed to be re-compiled (in most cases to none). Also, through the use of interpretive interfaces the system possessed a high level of dynamic reconfigurability. What is meant by "independent" is that each program had a specific task to perform in the sequence of the data processing, yet could be removed (disabled) from the processing pipeline without affecting the operation of the overall system.

User Interface

Most visible of all the **IWS** programs was the User Interface (**ui**) program. From this program the user communicated operations and commands to the **IWS**. The **ui** program was an interactive command line interpreter, which read its commands from the keyboard or optionally from macro files. All of the interactive **IWS** programs allowed an option that would redirect input from a file, or nested files containing sequences of often repeated commands. As was mentioned above, **ui** controls the interface to the **IWS**. It's from **ui** that the user initialized the **IWS**, putting it into a default state ready for data acquisition, analysis, and display. Following initialization, **ui** was used to customize the **IWS** configuration for the particular experiment. Most of the global system parameters were defined through **ui** commands, such as: enabling the front-end data acquisition hardware, defining which analysis algorithms to use, selection of a plot configuration for graphics display, and turning on the archiving of data.

Data Acquisition

The control of the front-end hardware was performed by a devoted program called **isdl**. **ui** passes commands to **isdl** by what in UNIX is known as a *pipe* . A *pipe* is an interprocess communication mechanism used to send data from one program to another. Its one method of accomplishing one-way communication between programs.

After **isdl** was done executing a command it read the *pipe*. If there are no commands in the *pipe*, **isdl** suspends execution until there is (referred to as read with wait or blocked read). However if **isdl** was performing a task, such as continously acquiring data, then the read on the *pipe* would not wait (unblocked) for new commands to arrive, but would proceed with acquiring the next data scan.

While in the continuous scan mode of operation, **isdl** would execute the incoming command before processing the next scan. **isdl** did not initiate each scan for the HP 3852's unless the single scan mode was in effect. That function was done directly on one of the HP 3852's, using its internal real-time clock to initiate a data scan, and at the same time send an external trigger to the other HP 3852's, synchronizing all the data acquisition. This technique of placing the burden of timing on the front-end equipment, assured better uniformity of the scan interval, placed less demand on the CPU, and resulted in better over all user response.

After collecting all of the data for a scan, the HP 3852's performed any additional processing (such as averaging), before writing the data into its output buffer. The HP 3852 then configured itself for the next scan and waited for the time interval to expire, or for a trigger in the case of a slave unit. **isdl** in the mean time was waiting to read the HP 3852's data, after which **isdl** applied the appropriate engineering units conversions, and wrote it to *shared memory* for access by other programs.

Analysis

Before **isdl** returns to read the *pipe*, it sends a *signal* to the analysis program **rvesp** telling it to start. A *signal* is also a UNIX form of interprocess communication that is efficient at interrupting a program as a result of an external event. **rvesp** functioned similar to a spreadsheet-type code allowing the calculation of derived data.

Most of **rvesp**'s mathematics and syntax was geared towards linear algebra and matrix operations; however scalar operations were also supported. **rvesp** had a powerful data manipulation function similar to an **if**-condition statement processor. **rvesp** contained a well rounded tool-kit for data processing including: macro files that had argument passing capabilities, and a variety of data base functions for manipulating the data matrix.

The commands to **rvesp** were first read in through **rvisp** the reverse polish command interpreter initializer program. **rvisp** functioned similar to a compiler. Commands were parsed and interpreted, expressions were converted to reverse-polish notation, and then everything was written to a large command buffer in *shared memory*. When **rvesp** was executed, it read all of the preprocessed (compiled) operations directly from *shared memory*. Because there was no disk access (as in the batch operation of the post-run version of the code), very good real-time performance was obtained from **rvesp**.

Graphics

After the analysis and creation of derived data, **rvesp** sent a *signal* to the plotting program Quick Look DRaW (**qldrw**). **qldrw** is similar to the **rvesp/rvisp** pair in that all of the user's plot selections are predefined and loaded into *shared memory* by a user interface routine called PLot SELect (**plsel**). Plots were the primary end product of **IWS** during the run.

Post-run plots of all the data over the entire run duration were provided in a run report. Plots were of such importance both in real-time and post-run that the plot configuration interface contained a great deal of flexibility. Plots were guaranteed to display the last 1024 points collected, assuming that many points have already been collected. The most basic of the options was the selection of the dependent and independent variables to be plotted. If multiple variables were to be displayed upon the axes (superimposed), the default option was to scale the axes to fit all of the data being plotted. Automatic axes scaling could be overridden, if the user desired.

qldrw allowed up to a maximum of eight variables to be superimposed upon a single plot. By default all of the plots had time as the x-axis. Time was represented as a decimal hours relative to the start of the experiment. As new data was added to the plot the display took on the effect of a strip-chart recorder. **plsel** was interactive, allowing the user to dynamically reconfigure the plots during the experiment. If the experimenter was interested in a particular signal or calculated value, then that information could be easily brought up on the screen.

An overview of the interaction between the software processes of the **IWS** are shown on Figure 2.

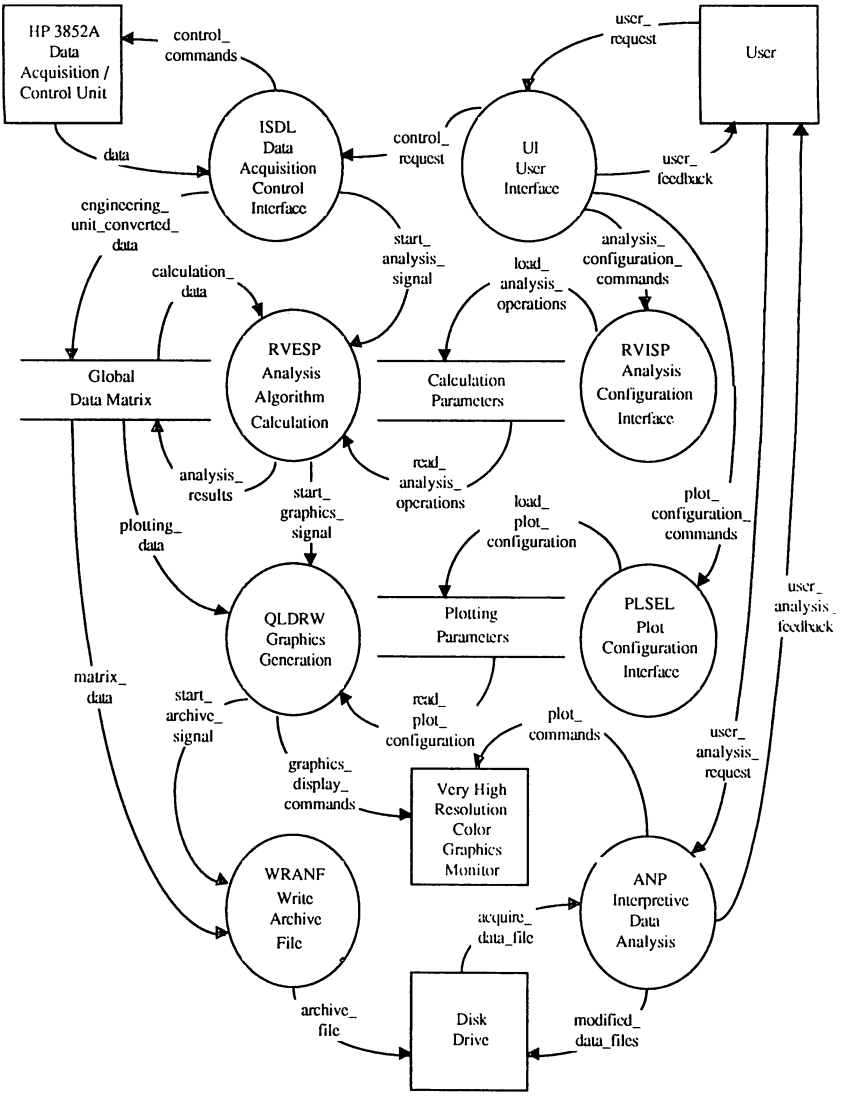


Fig. 2 Integrated Workstation Data Flow Diagram (Context Level)

Conclusion

By requiring all data entered into the IWS to be analog (0 to 10 volts) in nature, the flexibility and ease of adding new signals was almost as simple as plugging into a patch panel. Division of responsibilities between the providers of the analog signals and the IWS was mediated at the patch panel. Pre-run end-to-end checkout and test duties could be carried out independently. A disadvantage of this approach was if derived data was required from another workstation, this information must be converted back into analog signals for reacquisition by the IWS. In the process of doing this the calibration information was lost, along with possible significant figures in the derived 32-bit real number to 13-bit voltage conversion. This function was usually performed by hardware on the workstation, hence didn't steal cpu cycles, it did use up expansion slots. The HP 3852 easily accommodated other types of signals (temperatures, pulse counts) by adding the appropriate module.

The speed of the acquisition and display of data provided an accurate indication of the current performance and condition of the experiment. Powerful numeric processing available from the analysis spreadsheet allowed the physicists to observe the results of the experimental models applied to real data in near real-time. Knowing the current operating conditions of the experiment facilitated experiment scheduling decisions, assisted in the optimization of the process, and the development of algorithms. All of the data collected and calculated was archived and accessible to the experimenters for post-run analysis, modification, or report generation.

The AVLIS Program is scheduled for a major integrated enrichment demonstration in the early 1990s. The Full-Scale Demonstration Facility will fine-tune the scientific and economic concepts of the process to facilitate the transition to commercial use. Planned future work in support of this effort include: 1) moving more towards networked based data collection and distribution; 2) providing X-window user interfaces, and; 3) supporting distributed graphics for both the real-time trending and for historical data display.

This work was performed under the auspices of the U.S. Department of Energy by the Lawrence Livermore National Laboratory under Contract No. W-7405-ENG-48.

Paper Number 2067

POLLIT : A SYSTEM'S ADMINISTRATOR TOOL TO MONITOR A HETEROGENEOUS NETWORK

Najib Nadi
Dept. of Mathematical Science
Villanova University
Villanova, PA 19085
(215) 645 - 4852
nadi@tiger.vill.edu

Thomas A Savarese
AT&T Microelectronics
2525 North 12st, PO Box 13396
Reading, PA 19612
(215) 939 - 3419
savarese@tiger.vill.edu

I. INTRODUCTION

The proliferation of computer systems which are faster, smaller and less expensive has caused a corresponding proliferation of software applications for these systems. Networks have evolved around these systems so that individual systems may communicate with each other allowing users and applications to share and exchange information. The machine resources may be distributed over either a small geographic area of two floors of one wing of a building or a large area with nodes separated by hundreds of miles. However, the network allows access to programs, data, and peripheral hardware resources regardless of either the location of the user or the hardware/software.

Managing a complex LAN with computers from different manufacturers interconnected with bridges and routers from different vendors can be very difficult. This task is even more challenging when the different computers are running different operating systems, the computers are widely separated, and the users are intolerant of downtime.

The purpose of this paper is to present a simple solution that may be of some help in making this task a little bit easier at no extra cost in equipment.

II. PROBLEM DESCRIPTION

In today's environment, system administration is becoming an increasingly difficult task. The advent of small, powerful computers has propagated distributed networks consisting of heterogeneous hardware and software platforms at a rapid rate. The numbers and the remoteness of these system resources connected to the network has significantly increased the workload of the system administrator. The administrator must spend a great deal of time verifying that all network resources are working properly. The problem is further compounded because the scope of

user applications have become very narrow and focused within these small systems. The applications or functions supported by small computers typically cater to a small group of users or to a few specific tasks. This tends to give each computer a low profile within the entire network.

The administrator tends to neglect low profile applications and concentrate most of his or her attention on keeping high profile systems and applications running because of time constraints. Problems are not found on low profile systems until the user finds that some functionality, resource, or information provided by the system is not available. The administrator is not aware of problems with low profile systems until a crisis occurs when business cannot proceed normally. This reactive administration style causes frustration in the user community and wastes time and money. The user is forced to modify methods and procedures to continue business. This tarnishes the user community's perception of the administrative organization.

It's a misconception on the administrator's part to believe that low profile application systems are inherently unimportant. They may be critical to the business. Regardless, support of low profile applications does sway the user community's perception of the administrative organization.

III. PROPOSED SOLUTION

The first solution that was considered was to have each machine report its status periodically via E-mail to a key node that is constantly monitored. This approach is against the Unix† thinking "no news is good news". We only want to hear about machines out of service. The second solution adopted, *pollit*, does just that.

pollit was designed to aid the system administrator with the difficult task of monitoring the status of computers on a network. *pollit* proactively checks each node defined in its local database and will take some predefined corrective action when it finds a node which does not respond to the poll. *pollit* can aid the administrator by assuming the tedious network monitoring duties, and therefore, allow the administrator to concentrate on other types of work.

IV. FUNCTIONAL DESCRIPTION

1. GENERAL OPERATION

pollit is a command script which can be executed by either the Bourne Shell or Korn Shell command interpreters. *pollit* was designed to periodically poll a set of nodes defined in the ASCII data file *pollit.db* . Each node, which *pollit* is to monitor, must be assigned a unique role to play as a member of the set. The possible roles are either MASTER or ACTIVE or PASSIVE.

† Unix is a Trademark of AT&T Bell Laboratories

The MASTER provides a central location for the *pollit* configuration and database files, *pollit.cf* and *pollit.db* respectively. Only one MASTER node may be defined for a set of nodes being checked by *pollit*. The MASTER must be a Unix machine which has a compatible shell interpreter. *pollit* may be initiated as a background or foreground process on the MASTER either by *cron*(1) or interactively by the administrator.

The ACTIVE members are those machines which can assume temporary polling responsibilities should the MASTER fail. Like the MASTER each ACTIVE member must be a Unix machine which has a compatible shell interpreter. ACTIVE members do not have private copies of the configuration files used by *pollit*, eliminating periodic maintenance on ACTIVE members. All the required configuration files are sent from the MASTER to ACTIVE members after each successful poll. This ensures up to date network descriptions and a centralized location for configuration files.

The strength of *pollit* is its ability to handle PASSIVE members. The only requirement of a PASSIVE member is that it must respond with an ICMP ECHO _RESPONSE when explicitly polled by an ACTIVE member or by the MASTER. *pollit* uses the *ping*(1m) command to test if a member node is alive or dead. The *ping* command utilizes the DARPA Internet ICMP protocol's mandatory ECHO_REQUEST datagram to elicit a response from a given network node. This means that any hardware/software platform may participate as a PASSIVE member as long as the Internet protocol is supported on that computer. PASSIVE members may be gateways, bridges, routers, database servers, or machines running non-Unix operating systems. PASSIVE members are not limited to computers running Unix , unlike ACTIVE members and the MASTER.

When *pollit* is initiated on the MASTER, the script reads each node description entry from its database file *pollit.db* . Each node description entry contains a HOSTNAME , a ROLE , and a CORRECTIVE ACTION which is a legal shell command or shell script which can be executed by a compatible shell interpreter. Each node is subsequently polled by the MASTER using the ping command. If a node does not respond to the poll, the corresponding corrective action specified as part of the node's description entry is executed as a child process of *pollit*.

When the polling sequence is finished, *pollit* sends an executive script to all active members. The executive script contains a copy of *pollit* itself; the database file, *pollit.db*; the configuration file, *pollit.cf*; and the custom shell script file, *pollit.sh*, which is optionally provided by the administrator. The executive script is sent to all ACTIVE members using *remsh*(1). The remote shell subsequently schedules the executive script to be executed at a later time using *at*(1). The time interval is the same time interval used by *pollit* specified in the configuration file.

After *pollit* has sent the executive script to all ACTIVE members, the script exits.

When the specified time interval elapses, the executive script is unbundled into temporary files in the default directory of the *pollit* administrative account on ACTIVE members. The executive script then initiates *pollit* as a child process. When *pollit* is executed on ACTIVE members the MASTER is immediately polled. If the MASTER is alive *pollit* exits with a successful status forcing the executive script to exit without rescheduling itself. Otherwise, if the MASTER is dead, each ACTIVE member determines if it should assume temporary MASTER duties. This decision is made by polling higher ranked ACTIVE members with a simple rank order algorithm. The ACTIVE members which do not assume the responsibility of MASTER immediately exit with a failure status forcing the executive script to reschedule itself to run after the configured time interval using *at*(1).

The ACTIVE member which assumes MASTER responsibility polls each network node and initiates any necessary corrective action as described above. When these MASTER duties are fulfilled, the ACTIVE member exits with a failure status and its executor reschedules itself as described above.

2. SPECIAL FILES

pollit uses three support files: a database file, *pollit.db*; a configuration file, *pollit.cf*; and an optional custom shell script file, *pollit.sh*. These support files are centrally located beneath the HOME directory of the *pollit* administrative account on the MASTER.

2.1 *pollit.db*

pollit.db is a database file which contains an entry for each network node which *pollit* should monitor. Each entry contains a NODE , the node's ROLE , and a CORRECTIVE ACTION to be undertaken by *pollit* when the node does not respond to a poll. The ASCII file should have each field separated by whitespace. A whitespace may be any number of spaces and/or tab characters.

```
< NODE >  < ROLE >  < CORRECTIVE ACTION >
```

Figure 1. Format of an entry in *pollit.db*

NODE A node may be any valid official node name or aliases found in /etc/hosts

ROLE may be any of the following single character keywords:

- M means MASTER. The computer assigned to provide a central location for configuration files used by *pollit*
- A means ACTIVE member. The computer(s) designated to temporarily take over MASTER responsibilities should the MASTER fail.

P means PASSIVE member. A non-active participant in the *pollit* domain.

CORRECTIVE ACTION The corrective action may be any legal shell command or shell script which can be executed by the compatible shell interpreter chosen by the administrator.

An example of a *pollit.db* file which configures a *pollit* domain of five systems follows:

```
lynx   P  mailx -s "system lynx is DOWN" joe@tiger
puma   A  .  ./pollit.sh puma; echo "PUMA IS DOWN">/dev/console
tiger  M  mailx -s "system tiger is DOWN" joe
monet  P  cu 327-6001- - - - - - - -712*12
theBox P  mailx -s "system 192.82.109.99 is DOWN" joe@tiger
```

Figure 2. Example of *pollit.db*

2.2 *pollit.cf*

pollit.cf is a configuration file which contains a list of tunable parameters which may be customized by the administrator to modify the behavior of *pollit*. The meaning of each tunable parameter follows:

REMSH This is the name of the remote shell program which is distributed with the implementation of TCP/IP on the MASTER. The two typical names are either *rsh* on Berkley Unix operating systems or *remsh* on AT&T System V Unix operating systems.

MAXPINGS This is the maximum number of remote hosts to asynchronously ping at one time. This parameter ensures limits on the number of child processes *pollit* can spawn at one time. The default value is 5 processes. If more than MAXPINGS nodes belong to the *pollit* domain, the script will repeatedly spawn MAXPINGS polls then wait for the polls to complete before continuing.

POLLTIME This is the time between polling intervals. This parameter is passed as an argument to the *at*(1) command as described above. The default value is 15 minutes. This value should also be used as the time interval for spawning instances of *pollit* by *cron*(1) on the MASTER. This value should never be less than the time it takes to execute *pollit*.

PACKSZ This is the datagram length used by the *ping*(4) command. The default length is 512 bytes.

COUNT This is the number of ICMP ECHO_RESPONSE packets to be sent to each NODE in the domain by the *ping*(1M) command. There only needs to be one response returned

by the node being polled for it to be considered alive. The default value is 10 packets. This number should be kept small to keep the network load small.

POLLITSH This parameter determines when the executive script should execute *pollit.sh*. The custom shell script may be executed either BEFORE the *pollit* script is invoked by the executor, or AFTER *pollit* has succeeded, or NOT at all. If BEFORE is chosen, the *pollit.sh* script will always be executed on ACTIVE nodes. If AFTER is chosen, the executor script will only execute the *pollit.sh* script if the MASTER is alive. If NOT is chosen the *pollit.sh* script is ignored by the executor script. The default is NOT.

2.3 *pollit.sh*

pollit.sh is an optional shell script which the administrator supplies. *pollit* will send the shell script to active members and execute the script based on the value of the POLLITSH tunable parameter. The intent of this feature is to provide the administrator a tool for customizing *pollit*. Possible uses may be:

- To provide a set of corrective action shell scripts, one for each domain member, which can be invoked by the acting MASTER when a node is found to be dead. This provides a method to send standard scripts to all ACTIVE members eliminating the need to have the scripts stored on each of them.
- To provide the administrator an alternative of passing information to other monitoring packages using the *pollit* utility.

3. SETUP

pollit requires a unique administrative account on each ACTIVE member in its domain and also on the MASTER. The accounts should have the same Unix *uid* and *gid* to ensure that *remsh*(1) works properly. The utility also requires the administrative account on each node be permitted to use the *at*(1) command. This can be accomplished by adding the account's Unix login id to /usr/lib/cron/at.allow On the MASTER node the administrative account should also be permitted to use *cron*(1) by adding the account's userid to /usr/lib/cron/cron.allow The administrative account also requires permission to execute the *ping*(1m) command. One method for granting permission is by changing the owner of the /usr/etc/ping to root with *chown*(1), followed by setting the set user id bit on the executable using the *chmod*(1) command. The set user id bit is required so *pollit* can read and write to the device file used by ping.

4. USING CU TO CALL A BEEPER

cu(1) is part of the Basic Networking Utilities on AT&T Unix System V. The command is typically used to call up another Unix system, a terminal, or possibly a non-Unix system. It manages the connection by using two setup files, /usr/lib/uucp/Devices and /usr/lib/uucp/Dialers.

The following samples are designed to call a hardwired AT&T 2224 modem at transmission speeds of either 1200 baud or 2400 baud.

```
ACU tty101  - 2400 att2224B
ACU tty102  - 1200 att2224B
```

Figure 3. Sample Devices file entries for Automatic Call Units

```
##########
#  AT&T DATAPHONE II 2224B Modem
#
#    For normal operation dip switch S1 switch 2 should be OPEN.
#    This disables ENTER INTERACTIVE WITH <CR>.
#
#    Commands:        Explanation:
#    ---------    ----------------------------------------
#    =+-,         '+' for secondary dial tone, ',' for pause
#    ""           expect nothing
#    atT\T\r\c    enter command mode, tone dial the number
#                 (substitute 'P' for first 'T' for pulse dialing)
#    ed           expect "ed" (actually "Answered")
#
##########

att2224b =+-, "" atT\T\r\c ed
att2224B =+-, "" atT\T\r\c ed
```

Figure 4. Sample Dialers file entries for AT&T 2224 modem

cu can be used to dial a beeper to alert the administrator to some problem. The difficulty in calling a beeper from a modem is with the tone(s) immediately returned by the beeper which causes the modem to hangup. These tone(s) are returned to indicate to the caller that either a voice message or a key pad sequence may be sent. The indicator is for the benefit of human callers, but if the caller is a modem, the signal will tend to cause a disconnect to occur. The hangup caused by these returned tone(s) can be avoided by utilizing the *pause* feature of the *cu*(1) command, if the calling modem supports *pause*. The pause causes a modem to ignore any incoming signal or tone generated by the beeper being called for the time interval the pause is in effect. If enough pause directives are strung together, the modem will miss any incoming tone(s) generated by the beeper after it answers. All that remains is to send a meaningful string of digits to the beeper, simulating a human caller pressing a series of keys on the phone keypad. The following example shows how to call a beeper with a digital display and to send a numeric identifier to the beeper using the *cu* command.

```
cu -s 2400 3276000- - - - - - -712*12
```

Figure 5. Call a beeper using the 2400 baud modem and display 712-12

If a host is using Berkley Unix, the *tip*(1) command may be used in a method similar to the one described above. The choice of *cu* to dial out, ensures the use of the lock to access the phone line, and resolve any conflict with other applications that may request access to the dialup line.

V. CONCLUSION

pollit was built to assist the system administrator in monitoring the network. We believe it is a good tool that may improve the overall availability of the machines in a LAN with a minimum overhead. *pollit* configuration files are maintained in a single machine and dispatched to the participating nodes dynamically. This strategy maintains the integrity of the multiple copies of the configuration files.

The system administrator is kept informed of any malfunction of the monitored hardware via custom-built shell scripts. For the critical nodes in the LAN, the beeper feature provides an immediate alarm to alert the administrator of the failing network resource.

Because it is based on commands and utilities found in a generic Unix system, *pollit* can be installed in any Unix system with almost no changes, except for the configuration files. And best of all, it is free.

For those that need fancier monitoring schemes with more sophisticated graphical interface, there are products based on SNMP (Simple Network Management Protocol) and CMIP (Common Management Information Protocol) that achieve the same results and provide control from a central console.

Paper No: 2068
Title: ENGINFO - The Data Management Solution

K. Kannikeswaran, J. R. Krebs, J. F. McDonough
College of Engineering, ML#18
University of Cincinnati,
Cincinnati, OH 45221

ABSTRACT

It is difficult for complex organizations to provide access to numerous databases for large segments of their members. Incompatible systems make this data access task across the organization increasingly complex. The College of Engineering with its growing student enrollment, its rapidly expanding research activities accompanied by a myriad of information needs, confronted this problem and developed a generic interface to access all data on site. Allbase/SQL on a HP-UX platform has served as the information processing tool, in providing access to a comprehensive data bank of Relational Databases derived from a multitude of data sources ranging from Mainframe Databases to local PC based PC based worksheets. This paper describes the design principles and some features of this interface ENGINFO (with particular emphasis on Allbase/SQL, 4GL related issues). ENGINFO is gradually gaining acceptance as both a data processing tool (in managing such activities as a billing system) and as an information retrieval tool in meeting the complex information needs of an educational institution such as the College of Engineering, University of Cincinnati.

Key words:

ALLBASE/SQL, HP ALLBASE/4GL, HP-UX, Data Management System, Database Design Principles, Relational Databases

1. Introduction:

This paper describes ENGINFO, the data management system at the College of Engineering, University of Cincinnati. Sections 2 and 3 of this paper deal with the data processing environment at the College and at the University, the factors that led to the evolution of ENGINFO and the design principles involved. Section 4 deals with specific cases within ENGINFO and issues related to Allbase/SQL and 4GL.

2. Background

2.1 College/University Setting

The College of Engineering at the University of Cincinnati is one of the national leaders of Cooperative education in engineering. The college's current enrollment includes 2,100 full-time undergraduate students pursuing a five-year baccalaureate degree in one of 11 programs accredited by the Engineering Accreditation commission of ABET. Each of the six departments of the College offer programs of study and research leading to the degrees of Master of Science and Doctor of Philosophy.

Approximately 970 graduate students are working with about 130 full-time faculty and over 60 research and adjunct faculty through more than 15 research centers and institutes on funded research which exceeded $13 Million last year. The college is pursuing a dynamic growth path and is opening numerous new faculty lines in selected areas. The College of Engineering is the home of the Center for Robotics Research, the NASA designated center for Computational Fluid dynamics, the Center for Biomechanics, the Center for Membrane Technology, the Structural Dynamics Research Laboratory, the Polymer Research Center and many other laboratories and institutes.

The computing facilities within the College include a HP 9000/840, a DEC VAX 780,

numerous workstation and microcomputer labs, SUN 386i NFS file servers an IBM PS/2-80 OS/2 LAN Server and a broadband network integrated with the Campus ethernet backbone.

The University of Cincinnati is a state funded institution, and is one of the principal employers at Cincinnati, Ohio. The size of the university is a little over 35,000 students.

University wide computing facilities include 2 Amdahl Mainframes, a DEC VAX 6350, a DEC VAX 8650, several microcomputer labs and a campus-wide ethernet backbone integrated with Digital PBX system.

2.2 Data processing at the University of Cincinnati

The information needs of an academic institution such as the University of Cincinnati are very complex. The University has a comprehensive data processing system to take care of vital university Operations such as student registration, transcript processing, payroll processing, accounting/financial management etc. None of these activities are managed at the college level. The University's system pretty much takes care of all requirements in the aforementioned areas.

The Office of Facilities Management, manages data on space allotment and availability using their 'SMS' (Space Management System) , and the Division of graduate studies maintains a database on graduate students. In addition to this, the Office of International Affairs maintains a database on International graduate students. The Division of Financial aid handles undergraduate financial aid data while the Office of the University Dean for research handles data on research projects.

Data entry into the student registration system is carried out by the University's own data processing body. Personnel records are handled in a manner with some slight modifications. The College has very limited data entry authorization on the various student databases. Thus the University's student data management system with its data owned by the Office of student records operates in a highly centralized computing environment.

The CUFS (College and University Financial system) is accounting/financial management software, running on the mainframe. Transactions are entered at the college level into this system.

But for certain exceptions, communication between individual colleges and institutional databases is largely through 'paper mail'. For example, offers of scholarships are communicated to the Central graduate student database, housed at the University's Office of graduate studies through an 'A160' form, from which data is entered into the database by the division of graduate studies. They in turn provide the college with reports on financial aid etc. Thus, although a college may serve as the point of origin of data, data is not entered at the college level.

2.3 College specific data processing operations

Teaching and research are the basic functions of the college of engineering. The administrative units require information for decision making, reporting and operational needs. Some of the support operations carried out by the various units are i) Student academic record monitoring ii) Student record maintenance at the college level, iii) Management of scholarship/admission iv) Other administrative functions at the college level.

Typically, a very limited number of users have access to the Institutional databases. The update capabilities provided to the colleges are very limited, and most access is limited to look ups only. In addition, the user needs to maintain a good deal of information locally for various operational and reporting purposes. In several units, data is still largely stored in hard copy files, and in certain other units a number of spreadsheet based databases have mushroomed

over a period of time.

2.4 History of Data processing at the college of Engineering:

As described above, much of data processing prior to 1989 has been on a rather sporadic basis, and the first need for a comprehensive data management system for the college was voiced in 1988.

An intensive strategic study carried out in 1988 came out with a list of nineteen recommendations and suggestions for improving the quality of computing environment in the areas of academic, administrative and research computing. This report documented the fact that the existing institutional information management systems did not serve the needs of the College adequately and that the institutional data was not directly accessible by college administrative staff, was not available on a timely basis, was incomplete, inconvenient and difficult to use.

The report also acknowledged the need for i) a College wide centralized data management system for internal college information management and ii) reporting and analytical tools facilitating access to and analysis of both and institutional and college specific data.

The five year strategic plan published by the College of Engineering included in it the goal of creating an integrated and interactive database for all vital statistics of the college including academic administration, research awareness and productivity, faculty information, alumni and development information and financial information.

The first College-wide spreadsheet database was developed in 1988 to handle information on research grants and contracts. During the period 1988-1989 an extensive series of discussions were held with various representatives of the college (following the strategic computing study conducted in the summer of 1988). The nature of information required was studied and the data flow was documented. HP Today (Allbase SQL/4GL) running on HP 9000/840 emerged as the choice for a database platform, in March 1989. The first multi-user databases were implemented and tested out then and ENGINFO was formally inaugurated on January 16, 1990.

3 ENGINFO - The data management solution

3.1 ENGINFO - Mission

The purpose of ENGINFO is to aid the College of Engineering to the fullest extent in achieving Vision 2000 by

i) Providing decision makers with timely, reliable and accurate information
ii) Providing administrative staff with a variety of tools to automate operations.

This mission translates into the following objectives:

i) Meet all information needs of the College in a timely manner
ii) Eliminate redundant usage of resources and man power
iii) Achieve office automation to the fullest extent possible.

3.2 Information needs:

As mentioned before, the information needs of the college cover a wide spectrum of areas, which are not covered by the institutional databases. In other words, the data from institutional databases only form a subset of the range of information requirements for the College. For example, in monitoring graduate student academic progress, a graduate student advisor would require information on the student's course-work from the Institutional databases, and information on the students research activity which is more department

oriented while the college might want to look at a list of students advised by a given faculty member.

The information needs of the College can be classified broadly in the following manner:

Information for external reporting versus information for internal control:

The College needs to generate information which is reported to external agencies such as American Society for Engineering Education, NSF, Ohio Board of Regents, Engineering Man Power Commission etc.. College Publications such as newsletters, Profile Reports etc. also depend to a large extent on timely information on statistics pertaining to the College.

Examples of information requirements in the area of internal control include Space Management, Inventory Management, Budget Management, Room Schedule management, student progress monitoring etc.

Administrative vs. Operational

Information to aid management in decision making such as in financial management, in employee evaluation, in projecting future allocations etc. vs. information to aid administrative staff in routine operations.

Departmental level vs. College level:

Information is needed at all levels in order to carry out administrative and operational responsibilities. At the College level, a global picture, reporting gross statistics is typically the information requirement while at the departmental level, the information requirements are typically detail oriented.

3.3 Redundant usage of resources and manpower:

Redundant usage of resources and manpower is typically a result of one or more of the following factors:

i) Multiple databases at the departmental level, maintaining similar information on a variety of database packages

ii) Hard copy reports are often a source of redundant usage of resources. (Typing from a hard copy generated by a data source into another database is a typical example).

iii) Independently created local databases often pose a great problem in terms of data concurrency and often conflict with institutional databases which hold the same information possibly in a less accessible form.

3.4 Strategy

The strategy adopted in achieving the above objectives was to develop ENGINFO as a bridge between Institutional databases and local data in the following manner:

i) A centralized, integrated, multi-user data management system would cater to the college's information requirements

ii) This centralized data bank would use as its primary source the centralized records of the institutional databases on campus and would have minimal data input at the college level

iii) The secondary source of data for this databank would be college specific data

generated from within the college, data of no relevance to the institutional databases.

iv) Reporting and retrieving tools provided by the university would be utilized to the fullest possible extent both in report writing and in data transfer across databases.

Thus the primary functions of ENGINFO are to assemble, analyze and use data and to serve to provide simultaneous access to end users within the college through user friendly menus in order to enable them to retrieve information in a usable form.

3.5 Elements of ENGINFO

Figure 1 gives a schematic illustration of the approach adopted by the College in attempting to meet the objectives:

i) The primary data sources for the College are the Institutional Databases to which access is limited.

ii) The query and report writing tools provided by the University are the primary devices for extracting selected segments of the University Databases.

iii) The extracted data is stored along with the local databases on the College's Information processing system in the form of relational databases.

iv) Application programs run at the front end by users from within the College access the Central data bank, perform on line queries and generate reports.

The three primary elements within ENGINFO are:

3.5. Data Sources: Institutional databases are the primary data sources for the College. The University Student database is an illustration of such a data source.

3.5.2 Access Devices: Access to Institutional databases is usually very limited, for example, reports or extracts from the University's Student database can be run from only one account. The access tools used here are NOMAD and Answer-DB, a report writing product. In some cases arrangements are made with data owners for running batch jobs to create text files pertaining to the Collage of Engineering. These files are downloaded and then fed into the Central Data Bank.

3.5.3 The Central Data bank: This is a collection of Relational Databases most of which are based on the data from Institutional Databases. Also in this data bank are tables of information specific to the College of Engineering, generated from within the college. Allbase/SQL manages these databases, which are housed on the HP9000/840.

3.6 Design principles:

i) Operations oriented vs. Management oriented

Databases were designed keeping in mind the various operations performed at the user level and on the various data elements sought often for reporting purposes. The focus during the design and the development process has been on developing an information system that actively supports the activities of the operating staff.

ii) Information perspective

It would not be an exaggeration to state that the database design was 'Report driven', i.e. based on information that needed to be reported by the College in various reports.

iii) Multiple functions

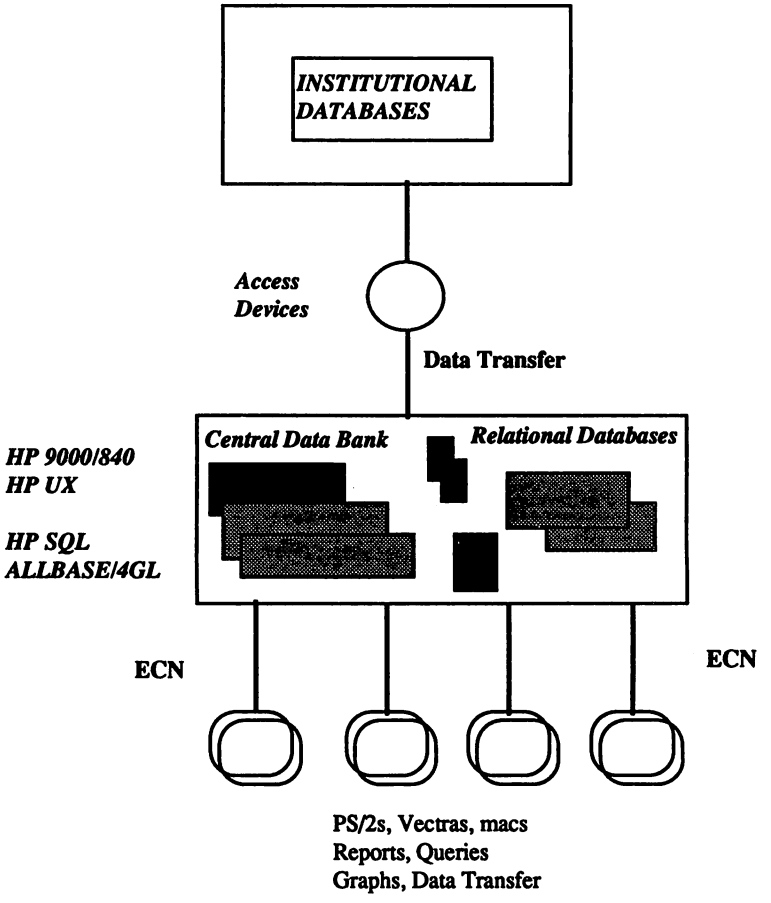


Figure 1: Elements of ENGINFO

The same data serves the College and the individual department in different ways, and this was taken into account in defining data dictionaries in order to achieve multiple outputs. For example, the data on research grants and contracts now is used to run the annual Faculty Activity Report (to be completed by the various faculty), as well as to provide an overall indicator of funding trends of the College.

iv) User input

User input has been sought at various levels right from data dictionary definition to screen painting. The growth of ENGINFO has been an evolutionary process with users defining needs at the various stages of development, with the systems developer working interactively with end-users in meeting the various needs.

3.7 Data Flow:

The attached diagram describes the data flow within the central data bank. The boxes represent the various databases while the circles represent application programs that access these database to perform queries/updates etc. Looking specifically at the faculty database segment of this diagram, the faculty database is related to the graduate student database through one of the advisor fields in the graduate student MS/PhD segments, and to the Funding agencies through the column 'principal investigator' in the research grants and contracts database.

The primary databases in the central data bank are:

i) Undergraduate Student database
ii) Fee collection system*
iii) Laboratory usage analysis database*
iv) Graduate student database
v) Research grants and contracts database
vi) Faculty database
vii) Course history database
ix) Operations management - Space database

(* not included in the original design, developed in 1990, 1991)

The databases listed above are independent entities in themselves and are related to each other only through the key fields mentioned above. ENGINFO capitalizes on these implicit relationships in meeting information needs by executing join operations on otherwise unrelated databases derived from different, totally incompatible sources (viz. Microvax - Mass11, Mainframe, local spreadsheets).

3.8 Application programs:

Application programs created using Allbase/4GL access these databases and perform online queries for the end-user who is unaware of the structure/setup of the databases. To the end-user, a query would fetch information from a number of data tables derived from a number of data sources. For example, if the user were looking up the profile of a faculty member, he would simply need to key in his name and the application program would scan the faculty primary data table, then the Research Grants and Contracts Database, then the Courses Database and then the Graduate student database. The user would then see a complete profile of the faculty member without even being aware of the existence of all these databases.

3.9 Access tools:

Access to ENGINFO is through terminals connected to the Engineering College network or through PCs using terminal emulation software such as Procomm or Reflections

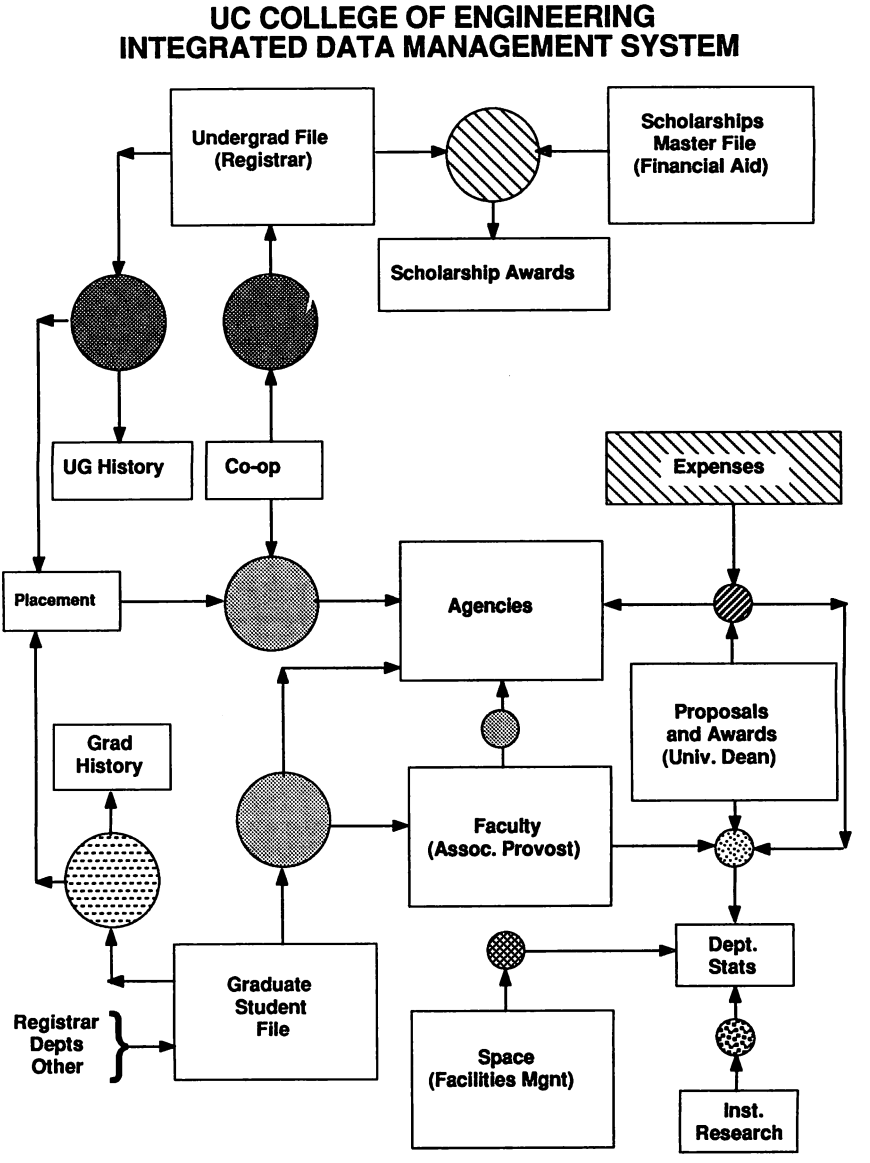


Figure 2: ENGINFO - Data flow

3.10 Data sources and access:

The following table summarizes information on the various data sources.

Database	Owner	Software/Hardware	Access tool
Undergrad Student Data	Registrar	Mainframe	Report Writer (Answer DB)
Graduate Student Accounts	Registrar	Mainframe	Report Writer (Answer DB)
Space	Facilities Management	Mainframe (Nomad)	Nomad Representative
Inventory	Facilities Management	Mainframe	Batch Jobs
Grad-Scholarships	University Dean	Mainframe	Batch Jobs
Research Projects	University Dean	Microvax	Mass 11 Manager
Professional Practice	Professional Practice	Revelations	Batch Jobs
International Studies	ISO	Dataease	Batch Jobs
Lablog	OCC	Lan Server	Mail
Courses	Registrar	Mainframe	Batch Jobs

3.11 ENGINFO - Summary

ENGINFO, described in a nutshell is nothing but a pseudo interface that provides users with access to a diverse collection of incompatible databases. It has evolved as

i) A comprehensive, college specific data management system within a larger university setting

ii) An assembly of data from several data sources providing users with a common interface to hitherto unavailable data

iii) A mechanism that enables institutional data to be used in conjunction with locally generated data for various reporting, querying purposes

iv) An information processing tool and not a data generating tool

v) An online transaction processing system

vi) A powerful analytical tool

vii) An office automation system

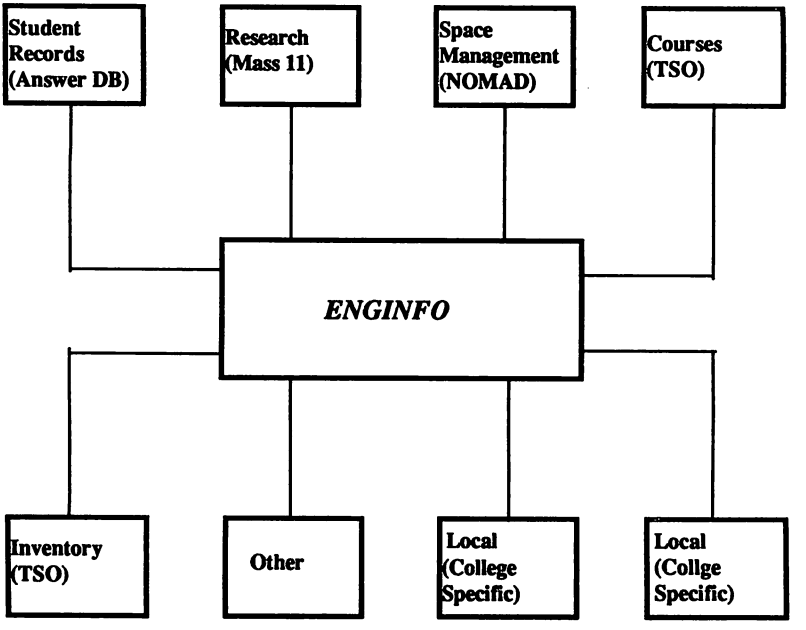


Figure 3: ENGINFO - Data Sources

4. ENGINFO - Details

The rest of this paper will be devoted to a discussion of some of the key aspects of ENGINFO viz: design approach, data update issues, data integrity issues, report generation, flexibility, online transaction processing, security, views ,graduate student data management, lablog analysis, pseudo front end processing and SQL related issues

4.1 Design approach

Behind the dataflow visible to the end users is the logical design of relational tables different in structure from the apparent end user view. The reason for this approach is the following. The central data bank has in it some fields which are entered/updated at the college level, some fields that remain fixed and some fields that are refreshed from the institutional databases. To facilitate this, data is stored in several different tables. For example consider the graduate student database. Fields that can be altered by the department are stored primarily in the MS and the PhD segments. Fields that are changed at the Institutional level are stored in the 'keys' segment. Records that remain unchanged are those pertaining to course history etc. and these are stored separately.

4.2 *Data update issues*

Data residing in the College's data bank can be classified into three broad divisions based on the update authority:

1. Data generated from within the college. The college has full update authority on this data. Updating or deleting data falling under this category has absolutely no impact on the Institutional databases. An example would be the field "grad_advisor" in the graduate student database.

2. Data generated within the Institutional database, over which the College has no update authority. The College is not empowered to change such data, and the data reported in the institutional database is the final word, and hence this category is also not of much concern. Examples would be the course grades or the QPA (quality point average) earned.

3. Data maintained in the Institutional Database, on which the College has update authority. This is an area of concern since updates on the College Database, regardless of the data residing in the Institutional database would lead to the existence of two different databases. A good example is the field "stud_major". If a student changes his major, if the college updates the local database and if the Institutional Database is not updated, it becomes a major problem.

Developing a rigid protocol for data updates is thus a major issue as far as data concurrency is concerned and hence the following policies have been strictly adhered to while developing applications :

i) 'Read only' restriction has been strictly imposed on all those fields falling under category 2 above.

ii) Users with the relevant update authorities will update fields under category 1. These could be users at the Departmental level as well as users at the College level. Logging mechanisms set up will keep track of the users that instituted changes in fields as well as the date and time during which the change was made.

iii) Update authority on potential problem fields will be severely restricted. Logging mechanisms will keep track of users that instituted changes. On a periodic basis, the system would run a report, which would inform the Engineering Administration of the changes made. Changes would then be made on in a centralized manner onto the Institutional Databases. For example, if a masters student completed his requirements and became a doctoral student, the department would make a change in degree goal

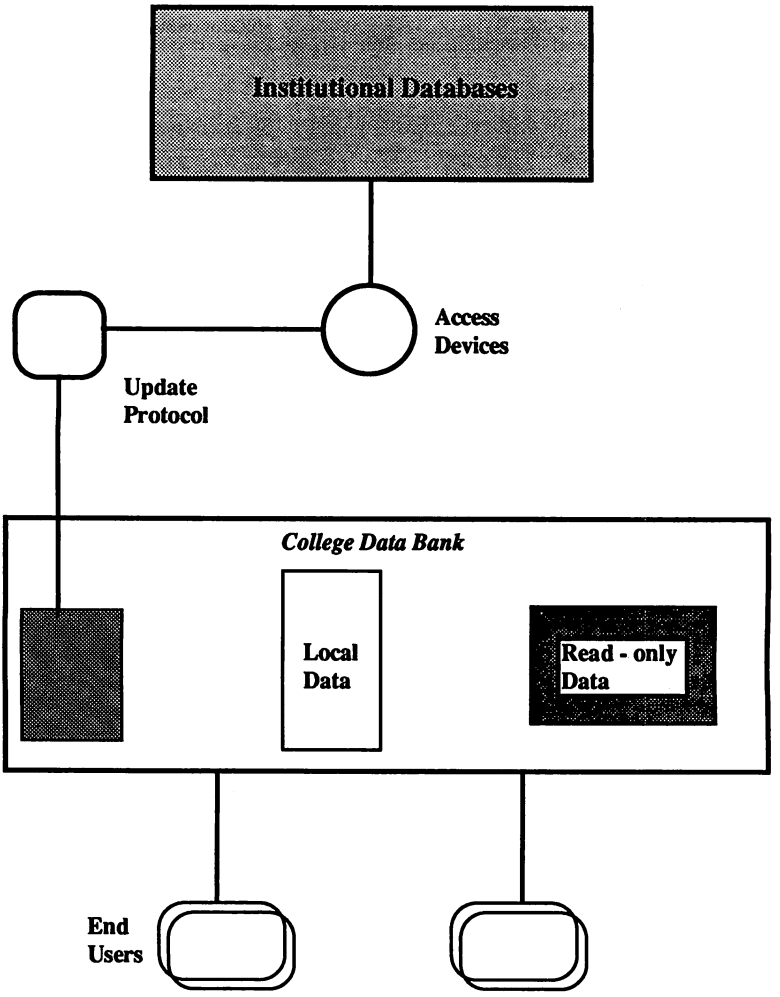


Figure 4: ENGINFO - Update Issues

in the College's database. This change would trigger a report, based on which the office of Academic Affairs would make a change in the field degree_goal in the Institutional database. In the past the situation was that, such a change would have been recorded in the student's file (hard-copy), but would not get recorded on to the Institutional database unless it was promptly and properly communicated. Thus this update mechanism overcomes the problems of limited access and data concurrency.

iv) The College database will be flushed periodically with the data from the Institutional database in order to update it. Thus in the Research Grants and contracts database, the number of entries would increase every time an update was made, while in the case of the student database, changes in registration and grade information would be reflected on the College database every time an update was made.

The optimal update cycles for each data base have worked out taking into consideration several factors such as the nature of the data, the nature of the changes and the frequency of changes.

4.3 Security

Four levels of security are provided in ENGINFO in order to prevent unauthorized access to information:

i) Without a valid HP/UX user-id and password a user will not be able to logon to the system and run the database programs

ii) HP SQL provides the next level of security in that the database administrator needs to provide the necessary connect, select, update, insert and delete authorities in order for the user (with a valid user-id) to be able to access and use the databases

iii) The third level of security is provided by HP ALLBASE/4GL wherein an ALLBASE user-id (different from the HP UX user-id) is provided by the database administrator in order to allow the user to run the database programs

iv) The fourth level of security is provided by HP ALLBASE/4GL at the menu level, in that certain menu options will not be accessible to certain users.

These four levels of security should guarantee the prevention of unauthorized access to information.

4.4 Report generation on ENGINFO

The combined strengths of HPSQL, Allbase and the design principles behind ENGINFO have rendered possible several reports run from menus, reporting information summarized in various forms. One such example is the monthly research award activity report. this report is driven primarily by the College's current faculty database. It processes the data from the research database which is refreshed on a monthly basis to provide a running tally on faculty performance over the last three years. Another example is the comprehensive Annual research activity report , a portion of which is reported hereunder, where the funding history of the College with various classifications of funding agencies is reported

4.5 Office automation

The report writer in Allbase/4GL has been used to automate the generation of letters of offers of scholarships to graduate students. This has been achieved by creating several data lines on

the report and then painting the report with the text of the letters of offer, with the data fields sandwiched between. This project has resulted in the elimination of redundant efforts in terms of keying in data several times in order to get these letters run.

4.6 Flexibility :

Advantage has been taken of the independence of data from the programming tools, in modifying data tables as and when required. ENGINFO has been steadily evolving and at several stages fields have been added to and have been taken from the databases. In one specific case one department required certain data fields from institutional databases while all others did not. This problem was resolved by creating a new table with just the specific fields entirely for the use of this one department.

It is a rather cumbersome operation to have these changes taken care of in the application program, as it involves a series of steps. The advantage of our approach in having seven different versions of the same application takes care of some of the flexibility problems.

4.7 Online Transaction Processing

Part of ENGINFO, is a fee collection/billing system which grew as an offshoot from the undergraduate student database. This functions of this system include i) blanket billing of all students registered during a term, ii) Menu driven system to record fee payments iii) Batch jobs to filter out lists of students with payments due past deadlines iv) Generation of letters indicating the above.

Online data retrieval for entering payments, is carried out using the SQL select command built into the 4GL application program and the data entry and update operations are fairly routine. Daily transaction reports are generated, and the totals reported are compared with the amounts collected by the college business office. Periodic batch jobs on the system pick out unusual transactions for further scrutiny.

The report writer is made use of in printing bills. Data from the student database is used in generating these personalized bills. There are only three lines of data elements in each bill, and the rest of the text is preprinted. These invoices/bills are run on a HP laser-jet printer.

4.8 Case discussion

4.8.1 Graduate student data management

Figure gives a representation of the data tables in the graduate student database. The operations performed on the graduate student database are described in figure .

The graduate student database serves the following functions:

i) Monitor student registration
ii) Monitor student academic progress
iii) Serve as a storehouse for information not provided in institutional database
iv) Process information for generating reports
v) Automate office operations

4.8.2 Design of applications/databases:

End users access these database only through 4GL applications. Seven different departments need to access these applications and accordingly there is a master application and six departmental versions. Users from various departments are provided access only to their data

GRADUATE STUDENT DATABASE - SCHEMATIC

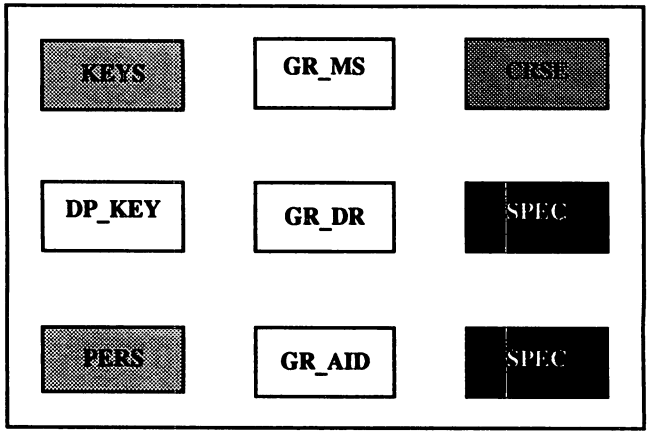


Figure 5: Graduate student database - segments

GRADUATE STUDENT DATABASE - SCHEMATIC

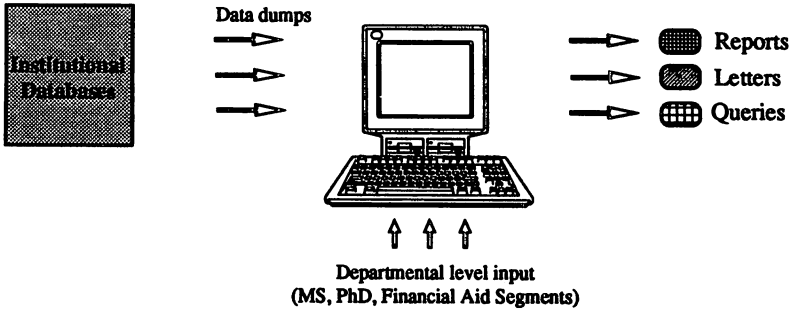


Figure 6: Graduate student database - Operations

through data views and security mechanisms built into the application programs on 4GL. ENGINFO in this regard deals with classes of users and not with individual users. Each class of users or each department is provided with a HP -UX account, and each of these accounts is treated as a group. Within each group are the individual departmental users. At the menu level, security is provided using Allbase's built in features.

The graduate student database tables have designed taking into account the complex nature of data. The various segments of this database are:

i) Graduate student - Key fields
ii) Graduate student - Academic information
iii) Graduate student - departmental key fields
iv) MS segment
v) PhD segment
vi) Financial aid
vii) Personal information
viii) Transactions

Segments i) and ii) are refreshed with data from institutional databases while segments iv), v) and vi) are entirely college specific.

4.8.3 Update mechanisms:

Segments iv) v) and vi)do not pose any problems at all as this data is owned by the College alone. ii) doesn't pose any problems either, since this data is not alterable by the college but the grey area represented in i) and vii) pose problems. These problems occur because, changes in data occur at the college level, and if they are entered into ENGINFO they would result in concurrency errors at the University level. This problem is solved by the mechanism discussed in one of the preceding sections.

To illustrate it further, when a student completes his Master's degree, his degree objective changes from a '3' to a '4'. The student's department is the first entity to receive this information, but this information does not become official until it is recorded into the institutional database. This problem is dealt with in the following manner. The department goes ahead and changes the students data in ENGINFO. This change triggers a report at the College's Academic affairs office which alone has update authority on the Institutional database. The college office, going by the transaction report enters the change onto the institutional database and the time-lag here doesn't have to be more than one day.

4.8..4. Office Automation

Awards of scholarships are made through ENGINFO and offers of scholarships are printed out on a HP Laser-jet printer, hooked up to the HP 9000 using Allbase. Report painting is cumbersome in this case, since modification of text is not all that sophisticated. However, it simplifies a good number of operations that need to be carried out otherwise.

4.8.5 Lablog analysis:

Problem description: The academic computing facilities at the college are used by the entire body of students from doctoral candidates through freshmen. Each user on the network is provided with an 8 character long id and a unique password. The userid is a combination of the users last name and his social security number.

It is necessary to keep track of the level of usage in various labs during various times of the day in order to determine peak load requirements and in order to plan on resource acquisition etc. Further, the College charges a computing fee to its undergraduate students, and hence .it is necessary to keep track of the various student segments that use the computing facilities. Students access machines through a file-server which logs each of the students' activities.

Weekly dumps of these files are mailed to the HP 9000. An extract of such a log is shown in the attached figure.

Such a data dump is loaded into an SQL database, and the aggregate functions in SQL are made use of in obtaining statistics pertaining to lab usage broken down by software, time of the day etc. Such a report is shown in the attached figure. This database is used again in conjunction with the graduate and the undergraduate student databases to obtain a summary of usage with respect to departments and classes.

4.8.6 Other databases and data flow:

The faculty activity report run once a year integrates data from the faculty data table, the course database and the research projects database both derived from institutional databases. The faculty data table is the only one that is updated at the college level.

4.9 Pseudo front end:

The data reported by ENGINFO in some cases is not the final product as further processing may need to be done on it. One such example is the lab usage analysis data. ENGINFO provides this data in the form of a report displayed on screen. (formatted as plain text without headers). This data is viewed by users using Reflections for Windows. The data reported on screen is highlighted, copied on to a clipboard, and then pasted onto Microsoft excel, and plotted out say in this example as a pie chart. Thus ENGINFO serves as a pseudo host in this case, providing usable, semi processed data to a pseudo front end, thereby running applications which neither EXCEL nor ALLBASE could have run by themselves.

4.10 Allbase SQL/4GL pros and cons

Querying: An SQL type of querying tool is ideal for ad hoc queries especially in computing aggregate functions on large databases. This advantage is well utilized in the laboratory usage analysis application, in research data analysis and in the various summary reports (such as the ethnicity report, man power report) that are run periodically.

Flexibility: SQL is flexible in terms of adding or dropping tables or in terms of adding or dropping fields from tables. However, to incorporate these changes in the application programs is quite cumbersome. Several steps are involved in adding a field to a database, especially if the added field is required to appear on a screen or on a report.

Database operations: Updates are not allowed on joins. For storage/retrieval efficiency it is economical to store data in several relational tables, however, when the application encounters a join operation, we run into problems as in the following case. The graduate student 'key segment' has the following fields (soc_sec, l_name, f_name, major, deg_goal etc.) while the financial aid segment has the fields (soc_sec, yr_qtr, aid_type etc.). It is to be noted that the fields l_name, f_name are stored only in one segment (table) of the graduate student database. However when financial aid information is to be updated, the user needs to look up the student's name as well. If we used a join operation to bring up the student's name on the screen, an update on the financial aid database would not be possible. We have overcome this problem by running to SQL logic blocks, one to retrieve l_name and f_name from the 'keys' segment (provided the user has the required select authority on the record), and another to bring up the financial aid information.

A detailed discussion on each of the problems encountered with SQL is beyond the scope of this paper. However, most of these problems will be discussed at the paper presentation.

5 Conclusions:

ENGINFO has been a success and is making strides towards realizing all of its objectives. Conceptually it is simple though unique in approach, and it has realized the objective of

Computer	Date	Time	User ID	Start/Finish	Application
129.137.012.056	910527	074724	chen3634	start	MATLAB - GPP
129.137.012.056	910527	074924	chen3634	finish	MATLAB - GPP
129.137.012.056	910527	075022	chen3634	start	MATLAB
129.137.012.056	910527	075353	chen3634	finish	MATLAB
129.137.012.056	910527	075430	chen3634	start	MATLAB - GPP
129.137.012.056	910527	075614	chen3634	finish	MATLAB - GPP
129.137.012.083	910527	075719	guo8945	start	WordPerfect 5.1
129.137.012.056	910527	075754	chen3634	start	MATLAB - GPP
129.137.012.056	910527	075958	chen3634	finish	MATLAB - GPP

Figure 7: Lab usage analysis - raw data

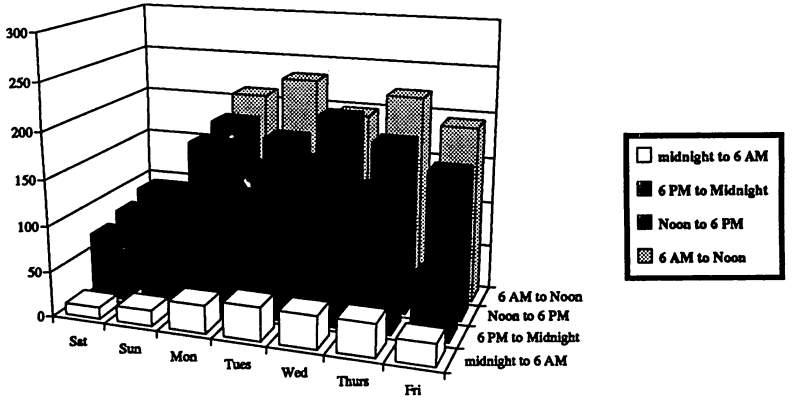


Figure 8 : Lab usage analysis, sample results

providing a uniform interface to several institutional databases distributed all over campus and it has provided a platform for the College to carry out a whole range of projects otherwise impossible. In the absence of ENGINFO each department would have resorted to a PC based database of its own, and there would be the usual scramble for information during the various reporting seasons.

The long term goals of ENGINFO are to continue to expand the range of data processing, information retrieval activities . In the graduate student data management area, provision will be made for computing QPAs and for processing applications for admissions through ENGINFO. Financial management is another of the venues under consideration and so is inventory management.

The University of Cincinnati is presently involved in the process of charting directions for a comprehensive SIS (student information system) for the university as a whole. ENGINFO will serve as a pilot for this project and will continue to fill the gaps in the SIS from the college's point of view.

References:

1. ENGINFO, Integrated Data Management System, College of Engineering, University of Cincinnati, Inaugural Report, January 1990.

2. College of Engineering, Strategic Plan, September 1989.

3. A strategic plan for computing within the College of Engineering, September 1988.

4. Towards an operations-oriented approach to information systems design in public organizations, James M. Tien, James A. McLure, Journal of MIS, 1990.

5. Specifying System Requirements: A framework of current techniques, Gayle J. Yaverbaum, Journal of Information systems management, Winter 1989.

6. The data consistency conundrum: deferred versus immediate checking, Tony Hatoun, Journal of Information systems management, Summer 1989.

7. Evolutionary Development of Information systems, Levent Orman, Journal of MIS, Winter 1988-89.

8. The importance of userware, Karen Gelman Strouse, Journal of Information systems management, Summer 1989.

9. Data systems to support student selection, placement and retention at the Univer sity of Illinois at Urbana - Champaign, Howard L. Wakeland, American Society for engineering Education, Annual conference June 1984.

10. The Johnson Space Center management Information System, Lloyid Erickson, Nomad User Conference, May 1990.

10. [illegible] Closer Union. May 1930.

9. [illegible] Journal of International Economics, Vol. [illegible]

8. The Economics of Customs Union. Basil Blackwell, Oxford. 1954.

7. [illegible] Journal of International Economics, Vol. [illegible]

6. [illegible] Journal of International Economics, Vol. [illegible]

5. [illegible] Journal of International Economics, Vol. [illegible]

4. [illegible] Journal of International Economics, Vol. [illegible]

3. [illegible] Journal of International Economics, Vol. [illegible]

2. [illegible] Journal of International Economics, Vol. [illegible]

1. [illegible] Journal of International Economics, Vol. [illegible]

TITLE: Introduction to Unix -- Part I

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2070

TITLE: Introduction to Unix -- Part II

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2071

TITLE: WA-6 The Multi-Language Approach to Unix Commercial Application Development

AUTHOR: Colin Bodell

Micro Focus

FINAL PAPER NOT AVAILABLE, HANDOUTS WILL BE PROVIDED AT TIME OF SESSION.

PAPER NO. 2072

Paper Number 2074

PERFORMANCE MANAGEMENT IN A DISTRIBUTED COMPUTING ENVIRONMENT

Dave Glover and Wayne Morris
Performance Technology Center
Hewlett Packard Company
8050 Foothills Blvd.
Roseville, Ca. 95678

Abstract

This paper examines the trend toward distributed, heterogeneous computing environments incorporating mainframes, distributed mini-computers, client-server UNIX[r] based workgroups, and PC local area networks. Current organization profiles, techniques and facilities used for distributed performance management are examined. The progress of measurement standards in the UNIX performance management community is described and evaluated as to how it contributes to managing this environment.

INTRODUCTION

Many organizations are experiencing, or are planning for, a dramatic shift from a centralized data processing center, to a distributed multi-vendor networked computing environment. PC local area networks, client-server UNIX based workgroups, as well as distributed, mini-computer systems are playing an increasingly important role in the corporate computing environment. Applications being implemented within this environment include "mission critical" applications which can affect the competitiveness and profitability of the corporation.

Performance and capacity management within this distributed, heterogeneous environment presents new challenges both technically and organizationally. The distinction between network and system management is becoming blurred with the introduction of Integrated Network Management Systems (INMS) from a number of vendors, such as HP OpenView Network Node Manager.

Technically, the performance and capacity management of this environment becomes more complex as more of the distributed resources are included in the corporate network. For example, application complexity increases as software and databases are partitioned and distributed. Resource utilization improves with dynamic job scheduling across CPUs in a workgroup environment, however this makes workload characterization more difficult. Problem diagnosis, resource management, system tuning and capacity planning within the distributed environment also include network components.

Corporations taking advantage of distributed computing environments are examining the effectiveness of their current organization profiles for network, system and performance management. The distributed network environment requires both centralized and distributed management elements. There is a trend toward centralized strategic management of the corporate or enterprise computing environment with decentralized operational management of distributed sub-nets, workgroups and LANs.

INDUSTRY TREND TO DISTRIBUTED COMPUTING ENVIRONMENT

Many industry surveys show growth rates for client-server workgroups and local area networks to be 30% - 40% for the next 3 to 5 years, while growths for the mainframe market are projected to be 7% - 10% for the same period.

Not only is there a trend toward a greater implementation of distributed computing, the nature of distributed client-server applications is also changing. A recent study of Fortune 1000 companies by the Business Research Group indicated that 80% of companies surveyed had implemented, or were planning to implement, client-server applications. Of these, 75% considered one or more of these distributed applications to be "mission critical".

Heterogeneous networked systems are made possible through the adoption of evolving communications standards. These heterogeneous systems are multi-vendor and utilize multiple operating system platforms. Distributed databases and dynamic allocation of jobs among heterogeneous systems are becoming realities.

As distributed systems and applications are implemented, management of the distributed environment becomes increasingly important. Performance management is one component of overall distributed environment management. Other components of distributed management which are often mentioned as key areas include: security, software backup and distribution, fault, configuration, and accounting management. All of these facilities will be required to effectively manage the distributed environment. As management becomes more centralized, the integration of these various components will increase the effectiveness of engineering and management personnel.

Another reason for the growth of distributed computing environments is the acceptance of UNIX as a viable commercial application platform. It is estimated that in 1992, the number of new UNIX commercial systems will be greater than 1.5 times the number of new UNIX technical systems. A recent IDC report indicated that this trend is a result of companies seeking hardware independence through open system environments, particularly as their computer networks grow in complexity and size. The majority of these UNIX implementations are used for distributed client-server applications.

CURRENT DISTRIBUTED PERFORMANCE AND CAPACITY MANAGEMENT

The distinction between network and system management is diminishing with new integrated network management system (INMS) capabilities. Performance management is being incorporated into these INMS products. This facilitates changes in the way network, system and performance management tasks are performed.

Currently, network and system performance management are quite separate. Integrated network management systems are available from a number of vendors which provide an umbrella under which integrated applications for distributed performance management can be developed. There are some products available for examining performance in distributed network environments, however they tend to either be proprietary in nature and don't operate across multi-vendor platforms, or are aimed at a specific aspect of the distributed environment, such as backbone network utilizations or Network File Server (NFS) subsystem performance management. These products provide information to help understand the network activity, but do not address the information needed to understand the "network of systems" comprising this environment. A number of vendors are seeking to extend these capabilities to encompass more of the distributed, heterogeneous environment.

The major inhibitor for these efforts is the lack of common performance metrics and access mechanisms in the heterogeneous environment. Currently, each computer vendor, performance software vendor, or large end user is faced with the task of operating system modification to collect the necessary data and develop a kernel interface to move the data for access by their tools. Since each vendor has their own unique method of access, performance management tools remain limited as to their portability in a multi-vendor environment.

FUTURE DISTRIBUTED PERFORMANCE AND CAPACITY MANAGEMENT

As the distributed environment gains acceptance and maturity, facilities for managing distributed environments will be improved. In the Open Systems Interconnection (OSI) model, performance and capacity management have been identified as key components of distributed management together with fault, configuration, accounting and security management. To address the needs of a standard measurement base, several vendors have joined together in a Performance Management Working Group to develop the requirements for a performance management subsystem for the UNIX operating system. This group includes the two major developers of open systems based platforms: Open Software Foundation (OSF) and Unix International (UI). The goal of this group is to help define standards for performance measurement interfaces and capabilities. These standards will then be implemented across multiple vendor platforms which would then support the development of network system management tools in a heterogeneous environment.

As the distributed computing environment becomes larger and more complex, traditional techniques for performance management become unwieldy and inefficient. Thousands or even hundreds of intelligent nodes can't be managed by waiting for user calls, monitoring individual nodes, or through periodic management reports. The network element of performance also becomes increasingly important.

Because of the interdependencies between network and system performance, performance management facilities should encompass both network and system elements, and be integrated within a single, graphical user interface. This will facilitate faster problem identification and characterization and facilitate handoffs between network and system management personnel.

The large number of monitored nodes will necessitate employing management by exception through the use of alarms based on defined thresholds for resource utilizations, response times and transaction throughputs. Expert systems will be used to relieve the user of low-level problem diagnosis, to filter multiple, related alarms to determine the base problem, and to provide guidance for the diagnosis of more complex situations. This will ensure management personnel are not overburdened with alarm messages and increase their effectiveness. A predictive alarming capability to warn of potential performance problems, based on the addition of a specified workload, would allow day-to-day performance management to be proactive rather than reactive.

The ability to logically partition the distributed environment for management purposes will be required. Varying levels of information detail will be available as the problem is isolated, characterized and resolved. During this process, one network or system manager may want to encapsulate a problem and pass it to another manager or analyst with a different expertize level.

To support performance management within the heterogeneous environment, common performance metrics will be required regardless of vendor platform or operating system. These metrics would be normalized to allow common interpretation rules to be applied across platforms. This will aid in data interpretation and simplify the implementation of expert systems for guided diagnosis. The common set of metrics would be supplemented with additional, platform-specific metrics. Registration of each metric and class of metrics in the Simple Network Management Protocol (SNMP) Management Information Base (MIB) would provide a registry of these metrics to be accessed by intelligent collectors in a networked environment. The MIB is simply a collection of object definitions that are required to control resources within the distributed computing environment. The standards efforts mentioned earlier should help resolve the availability of the data and the registration of the metrics.

As performance thresholds are crossed, and alarms triggered, a "trouble ticket" application would allow the user to capture pertinent information, and record the method and sequence used to isolate, characterize and resolve the problem. This information would then be indexed and retained for future reference by the user or by an automated guided diagnosis application.

Capacity planning and performance prediction will have to include aspects of network performance, and will be complex in client-server environments, where the impact of clients on servers and the interconnecting network will have to be incorporated into environment models. It should also be possible to model different network and system configurations to determine if historical alarms could have been avoided. Planning functions should also project future support loads by examining trends in generated performance alerts and recorded trouble tickets.

ORGANIZATIONAL STRUCTURES

There is a trend toward centralized strategic management of networks and systems, particularly for capacity planning and distributed resource management. Completely centralized management of distributed systems, LANs and workgroups may not be possible due to organizational, management, staff, or geographic constraints. Distributed performance management facilities will therefore have to also support decentralized operational management, as well as combinations of

decentralized and centralized management. In most corporations, some level of centralized management reporting and centralized management support will be desirable, particularly if there is limited remote expertise.

Network and system management capabilities will be more tightly integrated with responsibilities divided by function - fault, configuration, performance, accounting, security rather than by platform or geographic location. This transition will take some time, but already is becoming apparent in many organizations where network and system management staff are co-located and closely integrated, even if the two groups have separate reporting structures.

SUMMARY

The trend toward distributed heterogeneous environments is clear. Managing the performance of those environments will require changes in the network and system management organizations as well as new performance and capacity management capabilities and techniques.

The major differences between performance management in centralized environments and performance management in distributed environments are the integration with other management functions, such as configuration and fault management, and the complexities introduced by multiple system platforms, network elements and the large number of intelligent nodes to be managed.

This complexity requires the use of sophisticated performance management capabilities which are integrated with other network and system management facilities, and which operate on the principle of management by exception. Functionality will be required to automatically determine problem causes and either correct or offer guided diagnosis to performance management staff. New techniques for capacity planning will be required which incorporate network elements and the impact of client-server applications.

The key to the development of performance management in distributed environment is the identification and creation of common performance metrics and the standardization of the access mechanisms. The efforts by the Performance Management Working Group and the participating vendors and the efforts to get this work adopted by the POSIX 1003.7 will provide the springboard for the development of the necessary tools.

2075

A Talking Computer that Monitors Remote Computers

Tony Jones
Hewlett-Packard Co.
2 Choke Cherry Road
Rockville, MD 20850
(301) 921-6203

Abstract

HP NewWave provides the capability for users to develop an integrated application environment. As an example of this integration, this paper will describe a NewWave based personal computer in development that monitors disc space on HP 1000s, HP 9000s, and Apollo computers over a Local Area Network (LAN). If the NewWave computer detects that a remote computer is running low on disc space, it will place a telephone call to a designated system manager and announce the status of the specific computer's disc space. If the system manager does not answer the phone and a voice mail system is available, the NewWave computer can leave a message. The system manager can then call in for messages and address the problem of low disc space. In addition, the paper will also describe general purpose techniques for integrating the NewWave computer into existing environments. Finally, the presentation will include a demonstration of the NewWave computer monitoring remote computers.

Introduction to the NewWave Environment

The consistent and predictable graphical user interface makes the environment easy to learn and easy to use. Users can focus on results, not the tools used to accomplish them. The environment is based on Microsoft Windows 3.0.

Those working in the NewWave environment can move quickly and easily from one application to another because of the high level of integration. NewWave primarily consists of two components: the Object Management Facility (OMF) and the Application Program Interface (API). Data files and the executable programs which interpret the data are called objects. When the user accesses the object, the OMF starts the application and passes it the appropriate data file. Figure 1 is an example of the NewWave environment.

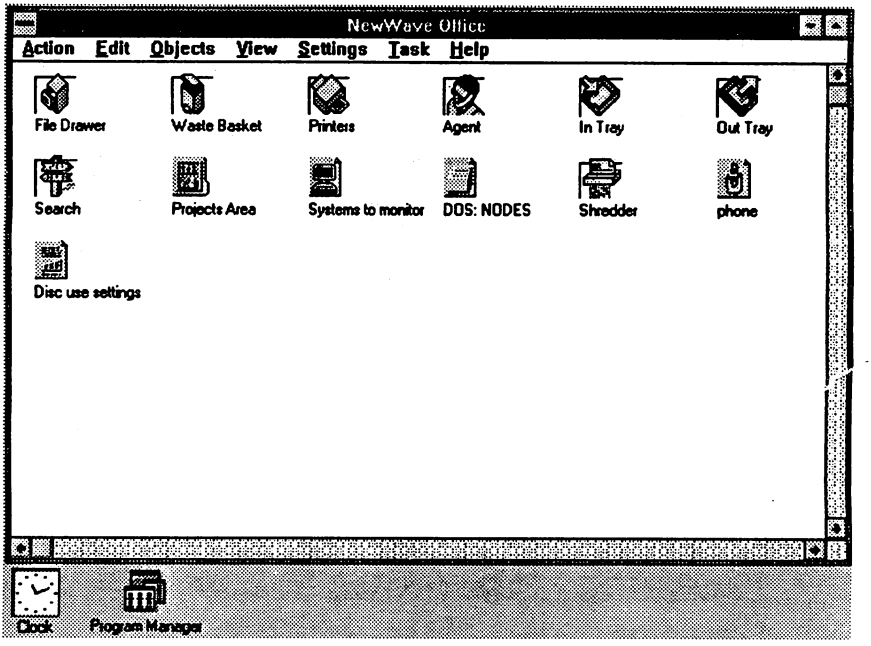


Figure 1: NewWave Environment

Objects are represented as icons. Each NewWave object has a title which is specified by the user. These titles can be

up to 32 characters, to provide a way to more clearly describe the item. This is in contrast to the typical PC environment where the user must fit a description into an 8 character file name. In addition, NewWave maintains a set of attributes for each object. These attributes include the creator, type of object, creation date, last modification date, and a comments area. Figure 2 is an example of the attributes area.

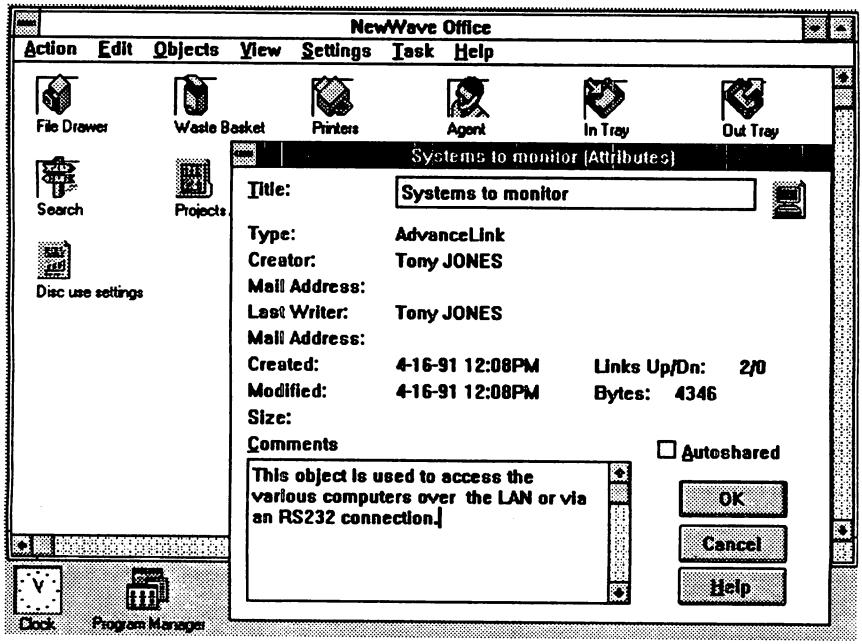


Figure 2: Object Attributes

Objects can be combined with other objects to create compound documents. The OMF keeps track of object relationships through information links. A user can link an object to multiple objects so they "share" the same information. If the information changes in the source object, the OMF notifies and updates all dependent objects. The user is not burdened with finding the files which need to be updated.

NewWave also allows objects to pass data to other objects through a data passing link. An object can focus on

retrieving information and depend on a graphics object to plot the information through the use of a data passing link. Visual links allow an application to provide a view of another object's data.

Application developers can take advantage of existing NewWave applications and reduce re-creation of existing software. Since the OMF provides a consistent architecture for data sharing among applications, newly developed applications will work with current NewWave applications. To facilitate development further, NewWave provides the Application Program Interface (API).

The API provides access to system wide services: context sensitive help (Figure 3), task automation by the Agent, and computer based training. The context sensitive help system is a complete facility for developing and displaying help messages.

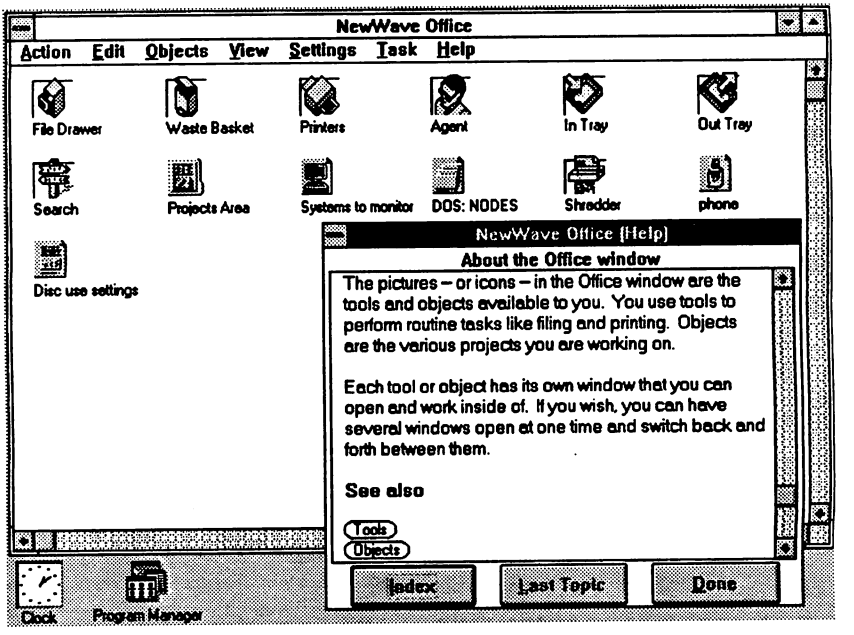


Figure 3: Context Sensitive Help

There are three methods for obtaining help: index, inputting

a topic, and an on screen pointer (question mark). The index can be used to scroll through the available help topics. If the user chooses to input a topic, the help facility jumps to the matching topic as the user enters each character. The third method is a context sensitive question mark where the user can place the pointer on a specific item in question. Developers can also take advantage of the hypertext-like facilities to provide links to related help topics. A related topic is designated by a group of words circled in the help text.

System wide task automation is provided by the agent. User actions are translated into commands and recorded in an agent task object or a menu task. For example, figure 4 contains a menu task that empties the waste basket.

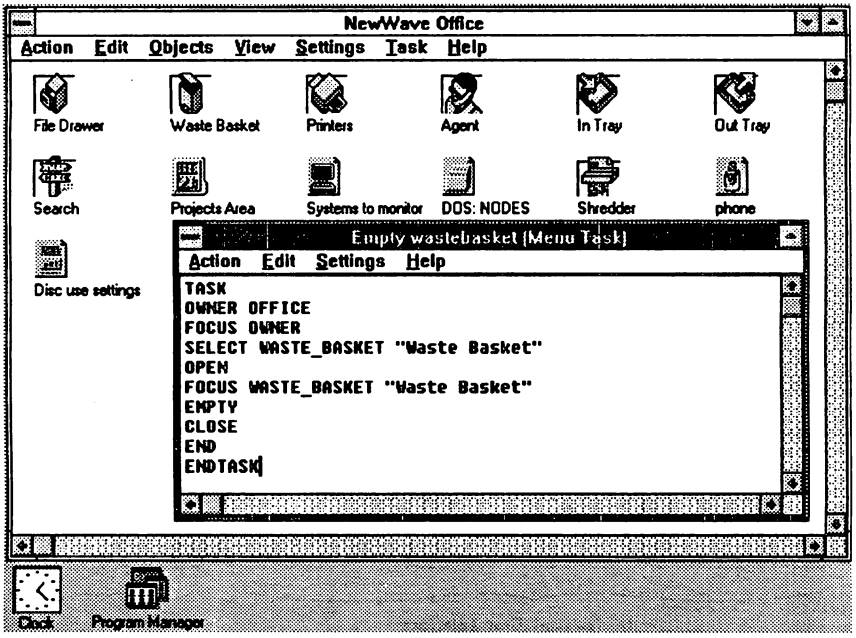


Figure 4: "Empty waste basket" Task

The agent can be triggered by an event or time, by an application call, or by user request. Developers can provide agent tasks which automate routine user activities. Since the API can monitor and control applications, NewWave builds on this to provide Computer Based Training (CBT).

CBT enables developers to provide interactive training to application users. Developers can design one application and have it operate as usual or have it controlled by a lesson. Furthermore, the lesson developer can work without knowledge of Microsoft Windows or NewWave programming. The course developer and application developer can work in parallel.

The Native Language Support (NLS) facility enables developers to provide applications which adapt to languages and customs of other countries. This means native language applications can be produced and maintained with less effort.

Finally, NewWave provides facilities for integrating current MS-DOS and Windows based applications into the NewWave environment. This technique is called encapsulation. Encapsulation refers to the process of building a software shell around an existing application that allows it to achieve higher levels of NewWave integration. Each level requires increased development effort (see Figure 5). Encapsulation is handled by the Bridge Builder.

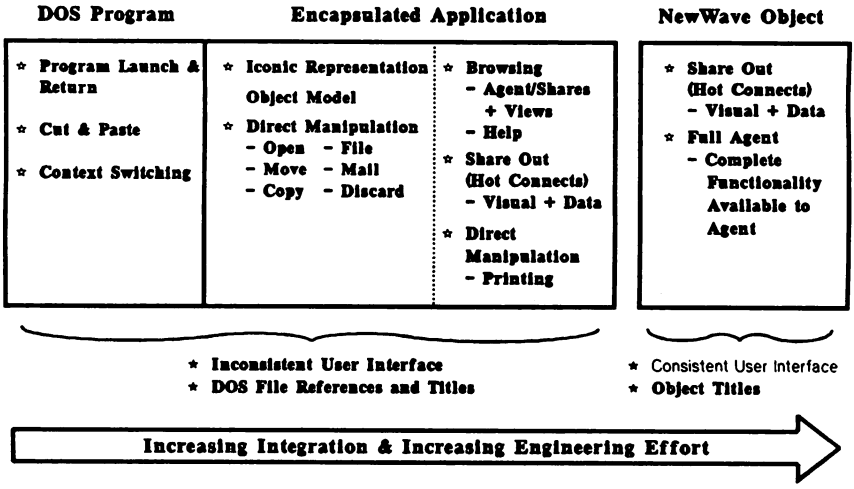


Figure 5: Levels of NewWave Integration

The Bridge Builder

The Bridge Builder is an object which assists in connecting DOS and Windows based applications to NewWave. Once a bridge is defined and installed, an application can appear as an icon. In addition, NewWave manages the location of files associated with that particular object. DOS or Windows applications can appear as DOS objects or tools. Each DOS object has up to an eight character title and associated data file(s). When a user opens the object, NewWave starts the application and passes it the file name(s) such that the user can immediately work with the information. Users can mail, copy, move, archive, share or delete objects.

DOS tools provide functions which are usable throughout the environment. For example, the Waste Basket tool provides a means for users to throw away objects. A user can only have one instance of any particular tool. In addition, tools cannot be copied or deleted. Bridges allow users to treat DOS or Windows applications as objects or tools. To bridge voice capabilities into NewWave, I developed two bridges for the Watson voice product.

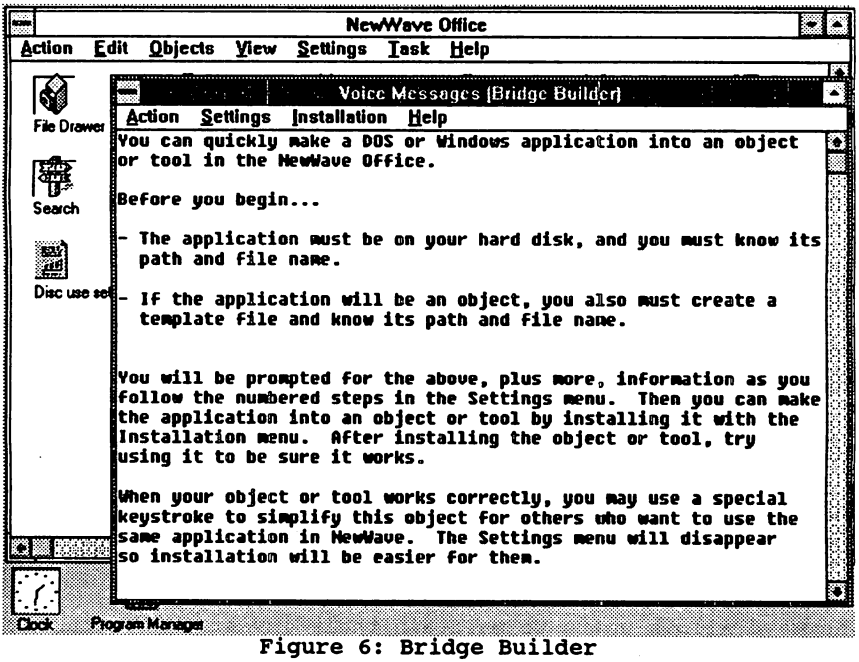


Figure 6: Bridge Builder

Watson Voice Product

Natural Microsystems Corporation produces an MS-DOS based voice product called Watson. Watson is a voice messaging system that includes either a 1200 or 2400 baud modem. The most common methods for using Watson include:

- Personal Voice Messaging. Watson manages incoming and outgoing messages similar to an answering machine, provides dictation capability with voice editing, provides phone book and telephone number dialing, and a personal calendar.

- Multi-User Voice Mail. A group of people can use Watson to send and receive messages.

- Modem. Watson provides 1200 or 2400 baud modem support which is accessible by data communication applications.

Watson uses card files to manage information, for example placing each incoming or outgoing message in a separate card. Five card files are included with Watson: incoming messages, outgoing messages, phone book, dictation, and calendar.

Using the Voice Object/Tool

Watson appears as two icons in NewWave. Icon 1 represents voice objects and icon 2 represents the phone tool. Managing voice objects is very similar to managing other NewWave objects. To create a voice message, select "Create a New..." from the Objects menu, choose a voice message object from the palette of icons and give it a title.

To record a message, open the object and select "Record Speech" then pick up the phone and speak. When you are through with your message, hang up the phone. Each message has a text area available for comments. In addition, each voice message object can contain a complete set of Watson card files. The standard Watson commands are still available for use.

The phone tool provides typical Watson functionality. For example, to use the personal computer as an answering machine or phone book with dialer, open the phone tool. Adding or deleting phone cards in the phone tool has no effect on voice objects.

NewWave allows users to maintain their investment in current

applications while providing an environment that assists in making more consistent application usage. By taking advantage of applications such as the phone tool, AdvanceLink for NewWave, MS cardfile, and agent tasks, one can produce a NewWave based talking PC which can monitor disc space on remote computers and use the phone tool to call system administrators.

The Application

The objective of the application is provide an automated tool that assists system administrators in monitoring disc space on computers throughout an organization. If the administrators can anticipate when they will run out of disc space then they can plan to obtain more space or reorganize current capacities. In addition, the talking computer may help in monitoring computers which are normally neglected on a day to day basis.

Talking Computer Feature Set

- Monitors disc space on remote computers
- Monitors HP9000, HP1000, Apollo
- Uses telephone to contact designated person (system administrator)
- Leaves message in voice mail
- Accesses computers over LAN or RS232
- Administrator can set disc space level for alarm conditions
- Administrator can set time interval between scans

Objects and tool used in disc space monitor application

- MS Cardfile which contains computer information
- NewWave Write object which contains initial disc space information and alarm condition settings
- AdvanceLink for NewWave terminal object used to access computers over LAN or RS232
- Agent task which contains disc space monitor procedure
- Phone tool which contains prerecorded messages for computer by nodename

Agent Tasks

- Add computer to monitor list
- Remove computer from monitor list
- Activate monitoring
- Deactivate monitor

- Change alarm limit parameters

To add a computer to the monitor list

- Add nodename card to cardfile
- Add phone card
- Record voice message for particular computer
- Log onto computer and execute command to get initial disc space usage and add alarm limits
- Set time interval between scans

Information required for each computer

- Nodename of computer
- Type of computer (HP9000, HP1000, Apollo)
- Contact person
- Telephone number
- Mail address
- Table of disc space items to monitor
- Table of alarm limits
- Time intervals between scans

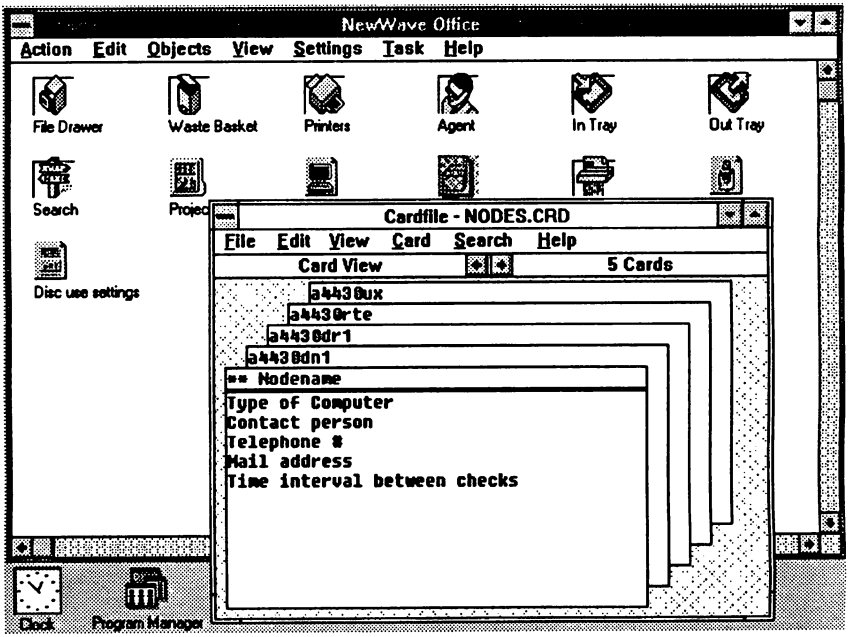


Figure 7: MS Cardfile used for monitor list

Typical message stored in voice message

"Hello. This is the NewWave disc monitor. System _____ is running low on disc space. Please address the situation as soon as possible. Thank you!"

Algorithm for monitoring disc space on remote computers

```
Open MS Cardfile titled "DOS: NODES"
Open AdvanceLink for NewWave terminal object titled:
     "Systems to monitor"
Open NewWave Write object titled: "Disc space settings"
For each card in the cardfile
     Select a nodename
     Retrieve computer type
     Based on computer type
          Select terminal object and connect to node
          Log onto node
          Execute command to get disc space information
             (HP9000: "bdf", HP1000: "FREES", Domain: "du")
          Log results to clipboard
          Log off node
          For each file system on selected node
           Compare current capacity values to alarm settings
     If any of the capacities exceed alarm settings
          Open "phone" tool
          Search for message which contains nodename
          Dial phone number
          If user answers phone
               Play message
          Else if voice mail is available
               Send touch tone commands to store message
               Play message
               Set to urgent priority
               Send message
          Endif
          Hang up phone
          Close "phone" tool
     Endif
Close MS Cardfile titled "DOS: NODES"
Close AdvanceLink for NewWave terminal object titled:
     "Systems to monitor"
Close NewWave Write object titled: "Disc space settings"
```

Future directions

- Store disc usage history
- Call beepers
- Call alternate contact

Conclusion

It should be emphasized that NewWave provides for integration across applications along with task automation. By taking advantage of the environment, users can automate the task of collecting information from remote locations and performing some function based on current data. In this case, monitoring disc space was the selected application. The developer can change the commands to produce other valuable applications. Users and developers alike can create new and exciting solutions by integrating feature sets of current DOS, NewWave and future NewWave applications.

References

Windows screen snapshots generated with "Tiffany" program by Alan Anderson, Washington.

Paper Number 2077

The Client/Server Cookbook : A Recipe for Success

Caroline Ellis and Steve Every

Pinewood Information Systems Division
Hewlett-Packard
Nine Mile Ride
Wokingham
Berkshire
England
Tel : 44-344-773100

1. Introduction

The prospect of implementing client/server computing can cause indigestion. High migration costs, write off of existing hardware and software, and dissatisfied under-productive users could be a recipe for disaster.

This client/server cookbook looks at your existing ingredients. It then shows how to mix those ingredients with additional flavours to produce the most palatable solution; implementation of a full client/server environment whilst minimising training and transition costs and maximising protection of investment.

The paper takes electronic mail and an example recipe. It describes a method for migrating from a pure host/terminal environment through a mixed environment to a full client/server solution.

2. What is client/server computing?

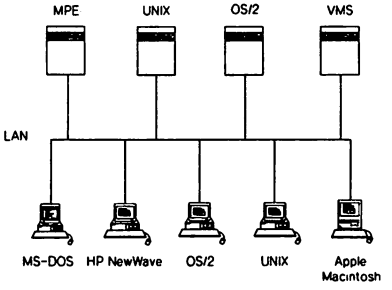
As with any recipe, we must keep a clear idea of the dish to be created. Otherwise, we may forget a vital ingredient and produce a totally different one by mistake. Forget the beef and the Bourguignon turns into soup!

So, let us define client/server computing.

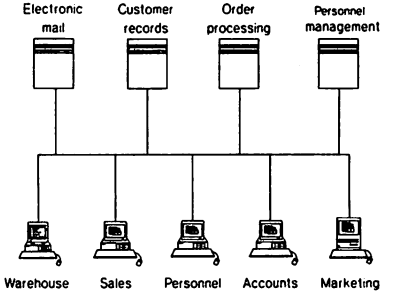
A CLIENT is a machine or software application which requests services or resources on a network.

A SERVER is a machine or software application which provides services or resources on a network.

The following diagram illustrates a typical client/server implementation:



The clients are various types of personal computer. These are connected via a local area network to a range of server minicomputers. For simplicity, printing resources are ignored.



The client workstations may run a variety of applications. Some, such as word processing or spreadsheet, may exist entirely within the PC. Others form the client part of a client/server application. For example, HP NewWave Mail is an electronic mail client which requires a server application, HP DeskManager or HP OpenMail, to provide a full solution. Most client machines will be MS-DOS based PCs, possibly with MS-Windows and HP NewWave in addition. OS/2, Unix or Apple Macintosh clients may also be used for some applications.

The server platforms are usually minicomputers, although this is not always necessary. The more powerful personal computers could perform this function in some cases. Server applications, like HP OpenMail or Oracle, may share a single machine.

Alternatively, in a large organisation, they may be distributed between several machines, each one providing a single function.

Connectivity is very important in the client/server environment. A user may need to access information from several servers, and may wish to run several applications at the same time. Serial connection may be adequate for the casual user, or for remote users who need access via a modem. However, client/server working usually involves significant amounts of data transfer, and serial connection would impose speed restrictions which may be unacceptable.

Local area networks provide the most suitable solution. Cabling is minimised, multi-tasking and multiple connections are simplified, and data transmission is much faster.

3. The benefits of client/server computing

Two examples of client/server computing illustrate the way in which such applications are used.

A data base application may divide the functions as follows:

Client	Server
User interface	Data base
Assemble record	Search for required records
Display record	Extract records
Format extracted records	Format extracted records
Sort extracted records	Sort extracted records
Print records	

Electronic mail may function as follows:

Client	Server
User interface	Message transport
Distribution lists	Directories
Create message	
Include PC files	
Store message	
Read message	
Print message	

In both examples, the user can perform some processing locally, within the client portion of the application and so has full access to the client data at all times. Personal files are stored locally within the client platform, while shared files are stored centrally. This reduces the need for the user to be connected to the server, and also reduces the loading and storage requirements of the server platform, leading to cost savings.

The same data base client could be connected to different servers, providing a consistant user interface and reducing training needs. This may also be true for other applications.

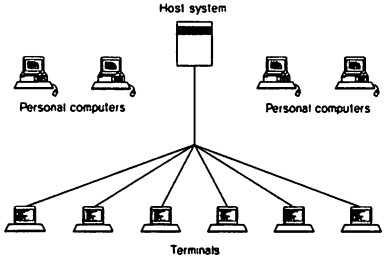
The user can run several client applications at the same time (if the client platform is capable of multi-tasking), either on a single or multiple server platforms. Information can easily be transferred between applications. Multi-tasking allows the user to work in one application whilst transferring data in another, etc.

Client/server computing can offer improvements in usability and flexibility, leading to better productivity. There are obviously financial benefits if productivity is improved, but these will vary from case to case and may be difficult to quantify.

4. Host/terminal computing

Most companies today run applications which reside totally on one computer. Host applications run on a minicomputer or mainframe and are accessed via a terminal. Some users have personal computers which are used for word processing, spreadsheets, etc. The user may also access host applications from the PC, using a terminal emulation program.

The following illustration is of a host/terminal system in which all host access is via dedicated terminals. Any personal computers are separate from this system. Choosing such an extreme case simplifies the comparison with client/server computing.



In this environment, all processing and data storage occurs on the host. The terminal provides a simple, single tasking user interface. The cost per user of the terminal hardware is relatively low compared to a personal computer client. However, the host computer must be larger than its server counterpart because of the greater processing and storage requirements. The user is totally dependant upon host availability for any activity.

Migration from a host/terminal environment shown above to the client/server environment previously discussed involves investment of time and money. If existing applications and hardware are simply abandoned in favour of new ones, the cost can be unacceptably high. There are several issues which must be considered if client/server computing is to be successfully implemented.

5. Migration issues

Investment

Software - it is likely that when moving to a client/server environment new PC software packages will have to be purchased. This cost can be minimised by upgrading software rather than completely writing it off and the cost can be spread over a long period of time, investing in the new software in stages, as and when new user groups are rolled to the client/server system. Old software can still be accessed by users remaining on terminals. It is also likely that host/terminal applications will have to be upgraded or re-written. At this point it should be considered whether to invest in a server product with an established upgrade path to protect investment in the future.

Hardware - a client/server environment may make it necessary to invest in new more powerful PC's, upgrade existing PC's and possibly upgrade or install new networking.

Write-off costs - as users are upgraded from terminals or old PC's these may need to be written off. Also, networks may prove incompatible resulting in write off of some network investment. Write off costs can be reduced by upgrading in stages and using the older equipment in areas that are unlikely to migrate to client/server.

Training

As users move from terminals to PC's and to new PC applications it will be necessary to invest in a training programme to ensure the users are as effective as possible in the shortest possible time. This training is likely to be costly and there will inevitably be a lag where users are not very productive or effective until they are trained and used to the new client/server environment.

Inertia

The training lag mentioned above could be made worse if users show inertia; they may not want to change to a radically new way of working. Education about the benefits of the client/server environment, and hands-on training are ways of minimising this problem.

Productivity

Overall productivity can be impacted during the roll-out of a client/server system. It will not be possible or cost effective to move all users to the new system immediately. This will result in incompatible users, for example electronic mail systems not being able to handle a file type from a new client word processor. Networks may become incompatible and stop communication completely for a time.

Control

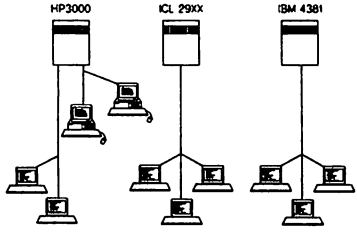
Controlling and managing the roll-out of a client/server system can be hit by problems. Deciding who to roll out first, why, how to minimise the impact of lost productivity, how to standardise and on what to standardise. Control can be the most important factor in the success of a client/server implementation.

6. Electronic Mail Example

This is a real customer example but the customer's name cannot be mentioned. This customer decided to aim for a full client/server electronic mail environment rolled out in stages on a demand basis. Different departments within this customers organisation had preferences for different vendors offerings. For electronic mail the customer aimed to standardise on Hewlett-Packard client/server systems, based on HP DeskManager, HP NewWave Mail, AdvanceMail and AdvanceLink, and then extend the client server idea further based on the success of this initial stage.

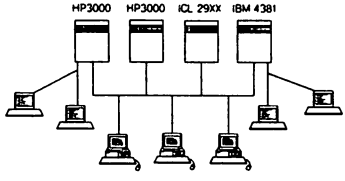
Customer Starting Point

In this example the customer started from a mixed hardware environment with 9 HP 3000 series 4X and 70's, 2 ICL 29XX's and 2 IBM 4381's. Most users were terminal users with a few exceptions using PC's accessing the hosts with terminal emulation. The customer had approximately 5000 users accessing the hosts, with 800 of these users being regular electronic mail users. The terminals were all connected via terminal servers and the PC's were mainly standalone, with a small number serially connected to the servers. Electronic mail (HP DeskManager) was widely used in a small number of departments, a larger number of departments used it on a very limited basis and a small number of isolated departments did not use it at all. All electronic mail users were direct terminal access or terminal emulation access from PC's.



Customer Goal

The customer's goal for the client/server system was to have approximately 10 HP 3000's series 70 and 9XX, 1 ICL 29XX and 1 IBM mainframe. Full client/server was to be implemented for about 80% of users, ie. only where necessary, with older PC's and terminals still made use of by the other 20%. The goal for number of regular electronic mail users was 2000.



Customer Migration Process

Phase One - the customer chose the departments already very interested in electronic mail and using HP DeskManager regularly as the first areas to be migrated to client/server. During the first phase the following activities took place.

Terminals replaced with PC's for all knowledge workers and secretaries.
Existing PC's run terminal emulation (AdvanceLink) into electronic mail with new PC's running full client/server mailing (NewWaveMail, or AdvanceMail for older less powerful PC's).
Other applications remain host based in short term.
Training in stages, all users receive low level training as soon as possible with in depth training to follow.
In the short term PC's are connected using existing cabling with new networking software phased in.

In this way the customer minimised the initial impact of moving to client/server, addressing many of the migration issues outlined earlier. Software and hardware investment cost was minimised in phase one with only necessary users moved to powerful new PC's, with existing PC's still in use. Old terminals were either scrapped or moved to areas where client/server will not be implemented and terminals are still necessary. The write-off cost was reduced as most of the old equipment was past its useful life and was already written off. Initial training costs were high - this was a conscious decision taken to try and minimise the cost of inertia and reduced productivity.

The first phase was a success. Users not yet migrated to client/server began to request the migration to take place in their department next as they began to appreciate the benefits it could bring. The number of regular electronic mail users had increased to 1500.

Phase Two - All departments that wanted client/server electronic mail began the migration. During phase two the following activities took place.

Database updated and client/server database access implemented.
Local area networks fully updated and installed.
PC hardware fully upgraded and further new hardware installed.
Additional training classes for new users.

The customer believes at this point that there is a diminishing rate of return for additional users to be upgraded and trained in the client/server environment. They still have some terminal users and some applications with terminal access only, and whole departments not yet interested in client/server. They feel there will be a stage three in their migration as the departments that have not yet moved to client/server do so. The number of regular electronic mail users has increased to 1700.

Success Factors

The customer has outlined some of the major reasons it feels the migration to a client/server environment has been a success.

- It was recognised from the beginning that old applications and terminals and other existing equipment would not necessarily be written-off. Not all users would be moved to client/server.

- Moving departments to client/server on a request basis after phase one minimised the loss of productivity and inertia caused by a new system. When users migrated they did so because they wanted to.

- Migrating people in stages, particulary using terminal emulation on the PC, helped the users accept the migration more readily than if they had, for example, been immediately switched to client/server from a terminal.

- An extensive training programme was in place so as soon as users migrated there was training available for them.

- User satisfaction and real productivity and effectiveness improvements were documented from the beginning to help prove the benefits of client/server. This enabled further investment to be justified in other departments and made users more inclined to accept the new system.

- Some protection of existing investment was achieved, for example HP DeskManager had been used for a number of years prior to this project and was moved very successfully to client/server.

This customer still has some way to go to meet the full client/server goal, but has accepted this is to be expected. The process so far has taken 2.5 years to implement, and they have achieved 1700 regular electronic mail users over 1000 of which are full client/server users.

7. Summary

The main migration issues facing an organisation moving to a client/server system are; financial, investment in software and hardware, write-off of existing equipment due to incompatibility or need for new more powerful PC's; productivity, investment in training, inertia of end users, productivity impact of rolling out client/server ; control, who to migrate, when, how to standardise in all departments, justification of costs, management of costs.

Moving to a client/server environment is not easy, it is not a fast process and it may not be justifiable for all organisations. But, the benefits are worth the indigestion along the way and the issues can be minimised with careful planning and mixing of the ingredients. The final dish can bring many benefits in terms of individual productivity, flexibility and usability, and organisation wide benefits with more effective use of people, more useful and timely information distribution and more effective decision making.

Biography

Steve Every spent the early part of his career as a development engineer in the telecommunications industry. He then spent 9 years with UK semiconductor and computer dealers, first in sales and then marketing. In 1989, Steve joined the Pinewood Information Systems Division of HP where he became the Terminal Emulation Product Manager.

Caroline Ellis graduated from Huddersfield Polytechnic in 1989 with a BA (Hons) in Business Studies. She then joined Hewlett-Packard's Pinewood Information Systems Division as the HP OfficeFax Product Manager. In 1990 she took the position of PC Mailing Product Manager. Caroline spent the third year of her business degree also working for HP in a marketing role.

Integration of the Telephony and Data Processing Industries

John Pickett

Hewlett-Packard
19420 Homestead Road
Cupertino, CA 95014
408/447-2906

ABSTRACT:
How can you reduce and control your telephone operating expenses while improving productivity? How do you stay one step ahead of the competition by providing better customer service? How can you provide existing customer with a better level of customer service? What industry are Industry Watchers projecting a 100% annual growth rate for the next 5 years? Answer: Applications integrating Computer and Telephone switches. This paper will explain how to achieve business goals through applications that take advantage of the integration between computers and telephone switches.

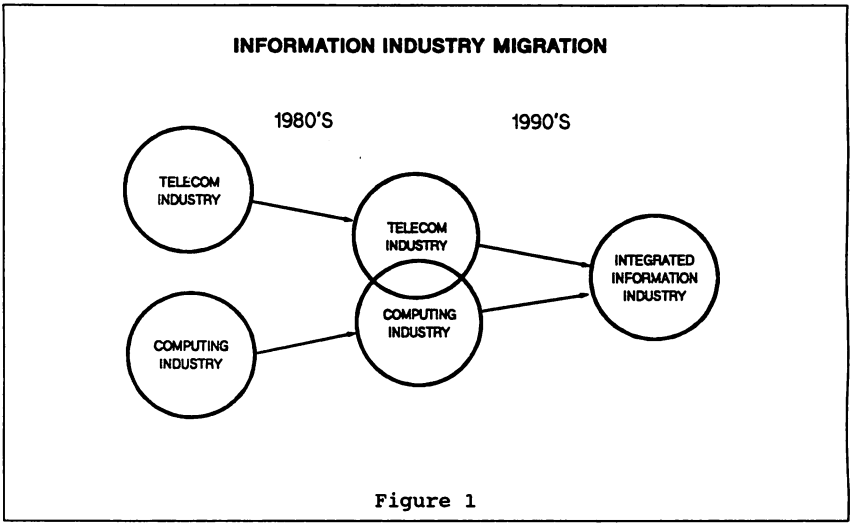


Figure 1

The last several years have seen significant changes in the voice and data processing industries, this is due to the fact that these two industries are merging as time goes on.

To illustrate this, let us take a look at a typical organization in the late 70s. You would find an individual responsible for the voice side of the house who would set up the phone service the company would use, and deliver that service to the employees desk.

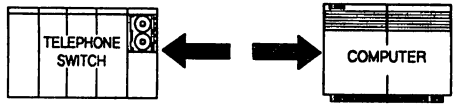
You would also find an equivalent on the data processing side who would handle the data communication needs for the company, connecting area offices to district sales offices, back to headquarters. These two individuals have traditionally reported up through different functional managers

In the early 80s we saw the advent of high speed phone lines, such as T1, that would carry the traffic for both the data processing and voice over the same telephone line. It was the technology that enabled us to do that, but it was the need to control costs and deliver voice and data effectively that accelerated the growth in that area.

Now we find ourselves in the early 90s where the integration of the voice and data processing technology is increasing at a rapid rate.

We will specifically discuss the integration of the voice and data processing industries through the use of Call Processing. Call Processing is intelligent communication between telephone switches and computers. This intelligent link is used to monitor and affect status message for phone calls coming in to the telephone switch. By having a message interface from a computer to a telephone switch, you can exchange command and status information. This enables you to use your computer to originate, answer and manipulate phone calls.

CALL PROCESSING



Vendor/Group	Technology Name
Hewlett-Packard	Applied Computerized Telephony (ACT)
Northern Telecom	Meridian Link / CompuCALL
ATT	Adjunct Switch Applications Interface (ASAI)
ECMA	Computer Supported Telephony Applications
ANSI	Switch Computer Applications Interface (SCAI)
IBM	CallPath
DEC	Computer Integrated Telephony (CIT)

Figure 2

There are different names for the computer to telephone switch links being used in the industry today. Some of them are:

1) Applied Computerized Telephony (ACT)
   Hewlett-Packard's product name

2) Meridian Link / CompuCALL - This refers to Northern Telecom's link coming from the telephone switch to the associated computer. Meridian Link is the PBX implementation and CompuCALL is the Centrex version.

3) Adjunct Switch Application Interface (ASAI)
   ATT's implementation of this technology.

4) Computer Supported Telephony Applications (CSTA)
   The ECMA (European Computer Manufacturing Association) name for this technology.

5) Switch Computer Applications Interface
   ANSI term for this technology

6) CallPath - IBM's implementation of this technology

7) Computer Integrated Telephony - Digital Equipments name for this integration

BEFORE APPLIED COMPUTERIZED TELEPHONY:

UNREALIZED CUSTOMER EXPECTATIONS

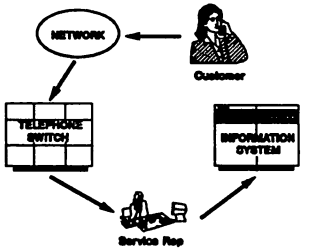


Figure 3

To fully understand, the advantage we receive from the integration of the computers and telephone switches, we must first look at how a standard phone call is handled in a typical environment. To illustrate, we will assume that a customer received an overdue notice in the mail regarding her account and wants to rectify the situation.

Agent: Thank you for calling XYZ, how can I help you?
Customer (concerned): My name is Sue Taylor and I received an overdue notice from you in the mail and I wanted to find out why. I mailed the check in over 2 weeks ago.
Agent: Mrs. Trailer, what is your account number?
Customer (slightly irritated): That's Ms. Taylor and it's 4678-4356-9985-23
Agent: I can't help you from here, let me transfer you to accounts deliverable department.

Customer (irritated): My name is Sue Taylor and I received an overdue notice from you in the mail and I wanted to find out why. I mailed the check in over 2 weeks ago.
Accounts Receivable: What is your account number?
Customer (cranky): 4678-4356-9985-23
Accounts Receivable: According to our records, that check was never received.
Customer (mad): Let me talk to a supervisor
Supervisor: Can I help you?
Customer (very angry): My name is Sue Taylor and I received an overdue notice, and mailed the check 2 weeks ago...

What started out as an innocent call to rectify a situation, turned in to an unpleasant situation for the caller. This phone caller could have been the company's largest customer and the company would not even have known it. I am sure that at some time in your life, you have placed a similar phone call.

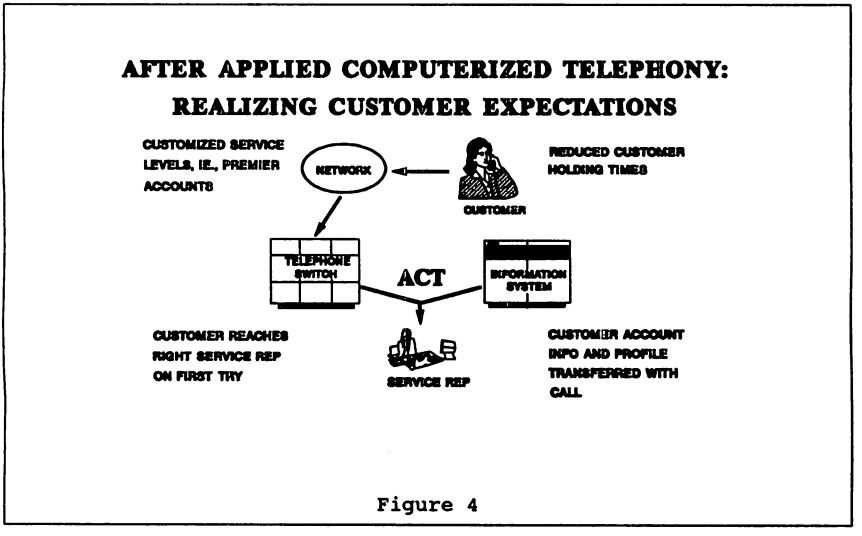


Figure 4

Now lets see what happens when an intelligent link exists between the telephone switch and the computer. We'll take the same example used above and describe how the phone would be handled differently had they had integrated their telephone and computer system with a product like ACT (Applied Computerized Telephony) from HP.

Agent: Thank you for calling XYZ, how can I help you?
Customer (concerned): My name is Sue Taylor and I received an

overdue notice from you in the mail and I wanted to find out why. I mailed the check in over 2 weeks ago.
<<The phone number of the person who is calling is used as a key into a database and the computer system pulls up customer information on the screen>>
Agent: Ms. Taylor, according our records, we received your check on June 17th and the notice you received was sent out the previous day. I apologized for any inconvenience. Is there anything else I can help you with?
Customer: Yes, I wanted to extend my credit limit?
Agent: I am not authorized to do that, but your account rep, Dave, can help. Hold on, I'll transfer you.
<<The agent transfers the call to the account rep easily via the computer application and types a message indicating the customer wishes to raise her credit limit. As the phone call arrives at the account reps desk, the message arrives along with customer profile information.>>
Account Rep: Hello, Ms. Taylor, I understand that you wanted to raise your credit limit. We can raise it to $5,000 if you'd like since your payment history is exemplary. By the way did you enjoy the golf shirt you ordered from us last month?
Customer: Great! $5,000 was what I wanted to raise it to and yes I did the shirt very much, thank you for your help.

Of these two scenarios, which one do you think would generate more sales in the future? Which scenario does your company currently use?

The group within your company that can benefit greatly with the integration of computers and telephone switches is a Call Center. A Call Center is categorized as an area where phone calls are distributed to a group of agents specifically trained to handle incoming and outgoing phone calls. Examples of this would be, a customer support group of an organization where agents(engineers) handle phone calls from their customers. Another example of a call center would be a group of agents that receive phone inquiries from advertisements in the newspaper.

One group of agents can handle phone calls for multiple products. For example, one agent can handle calls for a person wanting to buy a record album advertised on TV, or a miracle knife he saw advertised in the newspaper. The piece of information that indicates what product the agent represents is the phone number the caller dialed.

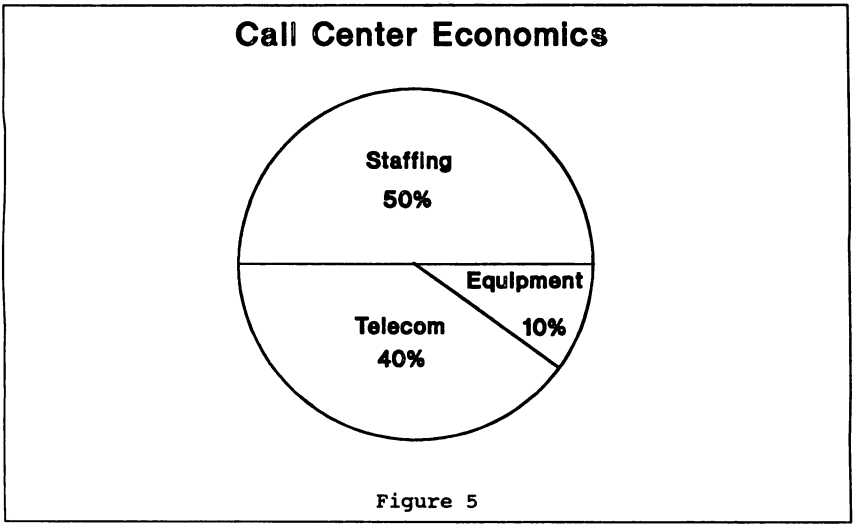


Figure 5

When we look at the economics of a typical call center, the three (3) pieces are:

1) Staffing (50%) - Includes salaries, benefits and overhead.
2) Telecom (40%) - Consists of telephone lines and associated distance sensitive charges.
3) Equipment (10%) - This is the telephone switch, computer and other ancillary equipment used to support the call center.

How much do you think a call center with 10 agents would cost for 1 year? $ 1 million! If you were a manager, how would you control a $1 million dollar call center? Since 50% of your cost is associated with staffing, you need to be sure that you are getting the most out of you people.

If you are looking at controlling your costs, the two major areas to focus on are Staffing and Telecommunication costs. Let's look at these two areas in a little more economic detail.

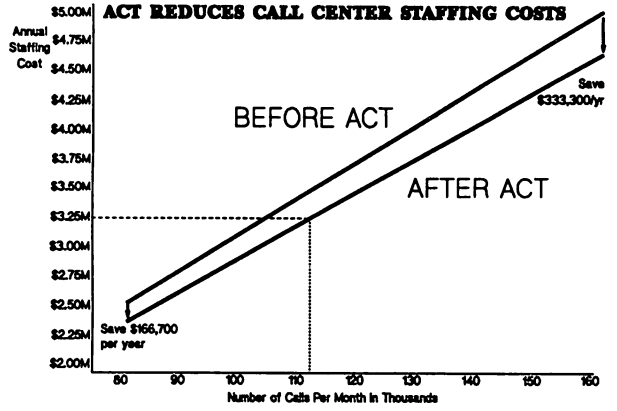


Figure 6

We have seen how the staffing costs are 50% of our overall call center budget, but how can the integration of telephone switches and computers help control, and in many cases, reduce your operating expenses?

Assuming that the duration of an average call is 6 minutes and a product such as ACT from Hewlett-Packard saves 30 seconds per call, there is significant savings. The first data point (savings of $166,700 per year) represents a call center of 50 agents. The second data point (savings of $333,300 per year) represents a call center of 100 agents. The savings are demonstrable, but we can illustrate how to reinvest those savings to generate additional revenue for the company.

To illustrate how to generate additional revenue, we assume that 65 agents, costing $3.25M annually, handle approximately 10% more phone calls using this technology.

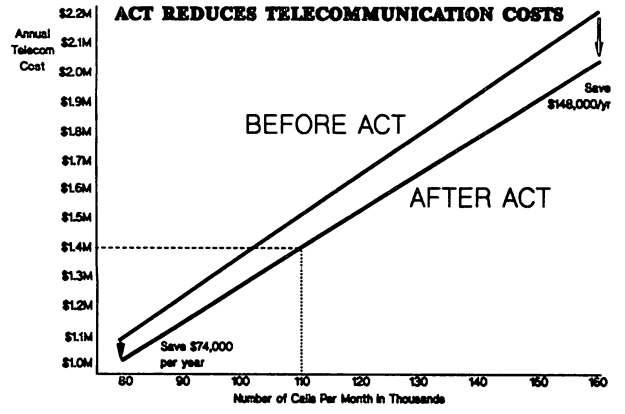


Figure 7

The Call Processing technology also allows you to control your telecom costs. Since the Call Processing technology saves you 30 seconds off an average phone call of 6 minutes, you can handle more phone calls with the same amount of resources.

Assuming an average telecom cost of $11.00/hour, integrating your computer with your telephone switch, you can save $74,000 annually handling 80,000 calls per month. If you are handling 160,000 phone calls per month there is a potential annually savings of $148,000.

You can also view the savings of 30 seconds per call as a way to handle more phone calls with the same resources. If you prefer, you can use the same amount of agents to mange approximately an additional 10% more phone calls.

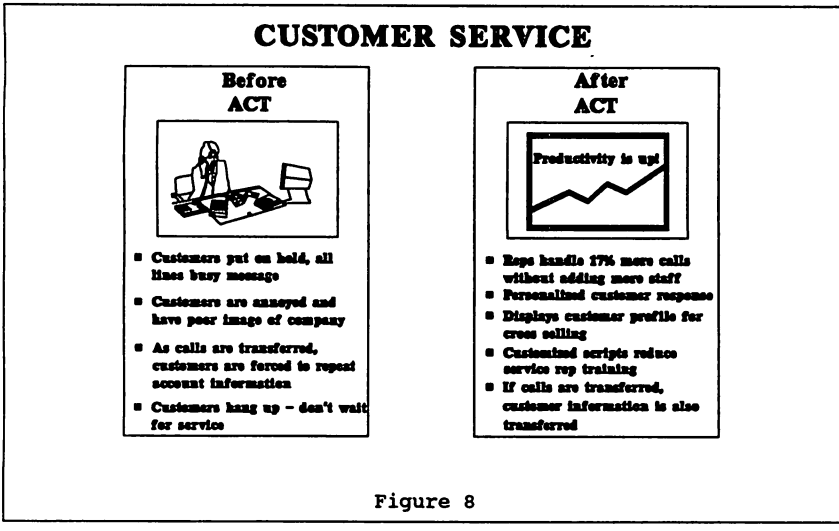


Figure 8

To illustrate how companies have used this technology to their benefit, we will look at two examples. The first is a customer service application. A customer service application is an example of end users calling for a variety of reasons. In the financial service industries, end users call to check their account balances, to query status on a loan application, to request a higher credit card limit, or to place a trade. In the distribution industry, end users are calling to order products from a catalog, or to query status of their order.

Before integrating computers with telephone switches customer were put on hold or given an 'all lines busy' message waiting to talk to an agent. Generally, having a customer wait to be helped does not promote good customer service. If the customer did not reach the correct person on the first try, then they would be transferred to the next agent where they had to explain their question over to the new person helping them. Many times, you would find that customer would hang up as opposed to waiting for service.

After integration, because the length of the phone call is decreased by approximately 30 seconds, the agent can now handle 17% more phone calls. This means that when a customer calls, there will be more available agents to assist them. Also, when we match the callers phone number to a record in our database, we can pull up important profile information that can assist us

providing better customer service. If for any reason the caller needs to be transferred, we can transfer screen information to the next agent they talk to so that the caller does not have to repeat why they are calling and what their account number is.

Identifying premier customers immediately is another benefit for using this technology. Based on the phone number of the person who is calling and the phone number the caller dialed, you can route the caller to an agent specifically trained to understand the callers business environment.

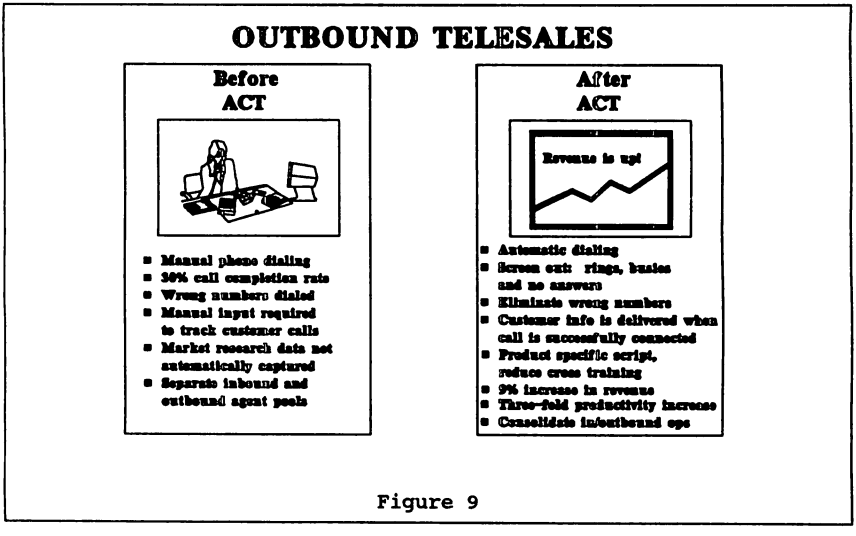


Figure 9

The second example focuses on using the Call Processing technology to generate outbound calls for their agents. The outbound telesales application is where agents place phone calls to their end users. A good example of this would be a stock broker calling his clients. Before integration of computers with telephone switches, agents would view their calling list for the day and physically dial the number of their clients. After listening to the ringing, busy signals and voice mail, they would finally talk to a client. 3 out of 10 times the stock broker would talk to their client. Much time was wasted even to get to that point and, especially for the stock brokers, time is money.

After integration, the application would automatically generate the calls for the stock brokers clients and after filtering out the busy signals and ring no answers, then connect the stock broker to their client. Simultaneously, client profile information was delivered to the brokers workstation screen to assist them in servicing their client. This method also eliminated wrong numbers dialed by the broker

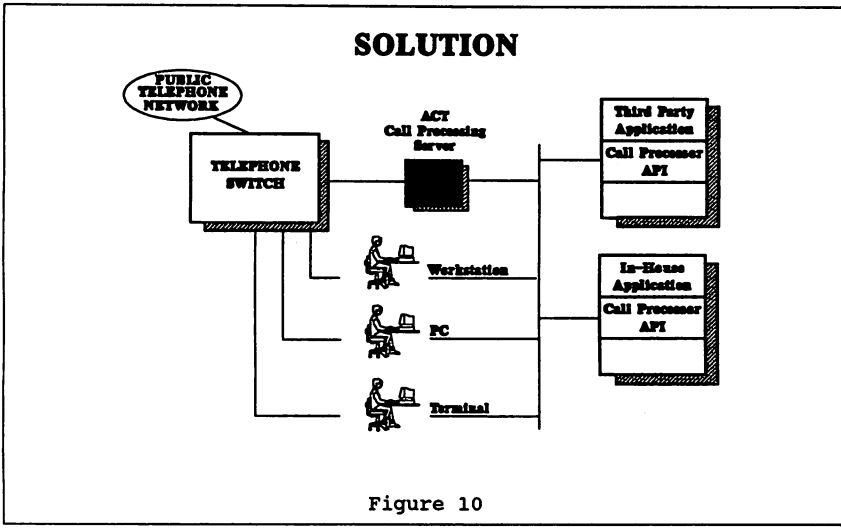


Figure 10

We have given several examples of how the technology is used. Now let's look at how this service is implemented. With Hewlett-Packard, the solution is called Applied Computerized Telephony (ACT). The solution consists of "5 Easy Pieces":

1. Public telephone network - This typically will be one of the major inter-exchange carriers (ATT, MCI, US Sprint) that deliver 800/900 service. They deliver the calling number and called number services which in turn may be used by the application.
2. Telephone Switch - The telephone switch can be a PBX, which is a customer premise telephone switch typically used in larger companies to distribute phone calls to their employees. Centrex which is a service offered by the Regional Bell Operating Companies (RBOCs) can also be used.

3. ACT Call Processing Server - The ACT Server is a UNIX processor that translates the messages from the telephone switch into a language easier understood by the application. The Server isolates the application from the telephone switch and the benefit is that an application developed on one type of telephone switch can be used with another type of telephone switch with only a modification to a driver on the ACT server.
4. Application Programming Interfaces (API) - The APIs are the subroutines used by the application that enable the application to talk to the telephone switch. Creating or modifying an application with the APIs is a straight forward procedure. The API functions enable you to Make a call, Answer a call, Transfer a call, Hold a call, etc. Anything you are able to do manually with a telephone set, you can do using the ACT APIs.
5. Application- The applications used in the integration of these two technologies fall into two different categories:
    A. Third Party Application - Value Added Businesses have modified their applications to integrate the APIs and provide an off-the-Shelf software package for their customers.
    B. In-House Application - Companies can create or modify application that they currently use to integrate the API to take advantage of this technology. To assist customers with this, consulting products and training courses are available within Hewlett-Packard. We have seen applications integrate the APIs in as little as half a day.

To find out if your company can take advantage of this technology that has proven itself many times over with increased productivity and enhanced customer service, take the following quiz. This technology is not for everybody, but if you answer YES to 6 or more of the following question, then you should very seriously look at implementing this technology:

Questions to Qualify

1. Do you use, or plan to use, 800/900 phone service?

2. Do you use, or plan to use, Automatic Call Distribution (ACD) software with a PBX or Centrex service?

3. Do people who answer phones use computers to retrieve customer information?

4. Do you have, or have plans for a Call Center type activity? (ie. Customer support, collections, telemarketing, etc)

5. Do you service customers over the phone?

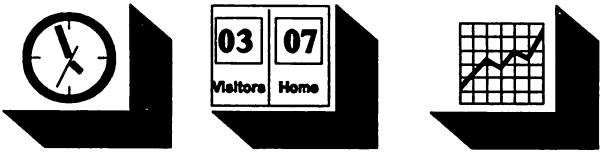
6. Would you like special handling of large important customers?

7. Do customers get transferred often when calling?

8. Do you have groups of people, larger than 15, answering phones?

9. Does your company receive phone calls for multiple products or service?

10. Does your company generate phone calls to specific groups of people (ie installed base, "active" account lists, etc.)



CAN YOU AFFORD NOT TO

ACT?

Figure 11

TITLE: Tutorial - Open Systems Networking In A Multi-Vendor Environment

AUTHOR: Steve Oppenheim
Hewlett-Packard Co.
C/O Ella Washington
19091 Pruneridge Ave.
M/S 46LK
Cupertino, CA 95014

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2079

PAPER #2080

GROUPWARE: SUPPORT FOR BUSINESS TEAMS IN PRODUCTIVITY SOFTWARE

Robert G. Brosseau
Manager of Training, Applix, Inc.
112 Turnpike Road
Westboro, MA 01581
508/870-0300

The power of the UNIX Workstation and the network can be used to achieve measurable gains in productivity by identifying and using effectively those features that can be included in the Groupware definition. For example, an effective set of communication tools, including -- but not limited to -- electronic mail and fax services, will allow users to share data, messages, documents and information in such a way that they are enabled as a business team, not just as individuals.

This paper examines some of the tools found in productivity software and identifies Groupware features that can support business teams. The information in this paper can help you to evaluate the software you use against the business team support criteria, or help identify those features that should be part of an evaluation and selection process for productivity software.

In his book on Groupware Robert Johansen defines Groupware as a generic term for specialized computer aids that are designed for the use of collaborative work groups. He goes on to state that Groupware can involve software, hardware, services and/or group process support.[1] It is not my intent to determine which of these factors is most important in the Groupware model. Certainly the social and interpersonal factors that impact on the use and effectiveness of Groupware could be the subject of another paper. I am limiting myself to an examination of Groupware features that can be identified in productivity software and provide support for business teams.

What is a business team? The answer depends on your organization and the type and duration of projects and tasks that you and others perform. Some companies form business teams that stay together for the long term while others rarely keep the same

groups together for more than a few days. Business teams can be formally defined in the organization or may come together informally. The individuals who make up a business team may be from the same or different departments, divisions or, in some cases, companies. In all scenarios, though, they are individuals who possess information or a talent that if shared effectively will cause increases in productivity and produce a better finished product as a result of a coordinated group effort. It also helps to keep in mind that the member of a group, being from different organizations, probably use different hardware platforms and software solutions.

For example, a group formed to bring a new software product to market is most likely made up of marketers, programmers and developers, designers, managers, technical writers, support and sales persons. Each of these individuals will need access to and in some cases will generate new information and/or documents that will benefit others. How often have you been part of a team and heard statements such as, "I thought we did this already", "Didn't we already discuss that", or "I know we did this but I gave my notes to...". It all too often follows that costly duplication of effort occurs after those statements are heard.

It is certainly the exception rather than the rule that software is employed for obtaining, storing, manipulating, editing and presenting information. It is often the case that the information has been stored in a variety of formats and on different machines throughout the network. There are now many good affordable software solutions that will work in or with largely dissimilar environments. The emergence of X technology has provided the base for much of this work, and standards such as NFS continue to play a key role.

The software that your organization employs should allow individuals that are part of business teams to use groupware-like features to empower themselves with timely information, and with ways to share, reacquire, update, edit and present that information in a variety of presentation styles and formats.

The remainder of this paper will identify some features found in productivity software that will help accomplish these objectives.

Open Software

The word "open" has different definitions and is used to define both hardware and software products. As I previously stated, given the diverse makeup of the business team, productivity software must be open to receive information created and stored in other software packages and send the information back in that person's preferred format. Though standards like ASCII and DCA can be used as a last resort, a new generation of productivity software employs filters that allow for greater transportability of information and documents. For instance, many spreadsheets allow direct transfer of data from Lotus document formats into their proprietary spreadsheets, and products in

the productivity solutions category will allow for the import and export of documents directly to tech-pubs writers using Frame or Interleaf on their desktop. These filters allow attributes and other format instructions to be part of the import/export process, saving time in the transfer process and making the information available sooner in final form.

Live Compound Documents

"Compound Documents are much more than an enhanced delivery and presentation system for users in electronic or desktop publishing. In the context of a distributed network computing architecture, compound documents become the mechanism for information access, integration, analysis, and dissemination across the network."[2]

The traditional view of a document has been as a snapshot of the information it contains at a point in time. A document with dated information is not the basis on which decisions should be made. The demand for up-to-date, even up-to-the-minute information is increasingly common in the organization of the 1990's.

Live compound documents, that is, documents that display information from other applications and sources and allow that information to be updated and incorporated immediately into the document, are obviously the better choice for business teams. Productivity software should allow multiple documents produced by team members to include data from the same document.

Some software packages lay claim to compound document features but lack the ability to maintain a link to the source information. Others allow only the linking of information that is stored in their proprietary format. There is obviously greater benefit in being able to include information in a document that is in a foreign format. For example, a product manager who is preparing a report on the last quarter's results and next quarter's expectations might well want to include revenue and sales information that accounting has stored in a Lotus spreadsheet. While the information could be exported and then imported to the report, that process takes time and does not allow for the numbers to be revised without repeating the export/import process. Being able to produce a live compound document, even to a foreign data type, will simplify the task for that marketing manager. The members of a business team could also use the compound documents -- and the better information it contains -- to make better decisions and to perform their work.

Electronic Mail

E-Mail provides the ability to quickly move information and documents to other members of the team or to those from whom the team needs information. It is relatively fast, generally reliable and the presence of national and international networks makes E-Mail widely available. E-Mail does have some drawbacks. There are format issues that are gradually being addressed with the uses of filters and gradual compliance with X.400 standards that address the movement of messages across systems. In some environments E-Mail may not be considered secure or provide the proper confidentiality. Business team members should have access to E-Mail, though in some cases it may not be part of the productivity software, but a separate and distinct product. Open Windows and X provide for some E-Mail capability as part of the basic product. An E-Mail product should be able to accommodate the movement of compound documents.

Fax Services Support

The fax provides the fastest way of moving a document from one place to another. It uses the largest communications network in the world, is inexpensive and does not require that information be received or sent by personal computer or workstation. The drawback is that the information is not readily available as data, only as paper. There are some products now that address that issue. Many productivity solutions provide users with direct sending and receiving of faxes from their workstations through fax-modems. The fax used this way is a powerful tool combining the best of E-Mail and fax services support for the work group.

Continuation Documents. Large Document Support

When a document is produced as a result of a group effort, it should reflect the contributions of all members of the group. However, the document should present a consistent appearance with common format and attribute features such as page numbering, headers and footers, section numbering, table numbering and content lists such a table of contents. Individual documents produced by members of the team should be able to be merged in such a manner by the productivity software.

Revision Control

Documents shared and revised by members of the team may change in content at any time during the project. The ability to compare different versions of the document and identify changes made since your last review will saves hours of time reading documents for the second and third time and avoid locating other members of the group for questions about the changes last made.

File Sharing

How are documents and information shared by group members? Unix productivity solutions built on top of the Unix file system offer the utility of the Unix directory structure and other capabilities to facilitate the sharing of files. Documents needed by the group as sources for data or produced by the group for review can be placed in common, logically named directories for easy access. Permissions can easily be used to restrict access to documents and data.

Extension Languages

Some productivity software offers the use of a customizing language that at a minimum provides macro functionality and when more fully employed allows tasks, such as the updating of source information, to be automated. One such product provides both shell and TCP-IP socket communication. TCP-IP communication is especially useful since it provides the group a way of receiving information as a live data feed, instead of having to go and get the information needed at periodic intervals.

Other Features

Recent enhancements to productivity software include voice annotation and the use of video images in documents. Voice annotation provides members of groups with the ability to include and forward recorded comments and other audio information.

Business team users will continue to benefit from the pace of innovation in productivity software tools.

# # #

[1] Robert Johansen, Groupware: Computer Support for Business Teams, New York, Macmillan, 1988, pl

[2] "Compound Documents in the Distributed Networked Environment," A White Paper, Applix, Inc., 1990

TITLE: Networking LaserROM for Multiple Users

AUTHOR: Bill Hassell c/o Ella Washington
Hewlett-Packard
19091 Pruneridge Avenue
Cupertino, CA 95014

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2082

TITLE: The Real Story About HP PowerPatch!

AUTHOR: To Be Announced

FINAL PAPER WAS NOT AVAILABLE AT TIME OF PRINTING.

PAPER NO. 2083

Migrating to Client/Server
(Paper # 8021

George Ferguson

Hewlett-Packard Co.
General Systems Division
19490 Homestead Road
Mail Stop 41AF
Cupertino, CA 95014
(408) 447-7481

Introduction and Scope of Paper

Client/server is a computing paradigm that can be viewed from many perspectives. One of the perspectives that has attracted a great deal of attention is the use of client/server for *knowledge worker information systems*. This paper focuses on how companies can save costs and achieve competitive advantage through migrating knowledge worker information systems to client/server. In so doing, the paper will begin by examining the purpose of knowledge worker info systems and the benefits of client/server. After examining the information systems currently in use and the users of those systems, the paper will then examine in detail the client/server application areas that are useful to knowledge workers. The paper will finally examine how these applications and supporting hardware should be rolled out within an organization.

Knowledge workers can be defined as people who analyze data and make recommendations and/or decisions. Knowledge workers include managers, supervisors, production planners, quality engineers, medical doctors, financial analysts, marketing analysts, strategic planners, human resource specialists, middle/high level managers, and many other types of employees.

One reason that information systems for knowledge workers are important is that they allow process-oriented work to be done at a lower cost to the company. This savings may result (1) from the automation of the knowledge worker's job or (2) from the knowledge worker interacting with an automated process that was previously handled by clerical or administrative personnel. Client/server examples, illustrating these two types of costs savings, may be helpful.

In the former case, a financial analyst might currently have to take a paper financial report printed off the company's corporate mainframe and re-enter the data manually in a PC Lotus spreadsheet. With client/server, the same financial analyst could automatically download the relevant data from the corporate mainframe into the PC spreadsheet.

In the latter case, a production planner might today draw on paper a graph showing the monthly increase in widgets manufactured and then send it to someone in the corporate office by addressing it and dropping it in the mail

pick-up basket for delivery by the mail room. With client/server, the production planner might create the graph on her PC from data on a local database and then send it via e-mail to the corporate office. In this case, the savings for the company would be in the downsizing of the mail room staff.

A second reason that knowledge worker information systems are important is that they help knowledge workers make better and more timely management recommendations or decisions. The information systems primarily do this by making it easier for the workers to get access to the *right* information. For example, an HMO hospital administrator might use an ad hoc database query tool to compare the average hospital stay of pneumonia patients receiving two different kinds of treatment. On the basis of the response, the HMO could make recommendations to its staff doctors so as to minimize medical costs.

What is Client/Server?

Client/server has been defined in many ways, but can generally be described as a variety of methods to distribute the user interface, the application, and the data. See exhibit 1 below:

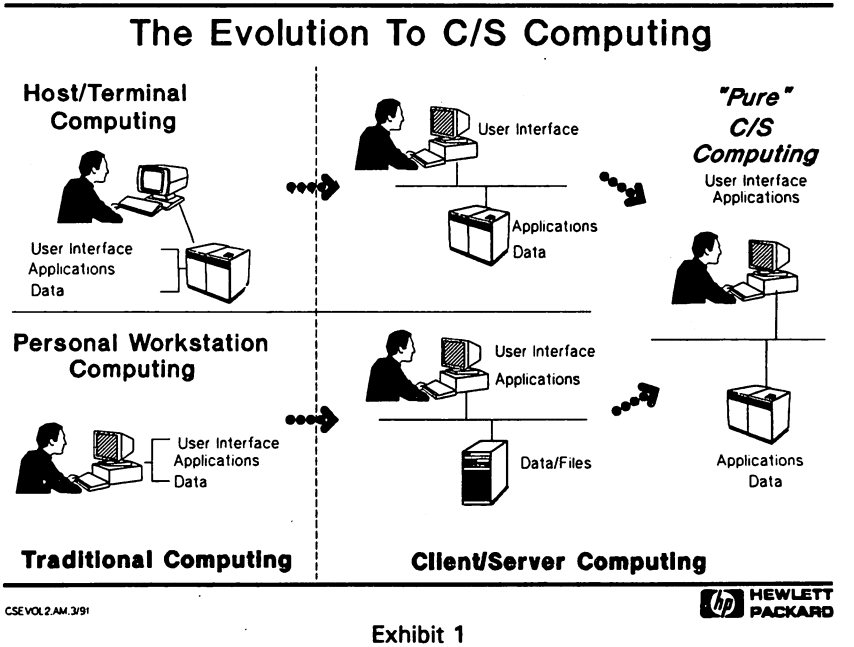


Exhibit 1

By the nature of their jobs, knowledge workers tend to use multiple computer applications and often need access to data residing on multiple computer systems. Both of these job aspects strongly point to the need for a client/server solution.

The use of multiple applications means that the workers need to go through a learning curve for each application they use. In order to minimize this expensive, non-productive training time, it makes strong sense to use a graphical user interface (such as MS-Windows or Motif) for all applications. Graphical User Interfaces (GUIs) minimize training time because *all* applications developed for them have a similar *look and feel* and because the overall interface is carefully crafted to be intuitive. GUIs are also important because they make even experienced users much more productive.

A recent study by an independent analyst (albeit financed by Microsoft) found that experienced GUI users were 57 percent more productive than experienced character mode users (with productivity being measured by the number of tasks completed and the percentage of errors made). Some users indicate that this percentage is exaggerated and that the gain is only 15 to 30 percent. In any case, however, basically all independent observers agree that the productivity gain is very significant. The same study also found that users of GUIs had a higher level of satisfaction with their computer than did character mode users.

Knowledge workers almost always need access to shared databases, files, and printers and in most cases need access to data residing on multiple computer systems (or servers). The only way that they can get access to such shared resources *and* to the benefits of GUIs is through client/server. Without client/server, you can only have GUIs on totally stand-alone desktop systems. (It is true that you can get GUIs and shared resources with X-terminals, but these are also a form of client/server, as shown in Exhibit 1.)

Evolution of Knowledge Worker Information Systems

Information systems for knowledge workers have evolved in the past from two alternate roots: the classic host-terminal configuration and the stand-alone PC. (See exhibit 2 below.) In the middle to late 1980s, these topologies were in many cases replaced with the host-PC (terminal emulation) and the PC LAN server topologies.

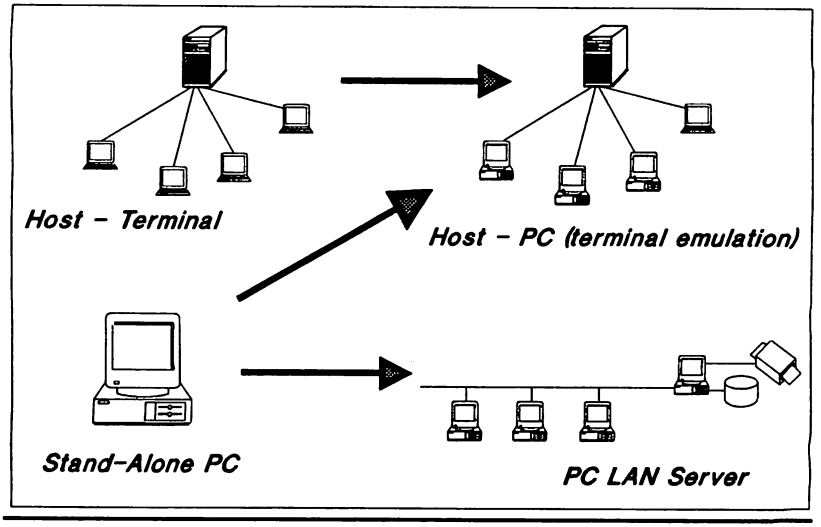
The host-PC topology allows the PC to be used with its PC applications *or* to be used in terminal emulation as a terminal to the host. It also normally provides file transfer capabilities between the PC and the host. Unfortunately, when the user enters terminal emulation, he effectively lobotomizes his $2000 PC into a $300 terminal. The PC and the terminal emulation environments are wrenchingly different and the file transfer between the two environments is unwieldy at best.

The PC LAN server, on the other hand, does a much better job of providing shared files and printers that look to the PC user like an extension of his PC environment. (Note: The term *PC LAN server* does not refer to simply any

computer system which provides file/print sharing services over a LAN to PCs. The term instead refers to a file/print server which is itself *an Intel-based PC.*) In spite of its strengths, the PC LAN server typically has poor communications with other computer systems, thus resulting in an island of automation. The users typically have little access to plant, division, or corporate databases and have no e-mail connections to users outside of their departmental workgroup. (Where users do have these capabilities, it is often through a mixed topology that allows a PC to connect *either* to a PC LAN server *or* through terminal emulation to a host--a very unwieldy environment.) Furthermore, since PC LAN servers are usually outside of MIS control, they often have poor backup and security procedures in place.

Both the problems of the PC LAN server topology and the host-PC (terminal emulation) topology can be resolved through client/server. In order to best explain the migration to client/server, however, it is helpful first to better understand the users who will be taking advantage of the system.

Evolution of Knowledge Worker Info Systems (mid-1970s to 1990)



HEWLETT PACKARD

Exhibit 2

Who are your Users?

Knowledge workers can be classified into three groups based on their information system needs.

The first group, which can referred to as *OLTP professionals,* is made up of

knowledge workers who have significant decision support responsibilities, but who also have at least occasional OLTP update responsibilities. For example, in financial services, many financial transactions entered by entry-level employees need to be approved by mid-level managers. Similarly, a purchasing clerk in a retail warehouse will occasionally need the help of a supervisor in handling a difficult transaction. OLTP professionals include managers, supervisors, medical doctors, area managers, production planners, etc.

The second group, which can be referred to as *quantitative professionals*, is made up of the non-technical personal-workstation power users. These users have a major need to download data from plant, division, or corporate databases to their PCs or workstations. On their PC or workstation, they will manipulate this data (usually in complex spreadsheets) in order to develop forecasts, budgets, cost justifications, etc. Quantitative professionals include financial analysts, accountants, controllers, and consumer product managers.

The third group, which can be referred to as *other professionals*, is made up of all other knowledge workers not included in the first two groups. The scope of these people's jobs and their accompanying information needs is often quite wide. These people need access to information (database, textual, and image) that is often scattered widely across many computer systems or that may not be readily available in computerized format (e.g. trade journals and general business periodicals). (Note that these same needs may also be present in the previous two groups.) The group of *other professionals* includes middle and high level management, strategic planners, marketing personnel, industrial engineers, quality engineers, etc.

Application Areas

Client/server can help meet knowledge worker needs in several application areas. Some of these areas apply to all knowledge workers, while others apply to only one of the groups mentioned above. Each of these application areas will be explored in detail, after which a technology rollout--including both software and hardware--will be recommended.

The four primary application areas are (1) file/print sharing, (2) e-mail, (3) review, approval, and aid, and (4) quantitative analysis, budgeting, and forecasting. These areas largely involve the automation of process-oriented work and are not overly difficult to cost justify. Three additional emerging application areas are (1) ad hoc database query, (2) text and document management, and (3) data analysis and executive information systems. These applications directly provide information to knowledge workers that should help them in making better and more timely decisions. The applications should potentially provide a large return on investment and significant strategic advantage to the companies that deploy them effectively. By their nature, however, it is difficult to cost justify these applications because the returns are uncertain and are based on many intangible factors, of which the technology use is only one. In spite of this difficulty, these application areas hold great potential for the companies willing to make the investment in them.

In examining the application areas and presenting the recommended technology rollout, the following material spotlights the use of DOS PCs as the client of choice. Although some of the technical material is specific to DOS, all of the basic points in the material also apply to UNIX workstations, Macintoshes, and OS/2 PCs. DOS is emphasized only because it is so predominantly used today by knowledge workers.

File/Print Sharing

File/print sharing has been widely deployed by many companies and is beneficial to all three groups of knowledge workers. The term file/print sharing refers to the capability of PCs or workstations connected to a LAN to store and access files located on a server's disk(s) and to direct output to a printer connected to the server. The server-based files and printer(s) appear to the user as if they were on the user's local PC. The server is generally either a high-end PC or a RISC-based mini. The most popular file/print sharing offerings on the market are Novell Netware, LAN Manager (for OS/2 servers), and LAN Manager/X (for UNIX servers).

File/print sharing has many great strengths, including the following:

- Files can be easily shared between users without requiring users to physically carry diskettes between PCs.
- Expensive printers (especially laser printers) can be made available to a group of people without putting one on everyone's desktop.
- Printers can be conveniently located in the office environment. This means that printouts are available almost immediately and that there is no need for printout delivery people.
- PC software application can be loaded onto the file server, thus making them available to all users, while avoiding the cost of purchasing the application for each user. (Note that this savings does usually come, however, at a performance penalty.)

For all of these reasons, it is reasonably easy to cost justify file/print sharing.

File/print sharing can be implemented in two different ways so as to avoid the problems previously mentioned with PC LAN servers. The first approach, which is particularly good for departments with PC LAN servers already installed, is to give the PCs access over the LAN both to the PC LAN server and to one or more other servers with e-mail and database capabilities. If the PC LAN server is running Novell Netware, this approach may require installing two "networking stacks" on the PCs, which can be done with products such as HP's *ARPA Services 2.1 for Netware*. This way the PC can talk Novell's proprietary networking to the PC LAN (file/print) server and talk standardized TCP/IP networking to the e-mail and database servers.

The second way to avoid the problems of PC LAN servers is to centralize file sharing on a RISC-based server controlled by MIS. The RISC-based server can be located in the office environment with printers directly hung off of it. In this case, MIS can simply administer the system remotely. In most cases, however, MIS will probably want to keep the server in a centralized, secure location. Printers then can be hung directly off a PC

LAN server in the office environment or new network printers (such as the HP LaserJet IIISi) can be directly connected to the LAN. The advantages of this alternative over the PC LAN server are that it provides good MIS administration, backup, security, etc. Furthermore, since both LAN Manager/X and Portable Netware run on RISC-based servers, any users migrating from PC LAN servers running LAN Manager or Netware will see no difference. In fact, they won't even have to have their PC software modified or updated.

Electronic Mail

Electronic mail (or e-mail) simply refers to the sending of messages between computer users. Although primitive e-mail systems allow messages to be sent only from one person to another person; most mail systems allow messages to be sent from one or more people to a full distribution list of people. E-mail does not require client-server; however for users with PCs or workstations on their desktops (i.e. most all knowledge workers), e-mail will be unwieldy and limited unless it supports client/server. The reasons for this are explained below.

E-mail, whether client/server or not, is important to companies because it saves considerable amounts of money both in external postage and in internal mail room personnel and delivery people. At the same time (and sometimes more importantly) it considerably speeds flow of information in organizations. The savings in postage and mail room costs can be directly quantified in justifying e-mail installation. The timeliness of information is more difficult to put an exact dollar value on, but in certain cases can be quantified. In any case, the value of quicker time-to-market and of easier communication between people working at different schedules and locations is obviously very significant.

Client/server e-mail allows PC users to send e-mail directly from their PC (and usually GUI) environment (instead of from a terminal-emulation environment). A PC user using terminal emulation will compose his message in one of two ways. The user may compose the message with the host or e-mail system's word processor (which is often very primitive and difficult to use). Alternately, the user may compose the message on the PC, transfer the file containing the message to the host, and then send it using terminal emulation from the host e-mail system. This is cumbersome, unwieldy, and time consuming. In either case, the host e-mail system's capabilities for handling anything besides bare ASCII text (without fonts, italics, underline, etc.) is usually very limited or non-existent. On the other hand, an ideal client/server e-mail package (such as NewWave Mail used with OpenMail), will allow full-fledged word processing documents, spreadsheets, graphics, images, or even videos to be sent from the PC environment. The CEO of Intel recently stated that such a client/server e-mail package (with support for documents, spreadsheets, graphics, and images) would be "the killer application" that would make client/server successful in the market.

The cost justification for client/server e-mail can be based (1) on the value of training time saved as the result of a GUI application, (2) on the value of time saved by knowledge workers who used to have to do file transfers, environment changes, and host logons every time they wanted to mail a

message, and (3) on the savings from items besides simple textual messages which would otherwise have been sent through internal mail rooms, external mail services (U.S. Postal Service and Federal Express), and fax machines.

Review/Approval/Aid

This is the area covered exclusively by the group previously referred to as OLTP professionals. The major distinguishing feature of this application area from standard OLTP is the strong need for its users to be working with multiple applications and to be able to switch on an interruption basis to and from OLTP with minimum disruption. This necessitates a GUI implementation with multiple window capabilities.

As given in the previous example for OLTP professionals, a warehouse supervisor might, for example, need to help a purchasing clerk with a difficult transaction. The supervisor does this occasionally on an interrupt basis in addition to his other work. He needs to be able to switch to an OLTP screen and back to his other work as easily as possible so as to keep his train of thought and avoid dead time. This is a good example of the *aid* function. On the other hand, a good example of the *review/approval* function is the previous example given of the financial services manager approving transactions entered by an entry-level employee.

The review/approval/aid application area is usually mission critical, in that it is part of the fundamental process whereby products or services are delivered by companies to their customers. The client/server implementation is needed to maximize productivity (through avoiding task switching costs) and to minimize training costs.

Client/server versions of OLTP forms can be retrofitted onto old applications through use of a client/server OLTP forms package or a programmable user interface toolkit. Client/server OLTP forms packages, such as Oracle SQL*Forms on the HP 9000 or VPlus/Windows on the HP 3000, allow old terminal-based forms to be moved under a GUI on a PC or workstation. Programmable user interface toolkits, such as Easel, allow a programmer to develop a completely new GUI interface that will interact with an old terminal-based application. In neither case does the old application have to be changed or modified.

For totally new applications, tools such as Ingres' 4GL/Windows, PowerSoft's PowerBuilder, or Unify's ACCELL can be used for developing client/server OLTP applications with a GUI.

Cost justification for the review/approval/aid application area can be based on estimates of the time lost under the current modus operandi (which requires knowledge workers to switch in and out of terminal emulation in order to handle their approval and aid responsibilities). Cost justification for new workers can also be based on estimates of the lower training costs resulting from the consistent graphical interface.

Quantitative Analysis, Budgeting, and Forecasting

This is the area covered exclusively by the group previously referred to as

quantitative professionals. Quantitative professionals need database tools that will allow them to download relevant data from plant, division, and corporate databases into their personal workstation spreadsheets and statistical analysis packages. The database queries/downloads are generally repeated on a periodic basis (for example, monthly or quarterly).

The accounting function that dominates this area is the fundamental way of measuring business success--both for external financial reporting and for internal decision making. The client/server implementation results in a major productivity gain for its users. Until very recently, the only way for this information to be manipulated and analyzed (by the users who needed it) was for the users to take a printed report from MIS and then re-enter the relevant data themselves into their spreadsheets or other applications. This resulted in tedious duplication, uneconomical use of valuable knowledge worker time, and user entry errors. For these reasons, it is very easy to cost justify the automation of this process.

A client/server implementation of quantitative analysis, forecasting, and budgeting might use an easy-to-use graphical PC tool such as HP's NewWave Access to query the necessary server databases (on an HP 9000, HP 3000, or IBM mainframe) and pass the data into a spreadsheet or other application. Alternatively, spreadsheets and statistical packages are beginning to have their own database query tools included with them (such as Lotus 123 Datalens and Microsoft Excel Q+E). These tools will also work with popular databases on the HP 9000 and HP 3000, although they tend to be less sophisticated and less easy to use than NewWave Access.

Ad Hoc Database Query

This application area, which is applicable to nearly all knowledge workers, is a much more general area than the previous one. The two areas, however, do have a great deal in common. While the database queries in quantitative analysis, budgeting, and forecasting are generally periodic, ad hoc database queries are totally out of the blue. For example, a marketing manager might ask "What percentage of product line Z sales in 1990 came from the European Community as opposed to the U.S.?", or a human resource specialist might ask "How has the percentage of women managers changed in our company from 1986 to 1991?" Ad hoc database queries allow knowledge workers to look for product/service problems and opportunities.

Since the users making the queries are unlikely to have a technical background (besides general PC literacy), the ad hoc query tools need to be very easy to use. NewWave Access or similar client/server tools are very appropriate for these users.

Although ad hoc query tools can potentially have a terrific payback for companies, it is very difficult to predict the financial return that will result from the investment. For example, queries such as the one above by the human resource specialist may help the company evaluate the success of its affirmative action programs and avoid an expensive lawsuit. Whether a lawsuit would actually occur, however, and how expensive it would be is unknown. Furthermore, strategic decisions are usually based on many points of data as well as on intuitive feelings. This makes it all the more difficult to trace the return to any particular investment. In spite of this,

companies constantly make outlays for strategic decision making--including the salaries of its high-level managers. Ensuring a better factual basis for those decisions through an information systems investment is a very reasonable business risk and represents a major opportunity for companies to achieve competitive advantage.

Text and Document Management

Text and document management systems are applications which allow users to locate textual documents, spreadsheets, graphs, images, or other file data on the basis of external or internal attributes. For example, a customer support engineer in a manufacturing company might request all documents relating to particular problem described by a user. Likewise, a financial analyst looking for a particular spreadsheet authored by a vacationing co-worker might request a list of all Lotus spreadsheets authored by the co-worker between March and June 1991 with "Expenses" in the title. Text and document management systems can be useful to almost all knowledge workers.

Text and document management systems have two levels of benefits. The first level is process-oriented. The systems measurably raise the productivity of knowledge workers through eliminating needless search time (for files, documents, etc.) This benefit is measurable and can be used for cost justification (although an estimate may need to be made of the time currently being spent doing searching). The second level benefit is strategic, resulting from the ability to look for product/service problems and opportunities that are not reasonable to spend time looking for, given current information systems.

Several of the market's top text management systems are currently offered on the HP 9000, including Verity's Topic and IDI's BASISplus.

Data Analysis Tools and Executive Information Systems

Data Analysis tools and Executive Information Systems (EIS) are a set of tools for analyzing and presenting database information that has been downloaded to a PC or workstation. The tools may or may not include the utility for actually extracting and downloading the database data. If this tool is not included, a separate ad hoc database query tool may be used. Traditionally data analysis and decision support tools have had more flexibility and have been targeted at users with better technical skills, while EIS has been more pre-canned and has been targeted at senior management with minimal technical skills. This distinction is melting, however. The tools have broad applicability to knowledge workers, but probably apply the most to the *other professionals* group.

The tools are primarily intended to help users identify product/service problems and opportunities. Consequently, they have all of the potential and difficulties that the other emerging application areas carry with them. Direct cost justification is difficult to *demonstrate*, but the strategic potential of these tools is great.

Integrated, high-end executive information systems on HP's platforms include the market-leading Pilot and Execucom offerings. Cognos'

PowerPlay and Channel Computing's Forest and Trees, on the other hand, are good examples of stand-alone data analysis/EIS tools that integrate well with other tools via HP's NewWave environment.

Technology Rollout

Having gone through the major application areas of knowledge worker information systems, it is important now to demonstrate the best manner in which these technologies (including necessary hardware and networking upgrades) should be rolled out within the organization.

The starting point for the knowledge worker information system rollout is an environment with either a PC LAN server or a Host/PC (*terminal emulation*) topology in place. All knowledge workers should have a PC; although many clerical workers (such as data entry clerks) may continue to use a terminal profitably for many years. Furthermore, the PCs should be at least '286 (AT class) systems with 640K memory. In order to take full advantage of a graphical user interface and the applications that have previously been described, '386 PCs with 4MB of memory are highly desirable. For this reason, all future PC purchases should be '386 PCs with the added memory. Obviously, however, most companies will have a mixture of '286 and '386 PCs for some time.

Migration for Sites with Host/PC Topologies

If a site is currently using a host/PC (terminal emulation) topology, the first step is to evaluate the e-mail system in use (if any). It is very important that the e-mail system support both client/server *and* enterprise-wide connectivity (even if neither of these functions is actually used right away). Open Mail (on the HP 9000) and DeskManager (on the HP 3000) both provide these capabilities. Without a good e-mail foundation, a company can find itself in a nightmare when it goes to extend its e-mail system. If no e-mail system is currently in use, an e-mail system with the preceding capabilities should be installed.

The second step is to implement file/print sharing using either LAN Manager/X or Portable Netware on the host. A LAN will need to be laid at this point and connected to the PCs. Printers in the office area may be provided either with RS232 lines from the server or with network printers (such as the LaserJet IIISi) hanging directly off the LAN.

Migration for Sites with PC LAN Server Topologies

If a site is currently using a PC LAN server, the first step is to determine whether to replace the current PC LAN server with a more powerful RISC-based server or to maintain the current PC LAN server and add a RISC-based application/communications server.

Sites running a DOS-based file server (such as OfficeShare or MS-Net) or a '286 version of Novell Netware should probably upgrade to a RISC-based HP 9000 running respectively either LAN Manager/X or Portable Netware. The HP 9000 will give them more robust file sharing capabilities, better administrative tools for backup and security, and the benefits of industry-leading e-mail and other applications. In this case, the old server may

continue to be used as a print server in the office environment or a new high-speed network printer (such as the LaserJet IIISi) may be directly connected to the LAN.

Sites running a Netware 386 server or a LAN Manager (OS/2) server may reasonably elect either to replace the current server with an HP 9000 running Portable Netware or LAN Manager/X or else to maintain their current server and add an HP 9000 for running e-mail and other applications. Replacement will consolidate administration and backup procedures, but will obviously come at a higher initial outlay.

The second migration step is to implement e-mail. It is important that the e-mail system should support both client/server and enterprise connectivity. Since most PC LAN e-mail systems are highly proprietary and have *very poor* connectivity, any such system should probably be replaced. Open Mail on the HP 9000 (and many other systems) is a very good choice for filling this need due to its excellent standards-based connectivity (through either X.400 or UNIX sendmail).

Joint Migration Strategy

The third step for sites starting with either host/PC or PC LAN server topologies is to implement client/server support for the application areas of (1) quantitative analysis, budgeting, and forecasting, (2) e-mail, and (3) review/approval/aid.

Up until this point, '286 PC clients with 640K memory could be fully supported (although 640K PCs running LAN software may have trouble running very large PC applications, such as Borland Paradox). These new applications will accommodate 640K PCs to some degree, but to take full advantage of the GUI-based applications, the PCs will need Microsoft Windows 3.0 with at least 2 MB and preferably 4 MB of memory.

Client/server support for quantitative analysis, budgeting, and forecasting requires installation of some tool for downloading data from the databases containing the desired information. Key factors to take into account in making the choice of the tool include what DBMS' currently hold the desired information and what spreadsheets or statistical analysis packages are currently in use by the quantitative professionals. These are important factors since the database download tools only support certain databases and certain desktop data formats.

The choice of a client/server e-mail package was made in the previous step, although only terminal-emulation access to the server e-mail system was available up until this point. Assuming that OpenMail (or DeskManager) is in use on the server, either NewWave Mail (for PCs running Windows 3.0/ NewWave) or AdvanceMail (for 640K PCs) should now be installed on the PC clients. Both NewWave Mail and AdvanceMail work hand-in-hand as client software with the OpenMail server software on the HP 9000. With NewWave Mail or AdvanceMail, the user never has to leave his or her PC environment in order to send mail. There is no more need to enter terminal emulation, log into a host, and then log into a host-based e-mail application. Furthermore, with NewWave Mail users can send text, graphics, spreadsheets, images, etc.

Client/server implementation of review/approval/aid entails providing a new GUI interface for an old application or developing new applications (presumably that work with existing databases). The various alternatives for doing this were previously explained, and the choice between the alternatives will depend on the terminal forms package that is currently in use, the DBMS in use, and the available MIS resources.

The fourth migration step for knowledge worker information systems is to implement client/server versions of ad hoc database query, text & document management systems, and executive information systems. Although some of these capabilities can be provided on 640K PCs, the full capabilities and benefits will require '386 PCs with GUIs. As with with the step 3 applications, choice of tools and applications will depend on the DBMSs in use.

General MIS Guidelines for Deployment

Most of the discussion up until now has focused on users, software applications, and topologies. A few additional general guidelines would be helpful to MIS in planning the migration to client/server so that both short-term and long-term objectives can be met.

Client/GUI Independence

Client/GUI information in this paper has centered on DOS PCs and MS-Windows, even though, as previously mentioned, most of the discussion is applicable to any type of desktop client system. Although DOS PCs are clearly the most popular desktop system, Apple Macintoshes also have a strong following among many knowledge workers, and UNIX workstations with a MOTIF GUI are beginning to move strongly from their technical roots into the commercial world. In spite of the desire in many MIS shops for a simple, single client-type world, users have a mixture of client types today and, almost assuredly in the future, will have even a greater mixture of systems on their desktops. For this reason, it is wise to evaluate client/server applications for their ability to work with multiple types of clients. Most of the applications specified by name in this paper either support multiple client/GUI types today or will do so in the relatively near future.

Application Integration

Given that knowledge workers use a number of different applications, it is very important that they be able to easily and intuitively integrate these applications. HP's NewWave environment provides an unparalleled means to do this, with support for compound documents with live links, inter-application supermacro capabilities (task automation), network distribution (LAN/WAN connectivity), and an intuitive desktop. NewWave now has the support of the three biggest players in the UNIX world (Hewlett-Packard and SUN--the market share leaders, and AT&T/NCR--the creator of UNIX). With this level of powerful support and with the expected endorsement of its core by the Object Management Group standards body, NewWave will soon be a de facto standard. NewWave supports all Windows 3.0 applications and advanced NewWave features are now available in applications such as

Lotus 123, Microsoft Excel, WordPerfect (available in 4q91), Lotus Ami Professional, Cognos PowerPlay, and Channel Computing Forest & Trees.

DBMS Independence

As was mentioned several times, various tools only support certain database management systems. The ability of knowledge workers to get access to the data necessary to effectively do their jobs is one of the keys to knowledge worker information systems. Even if a company is 100 percent successful in standardizing on one DBMS, the current corporate environment of mergers and acquisitions makes it wise to maintain some level of DBMS independence. For this reason, MIS should evaluate potential tool purchases on their ability to access the various popular DBMSs.

Network Security and Management

The security systems in use with current client/server applications and DBMSs are reasonably effective, but have poor usability due to the multiple layers of logons and passwords that users often have to go through in order to access server data. The OSF Distributed Computing Environment (DCE) will provide both a higher level of distributed security and better usability, when it becomes available in 1992. OSF DCE is being planned or implemented on many UNIX and proprietary operating systems including OSF/1, HP-UX, MPE, Ultrix, VMS, AIX, MVS, VM, OS/400, OS/2, and DOS. Industry analysts widely agree that DCE will be *the* de facto standard infrastructure for distributed computing.

Distributed systems management is also being considered by OSF with a selection of a standardized Distributed Management Environment (DME) being planned for this summer. Since the full implementation of such a standardized management scheme will probably not be available until 1993, HP is providing today PC software distribution and licensing software called HP Software Vendor. This software helps MIS to control versions and distribution of DOS and Windows applications in a LAN environment. Software Vendor has been heralded as a terrific means for MIS to regain control and order over their company's PCs.

Conclusion

Client/server implementations of knowledge worker information systems hold the potential of greatly increasing the productivity of knowledge workers through automating manual, process-oriented work, through shortening learning curves, and through helping knowledge workers get quick access to the right information so they can make better and more timely decisions. Those companies that aggressively implement knowledge worker information systems based on client/server can lower their costs and achieve competitive advantage over companies that do not do so.

Migrating A Turn-Key, Real-Time Test System From An HP/1000 (RTE) To An HP/9000 (HP-UX) Platform

James P. Langan
Kathleen M. Sagunsky

J.P. Langan & Associates, Inc.
2800 South Fish Hatchery Road
Madison, Wisconsin 53711
(608) 273-0428

BACKGROUND

In 1985 software was developed to collect, analyze and report nuclear containment integrated leakage rate test data. This software, written in FORTRAN, was developed for the HP/1000 A-700 computer running Hewlett Packard's Real Time Executive (RTE) operating system. The need to modernize this hardware and software platform demanded the migration of this test system to an HP/9000 series 300, HP-UX workstation programmed in C.

SOFTWARE DESCRIPTION

The data acquisition and analysis software, ExecTest consists of three main programs: 1) human interface (task scheduler), 2) report generator, and 3) data acquisition.

Human Interface and Task Scheduler, ExecTest

The Human Interface and Task Scheduler program (ExecTest) is a series of softkey menus which give an operator quick access to: data collection, test data, data analysis, and system status. When ExecTest first executes it requires as input a system configuration input file called a channel setup table. This file tells ExecTest: data acquisition specifics, hardware assignments, analysis methods, and input signal calibration curves. The channel setup table is read and compared to the actual system configuration. Any system configuration discrepancies are reported in detail. Serious discrepancies, such as interface not found, terminate further processing.

ExecTest's three main memory components are segmented to fit into 64 Kbytes of memory. The memory usage of these three components of ExecTest is: 1) main body 20 Kbytes, 2) system common memory (SCM) 500 bytes, and 3) eight segments of 40 Kbytes each. This constitutes approximately 341 Kbytes of

code and data.

Report Generator, ExecRprt

The report generator ExecRprt, is invoked spontaneously on operator command from ExecTest's softkey menu. The report generator input consists of: 1) starting and ending data scan as found in the test data log file, 2) type of report requested (data table or plot), 3) data analysis method, e.g., least squares fit, polynomial fit, Fourier transform etc... and finally 4) output peripheral. The report generator then creates a report using data extracted from the test log file. ExecRprt will finally update **SCM** with a status message of completion. ExecRprt consists of a main program with two segments. ExecRprt's main program body uses 20 Kbytes and two segments utilize 40 Kbytes of memory each. ExecRprt also uses 500 bytes of **SCM**, thus, giving a total memory consumption of 101 Kbytes.

Data Acquisition Program, ExecTake

ExecTest schedules the program ExecTake to start acquiring data immediately after the channel setup table is validated. ExecTake is a system level program operating at a real-time priority of 80. ExecTake reads every input channel (nominally 60 signals) at the required acquisition (scan) rate. After every acquisition, the collected data is written into a test log file and system common memory. ExecTake uses 500 bytes of **SCM** and 56 Kbytes is used for code totalling approximately 57 Kbytes of memory.

RTE DEPENDENCIES

ExecTest, ExecRprt and ExecTake together use many features of RTE. These features constitute two categories of system calls, executive communication (**EXEC**) and file management package (**FMP**). EXHIBIT A is a complete breakdown of those RTE features used by ExecTest and its associated programs. **FMP** is referenced as an ExecTest utilized RTE feature only. Conversion of **FMP** calls to their HP-UX counterpart is not discussed in this paper.

EXHIBIT A, RTE Utilized Key-Operating System Features

Executive Communication Calls (EXEC)

I. Program Control

1. Father-Son program scheduling
   a. exec (9) schedule with wait
   b. exec (10) schedule without wait

2. Swapping Control
   a. exec(22) data partition may not be swapped

3. Time retrieval request
   a. exec (11)

4. Stop Program Execution
   a. exec (6, 0, 2) terminate normally
   b. exec (6, 0, 1) terminate, save resources dormant

5. Initial Offset Scheduling
   a. exec (12, name, 2, scan time, hr, mn, sec, msec)
      1. schedule program, units seconds, at the scan rate, starting at hr:mn:sec:msec

II. Standard I/O

1. Device Driver Read
   a. exec (1)

2. Device Driver Write
   a. exec (2)

File Management Package Calls (FMP)

1. FMPread - read a file record

2. FMPwrite - write a file record

3. FMPopen - open file

4. FMPpost - update data control block onto file

5. FMPsetposition - specify data block and offset to read to or write from in file

ExecTest, ExecRprt Software Migration

Migration of software for those portions of ExecTest which do not perform data acquisition, e.g., test reporting, test analysis, human interface, utilized only code conversion techniques. Conversion of RTE FORTRAN into HP-UX C is not presented herein. ExecTest and ExecRprt do utilize one important RTE feature worth discussion, program scheduling. Program scheduling with and without wait are invoked with RTE system calls **EXEC(9)** and **EXEC(10)** respectively.

The HP-UX version of ExecTest implements RTE program scheduling calls **EXEC(9)** and **EXEC(10)** with the HP-UX function **system** call. **system** executes a single shell (**sh**) command, input as a runtime string, as though it were composed at a user's terminal. This runtime string must be carefully prepared since **system** does not perform any pre-processing prior to its passage to **sh**. **sh** will wait for the child process's completion unless an '&' terminates the runtime string. Simulation of an RTE **EXEC(10)** (schedule without wait) is made by invoking **system** with the runtime string argument terminated with an '&'. **system's** runtime string argument must not be terminated with an '&' when simulating an RTE **EXEC(9)** system call (schedule with wait).

for immediate execution without wait

```
system("ExecTake &");
```

for immediate execution with wait:

```
system("ExecRprt, start, stop, plotter, analysis ");
```

Although using **system** to spawn a new child process is very simple, this technique is not the preferred HP-UX implementation of child spawning. There are two problems with using **system** to spawn children. Using **system** is very slow because much CPU time is spent loading **sh** (Bourne shell) and the child process. This technique is also memory intensive because both **sh** and the child process are resident. ExecTest and its associated processes are memory locked. If not enough memory is available to the shell and child processes, ExecTest will wait until such memory is available. Under certain conditions this wait could be indefinite.

The preferred method for spawning an HP-UX child process is significantly more complex. This method requires two system calls, **fork** and **exec**. To implement this scheme, a parent process must first call **fork** to create a mirror image of itself. This mirror image child begins execution immediately after the **fork** call. The child determines that it has been **fork**ed by a return value of **0**. The parent is returned the child's process id number (PID) after the **fork** call. The mirror image child process must then call **exec**

to overlay itself with the desired executable process.

If the parent process must wait for its child's completion, the HP-UX **wait** call must be executed after the process is **fork**ed. The following is the **fork, exec, wait** sequence:

1. Parent process calls **fork** to create a child image of itself.

```
fork();
```

2. Parent process calls **wait** to suspend itself until child process completes.

```
wait(0);
```

3. Child process determines a **0** has been returned after the **fork** and calls **exec** to overlay itself with desired executable process.

```
execl("/usr2/ExecTest/ExecRprt", (char*) 0);
```

4. Parent is waiting for child process to complete.

ExecTake, Software Migration

EXHIBITs B and C are the RTE and the HP-UX versions of ExecTake respectively. The objective of migrating ExecTake from RTE to HP-UX was to maintain as much software continuity as possible. This was performed by replacing RTE features used by ExecTake with their HP-UX counterpart. In order to maintain program continuity it was decided that those RTE features which did not a have single HP-UX replacement would be simulated with multiple HP-UX calls or additional program logic.

In order to perform this conversion it is necessary to understand how ExecTake operates under RTE. Most RTE application software is assigned a priority of 99 or larger. ExecTake is initialized with a priority of 80 giving itself CPU precedence over most applications. Operating system programs which demand high CPU availability are assigned priorities between 0 - 80. To reduce the possibility of RTE crashes ExecTake's priority is set so it <u>does not</u> conflict with any system program.

ExecTake, like most data acquisition software, must be memory locked to avoid unacceptable disk swapping. This memory lock is performed by calling RTE **EXEC(12)**. ExecTake then creates, opens, and initializes a unique data log file by calling its subroutine

get_unique_file.

To complete ExecTake's initialization, **EXEC(11)** is called to obtain the current system time ti. t_i is incremented by two minutes to create a future time t_f which will be ExecTake's restart time. ExecTake programs t_f and t_a (acquisition frequency in seconds) into RTE's program scheduler with an **EXEC(12)** call. ExecTake will restart at time t_f, and execute repetitively every t_a seconds. For example, assume our current system time is 12:00:00 and the desired acquisition frequency is 15 seconds, ExecTake first acquires data at time 12:02:00 and continues to execute every 15 seconds: 12:02:15, 12:02:30, 12:02:45, 12:03:00 etc...

ExecTake then suspends itself with an **EXEC(6)** call and waits for RTE's scheduler. This suspension is performed to conserve CPU time when acquisition is not active. When ExecTake is awakened by the operating system, it immediately calls **EXEC(11)** to get the current system time. Subroutine **read_channels** is then executed to acquire data. The acquired data is then written to the log file via subroutine **update_file**. The program first loops to the suspend call **EXEC(6)**, then it suspends itself until ExecTake is rescheduled by RTE. Essentially this data acquisition is an infinite loop timed by RTE's scheduler. This loop exits when a system call to **ifbrk** returns a non zero value. This occurs when ExecTake's break flag is set by either an operator or other program. Once a break is detected ExecTake terminates itself by calling **EXEC(6)**.

EXHIBIT B, RTE ExecTake Source Listing

```
c
c.. program ExecTake
c..
c.. scan analog and digital instruments at regular intervals
c.. data collection frequency is between 5 and 60 seconds
c..
c system program priority of 80
program ExecTake(1, 80)
c include system common
$include exectest.inc
     integer name(3), timearray(5)
     data name /2hEX, 2hEC, 2hTA/
     call exec(22, 1)
     call get_unique_file
     call exec(11, timearray)
     timearray(3) = timearray(3) + 2
     call exec(12,name,2,iscantime,timearray(4),timearray(3),0,0)
100  call exec(6, 0, 1)
     call exec(11, timearray)
     call read_channels()
```

```
     call update_file()
     if(ifbrk(idum).lt.0) goto 200
999  goto 100
200  call exec(6, 0, 2)
     end
```

EXHIBIT C, HP-UX ExecTake Source Listing

```
#include <sys/rtprio.h>
#include <sys/lock.h>
#include <sys/types.h>
#include <sys/ipc.h>
#include <sys/shm.h>
#include <time.h>
#include <signal.h>
#include <stdio.h>

#define TRUE 1
#define FALSE 0

/* external globals */
extern int errno;

/* local globals */
int break=FALSE;

void set_break()
{
     break = TRUE;
} /* set_break */

void sys_error(strr)
     char strr[];
{
     printf("System call %-s generated error %d\n", strr,
          errno);
     printf("Program terminated \n");
     exit();
} /* sys_error */

main(argc, argv)
     unsigned int argc;
     char **argv;
{
     struct timeval cursec, nextsec;
     struct timezone tz;
     unsigned int seconds;
     int scan_time=5, eid, shm_id;
     char *shm_memory
```

```
     if((shm_id = shmget(key, 2048, 00006)) == -1)
          sys_error("shmget");

     if((shm_memory = shmat(shm_id, (char *)0, 0)) == -1)
          sys_error("shmat");

     if(open_3497(&eid) == -1)
          sys_error("open_3497");

     if(rtprio(0, 80) == -1)
          sys_error("rtprio");

     if(plock(PROLOCK) == -1)
          sys_error("plock");

     void signal(SIGTERM, set_break);

     while(!break) {
          gettimeofday(&cursec, &tz);
          readchannels();
          updatefile();
          gettimeofday(&nextsec, &tz);
          seconds = scan_time -(nextsec.tv_sec - cursec.tv_sec);
          sleep(seconds);
     } /* while */

     if(shmdt(shm_memory) == -1)
          sys_error("shmdt");

     if(shmctl(shm_id, SHM_RM_ID, 0) == -1)
          sys_error("shmctl remove id command");

     close_3497(eid);

     exit();
} /* main */
```

To insure proper response time and continuous acquisition it is necessary to execute ExecTake with a high real-time priority. ExecTake must have CPU priority so it does not get interrupted during acquisition. Most HP-UX programs are assigned a priority between 150 - 160. ExecTake calls system function **rtprio** to set its own priority to 80. It is necessary that ExecTake be spawned from a **root** session or indirectly from a **root** scheduled program. **rtprio** returns an error when accessed by any other user level.

To further insure ExecTake has enough available CPU time when acquiring data it must lock itself into memory. A system call to function **plock** is used to perform this memory locking. Much time is wasted by HP-UX without this call to **plock** because of HP-UX disk

swapping.

When compared to RTE, HP-UX is sloppy in its periodic time scheduling. RTE program scheduling was originally chosen because it allowed for periodic data collection without unnecessary CPU overhead. In order to keep ExecTake as consistent as possible with its RTE counterpart, it was necessary to somehow simulate program scheduling. The simulation of the RTE **exec(12)** call requires two HP-UX system calls **get_time_of_day** and **sleep**. Because acquisition timing is periodic, **get_time_of_day** is called just before acquisition starts (when ExecTake awakens), and again after acquisition and file logging completes. The difference between these calls subtracted from the desired data acquisition (scan) rate is the amount of time ExecTake must remain dormant. HP-UX system call **sleep** is used to make ExecTake dormant.

Essentially the while loop in ExecTake is infinite, however, at some point it is necessary to stop acquisition. In the original RTE version the system break flag was used to do this. In our example the break flag must be simulated or somehow replaced. This was very easily done by implementing an HP-UX operating system feature called **signals**. A call to **signal** is made with two arguments. The first argument is the signal which when detected, causes the service routine **set_break** to execute. The second argument is the address of the service routine **set_break**. A **signal** may be set either from a users session or another program. The process id number must be used to identify the process the signal is being sent to. ExecTake is programmed to terminate on a **SIGTERM** (terminate) signal.

Another important HP-UX feature used by ExecTake is error reporting via the system global variable **errno**. In the event of an error most HP-UX system and library calls return **-1**. Whenever system calls are executed the return variable should be tested for error occurrences. If an error has occurred an error service routine should be executed. System errors are further identified by the global variable **errno**. Function **sys_error** was created to service error conditions. **sys_error** is passed a character string which identifies the function and the system call which generated the error. **sys_error** then uses the externally defined global variable **errno** to identify the system error more specifically. Specifics to the error may then be identified in the correct HP-UX manual.

The RTE version of ExecTake utilizes system common memory for its inter-process communication. System common is created on the HP/1000 at generation time and resides as a portion of System Available Memory. RTE system common memory is allocated at link time to un-named common blocks defined in FORTRAN programs. Many problems can occur when using system common on RTE HP/1000s, e.g., system "hangs", software "lock ups", system crashes, system available memory unavailable. Fortunately HP-UX has a much

improved method of sharing memory segments between programs.

HP-UX shared memory is easily created, utilized and destroyed via simple operating system calls. Four calls are used by ExecTake; 1) **shmget** (shared memory create), 2) **shm_ctl** (shared memory control), and 3) **shm_at** (shared memory attach), and 4) **shm_dt** (shared memory detach).

shm_get creates a shared memory segment via the HP-UX kernel memory manager and returns a **shmid** (shared memory segment id number) which must be used in all future shared memory operatons. This HP-UX system call requires three arguments:

1) key identifier, which is specific to calling process
2) size of the requested memory segment
3) allowed access to memory

The key specifies the ownership of the shared memory segment. The size of the requested memory segment is in bytes and passed as **shm_get**'s second argument. ExecTake operates on a stand alone system so any program is allowed access although the shared memory access can be limited to specific users, groups, or other programs according to following table:

00400 read by user
00200 write by user
00600 read/write by user
00060 read/write by group
00006 read/write by others

shm_at is used to attach a shared memory segment to a program. Even though a program has created a shared memory segment, the program can not access the shared memory until it is attached. **shm_at** returns an address pointer to the memory segment. When shared memory is no longer required it is necessary to detach and purge the shared memory segment from the HP-UX memory manager. The shared memory is detached and purged via calls to **shm_dt** and **shm_ctl** respectively. **shm_ctl** must be given **SHM_RMID** as its second argument to properly remove the segment.

EXHIBIT D, open_3497 Source Listing

```
#include <fcntl.h>
int open_3497(eid)
     int *eid;
{
     if((*eid = open("/dev/HPIB/3497", O_RDWR)) == -1)
          sys_error("open_3497, IO Library open");

     if(io_lock(*eid) == -1)
```

```
          sys_error("open_3497, DIOL call io_lock");

     if(io_reset(*eid) == -1)
          sys_eror("open_3497, DIOL call io_reset");

     if(io_timeout(*eid, 3000000) == -1)
          sys_error("open_3497, DIOL call io_timeout");

     return(0);

} /* open_3497 */
```

Function **open_3497** (EXHIBIT D) performs the initialization on the HP3497 HPIB bus instrument. HP-UX system call **open** is used to obtain an entity identifier (eid). This **eid** is used in all future accesses between HP-UX and the 3497 instrument. **open** is given the device file name and path as it first argument. The device file makes the link between the hardware and the HP-UX kernel's interface driver. The second argument tells **open** that the device will have read and write access.

open_3497 next calls Digital Input Output Library (DIOL) function **io_lock** with the **opened** 3497's **eid.** This **io_lock** prevents other programs from accessing the 3497 during acquisition. Extreme caution must be given to implementing **io_lock.** It is important that system peripherals such as disks, tapes, system consoles etc... not be present on any HP/9000 interface which is **io_locked** to a process. In these situations a data acquisition failure could lock necessary system peripherals and crash HP-UX.

Next **open_3497** initializes the HPIB bus by calling DIOL functions **io_reset** and **io_timeout. io_reset** is used to reset the HPIB bus to a known state before io functions are performed. **io_timeout** initializes the HPIB bus timeout preventing null HPIB bus responses from "hanging" acquisition. An HPIB bus timeout of three seconds has been chosen for our purpose. This means, once a read command is issued on the HPIB bus, the instrument must respond within three seconds. If the instrument fails to respond within three seconds an io error is returned to the acquisition process.

EXHIBIT E, read_single_channel Source Listing

```
#include <fcntl.h>
extern int errno;
int read_single_channel(chan, eid, io_buff, value)
     int chan, eid;
     char io_buff[];
     double *value;
{
     strcpy(io_buff, "AC00\nVC2VF2VR5\nVT3\n");
```

```
     strl = strlen(io_buff);
     if(write(eid, io_buff, strl) == -1) {
          sys_error("read_single_channel, HPIB write");
          return(-1);
     } /* if */

     if(read(eid, io_buff, 3)) {
          sys_error("read_single_channel, HPIB read");
          return(-1);
     } /* if */
     return(0);
} /* read_single_channel */
```

Function **read_single_channel** (EXHIBIT E) is the simplest function used by ExecTake for data acquisition. This function sets up an HP3497 HPIB bus instrument for a single channel resistance measurement. HP-UX library calls **read** and **write** are used for this communication.

The 3497 is sent a command string via a **write** call. This call has three arguments; 1) eid number, 2) output buffer, and 3) output buffer length. The output buffer contains three individual 3497 commands. Each command in the buffer is separated by a newline character '\n'. These 3497 commands are: 1) set the channel to be accessed, 2) set the voltmeter and current source parameters, and 3) issue a software measurement trigger. Immediately the **read** command is executed to input the measured resistance. Upon a successful completion three data bytes are input from the 3497 instrument. The **read** command has three arguments: 1) eid, 2) receive buffer, and 3) expected byte count.

Notice in this example that error reporting has remained consistent with that used in ExecTake's other functions. It is possible to get errors with DIOL calls that can not properly be decoded by the system global **errno**. A call to DIOL function **io_get_term_reason** may be necessary to identify the reason for a failed DIOL call. Refer to the **HP-UX Device I/O User Interfacing, Concepts and Tutorials Manual** for further information.

Conclusions

Successful migration of ExecTest, ExecRprt and ExecTake to HP-UX was complete after approximately six man months of programming. After project completion, program efficiency and memory comparisons were made. This HP-UX implementation of data acquisition software caused problems with acquisition timing. The HP-UX **sleep** call has an accuracy of 1 second (no fractions), causing an acquisition timing error of up to 1/2 second. Because data acquisition frequency is between 15 seconds and 60 minutes, this acquisition

error is acceptable. Other applications requiring much more accurate timing will have problems using the **sleep** call in this manner.

One major feature of HP-UX which made these programs execute more efficiently is shared memory. This does seem to require much system overhead (exact amount unknown), however, the benefits over RTE system common are many 1) virtually unlimited space, 2) allocation on demand, 3) does not require special system configuration, and most important 4) easy to use.

Another significant difference between RTE and HP-UX is the program memory allocations. The following table describes the differences for each major module:

	Code Size (Kbytes)	
	RTE	HP-UX
ExecTest	341	643
ExecRprt	101	193
ExecTake	57	93

It is clearly seen that HP-UX based applications use significantly more memory than those operating on RTE. When RTE's memory usage of approximately 128 Kbytes of memory is compared to the HP-UX minimum requirement of 4 Mbytes it is clear that HP-UX is a major memory consumer. An X-windows front end was developed for ExecTest using the Motif development software to replace the original RTE based softkey menus. The memory consumption for this X-windows front end was 2.4 Mbytes, thus, giving more support to HP-UX's excessive memory usage.

This paper has described one implementation of converting an RTE data acquisition system to function on an HP-UX platform. This HP-UX implementation of ExecTest, ExecRprt and ExecTake may not be the most efficient usage of HP-UX features. These examples were selected because they demonstrated the closest overall software continuity between RTE and HP-UX.

Robert M. Hersh
Warren R. Weber
AGS Genasys Corporation
9710 Patuxent Woods Drive
Columbia, MD 21046
(301) 596-7410

Paper 8058

Making a Square Peg Fit into a Round Hole

The Impossible Dream? Can a Pascal-based system, running under RTE-A on an HP 1000 be converted to a C-based system running under HP-UX on an HP 9000? This paper details the trials and tribulations of this conversion process as the conversion team deals with such issues as: selecting a hardware/software configuration, meeting existing performance requirements, converting existing data structures, finding equivalent system calls, interfacing to a Sun workstation, and converting "escape code graphics" to XWindow.

The current system exists on an HP 1000/A900 and supports telecommunications test equipment on up to 25 serial ports and two HPIB buses. Additionally, 12 graphics terminals are supported along with an Ethernet interface to other computer systems. The converted system must support all these requirement as well as being able to have its data configuration down-loaded from a Sun workstation, provide alarm notification to a Sun, accept alarms from a Sun and/or AT&T systems, and be designed to its software can be distributed over multiple 9000s.

1.0 Introduction

In 1988 a paper was presented describing the HP 1000-based Automatic Monitoring System (AMS) at the Orlando Interex conference. Again in 1990, a paper was presented at the Boston Interex on further AMS developments, specifically accessing multiple Shareable Extended Memory Areas (SHEMAs). This multiple SHEMA access is being made available by HP with release 6.0 of the RTE-A operating system.

Finally we come to 1991 in San Diego. This paper will chronicle the events leading to the conversion of the HP 1000 based AMS to a Unix-based HP 9000 AMS.

1.1 What is AMS

AMS is a real-time telecommunications monitoring and control system. Through a variety of "monitoring devices" (like the HP 3852), the system receives telecommunication equipment statuses. Upon receipt, the information is filtered by user defined thresholds and reported graphically via "real-time" displays (RTDs). The RTDs are a series of graphic screens that allow a user to observe circuit statuses through the use of color coded acronyms. The user may also call up detailed information on a circuit showing equipments and other pertinent data, perform command and control operations or extract a chart showing the last five days of equipment statuses. Data is stored in an IMAGE/1000-II database for historical reference and report generation.

The addition of AMS to a telecommunications center means that an operator is notified of a problem in a communications circuit almost as soon as it occurs. This remedied a situation where the consumer had to notify the telecommunications center that a problem existed -- a very embarrassing situation for the communications center. With AMS not only is the operator made aware of a problem almost immediately, but often times he has corrected it without the consumer being aware that there even was a problem.

1.2 Original AMS

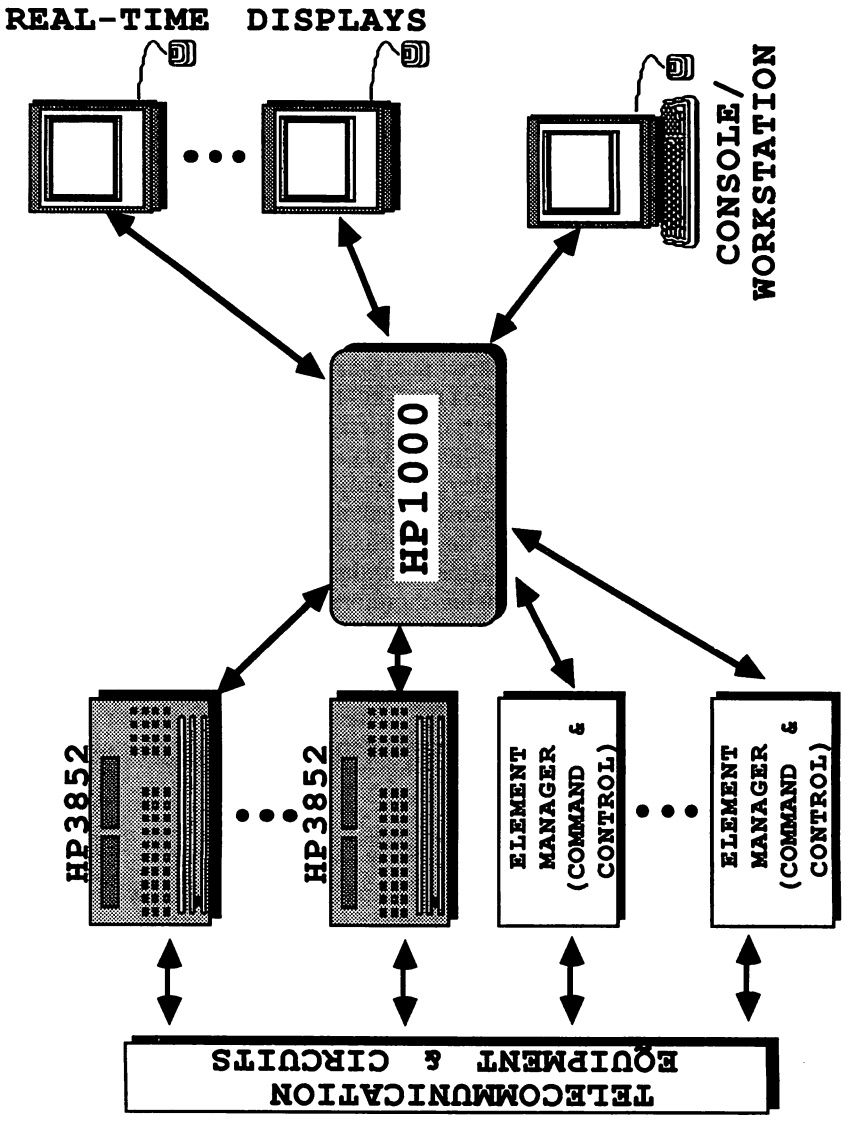
In 1984, the Department of Defense (DoD) contracted with the Genasys Corporation (now AGS Genasys) to participate in an R&D effort. The effort was to determine whether telecommunications equipment could be monitored in such a fashion as to provide fast and accurate information to the telecommunication control staff (tech control). From those humble

beginnings, AMS rose like the Phoenix from the ashes. In the last seven years, AMS has grown in scope, been modified to take advantage of new and better RTE-A and has been made a durable and tough system capable of handling status information from thousands of sources. Over the years, AMS has grown to approximately 90,000 lines of Pascal code.

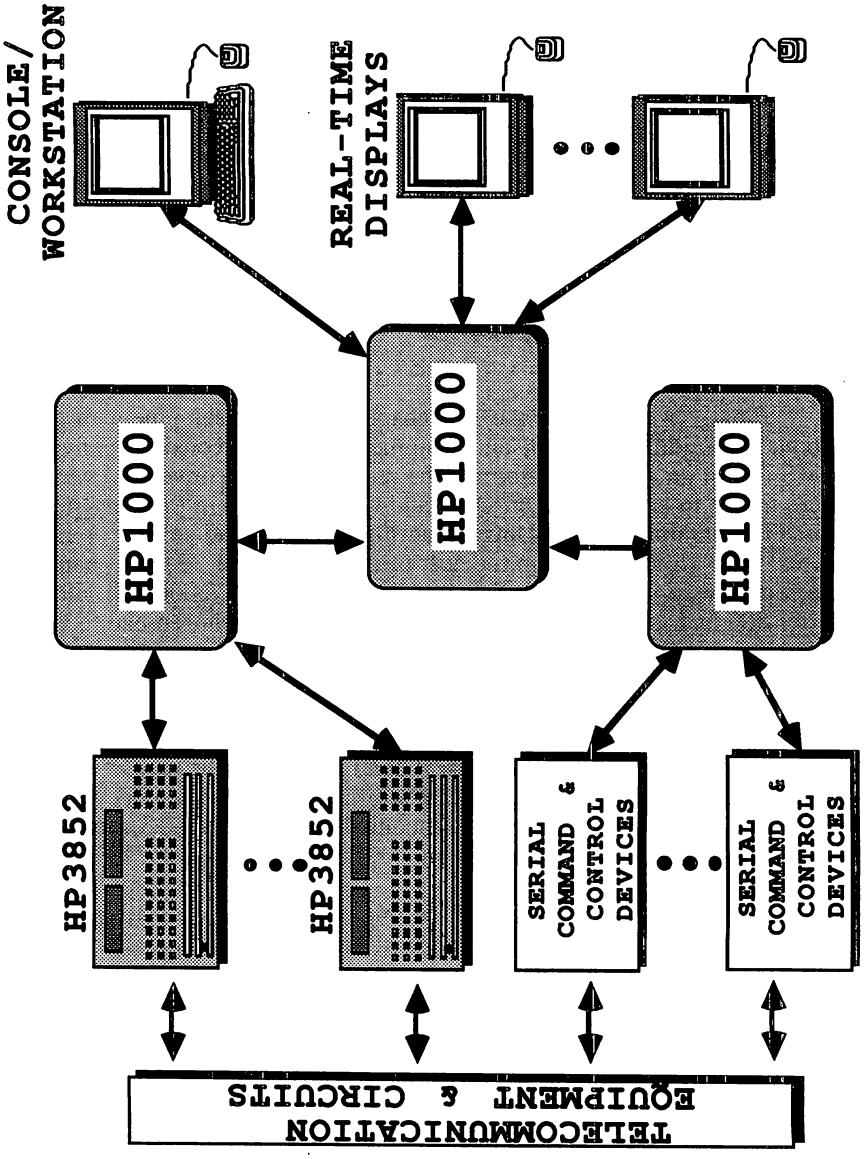
The original AMS is currently a production system at one large DoD site and four smaller DoD sites. The system has been in full operation for five years and is now an integral part of the day to day environment where it is installed. The system supports various combinations of HPIB and serial interfaces (generally either three HPIB and five multiplexer cards or four HPIB and four mux cards). Equipment statuses come in over the HPIB or serial lines and the alarm messages are processed through device-specific software (see Figure 1). The resulting information is then processed through a general purpose filtering mechanism using operator-defined parameters. Once the filtering has taken place, the alarms are sent to software that converts the device-specific information to a something that means more to a human (which piece of communications equipment is in alarm, what circuit(s) the alarm affects, etc.). This information is placed both in SHEMA data tables and the database for later use. If the circuit that had the alarm is being monitored by an operator, his RTD is updated to reflect the new status. The operator may then request more detailed information on the circuit to more precisely identify the problem. If, based on this information, the operator wishes to issue a reset command to a device, he asks AMS to send the request. AMS translates the request into a command the device understands and sends it out the appropriate LU.

1.3 Distributed AMS

Distributed AMS (DAMS) was implemented and is now operational to support increasing demands to monitor telecommunications equipment. Simply, the backplane of the HP 1000/A900 was fully populated and did not provide any expansion capability to connect new monitoring devices. A choice had to be made: use an expansion chassis or go distributed. The distributed option was selected because the current HP 1000/A900 was being pushed to its processing limits as well. Thus a distributed system was designed in which several HP 1000/A400 would be used for status collection and filtering and the filtered information would then passed on to an HP 1000/A900 via NS-1000. The RTDs, database, alarm conversion and data configuration software remain on the HP 1000/A900 (see Figure 2).



AMS Configuration
Figure 1



DAMS Configuration
Figure 2

The implementation of DAMS presented us with many challenges. First, because of the size of our system and the fact that it is memory based, we were using the largest SHEMA allowed under the RTE-A operating system. The structures to support DAMS also needed to be memory resident. Thus we devised our method for using multiple SHEMAs [see Paper 1001, INTEREX 90 for a solution to this problem]. Secondly, we had to overcome NS/1000. Under testing at our facility using two HP 1000/A400s all seemed to work fine. Upon porting to the operational system, we ran into several problems dealing with multiple nodes. These have been ironed out, the testing completed and DAMS is now being certified to become the production system.

1.4 Unix AMS

Finally we come to the last chapter in the development of AMS. About the middle of 1990, our customer wanted to consider converting AMS to run on a Unix platform. The reason for this conversion was they were trying to standardize on Unix and C. They felt, in the long run, the cost of maintaining a non-Unix platform (hardware and software) and the Pascal code would outweigh the cost of the conversion. Several hardware platforms were investigated: Sun SPARCs, IBM RISC/6000, the HP 9000 series 800 and the HP 9000 series 300. The machine chosen had to be fast enough to overcome Unix (a bit of a CPU hog for this application) and still provide some level of real-time processing. The HP 9000 series 300 was selected because of the MIPS rating, the real-time hooks in HP-UX and the assurances of HP that their machine could in fact do the job. The government decided to fund the effort and the project was to start in 1991. AMS would be converted to a Unix platform and the Pascal code would be converted to C. The effort was scheduled to take about 13 months and involve approximately 10 analysts and programmers.

The conversion to Unix will be implemented in two distinct phases, with the first phase resulting in a system that can be fielded at small sites and the second phase producing a large site system.

1.4.1 Phase I

About two-thirds of DAMS is being converted in this phase of the effort which will replace the existing small site systems. The Phase I system is being developed to be integrated with a Unix-based Management Information System (MIS) as shown in Figure 3. This presented a minor problem in that the MIS has been implemented on a Sun-4 workstation (whose SPARC processor uses different byte alignment and structure

padding formats than the HP 9000). The total integrated system will be fielded initially at five sites and is projected to be installed at more than twenty sites worldwide.

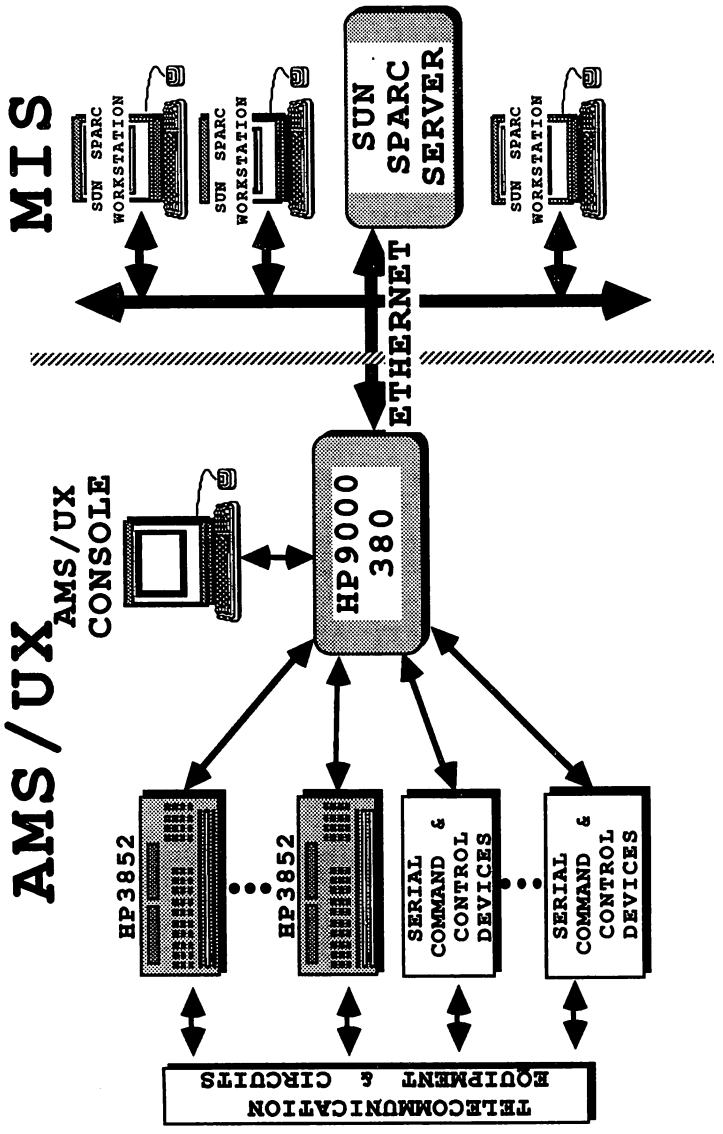
The DAMS portion being converted includes: the data capture, filtering, internal table management, alarm conversion and real-time display functions. The data configuration and database maintenance functions are to be converted in Phase II (the MIS is to provide these functions under Phase I).

The bulk of the system is being converted using a Pascal-to-C converter utility. The real-time display code is being rewritten from scratch to take advantage of the X Window environment. In addition, a system-to-system interface is being written to handle the data flow between AMS/UX and MIS.

Several opportunities (a.k.a. problems) have risen from this effort. Included in these opportunities: using a client-specified interprocess communication (IPC); talking across dissimilar platforms (MIS is on a Sun SPARC); converting RTE-A internals to Unix; and do Windows (as a contractor we do floors, toilets and yes, even Windows) for the first time. In addition, several design considerations had to be made in light of Unix and, of course, we had to work with the MIS contractor to develop an Interface Control Document (ICD). The final opportunity was using a Pascal-to-C converter that we were told would do the job in three months (six months later...).

1.4.2 Phase II

Under Phase II, the data configuration and database portions of DAMS are to be converted. This system will initially replace the large site system but will be configurable for any size site. The Phase II system will be modelled on the DAMS system in that there will be front end systems that will handle the device interface and filtering functions with a larger back end system that will handle the database and RTD functions. There are still design issues being worked with this system, among them whether a RTD server will be necessary given the volume of data the database will have to handle and the number of RTDs that the system will have to support. Included in this phase will be any enhancements to DAMS that were not included in Phase I. The most significant of these enhancements is the monitoring of communications networks (using N.E.T.'s IDNX T1 multiplexers).



Phase I AMS/UX
Figure 3

Making a Square Peg Fit into a Round Hole
8058 - 8

2.0 Design Concerns

As with any project the size of AMS/UX, there were a number of concerns. Among there were the response time of the Unix operating system, the ease with which the RTE system calls (primarily for the device interface software) could be converted to Unix, being able to quickly get the inter-host interface running and the differences between Sun's SPARC architecture and HP's 68040.

The overall intent of the conversion effort was to get the existing RTE/Pascal code moved to Unix/C as quickly as possible and getting it running without making it efficient or pretty. After the system was running, we would then be able to go back through the code and optimize where necessary. There were certain parts of the code that would port easily while others would have to undergo significant revisions (principally the device interfaces). Finally, there were portions of the system that would have to be written from scratch (the new display and MIS-interface programs).

The most important part of the system to get running early was the system undercarriage that would support the rest of the system. The undercarriage builds from and writes to files the memory-based configuration tables of the system we are monitoring, provides inter-process and inter-host communications and a host of system library functions.

System performance is a major concern in the design and implementation of AMS/UX. Our performance specifications require that the time from alarm receipt (from a monitoring device) to output to the operator be no more than five seconds. This has not been a difficult number to meet with even a heavily loaded RTE system, but it is a concern to us with a Unix system. We will be making use of Hp-UX's rtprio call to adjust the priority of the programs within AMS/UX so that the device interface programs run as realtime processes. This will require tuning as the system is integrated since we don't want to miss data nor do we wish to have the non-realtime processes get shut out of the CPU.

The graphics system AMS currently uses is based on escape codes being sent to a 2397A terminal emulating a 2627A. Moving from this environment to X Window is a rather large step so we are using all the tools we can to make the transition as simple as possible. The biggest help is the Architect screen design tool, which lets a developer rapidly prototype his screens and then generates the C code that creates the screens. By using

the Architect, we were able to quickly mock up our screens to get customer approval before we spent a lot of time working on the code behind the displays that does the "real" work (CallBacks and the like). Working in this manner, we were able to have our our screens all laid out in one month, which certainly helped out our schedule.

3.0 Conversion Issues

To port the code from Pascal to C, there were a number of issues that had to be resolved. Among these are converting the Pascal to C easily and reliably, inter-process communications, inter-host communications, shared memory, device-specific interfaces and data alignment problems between hosts.

3.1 Pascal-to-C Conversion

To assist the conversion of the Pascal code, we obtained the p2c (version 1.18) Pascal-to-C code translator (copyrighted by Dave Gillespie and distributed under the GNU License Agreement) and configured it to run on one of our Sun-4 workstations (and later on our HP 9000/380). The configuration process involved teaching the converter about the particulars of the target system (word size, data alignment, etc.), familiarizing it with the source Pascal (primarily the extensions to "standard" Pascal that are part of Pascal/1000), setting up equivalences between RTE system calls and Unix system calls (or writing library routines to take their place and instructing the converter to reference them in the converted code) and making stylistic decisions on how to deal with Pascal that could be converted a number of different ways (for instance, how to deal with enumerated data types and Pascal sets). The converter required constant attention while converting the first group of programs as new issues cropped up that had not been dealt with before (principally system calls not previously encountered). This type of problem gradually disappeared as we converted more programs and fewer surprises presented themselves.

While the converter saved a great deal of time, there were some areas that it was not able to handle and that required that a human become involved. There were a number of areas that, despite great effort with the converter configuration process, some did not convert well. This usually involved RTE system calls, particularly low-level I/O calls.

3.2 Inter-process Communications

In an effort to facilitate an easy integration with the MIS system and the MIS system's contractor, we were supplied with a set of library routines that were to be used for communications between the two systems. This Inter-Process Communications (IPC) package is based on the Unix System V message handling routines and was already in use on other systems. The major problems we encountered were that the library had only been tested in a homogeneous environment and never tested on an HP machine (it was originally developed on AT&T 3B computers and ported to Sun-3s and -4s). Getting the code to compile and link was simply a matter of locating the correct system libraries and adjusting the make file to account for the differences. Getting the library and its associated daemon to run took longer but was accomplished by modifying our file structure to accommodate the library's needs and remaking the HP kernel to allocate the necessary system resources.

Since IPC software was written to handle both inter- and intra-host communications, the decision was made to convert our existing intra-host communications to use its library functions. To implement this, we took the DAMS subprogram that handled the majority of our interprocess communications and replaced RTE's Class I/O calls with calls to the IPC library. Since, however, the IPC method of identifying hosts was different from the way the RTE system dealt with them (using the host name vs. a node number), a means of translating the node numbers to the corresponding host name was required. This was implemented in a simple lookup table that was initialized at system startup time using a host configuration file available from MIS using an NFS-mounted file system. Additionally, the possibility of having two programs with the same name on the same host will exist in the large site version of the system so an additional identifier was also required to allow the inter-process queue manager to locate the intended recipient.

3.3 Inter-host Communications

A part of DAMS that did not lend itself to using Class I/O was the inter-host undercarriage that allowed programs to share memory-based table information across hosts on the distributed system. To do this, we implemented a method for querying a remote node for the desired information (using NS/1000 calls), having the host perform the lookup, package the response (the size of which was often unknown until the data had been located in the tables), sending the response back to the requesting node and notifying the requestor that the desired information had arrived.

With the more sophisticated Remote Procedure Calls (RPCs) available under Unix, this inter-host interface was rewritten to utilize the capability.

The RPC is a communication mechanism that allows a client program to make a procedure call that is executed by a server program running on a different machine. To the client program, the RPC procedure call looks just like a call to a procedure in the local address space. RPC hides the details of of the underlying network from the client and server programs, making the procedure call transparent. RPC is often used to communicate between processes on different machines, but also works for communication between different processes on the same machine.

RPC presumes the existence of eXternal Data Representation (XDR), a standard for the exchange of data between machines with different architectures (very important to us since we are required to communicate with processes on Sun-4 workstations). XDR, in a process called encoding, converts data into a standard format that can be sent over a network. On the receiving host, XDR, in a process called decoding, converts the data into a format suitable for the receiving hardware. Using XDR, RPC can handle arbitrary data structures regardless of a machine's byte alignment or structure padding conventions.

Through the use of RPCs, we were able to replace about 9000 lines of code with 1000, remove an extra layer of interface software and allowed for more efficient use of system memory by allocating the exact amount of memory required for the returned information. Additionally, we did not have to concern ourselves with data alignment issues as we did with the IPC calls.

3.4 Shared Memory

Since AMS is a real-time system, the system configuration information and the current status of the devices being monitored are stored in shared memory tables for quick access. In the RTE version of DAMS, our space requirements had exceeded the limitations of a single RTE Shareable Extended Memory Area (SHEMA) which was not a problem with Unix. This allowed us to do some rewriting of our existing code to make use of a single shared memory area which increased our performance.

3.5 Device-Specific Interfaces

One area that required a great deal of work was the device-specific interface programs. All the RTE Exec calls had to be recoded into Unix system calls, the availability of information (or lack thereof) from the device drivers had to be dealt with and RTE's unique method of dealing with device and driver errors had to be changed. We replaced the RTE Exec 1 calls (used to determine both the device's and device driver's status for HPIB instrumentation) with an hpib_spoll command. This returns both the status byte from the device and an error code from the device driver (although not in the same format used by RTE). If the Exec 1 call encountered an error, it would return and execute the next instruction in the program; if there were no problem, it would return and skip the next instruction in an assembly language-like manner. This alone caused a great deal of the device interface code to be rewritten. The actual I/O to the devices required that the Exec 2 and 3 calls be replaced with Unix reads and writes. We encountered the same problems with the Exec 2s and 3s as we did with the Exec 1s -- the error return from the calls had the same error/no skip and no error/skip mechanism. This required rewriting a good portion of the interfaces. By the time the device interface programs had been converted to C, they had all shrunk a good deal.

Another aspect of the device interface code that required a great deal of work was getting Unix ready to talk to the devices. In RTE one simply addresses a command to a Logical Unit (LU) number, the operating system looks up the LU in its internal tables to find the select code and the device's address on the bus and sends the command goes to the device. For HPIB devices under Unix, we open the raw bus, reset the bus, set up the bus time-out, send out an All Untalk/All Unlisten/Bus is Talker/Bus is Listener command, enable bus interrupts (for SRQs) and store the bus file ID and address. To open the device itself we serially poll the device (through the open raw HPIB file), open the device file, send Device Clear and Local Lockout commands and store away the device file ID. At this point we are ready to talk to the device. To be certain that nothing has happened to the device since we last talked to it, we poll it before we attempt to write to it or read from it.

3.6 Performance Requirements

AMS/UX has two main performance requirements: never lose input from a monitoring device and the time from receipt of the raw alarm to having the filtered and converted alarm ready to send to MIS is to be less than five seconds. Of the monitoring devices, the 3852 is the most

forgiving in that it has a large buffer for alarm data storage. If AMS is not able to read it immediately, the 3852 stores it until AMS comes to ask for it. The other devices are less forgiving because they report their status only when polled. If AMS is not able to query them before the status changes, it is lost forever. These devices, however, process less transient information that the 3852 does. At this point in time, AMS has not had a problem with data loss. It is our design goal to make sure it stays that way.

System throughput is dependent on a number of factors, many of them out of AMS' control. It is up to AMS to handle all the curves that the telecommunications center can throw at it. If there are a large number of alarms at any given time, the device interfaces will bring them all into the system and then let the filtering software work them off as time permits. The delicate balance between having the device interface software running at a high priority so as not to lose raw alarms and the need to provide CPU time to the rest of the system to process the alarms is a very delicate one. This balance will be determined during AMS/UX integration testing.

3.7 Data Alignment

Since the HP 9000/380 is based on the Motorola 68040 and the Sun-4 is based on the SPARC processor, there were a number of points at which we could have run into both inter-vendor problems and inter-host incompatibilities. They were, to a large extent, avoided by making use of Unix capabilities such as XDR for the RPC calls. XDR allowed us to transfer data between the two machines without worrying about each host's internal architecture. There were places where this was not possible (primarily in the IPC library functions) where we had to verify that each message was structured so that the messages fell on boundaries that both systems could agree on.

4.0 Summary

As this paper is being written, approximately one-third of the code has been converted and the two new pieces (display and MIS interface) are well into development (we plan to be into integration testing with MIS by the time this paper is presented). We have encountered a number of problems (generally self-inflicted), discovered a host of "features" in Pascal/1000, RTE-A, C, Unix and SunOS, and tweaked the system design any number of times. Would we change the system design given the chance? Yes. Would we change our approach given a second chance? Probably. Would we present a different schedule to our customer given a chance to do it again? You bet! Would we sign up to do this job again?

Without a doubt! To say the least, this process has been educational (to say the most, parts have been a real pain!).

5.0 References

1. Arbogast, Charles R., Telecommunications Automatic Monitoring System, Technical Proceedings, Interex 88, August 1988.

2. Weber, Warren R. and Millford, Sandra M., Accessing Multiple Shareable EMAs in a Single Program, Proceedings, Interex 90, August 1990.

UNIX for the MPE Programmer

Michael L. Barrat
ELDEC Corp.
P.O Box 100
Lynnwood, Wa. 98046-0100
(206) 743-8615

So you have MPE (or MPE/XL) operating system experience and you're about to step into the UNIX world. Are you scared? Fear of the unknown. Don't be. You know more than you think you do. You need not look at UNIX as a whole new operating system, but merely a change in the words needed to perform the things you already know how to do. This paper the translation of the commands, structures, and methods that you know and are comfortable with to their UNIX counterparts.

History

In order for you to make the transition from MPE guru to UNIX guru it is first necessary to know a bit of the history of UNIX. The history portion of UNIX is probably one of the most confusing aspects of the operating system, due to the fact that UNIX is not UNIX, is not UNIX (i.e. AT&T UNIX is not quite the same as Berkeley 4.2bsd, which is not quite the same as XENIX, which is not quite the same as OSF/1, etc.) Comparing these is like comparing hamburgers from different hamburger joints, they all have the same basic ingredients but each hamburger has its own personality. The same is true of UNIX.

Since AT&T is where UNIX originated, I'll start with their UNIX mutations. UNIX was first written at AT&T in 1969 using PDP-7 assembly language to provide a superior environment for software development. It became operational as a single-user version in 1971 within Bell Labs. Ironically, UNIX took its name from a multi-user operating system from General Electric called Multics. Unix came from "uni" which was from the single user orientation of the first version. It wasn't until much later that UNIX became multi-user. It still today does not have some of the features a good commercial system provides. An example, is the relatively simplistic spooling system that comes with UNIX.

UNIX was rewritten in C in 1973 to allow for portability. The first commercial version became available in 1980, as Version 7. Now I'm not trying to purposely confuse the issue, but the next major milestone in AT&T's UNIX history was called UNIX System III, an update of Version 7. UNIX System III became available in late 1981. To confuse things a little more, AT&T's current derivative is known as UNIX System V (don't ask me what happened to System IV). System V was introduced in January, 1983, with some additional features to make it a more viable commercial operating system.

Before I further confuse you with the history of other UNIX variations, let me provide you some historical details of MPE. Development for MPE started in 1970. Its first commercial release was in 1972. As we all know MPE is HP's proprietary operating system, but what you may not know is that when MPE/XL came out a couple of years ago, it incorporated some of the concepts of UNIX, becoming more UNIX like. Examples of these features include a much more extensive command language interpreter, providing for command files, I/O redirection (in Version 2.1), the CHGGROUP command, and dynamic expansion of file lengths when needed.

HP tells us that they are planning a POSIX interface to the MPE operating system sometime in the near future. It will be interesting to see how they will bridge some of the remaining fundamental differences between the two operating systems, such as file structure.

Some believe UNIX is relatively new, but you'll note that UNIX is actually slightly older than MPE.

UNIX Variations

While UNIX is thought to be a "standard" operating system, it seems like there are more variations of UNIX than there are other operating systems. These variations are often only slight in nature and the differences are not always obvious to the novice. However, when you start working across the variations through porting or sometimes just through the command features, the differences become obvious (and sometimes obnoxious).

Berkeley UNIX - A source code license of UNIX was obtained by the University of California at Berkeley sometime in the 1970's and one of the most popular variations emerged through student and faculty development. Berkeley UNIX (also known as 4.1bsd, 4.2bsd, etc.) was responsible for vi, the powerful, and now virtually standard text editor. Their version included support for virtual memory on the VAX for large programs and also focused on file system performance. Berkeley UNIX is more than slightly different from AT&T UNIX.

HP-UX - HP's variation, primarily based on AT&T System V with some Berkeley enhancements.

XENIX - Microsoft's version based on Version 7 and System III which are relatively older versions of AT&T UNIX.

POSIX - IEEE standard definition. The people from HP claim that POSIX is more of a interface definition to the operating system kernel rather than an entire operating system definition. HP claims it is this interface definition that they will be providing jointly with the MPE/XL interface, in the not too distant future, on the HP3000. They state that POSIX won't be built on top of MPE but will provide co-resident access to the machine. Sounds kind of like another command interpreter to me.

OSF/1 - The Open Systems Foundation variation of UNIX. Sorry, that's all I know so far.

Ultrix - Dec's variation on UNIX. Built primarily on the Berkeley version.

AIX and
IX/370 - IBM's versions of UNIX.

UNIX Shells - Command Interpreters (CI)

Unlike MPE's single command interpreter (CI), UNIX offers several standard command interpreters. The first of these is called the Bourne shell, represented by the command or process name sh (note UNIX is case sensitive, so SH is not the same as sh).

The second command interpreter is the C shell called csh. Named after the C language like commands (command files in MPE/XL) that it interprets, it was developed in the late 1970's at Berkeley with special features for C programming. Unlike the Bourne shell it contains a history stack (REDO stack) of previous commands.

The Korn shell, ksh, is a more recent shell. It is said to include some of the best features of the two other shells including the history stack.

Choosing a shell is like selecting clothes. Everyone has different tastes. The nice thing about the shells is that if you don't like the one you're in for some reason, you can change to another one easier than changing your clothes.

UNIX Philosophy

Files, commands, etc. are built more modular. Many commands are built to be hooked together as "filters" (take data in a stream from one command and pass it on to the next command after processing it). A user is permitted, even encouraged by design to add to the system. A user may customize much of their own environment relatively easily. Many commands are not operating system resident commands, but shell scripts (command files) or programs. Some of these concepts are also true in MPE, but not as integrated and some only became possible with the advent of MPE/XL.

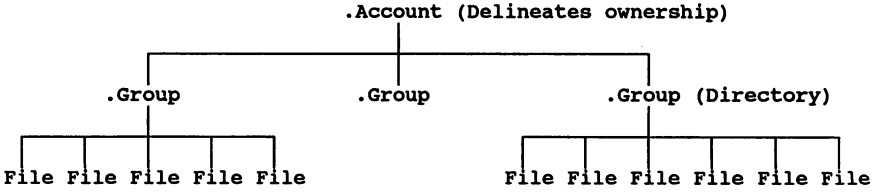
MPE vs. UNIX Directory Structure

One of the most significant differences between UNIX and MPE is the directory structure used by the two operating systems (reference FIGURE 1). UNIX offers a much more flexible directory structure. The tree structure used by UNIX is similar to a standard office paper filing system. Picture a file room (the root) full of file cabinets (the directories immediately below the

root, or first level directories). In each of these file cabinets are drawers (second level directories). In each drawer can be folders (directories) and/or papers (files). Within the folders can be more folders or papers, etc. Files and folders anywhere on the system can be owned by virtually user.

Managing files within MPE is not quite as easy. We are limited to creating only file drawers (Accounts) in which we can't directly place files without folders (Groups). We are also limited in MPE of the ownership of the groups and files within an account.

MPE's Semi-flat Structure



UNIX's Tree Structure

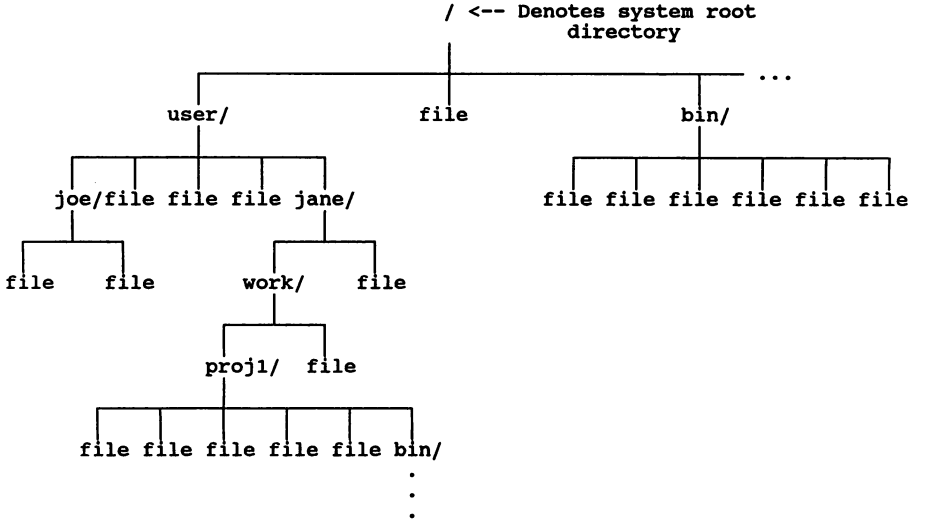


FIGURE 1

In UNIX all "files" (i.e. devices, binary files, ASCII files, etc.) are handled in one byte "blocks". This concept significantly facilitates data flow throughout UNIX because everything is handled in streams of single bytes. In MPE files are record oriented. Records in UNIX are more logical structures imposed in software while in MPE are physical. When the operating system fetches data in MPE, all of a physical record is fetched, as well as the rest of the records of the block. The result is that MPE is more efficient in retrieval of related data. UNIX is designed to be more interfaceable and modular.

File Naming

File names in UNIX are much more flexible than in UNIX. UNIX will allow file names of virtually any length although the system manager typically can opt for how many characters are significant. File name conventions in UNIX typically will use extensions to convey type. For you non-PC programmers, file types are a new concept. A file extension is typically a one to three letter suffix added to the end of the file name separated from the file name by a period. An example is, myprog.c, filename is myprog, with extension type c (for C language). Note, that the period is different than the period used in MPE which implies group and account delineation from the filename.

Standard Directory Nomenclature

Some of the standard directory names that you will see on most UNIX system structures are:

/dev - is the place for all device files. Devices are handled just like files (i.e. device names can be used wherever file names are used) and all devices typically reside in this directory). This is UNIX's method of assigning a device as a file as you would do with MPE's FILE equations.

/etc - (SYSOP.SYS) Utilities usually available only to the superuser (System Manager) reside here.

/usr - either used as the common user area or where the user home directories reside.

/lost+found - the place where files are put that are not properly linked to a specific directory when a fsck (file system check) is run. The UNIX file system is not as robust as MPE thus requiring the need for a lost+found. The internal structure of maintaining the file system with the use of pointers can come apart if the system fails while the pointers are being updated.

/temp or /tmp - a place for temporary files (TEMP) that is typically cleaned out every time the system is booted rather than at the end of your session.

/user - the typical place where user home directories reside if not placed in /usr.

/bin - is the equivalent of PUB.SYS, the directory for system supplied executable files.

/local - locally created files for common usage.

/include - include copylib files for inclusion in source code.

/lib - library functions or subroutines (LIB).

/contrib - contributed files (CSL tapes)

Following these directory naming conventions, the file names are often compounded to further specify a file. For example, a user contributed executable file would likely be found in /contrib/bin.

Tree Navigation

The following are important commands and shortcuts to help you climb (navigate), grow, and prune the tree.

pwd - list the present working directory (subset of SHOWME which tells which directory you're in)

cd - change to directory (CHGGROUP)

ls - list files and directories (LISTF or LISTDIR)

mv - short for move node, means rename (RENAME)

cp - copy command (COPY, FCOPY)

rm - remove file (PURGE)

mkdir - make directory (NEWGROUP)

rmdir - remove directory (PURGEGROUP)

. - short name for the pwd (present working directory) when used as a directory name or part of a directory name

.. - short name for the parent directory to the pwd (i.e. "cd .." would change you to the relative parent directory)

Some additional commands that are useful for a UNIX novice are:

cat file - display the contents of the file to the terminal display (FCOPY FROM=file;TO=$STDLIST)

more file - display a file with pauses every 24 lines and/or searches for specific strings and other features (PRINT in MPE/XL with additional features)

who - who is logged onto the system (SHOWJOB JOB=@S)

mail (or mailx) - a operating system provided generic mail system (quite functional).

pr file - a fancy FCOPY that gives page breaks and numbers

lp file - copy file to spooling (FILE X;DEV=LP
FCOPY FROM=file;TO=*X)

man command - HELP - on-line manuals, type "man man" (HELP HELP) for a full explanation on how to use the on-line manuals. The on-line manuals are a very nice feature of UNIX. They are identical to the paper manuals except are kept up-to-date by the tapes sent for operating system updates. Therefore, I think they're better. Besides the man command has many features to help you find what you're looking for.

Shell Metacharacters (Wildcard characters)

? - single character replacement (?)

[and] - designates range or option (i.e. abc[1-9a] indicates to include all files starting with abc followed by any number from 1 to 9 or an a)

* - anything replacement (@)

Now that you know the UNIX wildcard equivalents, be warned that UNIX's wild cards are not as protective of the user as MPE's. The command "rm *" will remove all the files in a directory without confirmation. If you intend to delete all files starting with abc and issue the command "rm abc *" instead of "rm abc*", UNIX will delete the file abc (if one exists) and then it will delete all the rest of the files in the directory. This is due to the fact that parameters in commands are always space delimited in UNIX (rather than comma delimited or in some cases space delimited) and that UNIX typically allows a command to be given multiple parameters.

Pipes and I/O Redirection

I/O redirection is the capability to redirect where the output to STDLIST is to go or what STDIN is to replaced with (typically commands from a file).

> outfile - placed at the end of a command redirects the output to outfile

>> outfile - placed at the end of a command redirects the output to be appended to outfile

< infile - placed at the end of a command takes input from infile

This is a one command JOB taking its input from infile and placing its output to outfile:

```
nohup command < infile > outfile &
```

A pipe is the mechanism used to connect the STDLIST from one command to the STDIN of another command. The symbol used is the | and an example is:

```
ls | sort | more
```

This will give a listing of the files in the current directory, sort the lines of output in ascending order, and display the lines, 24 at a time. This is the mechanism used to connect the modular "filters" of UNIX together.

I/O redirection in MPE is handled through the use of FILE equations (in MPE/XL Version 2.1 the UNIX method is copied). There is no identical MPE equivalent of a pipe that I know of.

Command Options

Command options are typically indicated in UNIX by placing a - in front of the option after the command. UNIX's commands typically have many options and there may several placed together after a command.

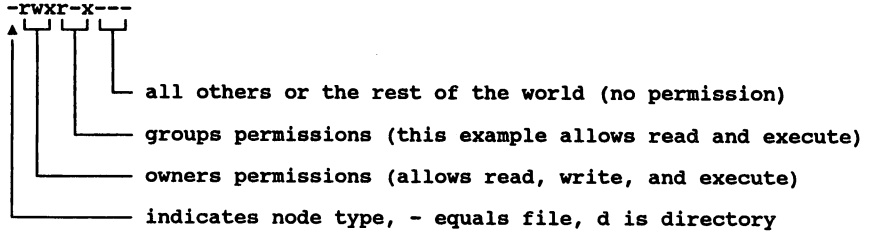
```
ls -l (LISTF ,2)

or  ls -al (two options)
```

The command "ls -l" would produce something like the following:

```
-rwxrwxrwx   1 mbarrat  corpdp     47104 Feb  6  1986 accfuncs
-rwxrwxrwx   1 mbarrat  corpdp       266 Nov 22  1989 acronyms
drwxr-x---   2 mbarrat  corpdp      1024 Oct 18 19:01 arc
-rwxrwxrwx   1 mbarrat  corpdp      1068 Aug 14  1989 artintro.doc
-rwxr-x---   1 mbarrat  corpdp       769 Aug 14  1989 charactr.ctl
drwxr-x---   2 mbarrat  corpdp      1024 Oct 18 19:01 configs
-rwxrwxrwx   1 mbarrat  corpdp     41984 Feb  6  1986 etsfuncs
-rwxrwxrwx   1 mbarrat  corpdp       247 Nov 14  1989 exclrtn.let
-rwxr-x---   1 mbarrat  corpdp       362 Aug 14  1989 exrc
-rwxrwxr-x   1 mbarrat  corpdp     53430 Aug 14  1989 fortunes
-rwxr-x---   1 mbarrat  corpdp     14231 Aug 14  1989 jal.let
drwxr-x---   2 mbarrat  corpdp      1024 Oct 26 14:21 memo
drwxr-x---   2 mbarrat  corpdp      2048 Nov 16 13:34 misc
drwxr-x---   3 mbarrat  corpdp      1024 Oct 18 19:00 source
drwxr-x---   2 mbarrat  corpdp      1024 Oct 18 19:01 xbarrat
```

Permission Bits (File Security)

Looking at the example above the first groupings of -, r, w, and x are called the permission bits. They are used to indicate the type of access permitted to files and directories. The permission bits are divided into three groupings with each grouping separated into three types of permission. The first grouping is the file owners permissions. The second is a logical grouping of users (called a group, similar to the grouping of users within an account within MPE). The third grouping is all others (i.e. those that don't qualify as the owner or a member of the listed group).



chmod - (stands for change mode) alters permission bits for an existing file or group of files (ALTSEC, RELEASE and SECURE are the closest MPE equivalents). The way that chmod works is that each permission bit grouping is composed together as an octal number and positional permissions that wish to be granted are done so by setting the corresponding bits on. The permissions for the file in the example above would be set with:

```
chmod 750 file  (owner, rwx; group; rx, all others, none)
```

umask - sets the default permission bits for files created by a user (usually set in the .profile file or .login file - logon UDC) (ALTGROUP; ACCESS=), The default umask for file above would have been:

umask 027 (note: the bits you want set are inverted (0) which is why it is a called a mask)

Examining the ls -l command output further, the column with the name mbarrat is the owner, the next column is the group, the next column indicates the number of bytes in a file or the number of blocks in a directory. The next columns indicate the file creation date and/or time. If created in the current year then the time of day is shown, otherwise the year of creation is shown. The last column is the name of the file or directory.

User Passwords

An interesting feature of UNIX system security is the capability to see everyone password (that is an encrypted password, of course). In the file /etc/passwd is the setup information for each user. For a user to change their password use the command "passwd<cr>" (ALTUSER;PASS=). The command will prompt you for what it needs to change the password.

User Processes and Process Control

How often have you issued a command in MPE and wished that you had made it a JOB since you'd now like your terminal back and you can't BREAK and ABORT the command because you've got to get the work done? Better yet how would you like to skip the need for a job altogether and just be able to start working on something else while that first command continues to run. UNIX allows just the type of control you've dreamed of through the use of background processing. Before I tell you about how background processing is done, lets look at some simple process management commands.

ps - show me all of my processes (done with SCOUT, GLANCE or equivalent).

ps -f - show me all of my processes with more detail

ps -ef - show me all processes running on the system

The following is a subset of the result of a ps -ef command showing the threading of a process tree for a single user:

```
    UID   PID  PPID  C    STIME TTY      TIME COMMAND
mbarrat 16113 15711 15 13:46:06 ttys0    0:00 ps -f
mbarrat 15711 15707  0 13:42:40 ttys0    0:01 -csh [csh]
   root 15707    94  0 13:42:38 ?        0:00 telnetd
   root    94     1  0  Nov 21  ?        8:48 /etc/inetd
   root     0     0  0  Nov 21  ?        0:06 swapper
   root     1     0  0  Nov 21  ?        2:06 init
```

This example shows the simple, typically MPE like example of a single threaded process tree in which each of the user processes has only one child. UID is user identifier (USER). PID is process ID (PIN). PPID is parent process ID. The process names are listed under the heading COMMAND. Looking at the example below, when the system came up, the first process was the swapper process (PID = 0) which fired off the init process (PID = 1), both for root (alias, superuser (system manager)). If we look at the PID and PPID values we can follow either up or down the process tree for any given process, in the case the leaf process ps -ef.

While MPE is a multi-user, multi-tasking operating system, for one user to fork (create) more than one child process in a session typically requires intrinsic calls from a program. UNIX on the other hand allows as many concurrent child processes as a user would like. From the user's foreground process (typically sh, csh, or ksh, the shell interpreters) the user fires off any additional processes into the background, and manages all the processes.

Background processing is sort of like a JOB but instead of having a separate session,background processes are spawned from the session from other processes in the same session. Within different shells are different background processing management capabilities but they all allow background processes.

The predominant way to start a process in the background is by simply adding an & at the end of the command. As in:

```
$ ls -al & > listing
3602
$
```

which will place into the file "listing" a list of files but before doing so, displays a process number (3602) to the user and then immediately returns control of the command interpreter back to the user.

There is one minor problem with the above example. If the user wants to log off the computer the background process will terminate when the session is terminated. In order to allow the process to run to completion, irrespective of the status of the parent process, the user simply needs to add the command nohup to the front of the command:

```
$ nohup ls -al & > listing      (nohup = no hang up)
3643
$
```

Now how do we get rid of an unwanted process? Simply with the use of the kill command (MPE - intrinsic kill pin). This is one command I wish MPE would inherit from UNIX at the command level.

```
$ kill 3643     (Note: For a stronger kill try kill -9 3643).
```

The UNIX vi (visual) Editor

The vi editor was developed at Berkeley. It now seems to be included as a standard feature of most versions of UNIX. As is the case with many such programs within UNIX, vi was built on top of earlier developed "modules". Vi was built upon the ex editor which, in turn, was built upon the ed editor.

Vi is invoked by typing "vi" optionally followed by one or more file names or wild carded filenames. When working within vi there are three modes (sub-environments) of operation:

command - single or multiple letter commands that cause several types of response including many types of cursor movement and placement, commands to jump into the other modes, text deletion commands, etc.

insert - like SCREEN mode of TDP (i.e. what you type is added to the file

ex - like the command mode of TDP or EDIT/where commands are primarily issued to work globally on the file.

One of the dislikes of vi is learning how to use it. Almost everyone, when first learning it, can't understand why an editor is so complex (particularly if they use editors from the PC world). All I can say is that vi was considered very powerful at the time it was developed (and still is, just difficult to learn).

One of the most difficult things about vi is knowing which of the three vi modes you are in. When you enter into vi, you enter into the command mode. In order to place characters into the edit buffer the user must enter the insert mode. If the user wishes to issue commands that are typically executed either a group of lines in the file or over the entire file, the user must issue the command in the ex mode. Sometimes the user can lose track of the mode they are in. In case that happens, the user can press the escape (Esc) key. If the user was in the insert or ex mode when the escape key was pressed, then the mode will change to the command mode. If the user was already in the command mode when the escape key was pressed then vi will just beep the terminal bell.

This concept is often confusing to the new user, so I have provided a simple flow chart in Figure 2 (that I'd wished I had) for knowing where you are when you are in VI.

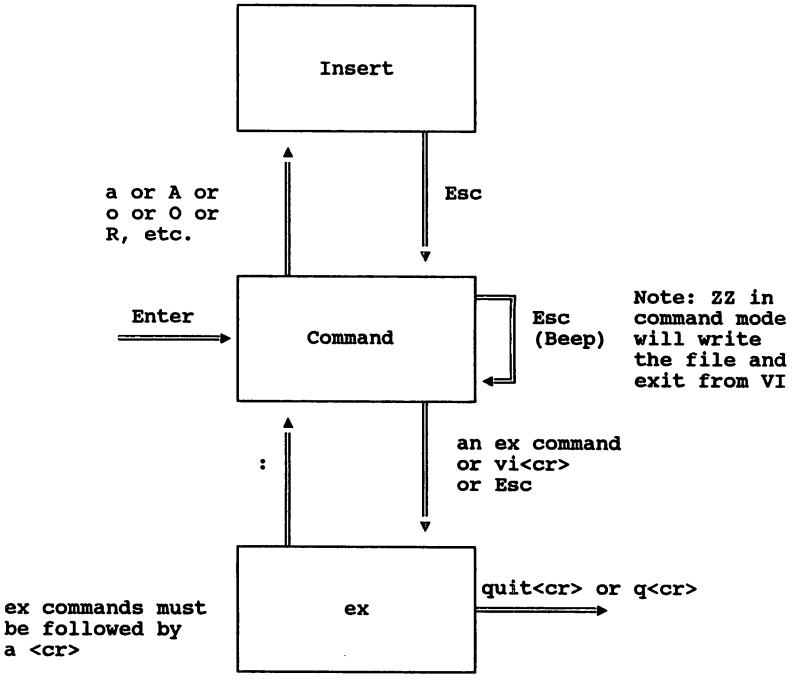


Figure 2

Compilers and Debuggers

C is the dominant language of the UNIX operating system. The operating system itself was written in C. The command for the C compiler is cc. The C compiler seems to have traditionally come with all versions of UNIX, but HP is saying they are about to stop bundling it as part of the operating system on some or all of the models. To compile the source file myprog.c (note: the .c extension is often required) a user types:

```
cc myprog.c
```

The output is placed in a file called a.out, which stands for assembler output, but the output is executable object. Why doesn't the output go into myprog.out or something equivalent? Because it doesn't. Seems kind of silly to me too.

Further options to the cc command (the same options work the same for the other compilers) allow the programmer to dump output to a chosen object file and listing file, automatically call the linker, indicate which libraries to use, etc.

```
cc myprog.c -o objectfile -llibrary
```

Other compilers are:

fc - Fortran compiler

pc - Pascal compiler

The debugger is called xdb and is also somewhat standard. In HP-UX if you call cdb (C debugger), fdb (Fortran debugger), pdb (Pascal debugger), etc. you get xdb.

Programming Tools

Some special utilities seem to have originated in the UNIX world for programmers. These utilities have become so popular that they have been copied and enhanced for other operating systems.

SCCS - Source Code Control System. SCCS is an automated configuration manager helping the programmer(s) maintain versioning control over their source code.

make - a utility that executes commands based on file age dependencies. For example, we know that typically in configuration management, if a source file has a file modified/created date/time stamp that is newer than the object file's date/time stamp, that the newer source version should be compiled to produce a newer version of object. The make utility compares file date/time stamps as specified by the user in the makefile and executes the commands (also specified in the makefile) to bring "out of sync" files into sync.

Additional UNIX Commands To Know About

One of the nice features of UNIX is the plethora of commands, utilities, and applications that come standard with the UNIX operating system. The operating system eliminates the need to purchase many utilities that might be purchased otherwise. Some of the more interesting ones are:

grep - search files for a regular expression (string pattern)

wc - word, line, and character file content counter

calendar - daily reminder service (sent through mail)

cmp - file comparator (SCOMPARE)

diff - differential file comparator (indicates which lines must be changed and how, to bring two files into agreement)

touch - set the last modified time and date of a file to the current date and time

kill - kill a process - (operating system level command equivalent of the intrinsic kill pin, desperately needed in MPE, kill -9 usually will stop a particularly stubborn process)

chsh - change your default login shell in the /etc/passwd file

sed - stream editor (a filter editor) (can take the command diff's output and create an updated version of a file)

news - prints the 'latest news' (those not previously seen by you) news items are in files found in /usr/news

stty - sets session only terminal options for your port (like SPEED with other terminal setting options)

spell - a spelling checker

sort - file sorter and or merger (SORT/MERGE 3000)

uucp - UNIX to UNIX copy, used to transfer files between UNIX systems (DSCOPY)

whereis - file locator (LISTF filename.@.@,2)

file - determines a file's type (i.e. ASCII, binary, C source, etc.). Good to use before copying a binary file to the screen.

cron - executes commands or 'jobs' at scheduled times (JOB scheduler)

at - executes commands or 'jobs' once at a specified time (STREAM)

Summary

Both UNIX and MPE have benefits that the other doesn't have. I have mentioned some of them. If you are sold on MPE, try learning about some of the things that UNIX does that MPE doesn't do, so that we can join together in getting HP to further enhance MPE with some of the niceties of UNIX. Otherwise you may try UNIX and decide to never move back to MPE despite some of the shortcomings of UNIX.

Acknowledgements

I gratefully acknowledge David White and Roger Dyckman for their comments and suggestions in support of this paper.

Bibliography

Groff, James R. and Paul N. Weinberg, Understanding UNIX - A Conceptual Guide, Que Corporation, 1983.

Holt, Wayne E., Editor, Steven Cooper, et al., Beyond RISC - An Essential Guide to Hewlett-Packard Precision Architecture, Software Research Northwest, Inc., 1988.

McGilton, Henry and Rachel Morgan, Introducing the UNIX System, McGraw-Hill Book Company, 1983.

Other Suggested References

Kernighan, Brian W. and Dennis M. Ritchie, The UNIX Programming Environment, Prentice Hall, Inc., ?.

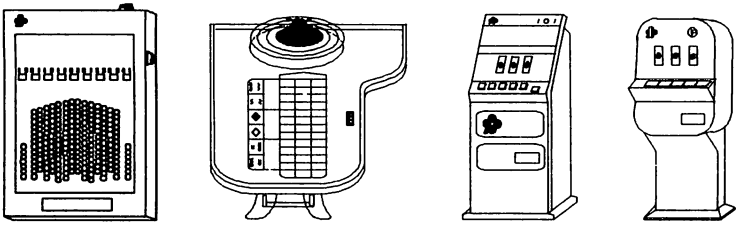
Mikes, Steven, UNIX for MS-DOS Programmers, Addison-Wesley Publishing Company, Inc., 1989.

MIGRATING FROM HP260 TO HP9000 - MIGRAINE OR NOT?

Pasi Riihilahti
Olli Lammi
The Slot Machine Association of Finland
Keilaranta 4, SF-02150 Espoo Finland
+358-0-43701

The purpose of this presentation is to shed light on the process caused by the need to move over from HP260 computer environment to some other equipment environment.

In particular we try to describe the things one should pay attention to when changing the computer brand and at the same time the operating system. This study is partly independent of which equipment is given up and which is taken into use. We therefore hope that our experiences could be utilized also in other environments as applicable.



PREFACE

The Slot Machine Association of Finland (abbreviation RAY) is an enterprise engaged in slot machine and casino game activities. RAY also manufactures slot machines, both for our own use and for export. RAY employs 1400 people; in 1990 the turnover was abt. US$ 400 million and the business profit abt. US$ 230 million. The profit is distributed to social and health organizations acting for public good.

RAY has 30 local offices and 30 smaller branch offices all over Finland. The local offices are responsible for the day-to-day game activities in their area. Computer experts employed by RAY all work in the headquarters.

Figure 1 presents the basic solutions of automatic data processing in RAY, excluding headquarters and the factory, at the beginning of 1990.

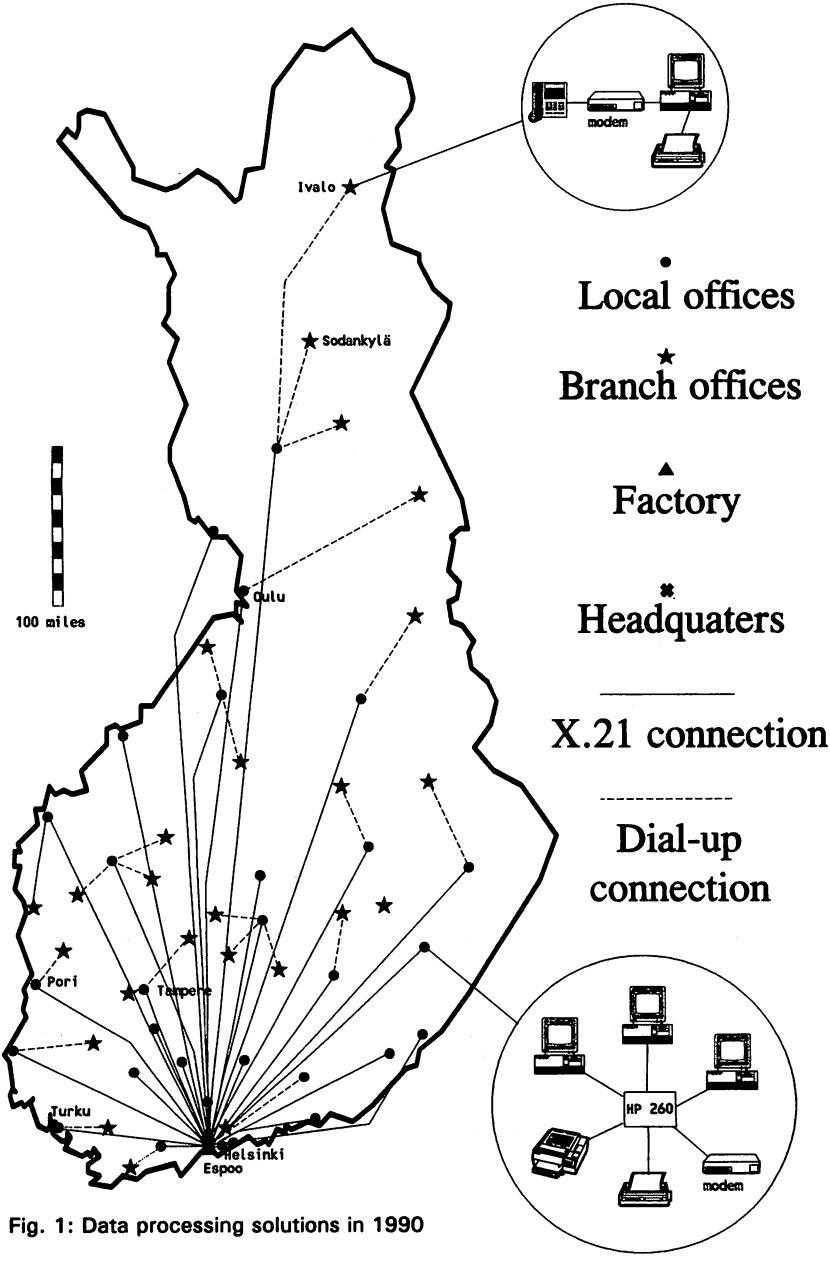


Fig. 1: Data processing solutions in 1990

The day-to-day operation of the local offices was based on data systems that were used for the operations and their control. Each local office had:

- its own HP260 equipment
- 20 data systems (maintenance, cash collection, etc.)
  the systems were self-made (abt. 10 man-years), 450 programs
- 104 000 program lines
- 4 data bases, total size 40 Mb
- x.21 and RJE as data communication links, also dial-up telephone connections were used.

NEEDS FOR AND OBJECTIVES OF DEVELOPMENT

The data systems of local offices served their purpose and the users wanted to maintain them as such. The performance and capacity of the offices' HP260 computers, on the other hand as well as their extension possibilities had already run out. More over, some new features were desired, eg. as far as data communication was concerned.

The most important objectives of the renewal process included:

1. To eliminate the performance and capacity problems.
2. To move all existing systems over to the equipment making sure the systems would operate as before.
3. To transfer all data bases.
4. To make it possible to connect the existing micro computers, terminals and printers to the new CPU so that as little new peripheral devices would be needed as possible.
5. To avoid additional data processing tasks that the users would have to perform because of the new technology. To minimize operating tasks.
6. To make the new computer system flexible so it would adapt to changes taking place in the business activities.
7. To make the new computer equipment compatible with the RAY data processing environment - architecture.
8. To make the moving over process take place fast and absolutely safely.
9. To make sure the most common application generators and CASE tools would be available for the equipment.

ALTERNATIVES

After a market study had been carried out, the following alternative solution models were considered:

1. Dedicated environments: HP3000, VAX3600
2. UNIX: HP9000, IBM RS6000, UNISYS 6000
3. SAA: IBM AS/400
4. Microcomputer environment

Correspondingly, the alternative procedures were:

- Recording of the systems using a 3GL tool
- Recording the systems using a 4GL tool
- Use of the conversion program JOIN/3000 (HP260 => HP3000)
- Use of the conversion program ELOQUENCE/9000 (JOIN/UX) (HP260 => HP9000)

For environments other than HP no appropriate conversion program was available.

The selection process progressed so that

- suppliers, equipment and their utility programs were studied
- the work amount required by the recording of the application programs was estimated
- the JOIN/3000 program was studied
- the ELOQUENCE program, which was not by that time a released product, was studied by making test conversions with limited material, by measuring the performance of the converted programs and by testing their functioning and finally, by taking part in the beta-test.
- In addition, the performance of HP9000 equipment of various models was tested.

SOLUTION

HP9000/832 was selected for the equipment, X.25/9000 and ARPA/BERKELEY for data communication products, and OpenMail, and for PCs, AdvanceMail for the electronic mail system. The conversion was made with the ELOQUENCE software.

The main criteria for the selection were the rapidity of the conversion and its reliability, the solving of compatibility problems with HP3000 and its electronic mail HPDESK as well as the strategic aim to create open systems.

We did not wish to select 4GL at the same time as the equipment. It could have endangered the entire project, at least it would have made the project longer. Recording with a 3GL tool would have taken approximately 4-5 man years and at least 1,5 calendar year.

As a result of the selection made, we got on main supplier previously known to us as well as the solution that had already been tested in practice. Owing to this testing, our own knowledge of the conversion program was good enough. The selection of the data base and the 4GL tool could be postponed.

The solution is described in figures 2 and 3.

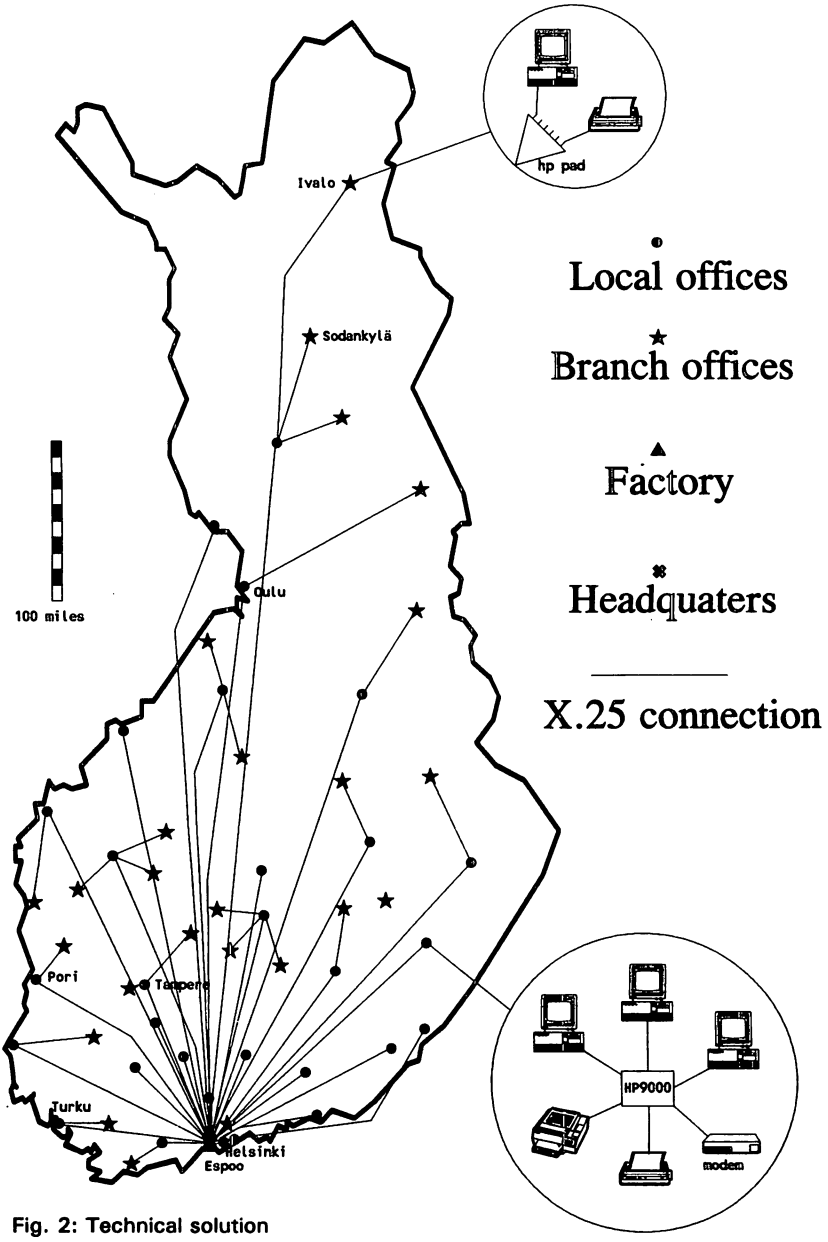


Fig. 2: Technical solution

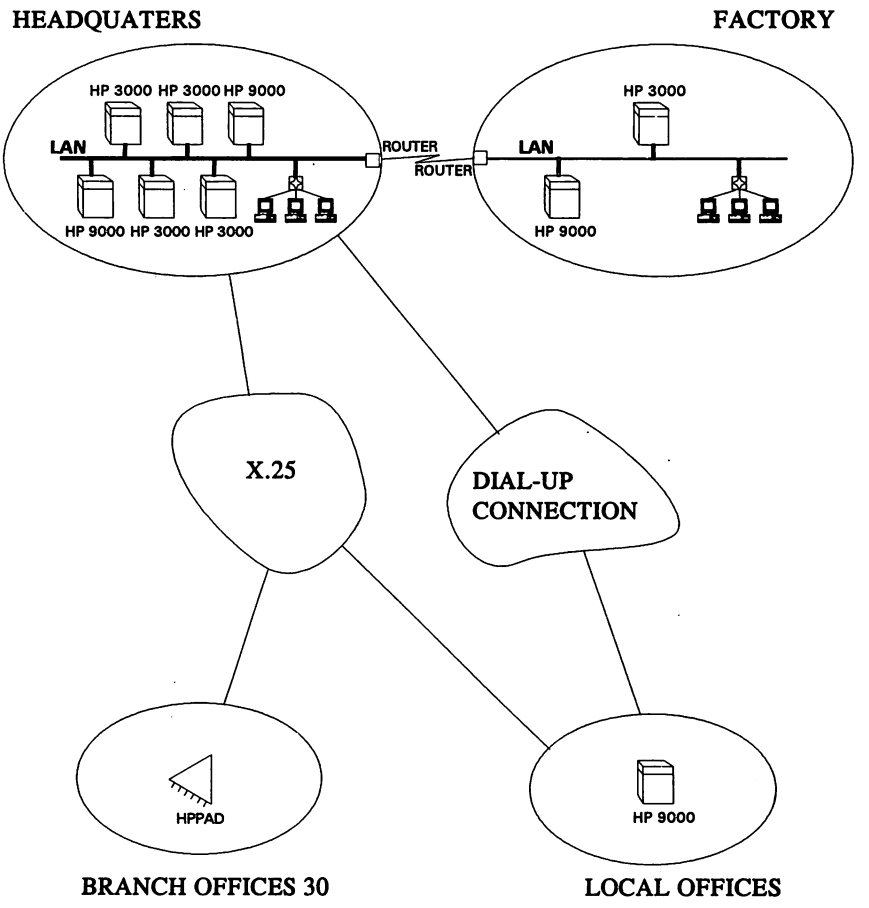


Fig. 3: Data communication solution

The solution also brought about some fears:

- is UNIX sufficiently reliable operating system, particularly in the end-user environment?
- lack of knowledge of UNIX in RAY, how to manage with a new friend.
- how to arrange data protection.
- how to fade out UNIX under the applications, how to stop the end-users from accessing the operating system level.
- tightness of schedule.

IMPLEMENTATION PROJECT

The objective (and requirement) was to replace 33 HP260 installations in the period during May and November, ie. within a period of six months. The down-time was to be at most 2 days in each office. Other requirements were listed above in chapter "Needs for and objectives of development".

The project was organized in such a way that a project control group operating under a supervision group was responsible for the practical measures. There were five subprojects. The conveying of information within the project was ensured by eg. the same person taking in several subprojects (with a different role in each of them) and by active using of electronic mail for communication. The protocols of the various subprojects, for example, were distributed to everybody via electronic mail.

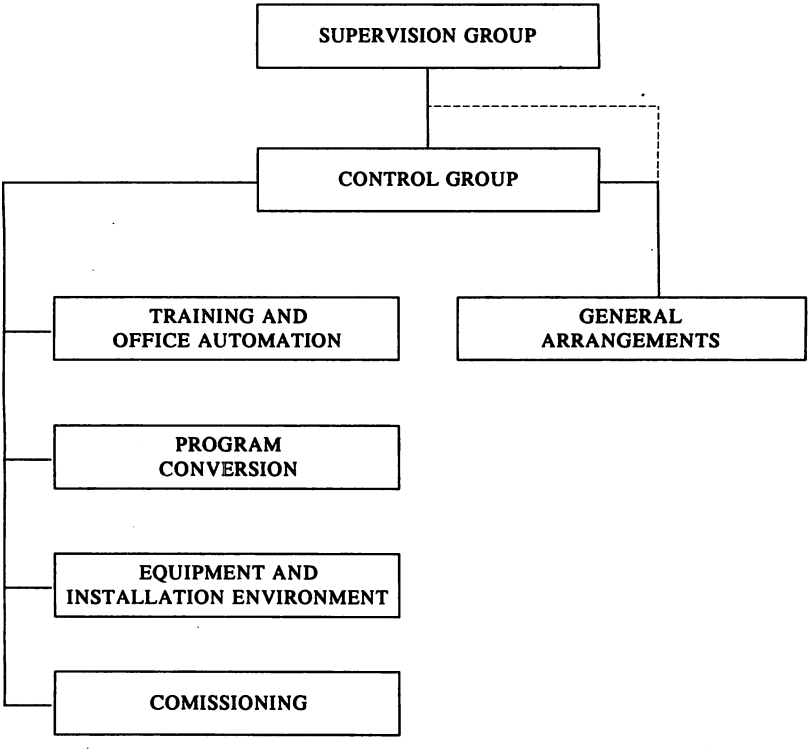


Fig. 4: Organization of the project

Figures 5 and 6 describes the task and their schedules carried out in each office.

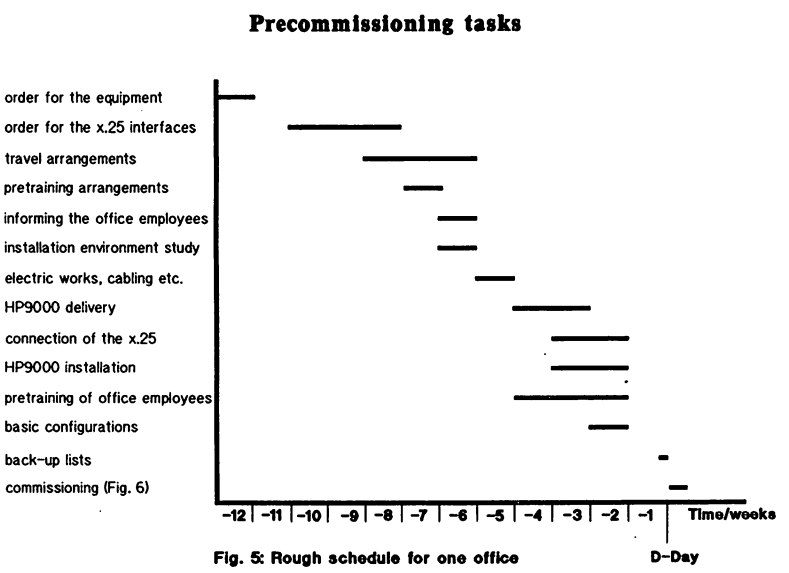


Fig. 5: Rough schedule for one office

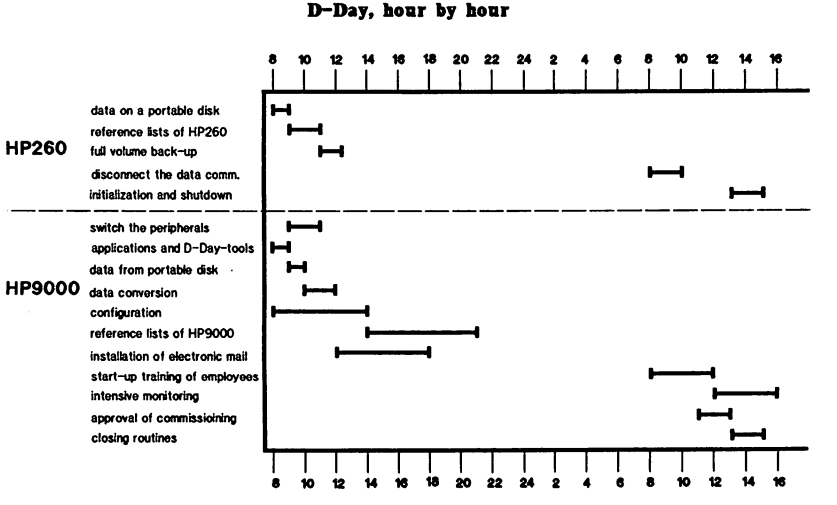


Fig. 6: Schedule for commissioning

Organization:

Supervision group

- general supervision of project
- protection of the user's interests
- processing and decision-making connected with changes made in project assignment and objectives termination of project

Control group

- executive group of subprojects
- assignments
- coordination
- schedules
- approval of plans and results

Subprojects:

Training and office automation:

Training and its planning:
- directed at:
- data processing department (UNIX, hardware, Eloquence)
- system designers
- inspection
- main users in the office
- end-users
- preparing of instructions
- pretraining separately in each area, final training in connection with commissioning

Office automation:

- planning the use of office automation
- installation of electronic mail
- changes in micro-computers

Software conversion

- definition of break.up point
- compulsory changes
- other changes
- connections with data communication project
- testing and documentation
- implementation of software conversions

Equipment and environment

- electric works, cabling, UPS
- delivery and installation of hardware and new peripheral devices
- ordering, delivery and installation of data communication equipment and related software cables, fittings
- configuration planning and preparatory measures

Commissioning:

- commissioning tools, planning and preparing them
- creating the application environment in the office computer
- connection of existing peripheral devices (possibly configuration)
- data base conversion
- implementation of operation authorizations
- making arrangements for and instructing in commissioning tests
- keeping on-duty personnel, 3000-group, operating group, settlement of accounts and others involved posted
- intensified on.duty manning

General organization

- pretesting in system development office
- sequence of procedure
- preparation of premises
- contact persons
- definition of operating authorizations (main users)
- coordination connected with arrangements
- commissioning tests and reporting of results
- acceptance of testings or correction requirements
- active follow-up measures
- communicating within the project, with offices and linkage groups

In the actual commissioning situation there were two or three computer experts in the local office to carry out the tasks connected with the commissioning. These experts were responsible for both commissioning and the training the following day. There were two of these commandos.

Once the project was well under way, HP9000 equipment was put into use in two offices every week. One spare week had been reserved in the middle of the project in case unsuccessful commissionings had to be repeated. Such did not occur so that week could be used to catch breath.

TOOLS USED

a) ELOQUENCE:

The product intended for transferring applications to the HP9000 is called ELOQUENCE. It comprises the following components:

1) programming language and development environment
2) run-time environment
3) data management
4) conversion tools

1) The development environment including the programming language corresponds almost completely to that of the HP260 equipment. Programs can be loaded, run, tested etc. in a similar way. The form files are created and maintained by means of the same programs. Some utility programs and commands have become unnecessary, either because of the dependency on the operating system or for some other reason, for example CONFIG and ROUTIL programs and RUN-ONLY command.

2) In traditional manner, the run-time environment is only intended for running applications, not for developing them.

3) Data management is based on the C-ISAM data base. Owing to the mapper (called DBMAPPER) that is implemented above C-ISAM, the data base looks to the programmer like an IMAGE/260 data base. If necessary, DBMAPPER can be by-passed and the C-ISAM base accessed directly. Also C and COBOL can be used as programming languages. Data base loadings can be made either at UNIX level by means of new tools or with DBUNLD and DBLOAD programs like before; the UNIX level is more sensible due to its superior speed.

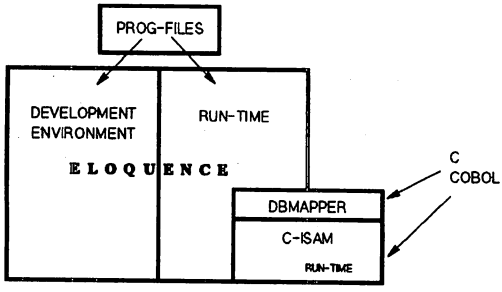


Fig. 7: Data base management

4) By means of conversion tools the files of an application can be transferred from an HP260 formatted disk or cartridge into UNIX directories. After the transfer, the forms, programs and data files can immediately be either used or maintained, depending of compatibility. The data in the data bases has to be exported into ASCII-files after which a ROOT file is created in a normal way from a schema and then further an empty data base. Finally, the data is imported into the data base.

As the way in which the programmer uses the peripheral devices is very different in HP260 environment than in UNIX, ELOQUENCE contains equipment level, group level and user level configuration files connected with these. It is possible to define which disk (HP260) corresponds to which directory (HP9000), in which port printer 11 is, etc.

b) Self-made tools:

Equipment configurations, installations of applications and data conversions were as far as possible automatized with our own shell programs etc.

RESULTS

The schedule was met for each office. Although there were some delays within certain subprojects, they did not affect the end result. The system users were satisfied with the implementation.

14 people took part in the entire project and the total working time was 430 man days.

In each office, down-time was at most one working day.

The need for training was minimized as the original and the converted application look identical. The equipment used to run the application cannot be determined on the basis of the display image. The most important issues for training were the use of login and electronic mail.

The amount of manual software conversion was smaller than expected as the more exotic features of HP260 (TIO, binary programs, etc.) were not used in the original application. The processing of secondary tasks as well as the CAT and PERFORM statements involved compulsory work. The distribution of the application into various directories, i.e. defining separate directories for data bases, programs, forms etc., was voluntary work.

Technical results:
- the performance increased 50-500 %
- the limitations of the capacity disappeared
  - the size of a program
  - the size of a data base and a data set
  - the number of transactions
- the new features
  - X.25
  - user interfaces (in the near future)

- application generators (in the near future)
- spooled printers
- etc.

DIFFICULTIES EN ROUTE

Data communication:

- Co-operation between the Finnish National Telecommunications Company and the local telephone companies was not very good and connection of these packet switching networks based on different technologies caused great problems. The plans for a closed user group had to be abandoned. All in all, data communication matters proved to be the most difficult ones.

- There was not enough time thoroughly test the terminal connections based on small branch office's dial-up telephone network or to find the right modem switches. In principle, connections were all right but in practice connections were broken or hanged. Connections were then changed to X.25-based connections as a follow-up project.

Mains supply back-up:

- If a terminal in UNIX environment is disconnected from current supply, the whole session will have been lost once supply is restored. Compared with HP260 environment in which blackouts cause no problems, this was somewhat a shock. In some parts, mains supply back-up was not arranged. Afterwards each office had to be provided with UPS devices (Uninterrupted Power Supply); it was attempted to cover as many terminals as possible with these.

ELOQUENCE:

- The ELOQUENCE licence only applies to certain equipment, determined on the basis of its serial number. HP had difficulties in delivering to us in time the right ELOQUENCE version on the right data medium. When the project had reached its mid-point, the software had been installed in the 9000 equipment by HP but although the version of this software was more recent, it did not necessarily work as well as the older version that had been tested and proven good for our purposes. We did not even get any information on the installation of the new versions. In two offices commissioning almost failed because of this but the support we got directly from the factory helped us solve the problems.

Organization:

- Generally speaking, organization was a success. For example, all "general" issues, and there are always lots of them, could be left to the general organizing group and the other groups could concentrate on their actual duties. One thing that did cause problems, however, was that the same people took part in several different subprojects and therefore these subprojects were seldom fully attended. On the other hand, this practice ensured good communication between the various subprojects.

DAT tapes:

- At the initial phase, there were such a deficiency in DAT tapes in Europe that each office could only have a few of them.

Equipment defects and deficiencies in deliveries:

- Although the schedule allowed for the inspection and testing of the delivered equipment in good time before commissioning, surprises could not be avoided completely. A defect noted in CPU, for example, was reported to HP but was not repaired in time. With respect to the number of devices, however, there were only a few defects; also, for some reason these occurred almost in a row and did not cause too much trouble.

Testing of results:

- Testing comprised comparison of outputs in the old and the new environment. Some outputs were made dependent on the physical sequence of entries in the data base (to speed up the program), and as this could not be kept unchanged in the conversion, sometimes quite great differences appeared. Outside the project the operation of the programs was made independent of the sequence.

Competence:

- Along the project, we became more competent in UNIX, ELOQUENCE and other tools, which meant that our own tools became faster and more troublefree with the time. Also, the overlapping of the various commissioning stages became more sensible. At the beginning, the first day of commissioning could extend well over midnight, whereas towards the end of the project normal working hours were enough in certain places.

- At first, we had difficulties in finding correct setting and configurations of UNIX.

Amount of work:

- Due to long geographical distances, a lot was demanded of the groups responsible for both commissioning and the equipment and the installation environment. Data communications, in particular, often did not work till the last minute, sometimes not till after commissioning.

- We had not reserved too much time for anything, on the contrary, the amount of work required for the equipment projects came as a surprise. Locations, cablings, electric supplies etc. needed working out.

- The fact that the planning phase coincided with the holiday season slowed down the progress of the subprojects.

UNEXPECTED BUT NICE!

Electronic mail:

- When completed, the OpenMail network of RAY was the second largest in the world. It opened up new possibilities that had not even been foreseen. For example, all program error messages appearing in an application can be directed to the electronic mail of the appropriate computer expert.

Packet switching network:

- In comparison with a dial-up telephone network, the packet switching network offers a troublefree, easy and flexible alternative.

SUMMARY

With just one step we have proceeded from one of the most dedicated environments into a world that in the future will give us at least some freedom to select the best possible solutions computer technology has to offer.

Looking back, the most important issues connected with the migration carried out in RAY included:

1. Mental factors.
   Firm belief in the rightness and goodness of the solution. It would be a great help if the solution really was a right and good one. A fundamental requirement for a successful process is that the faith, belief and commitment can be pushed through the organization.

2. Conversion tools.
   Nothing beats good conversion tools. They can reduce the costs, time, resources and problems of a migrating process to a fraction when compared with poor tools or manual implementation. In addition to commercially available conversion tools, also self-made tools are often required. It will be worthwhile to carefully study and learn the tools and the environment in advance.

3. Competence.
   One should have some knowledge of the new equipment environment as well as of the moving over process, eg. tools. Some persons need to have the knowledge; inside the company, the supplier, the consulting engineers. The main thing is that this knowledge is available for the migrating process. It would naturally be ideal if this knowledge could be found inside the company or it accumulates during the process.

4. Organizing of work
When migrating, right people should be in right positions, properly authorized and with a distinct division of responsibility areas. In big projects the manageability of the process and the detecting of independences create problems. By means of organizing one can ensure communication and thus, reduce the amount of errors caused by ignorance.

5. Training.
Attention shall be paid to whether the people to be trained are ready to accept and receive training. For example, if the end-user of a data system has not adapted to the change, is in the state of shock, withdrawn or angry, all training efforts will be useless. As soon as his attitude becomes one of passive acceptance or interest, training can be gradually started. People must be given time to mourn for what used to be.

6. Preparing for surprises.
One must prepare oneself and reserve resources for unaccepted situations as these will certainly occur. Data communication, configurations, compatibility of equipment etc. involve thousands of potential problems.

We feel that the migration we carried out was certainly worth it. It was an interesting challenge. And to answer the question of the heading - no nervous breakdown, no migraine, just a little headache.

DO I REALLY NEED A RDBMS?

Paper # 8066

3004 Hillegass Avenue
Berkeley, CA 94705 (800) 338-0898

I have decided to move to a Unix/C environment. As a long time Hewlett Packard customer, I will most likely choose the HP9000 as my jumping off point into this new universe. I see that the major RDBMS vendors (Oracle, Ingres, Informix, etc), have already ported their database/4GL products to the HP9000. I'm confused about how to interpret this phenomenon. Should I automatically assume the purchase of a commercial RDBMS? Are the two entities (a Unix/C software development environment and a commercial RDBMS) inseparable?

This is an important question for me, as a commercial RDBMS package can be a very high-ticket item, and will have a big effect on my budget.

ADVANTAGES OF AN RDBMS

There are many advantages associated with a high quality RDBMS. Among them are the following:

FOURTH GENERATION LANGUAGE SUPPLIED

Often a fairly sophisticated 4GL is a part of the commercial RDBMS package. Use of this 4GL by my programming staff could shorten the development cycle of my application significantly.

HIGH LEVEL INTERFACE TO MONITORING OF USER ACCESS PRIVILEGES

With a good RDBMS package, there is often a high level interface available which allows the Database Administrator to enroll and drop users of the database, give users privileges with respect to the database, and revoke or modify those privileges.

SOPHISTICATED LOCKING MECHANISMS FOR MULTI-USER APPLICATIONS

An RDBMS which supports multiple simultaneous users will have locking mechanisms in place which will reduce congestion and maximize output.

ABILITY TO MONITOR ACCESS TO DATABASE IN REALTIME

An RDBMS will often contain a user friendly, screen driven method of monitoring such activity as which users are currently accessing the database, which database tables are being accessed, and which locks are being held.

USER AUDIT TRAIL

By using various configurations of auditing, the database administrator can monitor user activity on the RDBMS. Among the activities which can be audited are such events as successful or unsuccessful attempts to log on or log off the database and successful or unsuccessful attempts to give or take away access privileges.

USER FRIENDLY MANAGEMENT OF FILE GROWTH

The RDBMS will often offer the capability to add file extents to tables in realtime, as needed. Until a maximum number of extents' parameter has been reached, file growth will be managed by the RDBMS, without the need for database administrator intervention.

AUTOMATIC LINKED-LIST MANAGEMENT

This feature is key. For indexed retrieval, linked list maintenance is automatically achieved for you by the RDBMS. This means that every time a record is added, updated, or deleted, any and all index pointers are managed by the database package itself, without programmer intervention.

MANIPULATION OF PHYSICAL DATA STORAGE WITH AN EYE TOWARD RUNTIME PERFORMANCE ENHANCEMENT

The number of actual disc accesses required relates directly to runtime performance for any application which reads and/or writes data with frequency. An RDBMS will often have features which allow high level interface to the actual physical manipulation of data on the disc in order to enhance runtime performance.

MANAGEMENT OF ACTUAL DISC WRITES WITH AN EYE TOWARD DATA CONSISTANCY AS WELL AS RUNTIME PERFORMANCE

An RDBMS may have a feature commonly know as Commit, which allows the programmer to manage programmatically when data is actually written to disc. This feature will ensure that a logical transaction is posted, rather than simply a physical transaction. For example, if my application requires a decrement to an inventory record each time a particular part is sold, I can write my program such that I do no actual write to the database until both the procedures for the update to the customer invoice to represent the sale to him of the part and the procedures which decrement the parts inventory dataset are completed. The RDBMS may also have a Rollback feature, which supports "unwriting" any data which may have been written to the database if a system crash occurred during the middle of the posting of a logical transaction.

The Commit feature can also be used as a runtime performance feature. I can programmatically hold off actual I/O to the disc until a substantial amount of data is ready to be posted. This feature is particularly useful if I am writing data which is physically contiguous on the disc.

ABILITY TO ADD AND DELETE INDEXES AND FIELDS UPON DEMAND

This feature is central to an RDBMS. Gone are the days of being locked into one database design, and one set of keys which cannot be modified without taking down the database, unloading the data, rebuilding the database, reloading the data, and recreating all indexes (not just those which might be new). With an RDBMS, a new field and/or a new index can be added and propagated with values without disturbing the integrity of the database.

DATA DICTIONARY

A very useful feature, the data dictionary can be utilized as a central repository for all data definitions. The dictionary will be interrogated at compile time by all programs in the application, and will enforce data consistency throughout the software design and development phase of the system life cycle.

DISADVANTAGES OF AN RDBMS:

If I decide to go with a commercial RDBMS, I will gain all advantages inherent in the product. However, there will also be negative baggage included, which I should take into consideration prior to making my final decision about the purchase.

PRICE

A good RDBMS can be very expensive. Remember that, even if you are starting out small on an inexpensive hardware platform, if you expect to grow and will want to expand to a larger, more powerful platform, you must eventually purchase the RDBMS for that new platform as well (unless you want to rewrite your application yet again). And, even if the RDBMS is inexpensive on the first platform you are using, it may well be very costly on the new, more powerful hardware.

PERFORMANCE

Examine those advantages of an RDBMS discussed above. They are all powerful features, affording a great deal of user- friendly interface. With many a commercial RDBMS, what you have is a very large piece of code which is (by definition) built to be as general as possible, encompassing the needs of as large a market of software developers as possible. (After all, the commercial RDBMS developer is, like any software developer, interested in the largest portion of market share he can capture). Rest assured that this is not a situation which will optimize performance for your particular application.

PORTING LAG

Each commercial RDBMS must be ported to each piece of hardware upon which it will run. This porting process can be a complex or a trivial process, but it must be done by the RDBMS developer himself. You have no control over whether or when the porting will take place. Although the RDBMS may already

be ported to the HP9000 (the hardware environment you may choose), take care to examine whether or not porting is in place or is planned for the hardware which is a part of your overall growth plan.

REALITY

Whether I purchase a commercial RDBMS or not, I will still need:

SOPHISTICATED DATABASE DESIGN

How many of us have seen an instance where a poor database design has led to poor runtime performance? Let's face it. Whether you go with an RDBMS or not, you will need a good database design for your rewrite, and that takes expertise, and that is expensive.

SOPHISTICATED PROGRAMMERS

It's been my experience that, no matter how sophisticated and user friendly the 4GL that comes with the commercial RDBMS, you invariably need good 3GL programmers to complete the job. Your users' requirements will not often jive exactly with the capabilities available in the 4GL, and you will find yourself escaping to the third generation language in order to satisfy certain of your users' needs. And don't kid yourself, a 4GL is no good in the hands of an uneducated user! Without a serious degree of programming experience and sophistication, writing an application in a 4GL can lead to abysmal performance. So, even if I go RDBMS/4GL, I will need good 3GL programmers.

In addition, if you will have a multiuser system, locking strategies within the 4GL will require a substantial degree of programmer sophistication.

SOPHISTICATED DATABASE ADMINISTRATION

Any RDBMS shop will require a talented database administrator. (Someone will need to manage the database using all those friendly features supplied by the product). And don't assume that a skilled database administrator will not be needed even if you don't go with a commercial RDBMS!

CAN I GET SOME/ALL OF THE RDBMS ADVANTAGES VIA WHAT IS OFFERED TO ME IN VANILLA UNIX/C?

FOURTH GENERATION LANGUAGE SUPPLIED

There will be no 4GL supplied in a standard Unix/C environment. However, there are a substantial number of low-level subroutines available to the C programmer, either through the standard C Library, or via the Unix kernel, which can greatly simplify the C programmer's task.

HIGH LEVEL INTERFACE TO MONITORING OF USER ACCESS PRIVILEGES

Vanilla Unix offers a standard set of commands which allow the systems administrator to assign and modify user file and program access privileges. The ease of use of these commands will depend upon the the friendliness of the systems administration interface supplied with the Unix you choose to purchase. Of course, you will also generally have the option of the purchase of a third party user interface product as well.

SOPHISTICATED LOCKING MECHANISMS FOR MULTI-USER APPLICATIONS

Unix/C offers a sophisticated set of locking mechanisms for multi-user applications. In a multi-user environment, use of these mechanisms requires a reasonably high level of programmer sophistication (as does the use of the mechanisms available via a 4GL supplied with an RDBMS). Keep in mind that, if you don't foresee the need for simultaneous multi- user access to data, you won't need this level of sophistication, with or without an RDBMS.

ABILITY TO MONITOR ACCESS TO DATABASE IN REALTIME

There are vanilla Unix operating system commands which allow monitoring of programs and/or resources in realtime. Ease of use of these commands will depend upon the degree of user friendly interface built in to the Unix you choose to purchase. Here again, you will generally have the option of the purchase of a third party user interface product.

USER AUDIT TRAIL

Vanilla Unix does not offer a standard user audit trail utility. However, a if you choose a Unix which is enhanced to meet the requirements of the C2 Class of Trust as defined by the Department of Defense Trusted Computer System Evaluation Criteria, (also known as the Orange Book), you will have available to you a sophisticated auditing feature which will allow various levels of tracing of user activity on the system.

USER FRIENDLY MANAGEMENT OF FILE GROWTH

File growth is managed automatically by the Unix operating system in realtime. There is no standard user interface in Unix to allow user management of file growth.

AUTOMATIC LINKED-LIST MANAGEMENT

This you won't get in standard Unix/C. However, there are relatively inexpensive third party software products on the market (directed toward the needs of software developers) which will work with standard Unix and which will offer subroutines to accomplish linked-list management.

MANIPULATION OF PHYSICAL DATA STORAGE WITH AN EYE TOWARD RUNTIME PERFORMANCE ENHANCEMENT

The features available in vanilla Unix toward this end consist primarily of the system administration capabilities of backup and restore of a Unix filesystem, which will reduce file fragmentation, and will subsequently boost performance.

MANAGEMENT OF ACTUAL DISC WRITES WITH AN EYE TOWARD DATA CONSISTANCY AS WELL AS RUNTIME PERFORMANCE

This task can be managed programmatically in the Unix/C environment, as in any programming environment.

ABILITY TO ADD AND DELETE INDEXES AND FIELDS UPON DEMAND

This feature is not available in a vanilla Unix/C environment. However, if your application data needs are fairly stable in terms of field and retrieval capabilities, you can live without this very attractive feature of the RDBMS. If your data field and data retrieval needs are not stable, you might want to examine various possibilities of batch processing of data and placing it into summary data structures to allow you new and/or extended reporting capabilities. If your reports do not require up-to-the- minute accuracy, a new or expanded summary data structure and summary batch algorithm may sufficiently satisfy your moving requirements.

DATA DICTIONARY

In vanilla Unix/C, there is no data dictionary as such. However, the concept of the C Include File supports the sharing of common data structures by every programmer who wishes to include the file in his program. (The C Include File operates along the same lines as the tried and true Cobol Copylib, allowing data definitions to be shared by everyone).

IN SUMMARY.....

There are many real advantages to a good commercial RDBMS, and there are some serious disadvantages too.

Many features of a good RDBMS are available to you via vanilla Unix/C. Some features of the RDBMS can be enhancements to vanilla Unix/C specific to the flavor of Unix which you have purchased. Other features of the commercial RDBMS (particularly user friendly system administration features) may be available to you via third party software products.

For those features which are just plain not available in an environment without an RDBMS (e.g.,. adding and deleting fields and indexes upon demand), various programming strategies (for example, batch runs of data summaries which run on a periodic basis and are then deposited into a small summary 'reporting' database), may be sufficient for the needs of your particular application.

Although use of an RDBMS (and it's accompanying 4GL) can require a smaller and less complex programming effort and and less technical expertise than programming in straight Unix/C, in either case

you will need a highly technical programming staff and systems administrator to accomplish your development goals.

INTEREX '91 - Paper #8073

From MPE XL To HP-UX & Back Again: Life With Open Systems

by
Gary Lowell

Allegro Consultants, Inc.
2101 Woodside Road
Redwood City, CA 94062
UUNET: glowell@allegro.com
Voice: (415) 369-2303
Fax: (415) 369-2304

1. Introduction

With the increasing interest in Open Systems many HP 3000 shops are acquiring their first UNIX machines, usually an HP 9000 running HP-UX, HP's variant of UNIX. The reasons are varied, ranging from long term strategic directions, to a need for a specific application that presently is available only under UNIX, to just curiosity. Software developers are increasingly finding customers want their HP 3000 application, but they want it on a UNIX platform.

This paper is about living in a mixed shop. The emphasis is on migrating HP 3000 commercial applications to an open system such as HP-UX, since this type of migration is the most common. We will also look at porting software from UNIX to MPE XL, as well as cross development.

To establish background for the discussion to follow, we will briefly look at Open Systems and standards, as well as differences between MPE and UNIX.

2. Open Systems

Open systems are a result of the maturing of the data processing industry. The beginning of open system standards was the AT&T System V Interface Definition, (SVID). AT&T required as part of the license terms, that any computer manufacturer that ported UNIX to their computer comply with the SVID. This meant that no matter who you bought a UNIX computer from, you could expect that there would be a common set of system calls and utilities that would be the same on any other UNIX computer. Each vendor also made proprietary extensions, the most common extensions being to incorporate features from the Berkeley version of UNIX.

There are now a number of non-proprietary overlapping standards that define what an open system should look like. At the bottom of the hierarchy are standards that define specific elements of the computing environment:

ANSI C	The C language and libraries
IEEE POSIX 1003.1	System Calls
IEEE POSIX 1003.2	Commands and utilities
OSI	Networking

There are a number of POSIX standards for more specialized areas such as real-time processing.

X/Open is a consortium of computer manufacturers that develops specifications for the entire computing environment by selecting and building upon specific standards. This specification is published as the X/Open Portability Guide, or XPG.

3. Comparison of MPE XL and UNIX

There have already been several articles published in INTERACT magazine comparing MPE XL and UNIX. Most find that UNIX has significant features in favor of the programmer, and that MPE XL has significant features in favor of production data processing. I would like to briefly review the differences concentrating on areas that affect software portability.

3.1 File System

The difference that is most readily apparent is the file system. MPE XL supports eight character file names within a two level hierarchy: account and group. HP-UX supports indefinite levels of hierarchy, with no limit on file name, except that the total path (filename and all intermediate directories) can not exceed 256 characters (some earlier versions of HP-UX had a 14 character limit on the filename portion). MPE XL places further restrictions on access of files between accounts, the most onerous being the inability to create a file in another account without privileged mode (or being the system manager).

MPE XL imposes structure upon files. MPE files have a number of attributes that affect how one uses them. HP-UX files are uniform. Any structure is imposed by the application that is accessing the file. This means that programs can do simple things to files, simply. A program to copy an arbitrary file is only a few lines long. In contrast, a similar program on MPE has to know about all of the various types of files and have a special case for each. Indeed, privileged mode is required to copy some types of files on MPE.

Access rights to files are handled similarly by both operating systems. Files have an owner and belong to a group. Access rights can be granted to the group, or to any user. XL access rights are more "fine grained" than in HP-UX, consisting of: Read, Write, Append, Lock, and Execute. HP-UX access rights consist of just Read, Write, and Execute. However, the HP-UX group is a more general mechanism than XL's groups, an HP-UX user may belong to more than one group, and have the associated group access rights.

The method the operating systems use to manage disk space is very different, not to mention their disagreement over the spelling of disk (disc). Ignoring private volumes, MPE XL file system encompasses all the active disk drives on the system. On HP-UX there can be multiple file systems, but each file system is restricted to a single volume, and can not be larger than 1 gigabyte. This restriction puts constraints on the size of a database, although third party Relational DataBase Management System (RDBMS) vendors work around this constraint by managing the raw disk space themselves.

The capability of a UNIX file system to encompass multiple physical disk volumes is provided in other versions of UNIX, and will likely soon be present in HP-UX. HP-UX's equivalent of MPE XL's private volumes is much easier to create and use. By using the Networked File System (NFS) it can be accessed by multiple computers.

The MPE XL file system also provides file equations; file equations can provide independence between an application and the real file name. Under HP-UX, the environment variable provides for a similar capability, but the application program must explicitly use the environment variable.

3.2 Utilities

MPE XL has a relatively small set of system utilities. These utilities often have inconsistent command syntax, and are not designed to work together. In contrast HP-UX has a very rich set of utilities with consistent syntax, that are designed to work together. This makes it possible to accomplish tasks under HP-UX by stringing together several utilities, that under MPE would require writing a program.

3.3 Job Control

MPE XL supports batch data processing with the ability to assign execution priorities, and to limit the number of batch jobs that may execute simultaneously. In contrast, HP-UX does not really support the concept of batch data processing. HP-UX does support executing a command in the background, or scheduling it to execute at some future time. HP-UX execution priorities are often ineffectual, and there is no way to limit the number of background processes. HP-UX does have the ability to specify that a process started from a terminal should continue to execute after the session logs off, or more likely, the modem drops the line.

3.4 Output Spooling

MPE XL has a robust print spooler that supports multiple print queues, priorities, and print job restart. HP-UX print spooling does not support priorities, and often will not detect exception conditions such as paper jam, which can cause a print job to be lost. HP-UX does have superior support for networked printers.

3.5 Command Files

MPE XL uses the same command language for interactive users as for batch jobs. So, too, does HP-UX. Since its introduction, the command language in MPE XL has seen increasing functionality, inspired by some of the features of UNIX. There are minor variations to the MPE XL command language depending on whether the commands are in a batch job, user defined command, or command file.

HP-UX command language is handled by a program called the shell. There are three different shells in common usage. The usage of the command language is uniform between interactive, background, or command files. The passing of command line parameters into shell scripts follows the same conventions as for programs.

3.6 Networking

While MPE XL networking is built upon a standard transport layer, the programming interface is still proprietary to HP. MPE networking is also relatively expensive, so it's distribution is therefore limited.

HP-UX networking has a standard programming interface: Berkeley or ARPA sockets. Sockets are accessed with some of the same system calls as a disk file, making it a very generalized facility. The low cost of UNIX networking, in some cases built-in, makes UNIX networking widely available.

HP-UX supports the Networked File System (NFS), which allows files to be shared transparently over a network.

3.7 Languages

We will only cover three languages common to commercial applications in this paper, COBOL, C and SPL.

There is no COBOL compiler common to both MPE XL and HP-UX. MPE XL has the HP COBOL II compiler, which generates quite good native code. HP-UX has the MicroFocus COBOL compiler which generates unreadable C code, which is then compiled into native code. Past releases of the MicroFocus COBOL compiler were very difficult to use. Many of these problems have been solved in the latest release of the compiler, and an upcoming release will support native code generation. Despite pleas from users, HP has indicated no interest in making their COBOL-II compiler available on HP-UX.

There is a very good ANSI C compiler for both MPE XL and HP-UX. The C library for the MPE version of the compiler does a very good job of hiding XL specific features.

SPL is supported on MPE XL in compatibility mode only. Native mode support is available via the SPLash! compiler from Software Research Northwest. Software Research Northwest has been using the SPLash! compiler to generate HP-UX code as well.

3.8 Database

MPE XL has two databases from HP (TurboIMAGE and ALLBASE/SQL), and has several third party RDBMSs available. TurboIMAGE currently offers the best performance, and has the largest installed base of applications on MPE. RDBMS are rapidly catching up in performance.

HP-UX supports ALLBASE/SQL, and a larger number of third party RDBMS. At present Ingres, Oracle, and Sybase are supported on both MPE XL and HP-UX.

Most RDBMSs include a fourth generation programming language, and a complete application environment, including user interface. While the third party RDBMS are available on a wide range of computers, they have many proprietary features. It is possible to become locked into an RDBMS just as it is possible to become locked into a proprietary operating system.

There is a standard underway for Open SQL, which would ensure a common core functionality in all complying RDBMSs.

Some of the features in IMAGE that are missing in RDBMSs are:

- chronological access to records
- use of status areas
- directed reads
- locking (item level locking doesn't exist, RDBMSs have implied locking based on transactions)

Of course, RDBMSs have features not available in IMAGE.

3.9 User Interface

The most common UI for MPE XL is V/PLUS, although many software developers have built their own character mode UI as well. V/PLUS supports only HP Block Mode terminals. Some thrid-party UIs are available such as WINGS from Software Research Northwest.

HP-UX supports both the XII motif graphical user interface and a terminal independent UI called "Curses," which is now a POSIX and X/Open standard. There are also a number of third-party UI packages, including WINGS from SRN and JAM and JYACC. Additionally, many RDBMS packages offer user interface tools (including three RDBMSs offered on both MPE and HP-UX).

User interfaces built using character mode I/O are generally considered to be more user friendly than block mode user interfaces, since the user receives feedback immediately, rather than after he has filled in an entire form.

MPE XL does not support character mode I/O, particularly the single character read, as well as it does block mode reads. Character mode user interfaces that rely too heavily on single character reads can severely degrade system performance. Without Typeahead turned on, MPE XL can lose characters while reading data one character at a time.

HP-UX buffers character data between terminal and application program in such a way that single character reads can be handled efficiently.

There is slightly more overload incurred by using the CPU to do buffer handling, as opposed to doing it in a terminal controller like the DTC used by MPE XL.

3.10 Security

MPE XL has fairly robust security designed in. MPE XL has several capabilities that can be assigned to users. The most significant of these are Privileged Mode, System Manager, and Operator capabilities. There are a number of tasks that the System Administrator needs to perform on a regular basis to maintain security of a system. There are third party packages that will assist with these tasks, such as EnGarde form InCase Corporation.

UNIX has an undeserved reputation for poor security. This has probably come about because UNIX tends to be employed in environments where security is not a concern, or is even considered to inhibit one's productivity, more than from any architectural limitations. As with MPE XL, there are a number of tasks that the system administrator needs to perform on a regular basis. Many of these tasks have direct analogs in MPE XL. There are packages that will do security audits on HP-UX systems.

Both MPE and HP-UX have extensions that allow them to meet Department of Defense (DoD) level C2 security, but neither has as yet gone through the formal certification process.

3.11 System Management

System Management under MPE XL is performed using commands that are built into the command interpreter, and with a couple of programs for building configurations files: SYSGEN for system configuration and NMMGR for Networks, and Terminals.

HP-UX system administration is performed using a variety of programs and by editing various shell scripts and configuration files. Configuration files under UNIX are readable ASCII files, rather than encoded binary files.

HP-UX provides a menu driven System Administration Tool called SAM. At present, SAM does not handle all of the necessary tasks, and even the most recent release has been buggy. This tool will eventually make system administration of HP-UX less intimidating. There is a POSIX committee developing standards for system administration.

3.12 Shared Devices

The most common shared device is the magnetic tape drive. MPE XL has facilities to control access to shared devices and to ensure that two programs don't write on a tape at the same time. These facilities are lacking in HP-UX. File system security can be used to specify which users can access a shared device, but there is nothing to determine which user gets access.

4. Application Design Issues

In this section we will look at differences between MPE XL and HP-UX that influence program design. Some these difference were described above, others are due to differences in the performance of various operations on the two systems.

4.1 File System

Under MPE XL, applications tend to be squeezed into one account. This often results in file names being cryptically encoded into eight characters.

HP-UX applications tend to have complex directory structures that are a reflection of the structure of the application. Since the directory structure is hierarchical, an entire subtree can be replicated for test purposes, or to have multiple production directories. The application programs can determine the root of the application directory at run time from an environment variable.

The creation of files under HP-UX does not have as much overhead as with MPE XL, so HP-UX applications tend to create large numbers of small disk files. This occurs most frequently with C language program source, where each group of related procedures will be put in a source file, as opposed to one monolithic file.

4.2 Process Management

Under MPE XL, a new process creation has a lot of associated overhead. Applications tend to create a process only when absolutely needed, and once created, tend to keep the process alive as long as possible.

HP-UX process creation is a very low overhead operation. One of the benchmark metrics used to evaluate UNIX performance is how many thousands of processes can be created per second. UNIX applications create processes not only when needed, but as a generalized programming technique.

An HP-UX process inherits the parents environment, including all open files and environment variables.

4.3 Interprocess Communication

MPE XL has several InterProcess Communication (IPC) mechanisms. There are message files, memory mapped files, NetIPC, and the PORTS facility. The PORTS facility requires the use of Privileged Mode, and HP has made a subset of it available through the Architected Interface Facility (AIF). All of these mechanisms have been used to create applications.

HP-UX supports sockets, which can be considered a generalized IPC mechanism as well as being the programming interface to Networking. HP-UX applications that use this IPC mechanism will often work in a network environment as well.

HP-UX also offer FIFOs, which have slightly different semantics than the MPE XL message file. HP-UX also has a Message Queue mechanism, and shared memory. HP-UX shared memory is a little closer in concept to MPE V extra data segments than to memory mapped files. Memory mapped files similar to HP-UX are available in other implementations of UNIX.

HP-UX provides a semaphore facility for the coordination of multiple concurrent processes. MPE XL has a similar semaphore facility internally, but HP does not make it available to the application programmer.

4.4 Scalability

Our observations of applications on a number of systems have led us to believe that MPE XL applications scale to larger processors fairly well.

HP-UX, on the other hand, is more machine efficient on smaller systems than MPE XL, but doesn't scale as well to larger CPUs. As one adds more users, even on a high end CPU, one tends to encounter limits in system configuration, or the application architecture, or in HP-UX itself, that limit full utilization of the CPU.

4.5 Client/Server

Client/Server architecture is really a formalization of a common sense design principle. MPE XL has supported this type of design in the past with message files. The AIF Ports Facility now provides a superior mechanism, and one that more closely matches the UNIX model. MPE XL is missing the combination of IPC and Network interfaces that makes distributed applications so easy to do under HP-UX.

HP has indicated it will be implementing distributed computing standards both in HP-UX and MPE XL.

5. Migration Issues MPE XL to HP-UX

This section will briefly discuss some migration issues including languages (COBOL, SPL), databases (TurboIMAGE), user interface (V/PLUS and character mode), command language, environment, third party tools, and performance considerations.

5.1 COBOL and SPL

HP COBOL II can be mechanically translated to Micro Focus COBOL with an editor script. Unfortunately, after doing this translation and getting a clean compile you still don't have an executable program. In a typical HP COBOL II program, there will be calls to V/PLUS, IMAGE, and other intrinsics that are not in HP-UX. We will look at how to handle these later.

One pitfall to watch for is mismatches between the COBOL FD statement and actual file width in bytes. It is common for programmers to ignore these messages when they occur. MPE, since it's I/O is record oriented, will truncate or pad the record as needed, resulting in the correct operation of the program. HP-UX, which is byte stream oriented, will read exactly the number of bytes in the FD statement.

SPL is often used in commercial applications for special purpose subroutines, or for interfaces to operating system facilities. In general, these will have to be re-written in C. However, there may be an alternative: Software Research Northwest has the SPLash! compiler for HP-UX.

5.2 DATABASE

If your application currently uses an RDBMS, then database migration will be less of a problem. However, the vast majority of MPE applications use IMAGE for their database. IMAGE can be fairly readily mapped onto a relational database, and the IMAGE calls mapped into SQL queries.

Depending on how the application is structured, the mapping can be trivial,

RDBMS	IMAGE
Table	Dataset
Row	Record
Column	Field

or it can be difficult, requiring the addition of supporting information in the database and supplemental access code. Some of the common IMAGE translation problems are:

a) use of information in the IMAGE status area
b) chronological access of records on a chain
c) directed reads
d) item level locking

Since most RDBMSs use implied locking based on transactions, porting spaghetti code is difficult. If the application has put all database access neatly between DB Begins and DB Ends, there usually is no problem.

Performance of the migrated applications is not as good as the application running under IMAGE, but it is not as bad as RDBMS' performance is rumored to be.

5.3 V/PLUS

One of the most common user interfaces for MPE is V/PLUS, which works only with HP block mode terminals. There is no support for block mode under HP-UX. There are, however, two third party packages that map V/PLUS calls into Curses: WINGS from Software Research Northwest, and Transport from Bitech.

5.4 Character Mode U/I

A common alternative to V/PLUS for MPE applications is character mode interfaces. A number of applications have built their own character mode user interface. These are very easy to convert to Curses, and one has the advantage of terminal independence provided by Curses.

5.5 Command Language

Most applications have a number of batch jobs using MPE commands. These batch jobs have to be translated into shell commands. The Bitech package, Transport, has the capability of interpreting a limited number of MPE commands.

5.6 Environment

We have noted above that HP-UX lacks facilities for batch processing and print spooling that many MPE applications expect. Some of the features of these facilities can be emulated with shell scripts. HP is offering products to make HP-UX more attractive to Commercial Applications. Some of these are:

a) HP Open Spool (for print spooling)
b) HP Omniback (for system backup)

There are also third party packages that address the issues of batch processing, print spooling, tape library management, and system administration.

5.7 Third Party Tools

There are several third party tools that assist in migration from MPE XL to HP-UX.

Software Research Northwest has a set of migration tools that they use internally for customer migrations. Two of these tools are offered as products:

> WINGS is a character mode user interface that supports windowing and pull-down menus. It runs on MPE XL, HP-UX, and MS-DOS. It is built upon the Curses standard interface, and provides for emulation of V/PLUS.
>
> SPL/UX is a compiler for SPL for HP-UX.

Bitech has a migration product called Transport that addresses many of the issues raised above:

- IMAGE to RDBMS migration
- V/PLUS emulation
- MPE like environment on HP-UX

Pantechnic's Data/1 product provides for migration from IMAGE to ALLBASE on MPE. It would be useful in a phased migration approach where the first step would be migration to an RDBMS on MPE XL.

5.8 Performance

The current release of MicroFocus COBOL has poor performance compared with COBOL II on the HP 3000. The upcoming release of MicroFocus COBOL 2 with native code generation should provide increased performance, and reduced executable size.

In the migrations that we have done, performance has been better than we had expected based on reports of other efforts. By using only a restricted subset of relational database features we avoided falling into some common new user performance traps. The nature of IMAGE database accesses, when mapped into SQL queries, does not take advantage of the possible optimization that SQL can do.

6. Migration Issues HP-UX to MPE XL

This section will briefly discuss some of the migration issues that arise when moving technical and commercial applications from HP-UX to MPE XL.

6.1 Technical Applications

Most technical applications and utilities on HP-UX are written in C. On MPE XL, HP's C/XL compiler and libraries do a very good job of hiding the differences in environment. Problem areas are likely to be in the areas of signals, IPC, and networking. Most of these problems can be worked around by using MPE message files, at some cost in performance. The upcoming release of ARPA Services and the POSIX extensions to MPE will address most of these issues. Although in the first release some functionality such as Curses will be missing.

6.2 Commercial Applications

The same issues regarding COBOL migration discussed above apply here in reverse. The COBOL programs can be translated into HP COBOL II with an editor script. System calls can be emulated by writing routines to map the calls onto MPE XL intrinsics. Unfortunately, the upcoming POSIX extensions will only be callable from C.

There are many more RDBMS packages available on HP-UX than on MPE. If your application is using one that is available on both platforms, then the migration should be easy. Otherwise, the ease of migration will depend extensively on what proprietary extensions have been used.

7. Cross development

As MPE XL is the best environment for executing programs, and UNIX is the best environment for writing programs, can we develop our applications on UNIX and run them in production under MPE XL? The answer is a qualified yes. Many firms are doing that very successfully now with technical applications and utility programs. HP's ALLBASE RDBMS is a good example. ALLBASE development is done on (and for) HP-UX, and is then ported to MPE XL. Applications developed in such a manner tend to be written in C.

For commercial applications the situation is different. For development with one of the 4GL/RDBMS packages that are available on both platforms there's no problem. However, programming in COBOL will require source changes to the program after moving to MPE XL in order to compile. Additionally, the problem of the user interface arises. If you are planning to use V/PLUS, you will need to simulate it for testing under HP-UX.

8. Portability

This section discusses what can be done now to ensure portability of applications in the future.

8.1 Languages

There will always be slight differences in COBOL between MPE XL and HP-UX. Coding standards can ensure that programs can be mechanically translated from one system to another with an editor script.

Application programming guidelines that have served in the past such as isolating all I/O and system functions into separate modules with well defined interfaces, will serve well again.

Consider re-writing all SPL code into C, as this provides a gateway into non-HP UNIX systems.

8.2 IMAGE database

Access to the database should be within well defined transactions. Code should not depend on the format of the IMAGE status area. If possible, nothing in the status area should be relied upon. One area of trouble is the "chain length" value, set after a DBFIND. This can be expensive to simulate in RDBMSs.

8.3 System Calls

Use only POSIX or XPG compliant system calls. HP identifies in the HP-UX manpages the level of compliance each system call has. Other UNIX vendors usually provide a "portability guide" that identifies how their version of UNIX complies with the various standards.

8.4 User Interface

Consider re-writing to use Curses, or plan to use a third party product that is available in both environments.

9. Summary

It is clear that MPE XL will continue to be around, even as UNIX becomes increasingly pervasive in the commercial application area. It is more likely that shops will slowly migrate to an Open System such as HP-UX, or support both environments, rather than do a mass conversion.

HP's increasing support of Open Systems Standards on MPE XL, may make the HP 3000 more attractive to shops concerned about open systems. However, the first release of POSIX extensions to MPE XL appears to be more oriented towards attracting UNIX applications to MPE, than to support commercial applications.

It has been our experience that the "old" computer never really goes away. And that while you may set out to migrate in one direction, invariably the issues of migrating in the other direction get raised. Hopefully we've presented some of the problems faced by migration between the two operating systems, and some of their solutions.

Application Migration between UNIX Platforms*

Paper #8081

by **Andy Feibus**

Professional Press
101 Witmer Road
Horsham, PA 19044
(215) 957-4281

I just recently migrated (ported) an application from a Sun Microsystems workstation (running SunOS version 4.0) to the Hewlett-Packard 9000 Series 700 (running HP-UX), the Hewlett-Packard 9000 Series 400 (also running HP-UX), the Digital Equipment DECstation (running ULTRIX), the IBM RS/6000 (running AIX version 3.1), and the Data General AViiON (running DG/UX).

The application, which provided digital image display and analysis, had been written specifically for a Sun computer and includes about 300,000 lines of FORTRAN code and over 30,000 lines of C code. The application contained many SunOS-specific and Sun compiler-specific code.

The original user interface (under SunOS) incorporates both SunView graphical windows and a standard ASCII terminal window. Since only Sun workstations run SunView, the user interface had to be changed to use X Windows and the Open Software Foundation (OSF) graphical user interface Motif.

The point of this paper is to provide some insight into the current state of portability and compatibility between the currently popular "UNIX"-compatible workstations. The "UNIX" is placed in quotes, because only the UNIX System Laboratories, Inc. and its licensees can distribute and name their operating system "UNIX". However, the operating systems discussed in this paper are all compatible -- in one way or another -- with UNIX System V.

---

*Portions of this paper originally appeared in the article "The State of Portability" (*HP Professional*, March 1991) and are included with permission of Professional Press, Inc.

Sun SPARCstation

The application had been written using many SunOS-only features. The first task in the migration process was to remove as much SunOS-specific code as possible and replace it with standard UNIX System V or POSIX code.

SunOS is based on the Berkeley Software Distribution (BSD) variant of UNIX, which is very different from UNIX System V. SunOS was not POSIX (IEEE 1003.1-1988) compliant until version 4.1.1 (which was released only recently). SunOS is also only compliant with System V features at the subroutine level.

Two examples of this incompatibility:

1. System V uses the **lp** command to spool print jobs; SunOS uses the **lpr** command. Since the application's printer interface was hard-coded to use the options available with **lpr**, this interface had to change.

2. SunOS includes many BSD-specific subroutine calls. Since the program had been initially written for SunOS, these subroutines were used throughout the application. For example, to control the program's terminal window, the application used the BSD terminal interface. For maximum portability, I rewrote the control routines to use the POSIX terminal interface subroutines (see **termios***(7)*).

Additionally, the application's original user interface was based on the SunView graphical user interface provided with SunOS. Since SunView is not available from any other major workstation vendor, a new graphical user interface had to be written using X Windows Version 11 and Motif (which is available on all target platforms). Creating a minimal subset of the original user interface required about four weeks. The final version required about eight weeks of development time.

The SunOS FORTRAN compiler supports more system interface subroutines (e.g., **fseek***(3)* is available from within FORTRAN) than available on other platforms. Since the application was not originally written for portability, several of these routines were used in the FORTRAN code. As a result, these subroutines had to be emulated on the other platforms using either C code or slower FORTRAN algorithms. This conversion took a few days and made the code much more portable as a result.

Another "trick" used in the conversion process was to use the C preprocessor (**cpp**) as much as possible with *both* FORTRAN and C code. For example, if a subroutine was provided with the SunOS FORTRAN compiler that was not provided with other compilers, two sections of code were created within the source code file. The preprocessor compiler directives (e.g., **#ifdef**) were then used to control which section of code to compile.

On some systems (like Sun), the C preprocessor is automatically called if you use the FORTRAN compiler to compile a file having a **.F** suffix (e.g., **codefile.F**). On other systems, the C preprocessor had to be executed manually to generate the **.f** file with the code specific to that particular system (and then the FORTRAN compiler is executed on the **.f** file).

HP 9000 Series 800 and Series 700

The HP 9000 Series 800 running HP-UX version 7.0 was the first target platform for the application migration. Once the HP 9000 Series 700 was available, the application was immediately ported to that platform without changes.

HP-UX is both POSIX- and System V-compliant. Additionally, HP-UX provides many of the subroutines from the BSD version of UNIX. Since SunOS is based on BSD, using these subroutines made porting the application quicker.

HP-UX includes X Windows and Motif. The Series 800 was the platform on which the new Motif-based user interface was created. The new interface was based on X Windows 11R3 and Motif 1.0, both of which were shipped with HP-UX version 7.0. As I already stated, re-doing the graphical user interface using Motif 1.0 required about four weeks.

Migrating the user interface to X Windows 11R4 and Motif 1.1 (which are included with HP-UX version 8.0) required only that certain new Motif widget resources be assigned the proper values.

Migrating the remaining code in the application (once I understood the application) required about 5 weeks to complete.

Since HP-UX (the System V-compliant implementation) was originally released in 1985, the operating system is the most mature of the UNIX-like operating systems on which I've worked. The compilers are fine and the optimizers are improving with every release. The only problem found during porting (as I already mentioned) was that the FORTRAN compiler did not provide as many system interface subroutines as the SunOS FORTRAN compiler.

The only negative comments about the Series 800 port concerned the slow speed of the X Windows-based user interface. SunView provides a graphical interface that permits applications to directly manipulate the graphics hardware (similar to HP's Starbase). X Windows is hardware independent, so it is slower when performing similar graphics operations.

Because of the compatibility between the Series 800 and Series 700, the application ported to the Series 700 in less than one day. The application runs faster on the HP 9000 Series 700 than on any other platform discussed in this paper. Once the application was migrated to the Series 700, support for the Series 800 was dropped.

HP 9000 Series 400

Following right on the heels of the Series 800 port, the Series 400 was a very simple port. Both the Series 800 and the Series 400 run HP-UX version 7.0; only the FORTRAN compilers are different and those differences were minimal. These differences were eliminated in version 8.0.

As with the Series 800 port, the only negative comments about the Series 400 port concerned the speed of the X-based user interface. The horsepower provided by the HP 9000 Series 300 (and the early Series 400's) is inadequate for a floating point-intensive graphical application. Newer Series 400 systems provide better performance.

This port only required 3 weeks, and that figure *includes* testing more than sixty programs that comprise the software application.

Digital DECstation

Porting to Digital's RISC-based ULTRIX was the next task. By now, the bulk of the FORTRAN code was portable and only the C code required any modification. Also, ULTRIX originated from BSD and then added System V and POSIX features. ULTRIX also supports X Windows 11R4 and Motif 1.1 (running under DECwindows).

This port involved finding the right combination of SunOS-specific and HP-UX-specific code. In general, I only had three problems with this platform, and these problems were only irritating (and not debilitating).

The first problem was that DEC does not (yet) consider the Motif user interface to be a standard part of their DECstation software; it's an add-on product. Their Motif implementation operates like a hybrid of Motif and DECwindows. The ULTRIX Motif **include** files and libraries are located in the directory **/usr/lib/DXM**; all other systems placed the Motif include files in **/usr/include** and the libraries in **/usr/lib**. I had to create numerous symbolic links to these directories before I could compile the user interface code.

The second problem concerned the compilers and linker. When linking the code, I kept receiving a message instructing me to use the **-Gnum** option. However, my documentation did not describe what this option did, why I needed it, or the proper value for **num**. After a few days, my Digital support engineer provided the following explanation.

The RISC architecture used in the DECstation product line includes a high-speed cache area for all external symbols (e.g., variables in FORTRAN **COMMON** blocks). If too many variables are declared external, you have to instruct both the compilers and the linker to limit which variables are placed into the cache area; this limitation is specified using the **-G** option during both compile phase and link phase. For most applications, the default **-G** value (**8**) is great. For the application I was migrating, the best **-G** value (determined through trial and error) is **2**.

The final problem concerned using the RS-232 interface provided with the DECstation. The ***stty****(1)* settings used to control the serial data interfaces on Hewlett-Packard workstations caused the DECstation serial interface to hang. Once the interface hung, any program subsequently attempting to access it would hang. Only rebooting the system would clear the condition. I still don't know why this happened; but, after a little experimentation, I found another way to set the interface to the desired settings and the problem disappeared.

In general, the DECstation provides one of the fastest X Windows implementation currently available. Only the HP 9000 Series 700 provides faster X performance.

This port, including testing, required about four weeks.

IBM RS/6000

Not only does AIX version 3.1 support POSIX, ANSI, and System V, but also BSD; additionally, with AIX you can choose which standards to which you want to adhere. Since the application code combines features of System V, POSIX, and BSD, I chose to use all of these standards. Unfortunately, one of the C modules would not compile when all standards were used; the standards, in certain cases, conflict.

AIX includes AIX Windows, which is IBM's implementation of the X Windows 11R3 and Motif 1.0. Both the C and FORTRAN compilers support the current ANSI standards, as well as providing certain AIX-specific extensions and compatibility support for the IBM RT computer.

The port to AIX was one of the most frustrating. During the port, I found several major problems in the compilers, the operating system, and the X Windows/Motif interface.

The first problem I encountered was that FORTRAN intrinsic routines with the same name as C library routines cause a conflict during a program's linking phase. For example, using the FORTRAN intrinsic **GETENV** created a linking incompatibility with the C library routine **getenv**. IBM support spent three days solving that problem.

The next major problem involved a bug in the X Windows libraries. The bug was simple to reproduce (but difficult to explain) and I passed a test program to IBM on-line support for immediate assistance. After a week they agreed that the problem was a defect and that fixing this defect was a **priority 1** (the highest priority) task. IBM support requested a telephone number where they could reach me 24 hours a day; reluctantly, I gave them my home number.

And waited.... Six weeks later (without an IBM engineer *ever* calling me at home), they finally shipped me a version of the X Windows libraries without the defect. Good thing I was a high-priority task....

I found the IBM C compiler to be relatively free of problems; the FORTRAN compiler, on the other hand, is another area of difficulty. In addition to the FORTRAN libraries conflicting with the C libraries, the FORTRAN optimizer produced invalid code. Code compiled with the optimizer enabled would generate code that operated differently from code that was compiled with the optimizer disabled. Unoptimized code is between 2 and four times slower than optimized code; however, optimized code that is invalid is useless.

In general, except for the problems mentioned above, the RS/6000 is as irritating to use as a Digital VAX running VMS. For example, the system arrived with very little printed documentation. Almost all of the documentation for the RS/6000 is on-line and is viewed with a utility called **InfoExplorer**. The problem with this scheme: *where do I, a novice user, start?* The on-line documentation is organized into a database, so I can easily use it as a reference manual; however, it's useless as a user's guide. I also haven't figured out how to print a whole manual (e.g., the *C Reference Manual*) from the on-line documentation.

Some other RS/6000 irritants: system administration is much more complex and less intuitive (just adding the system to an existing network took me several days; adding the HP 9000 Series 400 to the network took about an hour), the print spooler is mind-numbingly complex, the tape drive's programmatic interface is different from every other UNIX system I've seen, and so on. The most irritating part: *the differences weren't necessary!!*

Because of all the problems, migrating code to this system required almost four months of off-and-on work. In general, not much AIX-specific code needed to be created; the previous ports had handled the special cases I encountered with the RS/6000.

The performance of the system I used (the PowerStation 520) is less than I expected. Compiling the application code is about as fast as the DECstation 5000, but X Windows graphics performance is about half as fast as the DECstation 5000 Model 200.

Data General AViiON

The last platform to which this application was migrated was the Data General AViiON 300 Series computer running DG/UX version 4.3.

DG/UX is compliant with POSIX and System V and adds some BSD features. In some instances, both the BSD and System V functionality are provided, but are named differently to prevent naming conflicts. One example is **sigpause***(2)*, which is available as **sigpause** (System V) and **berk_sigpause** (BSD). The additional functionality is nice, but the (nonstandard) name change makes porting applications more difficult.

DG/UX also provides X Windows 11R4 and Motif 1.1. The "standard" C compiler is provided by GNU; additionally, the Green Hills C and FORTRAN compilers are available. For this port, the Green Hills compilers were used.

The FORTRAN compiler was the most vanilla compiler of the lot: no system interface subroutines were provided and you cannot access command-line arguments from within a FORTRAN program. Once these technical details were overcome, the port was pretty generic and required about four weeks to complete.

The only major problem I found while porting the application was that the X Windows defect found on the IBM RS/6000 was also a problem in DG/UX. After one week, DG sent me a tape with their next release of the X Windows software; the new software corrected the problem.

The AViiON workstation, although RISC-based, runs about as fast as a HP 9000 Series 425: not impressive, but acceptable for most work. I think I've gotten spoiled by the HP 9000 Series 700's performance.

Conclusions

Claims of POSIX and System V compliance do not entirely describe whether a platform is truly "standard" or whether an application can be easily ported to it. Most software applications require additional support beyond the vanilla standards; the best systems provided this support. A defect-free operating system and compilers are also required to successfully migrate a software package to another platform.