EDP ANALYZER

MAY, 1976
VOL. 14, NO. 5

APL AND DECISION SUPPORT SYSTEMS

One of the interesting phenomena of recent years in the computer field has been the growth in popularity of the APL programming language. APL is a mathematically-oriented language—but one of its big growth areas has been in business environments. We investigated and found that it often is being used in support of management problem solving and decision making—in short, as part of decision support systems (DSS). This led us into an investigation of DSS, and we came across some particularly meaty research studies of these systems that have been performed recently. Here, then, is what we found to be happening with APL in business environments as well as what others have found is happening with decision support systems in general.

Massey-Ferguson Limited, with headquarters in Toronto, Ontario, Canada, is one of the world's largest manufacturers of farm machinery, industrial and construction machinery, and diesel engines. Annual sales in 1975 were $2.5 billion and the company employs over 60,000 people worldwide. The company has about 70 plants in 26 countries of the world.

The head office of Massey-Ferguson has only about 130 people. Most of the efforts of this headquarters staff are spent on planning and decision making about where the company wants to go in the future. Headquarters does not attempt to decide *how* to do these things, as those decisions are left up to the decentralized operations.

Because of these planning and decision making activities, the staff has developed a large number of application systems for processing hypothetical data—that is, "what if" evaluations. These must be flexible systems with fast response, so that a large number of hypothetical situations can be tested. Batch-type systems do not work well in such an environment.

In 1972, a commercial APL service in Toronto offered Massey-Ferguson (MF) an opportunity to use APL services on a two week try-out basis. (We won't identify which users used which APL services in this report since we are concentrating on APL as a generic language and not on specific service offerings.) The purpose of the test was to see if MF could use APL in the analysis and planning activities. One of the staff analysts at MF tried a small benchmark problem for this test. He programmed the problem in both FORTRAN and APL and then executed the programs. The FORTRAN version took 20 minutes to program and about 25 minutes to get it debugged and executed the first time. The APL version took 5 minutes to program and about 8 minutes to get working. Development computer time for the FORTRAN version cost MF about $13 versus about $10 for the APL version. Subsequent runnings of the two programs cost about $4 for the FORTRAN version versus about $8 for the APL version. The conclusions from this benchmark (which the staff analyst still thinks is valid) is that APL is better for the devel-

opment of programs and for short jobs, while FORTRAN is better for frequent long running jobs.

At MF, staff analysts who have been trained in the use of APL are the ones who sit at the terminals. These analysts report to the decision makers. Very rarely do senior executives themselves use the terminals, although there are some who are capable of doing so. Two types of APL training are used. One type is for simple retrieval, for invoking catalogued analysis and planning routines, and for reporting. This training is fairly simple and requires only a few hours. The other type is true APL programming training. Not many MF analysts have asked for this type of training, so the APL programming falls to the few that have had it.

To make the use of APL easier, MF people have built an English-like macro language for many common functions for analysis and planning. In addition, the APL service they subscribe to is very user oriented. The APL interpreter is quite forgiving when the user makes a mistake. It tells the user what was entered wrong and asks that the correct item be entered. In addition, the macro language provides a "help" feature. When the user does not know what to do next, he just types in "sos" and the system responds with easily readable instructions on what to do.

As an example of MF's use of APL, a controller at an operations unit in Europe requested that the head office develop a system for his unit for forecasting sales and direct variable costs, and for the allocation of overhead. The unit had been using a batch system and was not satisfied with it; the response time was too slow. For instance, it was taking three weeks just to forecast the sales figures, and the controller wanted to be able to test a larger number of values of the critical variables.

To meet a budgeting deadline, the controller wanted the system in seven weeks time. Headquarters agreed to help the unit do it in that time and the unit assigned a systems analyst to the job who knew the business well but who had just learned to program in APL. He was able to draw on the APL experience of other analysts, of course. The programs for forecasting sales and direct variable costs, and the data needed for the forecasting, were ready to go by the end of the seventh week.

So APL is performing a useful service for Massey-Ferguson in support of their management problem solving and decision making activities.

Xerox Corporation

Xerox Corporation, with headquarters in Stamford, Connecticut, is the leading manufacturer of photocopy equipment. Sales are in the order of $3.6 billion per year and the company employs over 100,000 people world-wide.

In the late 1960s, staff members at the headquarters of Xerox's Information Systems Group in Rochester, New York, were developing computer-based planning models for planning photocopier production and marketing activities. These models were programmed in FORTRAN and run in a batch mode. However, management was not satisfied with the inflexibility and slow response of this method.

In 1971, a staff member at Xerox Canada started using the APL service offered by a Canadian time sharing company. He demonstrated that he could write programs in APL in less time than was required for FORTRAN, and that he could get a very fast turnaround when running the models. Other people within Xerox heard about his experiences and subscribed to the same service. Then people at Group headquarters started using it. By 1972, the use of APL for analysis and planning purposes had built up to significant proportions. A proposal was written to obtain an in-house APL service. Since 1972, the Information Systems Group has used both in-house and commercial APL services.

Xerox now makes a wide variety of uses of APL. These uses include the analysis of both revenues and expenses, to track down causes of variance from planned revenues and expenses. Also, a large number of planning activities are performed, with APL as the programming language used; these include such things as market forecasting and production planning. Staff members have been working on a world-wide long range planning system for Xerox, covering five and ten years in the future. This system will include a large number of models, all inter-related. Some are forecasting models, others deal with the profitability of the forecast, and still others deal with the long range plans.

A good many of Xerox's models are user oriented. With these models, the user communicates with the system in an English-like macro language, which is automatically translated into the APL commands which are then interpreted and

executed. The user does not see the APL commands themselves.

Typically, we were told, there is an APL data base that is fed from two sources—a plan data base and a current operations data base. The plan data base covers the next year and is updated quarterly. The current operations data base is maintained by the regular data processing systems, which are usually programmed in COBOL and run in a batch mode. Data for the APL data base is extracted from these two sources.

A number of analysts at each using site have been trained to use the APL terminals for the retrieval of data, for invoking programmed routines from the library, and for obtaining output reports. In such cases, the APL terminal has replaced the desk calculator on the analyst's desk.

In addition, some sites have two or three good APL programmers. If a staff analyst needs a new program of some complexity, he may ask an APL programmer to write it for him. For models with company-wide significance, design by a central APL group is recommended.

APL is thus used for the analysis of current operations and for the planning of future operations. For planning, models are developed. In order that these models will be accepted, the people at Xerox have found that the models must be meaningful to management. So their models tend to be very straight forward, using techniques such as smoothing and curve fitting of time series data. The factors used in the models have to be recognizable and understandable, as far as management is concerned.

What have the people at Xerox found out about using APL for analysis and planning? Here are some of the points they made to us.

APL has reduced the time and cost of developing analysis and planning applications, as compared with, say, FORTRAN. It has also reduced the time and cost of system maintenance, but with a proviso attached. Each APL command can, in general, do so many things that one person has a hard time determining the intent of another person's APL program. If a program is not well documented, maintenance costs can go up rapidly. Another beneficial use has been to develop some batch-type programs in APL which are later converted to COBOL.

As with any powerful tool, control of usage is needed. Several points were made to us. The language is very flexible and a programmer can do things quite inefficiently if he does not understand what he is doing. It is possible to build up high usage charges in a hurry, if usage isn't monitored. The fact that APL commands are interpreted, not compiled, adds to this. Also, the analysts tend to do things "quick and dirty," they sometimes want to be clever in how they use APL, and they tend not to document their programs. All of these practices can lead to the scrapping of programs. So at Xerox, APL users are encouraged to use conservative programming techniques and to document all of their programs.

One other factor to remember is that APL is an interactive system. If an in-house APL service is provided, it will require a telecommunications network. The costs of the network must be carefully monitored.

As at Massey-Ferguson, Xerox is finding that APL is doing a good job for them in their analysis and planning activities.

American Airlines

American Airlines, Inc., with headquarters in New York City, is a major airline serving some 60 cities in the U.S., Mexico, and the Caribbean. Annual revenues are about $1.6 billion and the company has some 35,000 employees.

A few years ago, American Airlines consolidated all data processing operations at their maintenance base in Tulsa, Oklahoma. These operations include the Sabre reservation system, maintenance control systems, flight planning, and various administrative systems. However, they have provided additional computing services in support of analysis and planning activities via two commercial APL services. These services are available not only at the New York headquarters but also at major operating points such as Dallas and Tulsa.

Using APL, staff members have developed an English-like macro language, a library of routines, and a data base which together make up the system they call AAIMS. AAIMS is a modeling and reporting system with which non-programmer users can solve their particular problems. AAIMS started as a small project, consisting of several "cooperating" APL routines. It has evolved into a system with about 40 key words and a data base of 150,000 time series.

The AAIMS data base is unusual in that it con-

tains nearly as much information about competitors as it does about American's own operations. All airlines must submit extensive monthly and quarterly reports to the Civil Aeronautics Board. These reports provide a multitude of operating statistics—available seat miles, revenue passenger miles, departure and arrival statistics, etc. This data becomes publicly available once it has been submitted to the CAB. So American carries this data in time series form, generally beginning in 1968 but with some going back to 1961.

The original intent of AAIMS was to assist forecasting efforts but the initial usage was oriented around management reporting. At first, the daily reporting of traffic and capacity performance was emphasized. Each day, passenger load information was obtained for all of AA's route segments and entered into the AAIMS data base. Then daily reports were prepared, showing actual versus forecast, load factors, and the variance from plan and last year.

AA's use of AAIMS has expanded to cover the monitoring of historical trends and the analysis of relative competitive position. For instance, one application reports on trends in American's revenues and expenses. All facets of the airline's operation are covered—aircraft operating expense (including crew, fuel, depreciation, etc.) by aircraft type, landing fees, passenger services, etc. Also, each expense item is shown as a unit cost, by dividing cost by volume; this step helps separate price changes from volume changes.

But the analysis does not stop there. Because data is available and easily manipulated by AAIMS, analysts can generate reports pertaining to other airlines. By examining the cost structures of competitors, after adjusting for factors such as routes served, the analyst is able to identify areas where competitors may be doing a better job than American. These differences are then analyzed by the company and if found to be valid, appropriate action is taken.

American Airlines, in conjunction with the Air Transport Association and a commercial APL service, markets AAIMS to other companies. Other users include airlines, airframe manufacturers, and financial institutions interested in the airline industry.

APL has been useful to American Airlines. Virtually all of AA's time sharing activities are now done with APL, replacing applications that a few years ago would have been done with FORTRAN or BASIC. APL-based AAIMS, as well as APL alone, have gained extensive support in the areas of business analysis, planning, and budgeting whereby users can make use of the computer without becoming programmers.

The status of decision support systems

We continue to see references (primarily in sales literature, it turns out) to the fact that hardware and software are now available which will allow the executive to use the computer directly, for his problem solving and decision making. The impression is given that the manager or executive will sit at the terminal and will use the data base and the library of routines for problem solving in an interactive manner.

The concept here is intriguing and we have been looking for examples of it for over ten years. We have found a few cases, but they seemed to be like snow in Southern California—if they appeared at all, they disappeared soon after they appeared.

Then, when we began to hear about the good acceptance of APL in business environments, we figured that perhaps it was providing the missing link by which managers would use computers interactively. So we checked into it. What we found is illustrated by the three cases just described. In general, we found that managers are not sitting at terminals to do interactive problem solving. True, line managers in production plants (for instance) use terminals to get answers to production questions, such as the location of critical shop orders. But we did not find managers operating the terminals to develop budgets, marketing plans, or such.

What we did find was that computers *are* being used in support of the managerial problem solving and decision making process. That is, decision support systems (DSS) are in existence and in use. Another thing we found was that in just about all of the cases, some *intermediary* who works for the decision maker is the one who operates the terminal.

In addition, we came across two research studies that bear upon how decision support systems are used. Alter (Reference 1) prepared his doctoral thesis at M.I.T. on computer aided decision making in organizations. He studied in some depth how 56 DSS's in 25 organizations were being

used. We will draw heavily in this report on Alter's findings. Weber (Reference 2) had some 50 personal interviews at large banks on their use of computer assisted decision making and then made a questionnaire survey of over 100 users from other organizations on the same subject. Both of these men were able to go into the subject in much greater depth than we have had time to do. But what we encountered in our interviews seems to be in harmony with what they have found.

Before getting into a discussion of the major results of these studies, it would be well to define the types of roles that people play in connection with DSS. Most of these definitions come from Alter and Weber.

PEOPLE ROLES IN DSS

1. Sponsor—a person interested in the DSS who provides support by allocating resources or otherwise.
2. Advocate—one type of sponsor; one who uses persuasion to gain support for the system.
3. Evangelist—a person dedicated to persuading others to install and use the system.
4. Initiator—the person who first indentifies the need and requests the procurement of the system.
5. Implementor—the person or group that builds the system.
6. Maintainer—the person or group that keeps the system up to date.
7. Feeder—the person or group that provides data for the system and who often derives no direct benefit from the system.
8. User—the person or group that communicates directly with the system, generally by operating a terminal.
9. Decision maker—the person or group that makes the business decisions.

Note that one person can play two or more of these roles. For instance, the same person can be the initiator, implementor, feeder, and user.

Now, what were the results of these two studies, as far as the use of DSS was concerned. In brief, Alter found that, while there was relatively little direct use of DSS for decision making, these systems still supported decision making in many ways such as by performing clerical work, by assisting with learning, training, and communications, and by providing models for evaluating decisions. In many cases, the systems were used by intermediaries who knew the systems and the data bases. These intermediaries shielded the decision makers from the details of the system and the data base, from changes in the system, and from the use of sophisticated math models. In general, no formal problem solving process was used; the use of a particular system depended on the way that people played the various roles. If an interactive feature was present, it was used more for convenience than for interactive problem solving. He found that the DSS did not form a homogeneous set. There were many types of systems, different types of uses, and different types of support. Even if a DSS were available, some people in the organization would use it, others would not. Of those that used it, some used it intelligently while others did not; further, some used only those parts of the DSS that were designed for them.

Weber found that most of the use of DSS was for routine data retrieval or simple data manipulation. Even with these simple types of use, only about one-quarter of the executives claimed to use a DSS frequently. One-half of the executives said they used a DSS occasionally, while the remaining one-quarter said they never used a DSS. (Weber surveyed major banks and corporations who have been using computers for years.) Only 10% of the decision makers claimed to have had any direct use of a DSS, and only 3% had written programs with any regularity.

For one commercial media analysis system, Alter found that about 50% of the use of the DSS was by secretaries or junior analysts, who entered requests for pre-defined reports which they then gave to their superiors. Another 35% of the use was by intermediaries who could make a limited range of decisions. The remaining 15% of use was by decision makers. This system was the unusual case, in the amount of use by the decision makers, but even here the bulk of the use was by intermediaries who had little or no decision making authority.

As mentioned earlier, some DSS were mainly operational in nature, such as work-in-process file systems in production organizations. Dss use of these systems was often simple data retrieval, such as locating specific shop orders, and shop foremen and managers would often make such use. In fact, in some instances analysis capabilities were provided (say, for analyzing when specific orders would be completed under their present priorities and total shop load); these capabilities too were used by shop foremen and managers. But as the time horizons moved out further into the future, the less likely it was that managers themselves used the DSS's.

Another point made by Alter was that the like-

lihood of direct use of a DSS by a decision maker seems to be inversely proportional to the amount of staff help that the decision maker has available. One reason for this might be illustrated by the comment of one intermediary: "I waste a lot of time sitting at a terminal." Since interactive problem solving is not used very much, the decision makers would prefer to use their time (perhaps) more productively and let staff members operate the terminals.

In addition to decision making and problem solving, Alter found a number of different manifest purposes of these systems. Some installed these systems primarily to perform clerical work—that is, maintenance of records and retrieval of data, such as in the case of the work-in-process files. In some cases, the main purpose was more effective interpersonal communications. Learning or training was often mentioned as one of the purposes of a system; new-to-a-position employees often found such systems very helpful in learning their new jobs and becoming productive quickly. Another manifest purpose was to improve overall control of a complex situation, by bringing all of the data pertaining to that situation together in one system. Monetary savings were mentioned as a purpose but Alter considered this almost to be a debatable point since it was often very difficult to point to any tangible savings. Seldom, he says, was problem solving and decision making assistance claimed to be the *sole* purpose of the system.

In addition to these manifest purposes, Alter felt that there were probably some underlying purposes for installing DSS's. These included individual power struggles, attempts to become visible in the organization, attempts to impress people, and so on. But there is no way to really study such behavior in a large number of organizations in a reasonable time, so he did no investigation on this subject.

Alter's main finding, which we will discuss in more detail later in this report, is that the likelihood of successful acceptance and use of a DSS is high for *simple systems that can be planned in advance with the active cooperation of their few potential users.* There is a lot of information in what has just been stated and it will take much of the remainder of this report to describe it.

But first, let us consider whether the advent of APL might change the situation.

The role of APL

APL (which stands for A Programming Language) was created by Dr. Kenneth Iverson in the early 1960s, while working at Harvard University and later at an IBM Research Center. It did not catch on as a programming language in a batch environment. But in the late 1960s, it was implemented as an interactive language and interest in it has soared since then.

As mentioned earlier, we are not singling out any particular APL service offerings in this report. Reference 5 lists many of the U.S. and Canadian commercial time sharing companies and indicates which ones offer APL services. Some of the leaders are I.P. Sharp, Scientific Time Sharing Corp., Boeing Computer Services, APL Services, Inc., and Time Sharing Resources, Inc. Some of the computer manufacturers offer APL interpreters as part of their software. And it is likely that APL will be offered with mini-computer systems—as is the case with IBM's new Model 5100.

APL was designed for the handling of matrices, arrays, and vectors, and scalars—in short, it is a mathematically oriented language. But it so happens that a lot of business-type data is appropriate for APL, such as time series data (a vector) or multi-row multi-column financial report data (an array). APL has powerful operators for handling these types of data. For instance, with a single command, two time series can be added. Because of the power of these operators, a lot can be done in a short time, sitting at an APL terminal.

The APL language looks complicated because special symbols are used for many of these operators. Actually, we gather that APL is considered to be one of the simpler, cleaner languages; it is constrained to do certain types of operations on certain types of data, and it does these things very effectively.

The main difficulty with APL is probably due to the power of the operators. The operators can do so many things that the intent of the programmer often is not clear from a line of code. In fact, a complete "program"—a very small but logically complete job—can be written in one coding line, we have been told. Without explicit documentation, one person has a hard time understanding another person's APL program; in fact, the original programmer may have a hard time understanding it when he looks at it some time after he

has written it. APL probably is not suitable as the main programming language for data processing purposes. But it appears to be very useful as a supplementary language, such as in a DSS environment. (It can also be useful in engineering, scientific, and research environments.)

Is APL the missing link for getting decision makers to directly use the computer? Mock and Vasarhelyi (Reference 4) think it is. Their book was written for a management audience, to explain how APL can be used for management science applications. They assume that APL will be used *by* managers. They describe how APL can be used for programming solutions in a management science environment. Types of models discussed include linear programming, networks (PERT and CPM), inventory theory, transportation problems, statistical analysis and forecasting methods, financial statement analysis, time value of money, and so on. APL is an appropriate and powerful programming language for these types of problems, say the authors.

On the other hand, Alter seems not to think that interactive programming languages are the key to use of a DSS. He says there are a number of other equally important problem areas that need solving, such as how to build and verify models, how to insure data accuracy, how to efficiently access large data bases, and so on. But even beyond these considerations, there is a more basic question: *should* decision makers be encouraged to use complex systems which they do not understand? Alter feels they should not be so encouraged, because there is just too much chance of misuse. The staff person who understands the system (particularly if a complex model is used) and the data base knows what the legitimate uses are and what the limitations are. Instead of worrying about programming languages for managers, says Alter, worry about improving the computer interface for the staff members (the experts in using the DSS), so they are not plagued with a lot of computer-related details. It seems to us that *this* will be the role for APL in decision support systems—making it easier for the staff analysts to communicate with these systems.

We do not wish to give the impression that the use of APL in business environments need be limited to DSS-type use. If a staff member is now using a desk calculator or a conventional time sharing terminal, that analyst might be able to use an APL service even more effectively. This is particularly true if the analyst is often working with time series data and/or arrays of numbers.

So while APL might be a desirable facility for an improved analyst-computer interface, it is not a necessary facility nor a sufficient facility for assuring the effective use of a DSS. What then *is* the key to effective use of a DSS? Let us consider some of the problems encountered in trying to use a DSS.

Problem areas in DSS

Our discussion of problem areas associated with DSS is drawn largely from Alter, but we will also cite several authors whose papers are published in Reference 3.

Of the 56 DSS studied by Alter, he found that they divided into two major types—data-oriented systems and model-oriented systems. These in turn sub-divided into four generic systems each. The four data-oriented systems were: retrieval only, retrieval plus special analysis procedures, retrieval plus general analysis procedures, and special data bases plus special analysis procedures. Special analysis procedures are tailored to the specific type of application; general analysis procedures, like statistical analysis packages, can be used for a wide range of applications.

The four model-oriented systems were: accounting models, simulation models, optimization models, and what Alter calls "suggested action" models. The accounting models sometimes use regular accounting programs, together with hypothetical data, to test what financial results might be under different sets of conditions. The suggested action models are those where a series of calculations lead to a suggested action—such as a credit scoring model for determining if credit should be granted and what the credit limits should be.

Of the 56 systems, Alter found that with 15 of them, the potential users showed significant unwillingness or inability to use the systems. The usage was disappointingly less than originally expected, so such DSS might be considered as relatively ineffective.

What explains the relative ineffectiveness of these 15 systems, as opposed to the other 41? Alter analyzed the information he had gathered and came to some conclusions, primarily related to the "user."

Two of the main factors that influenced suc-

cessful use were user initiation of the project and user participation in the project. Following are the four situations identified by Alter, ranging from the most likely to be successful to least likely to be successful (and where "high" means a high degree of user initiation or participation):

First: high initiation, high participation;
Second: high initiation, low participation;
Third: low initiation, high participation;
Poorest: low initiation, low participation.

In the best situation, users keenly wanted the system, saw the need for it, and actively participated in the project. The next best situation was where the users wanted the system and initiated the project, but turned the building of the system over to the implementors. In the third situation, the users really did not feel the need for the system; they were either "sold" the concept by an "evangelist" or had the idea imposed on them by management, after which they actively participated. In the poorest situation, the users just were not interested.

Note that the comments apply to the potential user group as a whole. If only a small fraction of a large potential user group is interested in and participates in the project, the bulk of the potential users may well display a lack of interest.

Alter found some interesting characteristics of user-initiated projects as contrasted with those initiated by others.

User-initiated projects. Twenty-five (45%) of the projects were initiated by users. These projects tended to be smaller, tended to mechanize existing practice, but at the same time tended to be used by more of the potential users than the other systems, he found.

Non-user-initiated projects. Thirty-one (55%) of the projects were initiated by people other than users. Alter found that these projects tended to be larger, covered a longer time span, tended to be more innovative, and tended to have a higher incidence of ineffectiveness. Some of these projects were almost completely initiated and conducted by management science groups within organizations.

In his interviews, Alter was told about other management science initiated projects that had ended up as disasters—over schedule, over budget, users unwilling to use system because of the non-understandable methods used. The whole management science staff might be told to seek employment elsewhere, under such circumstances.

So projects initiated by others—and particularly by management science staffs—tended to seek "big gains" by being innovative. But there were real risks involved, perhaps the largest being that users would not feel comfortable in using the system when it was finally ready for use. A manager has a responsibility to his organization. One of the most important things he does is make decisions—on what to do, when, and by whom. If someone tries to impose a mechanized decision-supporting system on him which uses methods that are completely baffling to him, it is natural that he is going to resist using that system.

Wagner (Reference 3) points out some of the apprehensions that managers have, when someone is trying to sell them on the idea of a new system. They have doubts about the *claimed benefits*; is the system too big, or is it too little, or is there a better approach, or are the target benefits realistic? The managers also have fears for their *careers*; will the new system really enhance their positions, who will be blamed for a failure, and so on? They have apprehensions about the *development timetable*; how reliable are the time estimates, and will the project ever be completed? They are concerned about the *usefulness of the design;* will the organization be capable of using the new system, will the outputs be familiar to the people, or will a new management approach be required? Finally, they have real concerns about a possible *end-of-the-road disaster*; where do the risks of disaster lie, and can the success of the project be thwarted by people in other groups?

There are a number of factors that contribute toward a successful system, says Wagner. One factor is that the managers are heavily involved in the development of the new system. Further, that system should not be a substitute for an old, well known system upon which the users can fall back. Chances of success are better if the organization is in an economic crisis and the economics of a wrong decision are severe—then everyone will realize how important it is to make the new system work. It helps, too, if the project is relatively short (under one year in length), if the system has been installed by someone else (is within the state of the art), and the designer has had considerable experience with similar systems.

Minimizing the risk of failure

Alter has addressed the question of how risk of failure might be minimized, and in an interesting manner. First, he describes an "ideal" situation, where the risk of failure would seem to be minimal. Then he describes a number of ways in which an actual project might deviate from that ideal. Then he proposes a general rule which organizations might follow, after they have identified the specific ways in which a proposed project deviates from the ideal. We will briefly review his approach.

Here is Alter's "ideal" situation:

IDEAL SITUATION FOR DSS SUCCESS

1. The DSS is to be produced by a single implementor for a single user; they may be the same person.
2. The user anticipates using the DSS for a specific purpose which can be specified in advance.
3. The person who will maintain the system and all other people who will be involved in one way or another understand the impact that the new system will have on them and accept what that impact will entail.
4. All people involved have had prior experience with this type of system.
5. The system will receive the necessary support (management support, resources, and so on).
6. The technical design of the system is feasible and cost effective.

It is interesting to note, as pointed out by a reviewer of this report, that these same conditions apply to the successful installation of *any* significant information system.

But the trouble with an ideal situation, says Alter, is that it hardly ever happens. Most actual projects will deviate from the ideal in some manner. *It is essential that the initiator of the project identify all of the ways in which the project deviates from the ideal,* at the outset.

What are these possible deviations? Alter lists eight that he came across in the course of his study:

DEVIATIONS FROM THE IDEAL

1. Non-existent or unwilling user.
2. Users and/or implementors are members of large groups.
3. Original users, implementors, and/or maintainers "disappear" during the course of the project and are replaced in their positions by new people.
4. Users and implementors cannot specify the purpose of the usage pattern of the new system in advance; they have only a hazy idea of the purpose and/or usage pattern.
5. Users and implementors cannot predict the impact of the new system on all other parties and hence cannot cushion that impact.
6. The support that is assumed will be supplied turns out not to be supplied.
7. The people involved have not had prior experience with this type of system.
8. There are, in fact, some unsolved technical problems and/or the cost effectiveness of the system is based more on hope than on reality.

These are just examples of possible deviations from the ideal, but they give an idea of the types of deviations to look for.

The initiator and/or implementor must scrutinize the project plans and test all of the assumptions inherent in the plans. Will *all* prospective users participate in the project? How likely is it that some or all of the users, implementors, or maintainers will move on to other jobs in the course of the project? Who will play the role of feeders of data for the system? What direct benefits will the new system provide to these feeders? Who all will be impacted by the new system—who will have to change job practices to some degree, or whose jobs will be threatened, or whose "art" in job performance will suddenly be replaced by a mechanized system? This scrutiny of the project plans might well be the most important step in the whole project so it should not be done casually or hurriedly.

Then Alter proposes a general rule:

First, list every way in which the actual situation differs from the ideal situation;

Next, for each deviation, design a course of corrective action;

Finally, if there is even one deviation for which a course of corrective action cannot be designed, recognize that the risk of implementation difficulties may be quite high. One possible reaction is simply to drop the project, explain the problem to all interested parties, and go on to something else.

An extreme position? No, we don't think so. We have been witnessing attempts to use management science techniques for over twenty years. Acceptance of these techniques has been far less than their advocates were predicting twenty years ago. Where acceptance has been gained, as far as we have observed, it has been where simple, understandable techniques have been used and where the users have really wanted the new systems.

Wagner, McCoubrey, and Gomersall, all writing in Reference 3, give advice in support of Alter's position, in our opinion. There is a

reasonable chance of success if the DSS is designed for the use of a small group of potential users, is built by a small, competent team, and is used for a specific purpose that is precisely specified in advance. The project should be relatively short term, certainly less than one year in length. It will use simple, understandable techniques and will avoid the use of esoteric management science techniques. For example, it might use straightforward simulation methods, curve fitting, and such, but might avoid the use of linear programming, Monte Carlo techniques, and so on.

Alter makes two points about the less risky projects that should be mentioned. One, these projects tend to aim at short term gains, he says; they may contain little innovation, they may have little impact, and there may be little possibility of major gains. This may be true—but there is another aspect to consider, as evidenced by the installation and use of EDP systems. The big, multiyear projects seek big gains but these are the ones that all too frequently end up with big troubles. For the installation of systems using advanced technology, we have been advocating what we call the "progressive' approach" for a number of years. We described the progressive approach for installing fast response systems in our October 1970 report, for instance.

The progressive approach consists of a long range plan which is broken up into a series of short term, free standing projects. Each project should last no more than nine months and preferably would be much shorter. Each would be designed to maximize the chances of success. A structured approach, with a creeping commitment, would be used for each project, as we discussed in our May 1973 report.

If there is a long term goal, and if it is approached by way of a series of short term projects, a very impressive accomplishment can be achieved in three to four years. Management becomes much more pleased with the performance of the data processing staff, and the whole environment for change is enhanced.

So it seems to us that short term projects need not, per se, have little impact. If they are part of a long range project, the overall impact of several of them can be quite impressive. But we would agree with Alter's position if the short term projects were isolated ones seemingly picked on a random basis. For any major impact, we believe they have to be part of a larger program.

The other point of Alter's that we wish to discuss is the importance of the role of the feeders. Generally, these people are asked to do something additional in their work, in support of the new system. Usually they must supply data for the new system, of a type or in a format not previously supplied. Moreover, they soon are pressured to supply this data accurately and on time. Typically they get no direct benefits from the new system. So the attitude of the feeders may soon range anywhere from "why bother" to outright belligerency.

Much more attention must be given by the system implementor to the role of the feeders, says Alter. If at all possible, the implementor should find a way to give the feeders some direct benefits. This means the implementor must study the jobs of the feeders and find out how the system can supply information to them that will help them perform their jobs.

If the feeders adopt a negative attitude toward the system, the data they supply may turn out to be inaccurate, untimely, missing, or such. The whole system can fall into disrepute under such circumstances. So it is important that the system serve the feeders so that they will support the system.

These problems explain why the use of APL (or any other programming language, for that matter) is not the necessary and sufficient condition for the successful use of a DSS. The programming language used is just one factor. There are numerous other factors that are equally or more important as determinants of success.

Once the deviations have been identified, what courses of corrective action can be used?

Possible corrective actions

Alter lists and discusses 16 possible courses of corrective action, for overcoming the deviations from the ideal situation. We have divided these into four categories, and will give a brief overview.

Break up the project. Big projects involving the use of untested techniques can bring problems, arouse management resistance. So break up a big project in one of several ways. One way is to use a prototype system, that performs the complete handling of a portion of the overall job. Or use an evolutionary or modular approach (perhaps along

the lines of the progressive approach we discussed above). Or, instead of developing a system, develop a series of tools (software packages, data files, etc.) that can be used for decision support.

Keep the solution simple. The approaches discussed by Alter include using simple techniques as much as possible, hiding complexities when their use cannot be avoided, and in general avoiding change. Again, Alter expresses concern that the simple techniques may not really be able to do much. If powerful but hard-to-understand techniques are used, he says, consider using intermediaries who will hide these techniques from the decision makers. Avoiding change means mechanizing existing procedures.

Develop a satisfactory support base. Alter lists things that the implementors should seek: user participation, management support, personal commitment. All of these have their benefits and their problems. For instance, management support is usually sought for one of two reasons, says Alter: to obtain authorization and funding for the project, or to force someone to do something. In the latter case, if management becomes enthusiastic without the middle and lower management levels also becoming enthused, real problems can arise. Alter discusses the question of insisting on mandatory use of the new system (which might be needed for a large group of users) as opposed to permitting voluntary use (more appropriate for individualistic type of use) or relying on diffusion and exposure (not really very effective).

Meet the users' needs. One important corrective action is to design the system to suit the capabilities of the prospective users. And once the system has been built, the implementor must provide a user training program as well as on-going assistance in the use of the system. Alter frequently makes the point in his thesis that it may be necessary for the implementor to describe *exactly* how the users can use the system, even to the extent of showing how it can be applied to on-the-job problems. If the implementor adopts the attitude: "here is a powerful new tool that you can use on your job, so just figure out how you can use it," then there is a good chance that the system will not be used.

In short, there is no general solution, no "magic formula" that will guarantee the success of a decision support system. The chances of success are increased by identifying the ways in which the project deviates from the ideal situation and then selecting courses of action to overcome those deviations.

In summary

Over the past 12 to 18 months, we have been examining the question, "Are managers actually becoming hands-on users of computers, to aid them in their problem solving and decision making?" We have come across studies by others that address this same question and have cited these studies in this report. The answer to the question is, in general, a resounding No.

Then we investigated a related question, "Is APL the key to getting managers to become hands-on users of computers?" Again, the answer is No.

But decision support systems *are* being developed and *are* being used. There is a wide variety of types of systems in use, in a variety of applications. Of the installed systems, there has been a wide range of acceptance and use.

In the more meaningful systems (not just simple data retrieval systems), there have been staff intermediaries between the decision maker and the DSS. These staff intermediaries typically work for the decision makers. Some are capable programmers while others have developed computer expertise. It seems to us that APL *can* play a role for these intermediaries, by giving them more powerful operators for the handling of time series data, data in array form, and so on.

The message that we obtained from this study is as follows. It makes sense to use a progressive approach for building and installing a DSS. That is, lay out a long range plan for accomplishing something significant in the way of decision support. Then break up this long range plan into a series of short term, stand-alone projects. Each project should be aimed at a small user group (one to five people) and implemented by a small team (one to five people). Each project should be designed to maximize the chances of success—user initiation, user participation, provide direct benefits to all feeder people, use simple, understandable techniques, and so on. Use "creeping commitment" for each project, so that if a project starts turning sour, it can be terminated at the earliest possible date.

Let's face it: computers *have not* been used in support of management decisions nearly as widely as people were predicting 15 to 20 years

ago. Much the same has been true of management science techniques, and for much the same reasons. But by following a policy such as outlined above, we think the chances are good that you will get some most interesting results in three or four years' time.

REFERENCES

1. Alter, S. L., *A study of computer aided decision making in organizations,* unpublished thesis, June 1975, Massachusetts Institute of Technology, Cambridge, Mass.
2. Weber, R. J., "Computer-assisted decision making," *Bank Administration* (303 South Northwest Highway, Park Ridge, Illinois 60068), Part 1, June 1975, p. 16-21; Part 2, July 1975, p. 39-45.
3. Keen, P. G. W. (ed.), *Proceedings of conference on the implementation of computer-based decision aids,* Center for Information Systems Research (M.I.T., 50 Memorial Drive, Cambridge, Mass. 02139), 1975, price $15.
4. Mock, T. J. and M. A. Vasarhelyi, *APL for management,* Melville Publishing Co.; order from John Wiley & Sons, Inc. (One Wiley Drive, Somerset, N.J. 08873), 1972, price $9.75.
5. "Datapro report on remote computing services," Datapro Research Corporation (1805 Underwood Boulevard, Delran, N.J. 08075), price $10. This report may be updated about once a year; ask for most recent update.

One of the lessons of the first two decades of computer use has been: big systems often mean big trouble. So there has been a lot of attention given to ways to sub-divide large application systems—into sub-systems, into modules, and now into distributed systems. Next month, in the first of two reports, we will discuss some user experiences with distributed data systems. These are systems where large amounts of stored data are involved but the data is not necessarily organized as a data base. In the subsequent report we will consider network structures and protocols for distributed systems.

SUBJECTS COVERED BY EDP ANALYZER IN PRIOR YEARS

1973 (Volume 11)

Number

1. The Emerging Computer Networks
2. Distributed Intelligence in Data Communications
3. Developments in Data Transmission
4. Computer Progress in Japan
5. A Structure for EDP Projects
6. The Cautious Path to a Data Base
7. Long Term Data Retention
8. In Your Future: Distributed Systems?
9. Computer Fraud and Embezzlement
10. The Psychology of Mixed Installations
11. The Effects of Charge-Back Policies
12. Protecting Valuable Data—Part 1

1974 (Volume 12)

Number

1. Protecting Valuable Data—Part 2
2. The Current Status of Data Management
3. Problem Areas in Data Management
4. Issues in Programming Management
5. The Search for Software Reliability
6. The Advent of Structured Programming
7. Charging for Computer Services
8. Structures for Future Systems
9. The Upgrading of Computer Operators
10. What's Happening with CODASYL-type DBMS?
11. The Data Dictionary/Directory Function
12. Improve the System Building Process

1975 (Volume 13)

Number

1. Progress Toward International Data Networks
2. Soon: Public Packet Switched Networks
3. The Internal Auditor and the Computer
4. Improvements in Man/Machine Interfacing
5. "Are We Doing the Right Things?"
6. "Are We Doing Things Right?"
7. "Do We Have the Right Resources?"
8. The Benefits of Standard Practices
9. Progress Toward Easier Programming
10. The New Interactive Search Systems
11. The Debate on Information Privacy: Part 1
12. The Debate on Information Privacy: Part 2

1976 (Volume 14)

Number

1. Planning for Multi-national Data Processing
2. Staff Training on the Multi-national Scene
3. Professionalism: Coming or Not?
4. Integrity and Security of Personal Data
5. APL and Decision Support Systems

(List of subjects prior to 1973 sent upon request)

PRICE SCHEDULE

The annual subscription price for EDP ANALYZER is $48. The two year price is $88 and the three year price is $120; postpaid surface delivery to the U.S., Canada, and Mexico. (Optional air mail delivery to Canada and Mexico available at extra cost.)

Subscriptions to other countries are: One year $60, two years, $112, and three years $156. These prices include AIR MAIL postage. All prices in U.S. dollars.

Attractive binders for holding 12 issues of EDP ANALYZER are available at $4.75. Californians please add 29¢ sales tax.

Because of the continuing demand for back issues, all previous reports are available. Price: $6 each (for U.S., Canada, and Mexico), and $7 elsewhere; includes air mail postage.

Reduced rates are in effect for multiple subscriptions and for multiple copies of back issues. Please write for rates.

Subscription agency orders limited to single copy, one-, two-, and three-year subscriptions only.

Send your order and check to:
EDP ANALYZER
Subscription Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-3233

Send editorial correspondence to:
EDP ANALYZER
Editorial Office
925 Anza Avenue
Vista, California 92083
Phone: (714) 724-5900

Name_____

Company_____

Address_____

City, State, ZIP Code_____