OPERATION AND EVOLUTION OF ORGANIZATIONAL INFORMATION SYSTEMS

Gerald P. Learmonth and Alan G. Merten
Database Systems Research Group
Graduate School of Business Administration
University of Michigan
Ann Arbor, Michigan 48109/USA

1.0 Introduction

In recent years, users and data processing personnel have become increasingly aware of the cost of building and maintaining computer-based information systems. In addition to recognizing the cost and complexity of these systems, organizations have recognized that these systems are built and maintained over a long period of time and that the life of a computer-based information system can be viewed as consisting of a series of steps or phases. During each of these phases in the life cycle of a system, different activities are performed, various user and data processing personnel are involved, and different degrees of time and materials are expended.

Traditionally, the construction or programming phase of the life cycle received the most attention of practitioners and researchers. Because of the failure of many large systems to meet the requirements of the end users, much attention during the past few years has also been directed to the requirements analysis and design phases. Software engineering, structured design, and structured walkthroughs are but a few of the techniques that have been discussed, developed, and used in an attempt to improve the quality of these pre-implementation phases.

This shift in emphasis from programming to requirements and design will most certainly effect the quality of systems. However, in spite of how systems are built in the future, the importance of post-implementation activities will continue. These phases and activities are important because of the large number of complex systems in existence and the large number that are about to become operational (most of which are being built without the benefit of structured design, sufficient documentation, etc.). Daly [1977] characterized these post-implementation activities as follows: "Although many feel that the management of large software development is mysterious, or at least little understood, the long term control and maintenance of large programs is even more mysterious."

The purpose of this paper is to survey the phases of the life cycle which occur after the design, construction, and implementation of a system. These phases will be referred to as the operation phase and

the maintenance and modification phase.

Each of these phases consists of a number of sub-activities. With respect to some of these sub-activities, the paper will address the following:

a) the main problems that occur
b) procedures and techniques used by organizations during these activities to reduce the impact of the problems
c) research results and approaches which could effect the activities of these phases
d) activities which may be performed in earlier life cycle phases to reduce the number of problems during operation and during maintenance and modification

Section 2 will formally define the problem area and highlight the difficulties currently encountered during these two phases. Section 3 will present, discuss, and classify the various types of observation and analysis that occur during the operation phase. Section 4 will discuss maintenance and modification activities. Section 5 will present a summary and conclusion.

2.1 Problem Definition

The phases of the life cycle of a computer-based information system have been defined by many authors and operating organizations. Depending on the originator of the definition, the number of phases, their individual names, and the sub-activities differ widely. For the purpose of this paper, we will assume that the life cycle phases are specified as follows:

a) Specification of General Need
b) Feasibility Study
c) Collection and Analysis of User Needs
d) External Design (General Systems Design)
e) Internal Design (Detailed Systems Design)
f) Construction and Conversion
g) Testing
h) Operation
i) Maintenance and Modification

The detailed description of the sub-activities, problems, and solutions associated with phases a) through g) is the subject of many papers. While the amount of resources expended in these phases is often large, many people now believe the major portion of software manpower is spent in maintenance rather than in development (Boehm [1973], Branscomb [1976], Lientz, et al. [1978]). Similarly, because of the many

large systems now in development that will soon transition to the operation/maintenance/modification phases, it is expected that efforts in both of these phases will increase, at least in the short and mid-term range. Coupled with the high system longevity, some predict a five or ten to one ratio of operation/maintenance/modification cost to development cost when viewed over the total life cycle (Gansler [1976]).

The complexity of the post-implementation phases is de-emphasized in the "linear" presentation of the life cycle steps as presented earlier in this section and as presented in most system development manuals. A more representative description is given in Figure 1. The operation phase and the maintenance and modification phase are on-going, cyclic activities. In addition (but not represented in Figure 1), these phases often occur simultaneously, that is, while the system is in operation, maintenance and modification activities take place. The figure explicitly identifies the "analyze" activity and the "observations" and "specification of changes" entities. These entities and activities are important components of the transition between operations and maintenance/modification but are often ignored in the life cycle management of organizational information systems.

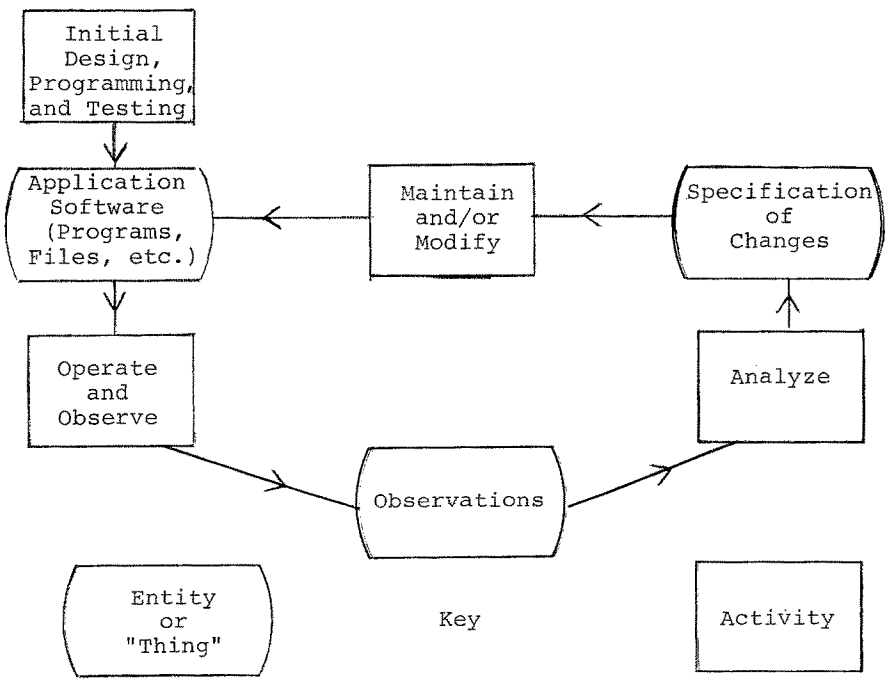

Figure 1. Cyclic Behavior of the Post-Implementation Activities of the System Life Cycle

3.0 Observation and Analysis

3.1 Introduction

To effectively manage a computer-based information system, thorough and careful observation and analysis of system behavior and performance must be undertaken. This section will review existing methodologies for observing and analyzing a system. Both external, off-line observation and analysis tools as well as internal, on-line, tools will be discussed. Several promising new ideas will also be examined.

3.2 Performance Management

With the ever growing complexity of computer systems, the problems related to computer performance measurement and evaluation have received considerable attention from researchers and practitioners alike. Most of the concern in this area, however, is related to issues of hardware and system software performance. Measurement and evaluation of such criteria as CPU utilization and scheduling policies are aimed at decisions regarding hardware enhancements or system software tuning.

The effective operation and evolution of an information system requires observation and analysis which transcends the hardware and system software. The term "performance management" will be used to describe the activities of observation and analysis both off-line and on-line which are necessary for managing an information system throughout its operational life-time.

Observation requires the systematic recording of pertinent performance measures. The analysis activity involves comparing these observed performance measures with certain predefined standards of performance. If these standards are not met, then maintenance and modification activities are undertaken.

Traditional computer performance measurement and evaluation has developed a common set of performance metrics which are easily observed and analyzed. When considering performance management of an information system, standards of performance include not only the quantifiable measures but also a considerable number of intangibles. It is extremely difficult, if not impossible, to measure and quantify such standards as user satisfaction, system security, data integrity, and system recoverability after unforeseen hardware or system software failures.

The next two sections will address existing tools and highlight some novel approaches to implementing an effective performance management program for an operational and evolving information system.

3.3 External Performance Measurement

In the development phase of an information system, alternative system designs are generally compared and the design with the least cost and greatest potential benefit is chosen. To make this decision, various organizational costs are identified and estimated. Among these are personnel costs, equipment costs, cost of vendor software, file conversion costs, and perhaps the opportunity cost of undertaking different projects. To the extent that these costs are able to be estimated, they provide quantifiable standards against which development progress can be measured.

A more difficult task is to quantify benefits which will accrue from an information system. Certainly some benefits can be assigned monetary values such as a reduced billing cycle and improved inventory management. Information systems are, however, often designed and implemented to provide certain intangible benefits to the organization. Improved managerial control, increased data security, improved employee morale are all benefits which are immeasurable yet constitute design goals for an information system.

The methodology of cost/benefit analysis is well defined and widely used in the development of many corporate and social systems. In a recent survey paper, King and Schrems [1978] have adapted cost/benefit analysis to information systems development and operation. Their survey carefully describes the stages of cost/benefit analysis as it would apply to an information system. For information system development, they enumerate many of the costs, both obvious and "hidden," which must be considered. They also provide suggestions for assigning values to many of the heretofore intangible benefits of information systems.

A comprehensive cost/benefit analysis of an information system provides standards against which the development of the system can be measured. More importantly, the cost/benefit analysis performed during system development can provide the standards for the performance management of an operational system.

The environment in which an operational information system exists during its life-time is a changing one. Workload requirements change, data files grow, and hardware is enhanced or replaced. By periodically reassessing the identified costs and benefits of the information system, maintenance and modification tasks may be initiated to ensure satisfactory system performance in light of a changing environment. This application of cost/benefit analysis as an auditing tool can pro-

vide significant external information to the process of performance management.

Lastly, cost/benefit analysis as an on-going procedure can provide the necessary information for the all important decision of when to initiate a complete system redesign. Information systems should enjoy only finite life-times. Costs of operation and associated benefits to the organization will eventually exceed reasonable limits and maintenance and modification activities will be insufficient to correct these problems. At such a time, the life-cycle of an information system must be reinitiated.

A novel approach to the observation and analysis of a continuing system is provided by Grochow [1971] who attempted to measure and tune the performance of a time-sharing operating system. Grochow's approach was not to tune the system to maximize CPU utilization or minimize page faults, but rather to maximize user satisfaction. To accomplish this, he devised a utility function which addressed the user-oriented issues of system availability, system reliability, and response time to different classes of tasks. He estimated the shape of the utility function by interviewing users of the system and was able to assess their levels of satisfaction over parameter ranges of three variables. With this empirical utility function, anticipated modifications to the time-sharing system could be measured with respect to their impact upon the user community. In much the same way, the performance management of an operational information system must be sensitive to the user community. External, off-line, measures of user satisfaction with the information system are important factors to consider during operation and may, after analysis, initiate maintenance and/or modification activities to insure user acceptance.

3.4 Internal Performance Management

Effective performance management of an operational information system must combine external and internal measures. In contrast to the external performance measures which are somewhat ad hoc in nature, the internal performance measures discussed in this section are more highly developed, readily available, and easily observed and analyzed.

Widespread interest in computer performance measurement and evaluation has resulted in the development of numerous methods for the collection and analysis of computer performance data (General Services Administration [1977]). These methods vary from sophisticated and expensive hardware attachments to vendor supplied software monitors and lastly to the often overlooked, operator's log and system malfunction reports.

Caution should be exercised in the use of these measures as it has been demonstrated by Stevens [1978] that many of the indicators in widespread use are actually misleading.

Hardware monitors are stand-alone electronic devices which are attached at various points to the main computer system hardware. They have several advantages over software monitors in that they do not induce any unwarranted overhead to the operation of the computer system and they can probe aspects of the hardware performance which would otherwise be unobservable.

Hardware monitors are used to observe device utilization, such as the CPU, channels, and secondary storage devices. They do not observe individual tasks operating in a multiprogrammed environment. They can, however, still be useful in the observation and analysis of an operational information system. It can safely be conjectured that a limiting factor in the performance of an information system will be its efficiency in I/O performance. Possible uses for hardware monitors then might include observation of activity to devices upon which information system files are stored. Maintenance activities might involve a redistribution of files across secondary storage devices or channels to decrease device/channel contention within the information system. Additionally, for on-line information systems supporting a large number of remote terminals, hardware monitors may be used in the analysis of communication bottlenecks. If such problems exist, hardware and/or software modifications may be made to alleviate the problems, thereby prolonging the operational effectiveness of the information system.

Software monitors differ from hardware monitors in that they are typically event driven software systems which log various occurrences during the operation of application programs. These monitors are often supplied by the computer system vendor as part of the operating system. The individual installation has the option of using a software monitor and may select those events which are to be logged. Typical events which may be recorded are job initiation, job termination, interrupts, and various input/output operations.

Software monitors do incur a penalty of increased operating system overhead and may be a source of performance degradation. Most software monitor systems may be turned "on" and "off" to afford periodic snapshots of application program performance with minimum interference.

The analysis of software monitor logged data should be made by an experienced systems programmer. In spite of the most careful detailed

system design and testing, the true operational characteristics of an information system will not become evident until the system is in full operation. In analyzing software monitor data, various maintenance and modification projects may be deemed appropriate. For example, different subsystems within the information system may access the same files and consequently cause a deadlocking situation to arise. If the subsystems can be scheduled independently, the problem may be avoided. Additionally, in virtual memory computer systems excessive paging may be a source of overall performance degradation. A simple re-linking of system components may alleviate this problem.

As part of the performance management program for observing and analyzing information systems, these performance monitors, if available, can provide a great deal of insight into actual system performance. Because the environment of an operational information system is continually changing, periodic observation and analysis should be carried out.

A final source of information to a performance management program is the operator's log. This manually maintained data is often overlooked and the information it contains may not be reported to the individual responsible for the information system. In most computer centers, operators record events such as system crashes, disk volume mounts, tape mounts, and hardware malfunctions. If any of these events concern the operational information system, then maintenance and modification activities may be warranted.

The internal performance management tools described above can and should be used to continually assess the overall performance of an operational information system. Recently, attention has been focused on analyzing the performance of a part of a total information system -- the database. It has been recognized that the most volatile aspect of an information system is its data files. While user requirements are sure to change over time, they do so relatively slowly with respect to the changes that take place in files through insertion, deletion, and update of stored records.

One of the most common maintenance activities performed during the operational life-time of an information system is the periodic reorganization of a database. In many information systems, the storage structure and access methods used handle insertion of new record occurrences by means of overflow areas on secondary storage devices. As these overflow areas begin to fill with inserted records, system performance for retrieval applications degrades. Similarly, deleted records are often merely flagged and the space is not recovered. An

important consideration for file systems of this type is when should the data files be reorganized in order to move inserted records to their proper positions and recover deleted space.

During observation of an operational information system this problem may be detected in a number of different ways. Externally, users of the information system may express dissatisfaction with progressively deteriorating response time to queries or increased run times for application programs. Internally, performance monitors may indicate poor performance by recording excessive numbers of I/O requests as the overflow areas become more heavily used. Additionally, many file systems maintain logs of all insertion, deletion, and update transaction for backup purposes. Analysis of the log tape will indicate the volume of insertions and deletions which may be used as a measure of file degradation.

This problem has been known for a long time and affects many operational information systems independently of their particular application. The solution to the problem is proceeding along two different fronts. First, there is a cost associated with reorganization of a file and there is a cost incurred by user applications when overflow areas are used. The solution then becomes one of determining when these costs are equal. Shneiderman [1973], Yao, Das, and Teorey [1976], and Tuel [1978] have examined the problem and have postulated appropriate cost functions. With these cost functions they analytically derive optimum reorganization points for data files. To adapt their work to a continuing performance management program, the various cost functions must be parameterized to reflect the particular information system, file system, and hardware being observed.

The second approach to solving this operational problem is the more practical in that it removes the problem completely. Recent advances in storage structure and access method design allow for efficient insertion of record occurrences in their proper places and recover deleted space automatically. Storage structures based on the B*-tree developed by Bayer and McCreight [1972] represent approaches to the solution of the problem.

During the internal design stage of information system development, decisions are made regarding the logical structure of a database, the physical placement of record occurrences, and the determination of which access paths will be maintained. These design decisions impact the operation of the information system throughout its life. System designers have at their disposal certain tools to assist in physical design. Among these are the File Design Analyzer (Teorey

and Das [1976]), IBM's Database Design Aid (Hubbard and Raver [1975]), and the Database Design Evaluator (Teorey and Oberlander [1978]). The designer estimates system parameters and anticipated user activity and a proposed design is produced. During actual system operation, it may become apparent that certain parameters were estimated incorrectly or that important considerations were overlooked. Also, information systems are subject to change over time. Files grow in size and patterns of usage change thereby invalidating the basis of the original design.

A very promising area being investigated now deals with the question: What can be done during an information system's operational life-time to aid the system in adapting to a changing environment? The answer lies in the continuing observation and analysis of the operational system's characteristics and the use of the results of the analysis to guide the evolution of the information system.

The extent to which an information system may respond to a changing environment partially depends upon the level of logical and physical data independence. With a high degree of physical data independence, self-organizing file schemes may prove beneficial in organizing the "active" portion of the file into a small number of contiguous blocks thereby reducing costly I/O activity. If and when user activity shifts to a different portion of a file, the filing scheme will respond accordingly.

At a higher level of consideration, the maintenance and use of secondary access paths to shared records affects system performance. Schkolnick [1974], King [1974], and others provide design aids which suggest optimum sets of access paths. A cost model which balances the cost of secondary index maintenance with the benefits of reduced I/O activity is used. These suggested designs also tend to lose their effectiveness after the information system has been in operation.

Continuous observation and analysis of an operational system can yield valuable insight into changing requirements. Hammer [1976] has proposed a methodology for collecting and analyzing access information and then allowing for the dynamic creation and deletion of secondary access paths. This methodology is quite promising for future systems. The present drawback is that this type of self-adaptive behavior demands a very high level of data independence. Present day database management systems typically allow the user to be aware of secondary access paths, and any scheme which dynamically creates and deletes these paths would seriously affect applications.

The last topic for discussion concerning observation and analysis of operational information systems deals with the maintenance of software.

Recently, Halstead [1977], Kolence [1975], and others have investigated the measurement and prediction of software development and quality. An excellent review of this recent work is given by Fitzsimmons and Love [1978].

Halstead's theory of software science is particularly applicable to the consideration of large-scale software development as well as maintenance. By cleverly adopting long known physical laws, particularly from thermodynamics, Halstead has postulated a set of "laws" governing program length, programming time, programming effort, and a predictor of program design bugs.

In the survey by Fitzsimmons and Love, numerous experiments are reviewed and a remarkable correlation between Halstead's predictions and actual software practice are cited. Perhaps the most interesting result concerns the prediction of the number of bugs to be expected in operational software systems. The predictors involve easily measurable quantities derived during system development. At the time of system implementation, the number of residual bugs may be estimated and may be used as a measure of the maintenance and modification effort which will be required during the remainder of the system life cycle.

4.0 Maintenance and Modification

4.1 Introduction

The cost and manpower effort expended on maintenance and modification of software system has often been much larger than expected, hidden in various budgets, and hard to estimate beforehand and/or determine after the fact. It is now commonly recognized that the cost of maintenance and modification exceeds that of initial design and construction. Estimates of the rate of maintenance and modification cost to design and construction cost range from two to fifty.

Lientz, et al. [1978] described the results of a survey of 69 organizations regarding maintenance and modification activities. The results of their survey were as follows:

a) maintenance and modification do consume much of the total resources of systems and programming groups
b) maintenance and modification tend to be viewed by management as at least somewhat more important than new applications software development
c) problems of management orientation tend to be more significant than those of technical orientation

d) user demands for modifications constitute the most important management problem area

The purpose of this section is to address the issue of maintenance and modification in the following manner:

a) present working definitions for "maintenance" and "modification"
b) describe some reasons why the maintenance activity is viewed as undesirable by programmers
c) present some suggestions to improve the maintenance function
d) describe what may be done in earlier phases of the life cycle to reduce maintenance and minimize the impact of modification

4.2 Definition of Maintenance and Modification

The dictionary provides the following general definitions:

maintain - to keep in existence or continuance; preserve; retain
modify - to change somewhat the form or qualities of; alter partially

With respect to computer-based information systems, the distinction between "maintain" and "modify" is often not as precise as may be indicated by the general definitions. When an activity ceases to be maintenance and becomes modification is not always clear, and what one organization calls maintenance may be called modification in another. One area of disagreement is in the classification of changes which improve the performance of a system. Most often, maintenance activities are those which correct programming and design flaws and those which change portions of a module or program in response to a requirement change. Swanson [1976] provided a classification of three types of maintenance activity. Briefly, these activities are: corrective maintenance (performed in response to the assessment of failures); adaptive maintenance (performed in anticipation of change within the data or processing environments); and perfective maintenance (performed to eliminate inefficiencies, enhance performance, or improve maintainability). Modification activities usually include sub-system or system design changes and result in the addition of functional capability.

System maintenance and modification involves users, programmers, and analysts, the same people that are involved in development activities. For many years, the data processing industry has implicitly operated as if development was hard and challenging, and maintenance and modification were easy. The definitions given above and past experience point out that it is not the magnitude or complexity which determines whether something is development or maintenance/modification, rather it is the purpose and/or point of initiation.

Just as organizations do not always agree as to the distinction between

maintenance and modification, they do not agree as to the desirability of a separate unit within the systems and data processing department for the performance of maintenance and/or modification functions. Some organizations require that the developer performs maintenance but have a separate group perform modifications. In most cases, when more than one group is involved, the maintenance and/or modification group as well as the user signs off on the system prior to implementation.

4.3 The Image of Maintenance

Historically, the task of software maintenance has been viewed as an undesired duty and responsibility by analysts and programmers. Some of the reasons for this attitude are real, some are only perceived. The reasons include the following:

a) the name of the function or activity. "Maintenance" implies a slight adjustment to something which someone else created. The adjustment is most often perceived to be only necessary to correct an error in someone else's work.

b) maintenance activities occur in a crisis environment. The system that needs to be maintained has been operational and therefore users are depending on it. While users are somewhat patient during modification because of their perception of the complexity of the task, maintenance is perceived by the user as easy and therefore to be completed immediately.

c) the system to be maintained is most often poorly documented. Liu [1976] claims that documentation is either non-existent, misleading, or insufficient.

d) both development and maintenance programmers most often over-emphasize program efficiency as a design or redesign criteria. This situation can have two types of negative effects. On one hand, the maintenance programmer is hindered by the fact that since "efficiency" and "maintainability" are often conflicting objectives, the program to be maintained is difficult to understand. On the other hand, maintenance programmers are often frustrated by inefficient programs and often spend too much time attempting to increase the efficiency of the program as opposed to concentrating on the task of maintenance.

e) the maintenance function is often viewed by management as a good place to provide "on-the-job" training for programmers. Because of this, the background and experience of maintenance programmers is too often limited. Consequently, many members of a maintenance team are often unaware of such issues as the long-term objectives

of the organization, the relationship of a system to others within the organization, and the evolution of a particular system.

In some sense, the current emphasis in organizations and by researchers on such techniques as software engineering and structured design is having a negative effect on the awareness of the importance of maintenance and modification activities. Some of the claims for these techniques lead one to believe that maintenance and modification problems will automatically be substantially reduced if the system is properly designed and constructed. However, in many cases in the past, it was not clear whether a system that was difficult to maintain or modify was in that condition because of events that occurred during design and development or because of events that occurred after initial operation.

In addition to reducing the cost and complexity of maintenance and modification through improving the techniques and procedures used in developing and testing (see Section 4.5), it is also possible to reduce this cost and complexity by improving the techniques and procedures used in the operation phase and the maintenance and modification phase themselves. The interest in changing the activities of these phases is motivated by the fact that many systems are often difficult to maintain because of events in an earlier iteration of the operations phase or the maintenance and modification phase. Often programming and documentation standards that were followed prior to initial operation are not followed during maintenance and modification. Similarly a system may have been easy to maintain when it became operational, but when efficiency problems arose during operation, the system was changed without regard to future maintainability.

4.4 Improvements to the Maintenance Function

These problems and negative attitudes toward the maintenance function have been addressed by some organizations (Daly [1977], Liu [1976], Izzo [1978], Podalsky [1977]). Some suggestions include the following:

a) the name of this function should be changed to something more positive such as "product support," "application system support," or "software support." This type of name more accurately conveys the importance and the need for creativity in this organizational function.

b) senior programmers should constitute at least 30% of the maintenance staff.

c) assign maintenance programmers to active design tasks for 50% of their working time. This suggestion and the previous one would

tend to increase the expertise of the maintenance staff as well as potentially lead to systems that would be designed with maintainability as an objective.

d) the project should not be turned over to the maintenance staff until it has stabilized through a number of iterations. During these first few iterations, the quality of the system and its documentation would hopefully be improved by the designers themselves. Placing them in a maintenance function for this short period of time may help them to realize that while the system is functionally correct it is also difficult to maintain.

e) develop explicit procedures for testing a system after it has been "maintained" but before it has been put back into operation. These testing procedures should address the unmodified portion of the system and the documentation as well as the modified portions of the system.

These suggestions and others are directed to the staffing and execution of this function. A broader management concept has been proposed, called software configuration management, which suggests that, beginning with the life cycle phase of testing, the system be under the control of a separate organizational function (Daly [1977]). Under configuration management, all code and documentation (original and proposed changes) must be accompanied by the proper supervisory approval and must include a test plan. This type of management structure in some sense applies the quality control normally used by hardware and system software manufacturers to manage system software to the management of application software within user organizations.

4.5 Changes to Previous Life Cycle Steps

The size and complexity of the maintenance and modification function have led many organizations and some researchers to address what can be done to reduce the required maintenance and modification of a computer-based system. Much of the work of structural design and programming (Peters and Tripp [1977]) and that of requirements definition and analysis are directed toward this goal.

Many of the current problems with computer systems can be traced back to the design objectives of a system. Most often these objectives are one or more of the following: produce a system that meets the functional requirements with minimum expenditure of resources, and/or produce a system that, when operating, consumes a minimum amount of machine resources. In order to produce systems that are easier to maintain and/or modify, the design objectives may have to change

(Daly [1977]). Two alternate design objectives are as follows:

a) perform the desired functional operations and minimize expected on-site maintenance costs
b) perform the desired function operations with minimal expenditure of resources (manpower and material) over the entire system life cycle.

The high cost of the maintenance and modification activities, have led many organizations and authors to question the basic structure of the life cycle as illustrated in Section 2 of this paper. Brooks [1975] and Podalsky [1977] contend that the life cycle is itself recursive and that, in addition, organizations often throw away systems sometime during the life cycle and start all over. While many organizations recognize these and similar facts, very few organizations have revised these life cycle management policies and procedures to reflect them. A related topic of increasing interest to organizations is the concept of "bread-boarding" or prototype development and use. The experience with these approaches is limited. However, their effect on the life cycle in general and maintenance and modification in particular could be substantial.

5.0 Summary and Conclusion

The cost and complexity of the process of building and maintaining computer-based information systems is now receiving widespread acknowledgment. Initially, each system was built and maintained based solely on the experience of the specific users, managers, analysts, and programmers assigned to the system.

Gradually, certain management, engineering, and scientific principles have been and are being developed by the data processing community or by a given operating organization which have the effect of bringing more structured, systematic approaches to the entire life cycle. These systematic approaces initially appeared in the middle phases of the life cycle, then in the early phases, and now, to some degree, in the later phases.

The changes within an organization with respect to life cycle activities are themselves often subject to an evolutionary cycle or series of phases. First, organizations recognize the problem in terms of excessive costs, delays, or redundant efforts. For example, in the early days of computing, the manpower inefficiency of writing programs in machine languages was recognized. Second, generally accepted rules, policies, and procedures are developed to organize or structure the specific life cycle activities. Third, formal methodologies and, in

some cases, computer-aids are developed and become the tools of the user and/or data processing personnel involved in the step. With respect to the programming phase, compilers and operating systems clearly place the industry in the third phase. With respect to design (internal and external), software engineering approaches moved the industry to the second phase, and recent advances indicated a possible transition to the third phase. With respect to operation, maintenance, and modification, the industry appears to be in the second phase although certain computer-aids are under development in the research laboratories (Su and Reynolds [1978] and Swartwout, Novak, and Fry [1977]).

Finally, just as automobiles and hardware have an identifiable useful life span, all software systems will at some point in time be no longer cost justifiable. The transition from the maintenance and modification phase of one system to the specification of user need for another system must be more systematically addressed. Recent work on both problem recognition and proposed solutions has appeared (Oliver [1978] and Fry, et al. [1978]). Formal approaches and, possibly computer-aids, hopefully will soon evolve both for more of the operation/maintenance/modification activities and the inevitable conversion step.

REFERENCES

Bayer, R. and McCreight, E. M. [1972]
"Organization and Maintenance of Large Ordered Indexes", Acta Informatica, Volume 1.

Boehm, B. W. [1973]
"The High Cost of Software", Proceedings of the Symposium on the High Cost of Software, Monterey, California, 1973, pp. 27-40.

Branscomb, L. [1976]
"The Everest of Software", Proceedings of the Symposium on Computer Software Engineering, New York, New York, 1976, pp. xvii-xx.

Brooks, F. P., Jr. [1975]
The Mythical Man-Month, Addison-Wesley, Reading, Massachusetts, 1975.

Daly, Edmund [1977]
"Management of Software Development", IEE Transactions on Software Engineering, pp. 231-242.

Fitzsimmons, A. and Love, T. [1978]
"A Review and Evaluation of Software Science", ACM Computing Surveys, Volume 10, Number 1, March 1978, pp. 3-18.

Fry, et al. [1978]
"An Assessment of the Technology for Data - and Program-related Conversion", AFIPS Proceedings of the National Computer Conference Volume 47, pp. 887-907.

Gansler, J. [1976]
"Keynote: Software Management", Proceedings of the Symposium on Computer Software Engineering, New York, New York, pp. 1-9.

General Services Administration (GSA) [1977]
Management Guidance for Developing and Installing an ADP Performance Management Program, Washington, D.C., July 1977.

Grochow, J. [1972]
"A Utility Theoretical Approach to Evaluation of a Time-Sharing System", Statistical Computer Performance Evaluation, edited by W. Freiberger, Academic Press, New York, New York.

Halstead, M. H. [1977]
Elements of Software Science, Elsevier North-Holland, New York.

Hammer, M.
"Self-adaptive Automatic Database Design", AFIPS Proceedings of the National Computer Conference, Volume 46, 1977, pp. 123-129.

Hubbard, G. and Raver, N. [1975]
"Automating Logical File Design", Proceedings of the First International Conference in Very Large Databases, Framingham, Massachusetts, pp. 227-253.

Izzo, J. [1978], personal communication.

King, J. L., and Schrems, E. L. [1978]
"Cost-benefit Analysis in Information Systems Development and Operation", ACM Computing Surveys, Volume 10, Number 1, March 1978, pp. 19-34.

King, W. F. [1974]
"On the Selection of Indices for a File", IBM Research, Report RJ1341, San Jose, California, January 1974.

Kolence, K. W. [1975]
"Software Physics", Datamation, Volume 21, Number 6, June 1975, pp. 48-51.

Lientz, B., Swanson, and Tompkins, G. [1978]
"Characteristics of Application Software Maintenance", Communications of the ACM, Volume 21, Number 6, June 1978, pp. 466-471.

Liu, Chester C. [1976]
"A Look At Software Maintenance", Datamation, Volume 22, Number 11, pp. 51-55.

Merten, A. and Fry, J. [1974]
"A Data Description Approach to File Translation", Proceedings of the ACM SIGMOD Workshop on Data Description, Access and Control, Ann Arbor, Michigan pp. 191-205.

Oliver, P. [1978]
"Guidelines to Software Conversion", AFIPS Proceedings of the National Computer Conference, Volume 47, pp. 877-886.

Peters, L. J. and Tripp, L. L. [1977]
"Comparing Software Design Methods", Datamation, Volume 23, Number 11, pp. 89-94.

Podalsky, J. [1977]
"Horace Builds a Cycle", Datamation, Volume 23, Number 11, pp. 162-168.

Schkolnick, M. [1974]
"The Optimum Selection of Secondary Indices for Files", Research Report, Department of Computer Science, Carnegie-Mellon University, November, 1974.

Shneiderman, B. [1973]
"Optimum Database Reorganization Points", Communications of the ACM, Volume 16, Number 6, June 1973, pp. 362-365.

Stevens, D. F. [1978]
"How to Improve Your Performance Through Obfuscatory Measurement", AFIPS Proceedings of the National Computer Conference, Volume 47, 1978, pp. 425-431.

Su, S. and Reynolds, M. [1978]
"Conversion of High Level Sublanguage Queries to Account for Database Changes", AFIPS Proceedings of the National Computer Conference, Volume 47, pp. 857-875.

Swanson, E. B. [1976]
"The Dimension of Maintenance", Proceedings of the Second International Conference on Software Engineering, October, 1976, pp. 492-497.

Swartwout, D., Deppe, M., and Fry, J. [1977]
"Operational Software for Restructuring Network Databases", AFIPS Proceedings of the National Computer Conference, Volume 46, pp. 499-508.

Teorey, T. J. and Das, K. S., [1976]
"Application of An Analytical Model to Evaluate Storage Structures", ACM SIGMOD International Conference on Management of Data, 1976, pp. 9-19.

Teorey, T. J., and Oberlander, L. B. [1978]
"Network Database Evaluation Using Analytical Modeling", AFIPS Proceedings of the National Computer Conference, Volume 47, 1978, pp. 833-842.

Tuel, W. G. [1978]
"Optimum Reorganization Points for Linearly Growing Files", ACM Transactions on Database Systems, Volume 3, Number 1, March 1978, pp. 32-40.

Yao, S. B., Das, K. S., and Teorey, T. J. [1976]
"A Dynamic Database Reorganization Algorithm", ACM Transactions on Database Systems, Volume 1, Number 2, June 1976, pp. 159-174.