General Disclaimer

One or more of the Following Statements may affect this Document

- This document has been reproduced from the best copy furnished by the organizational source. It is being released in the interest of making available as much information as possible.

- This document may contain data, which exceeds the sheet parameters. It was furnished in this condition by the organizational source and is the best copy available.

- This document may contain tone-on-tone or color graphs, charts and/or pictures, which have been reproduced in black and white.

- This document is paginated as submitted by the original source.

- Portions of this document are not fully legible due to the historical nature of some of the material. However, it is the best reproduction available from the original submission.

Produced by NASA STI Compliance and Distribution Services

77-1380

A MANAGEMENT APPROACH TO THE DEVELOPMENT OF COMPUTER-BASED SYSTEMS

R. Turn*, M. R. Davis and R. N. Reinstedt
The Rand Corporation
Santa Monica, CA

Abstract

Many organizations have experienced serious difficulties in developing complex computer-based systems, especially their software components. The problems include large cost overruns, schedule slippages, inadequate performance, and inability to use the system as originally envisioned. One major reason for such lack of success has been the inability of the management of the organization or the development effort to understand the need for a total-system management approach. In particular, acquisition of software and hardware separately with the hope of integrating them later does not work in complex systems. This paper outlines a management approach to acquiring computer systems which encompasses the whole system, with emphasis on the software, from the initial concept formulation to the support of the operational system. Expected improvements in the development process and organizational implication of this management approach are discussed.

MANAGEMENT PROBLEMS AND MYTHS

Many organizations have experienced difficulties in developing complex computer-based systems, particularly the software. Fred Brooks, in *The Mythical Man-Month*, likens large software development efforts to the struggle of prehistoric beasts trying to escape the tar pits: "Large and small, massive and wiry, team after team has become entangled in the tar. No one thing seems to cause the difficulty—any particular paw can be pulled away. But the accumulation of simultaneous and interacting factors brings slower and slower motion. Everyone seems to have been surprised by the stickiness of the problem, and it is hard to discern the nature of it." [1]

In the metaphorical tar pit have been systems such as airline reservation systems, operating systems developed by computer manufacturers, and a variety of systems developed by the military services. The experience with the latter, in particular, have contributed much of the rationale for the management approach advocated in the present paper. This approach is a generalization of conclusions reached in a study of computer resource management of the United States Air Force in which the authors participated. [2]

One major reason for the stickiness of the computer systems' development problem, especially in large military systems that include computers as subsystems, has been the inability of the top management and managers of the development effort to appreciate the subtleties of integrating computers with other systems. They have failed to appreciate the need for a total system management approach. Instead, they have developed hardware and software separately, hoping to integrate these successfully later both with each other and with the rest of the system. Some of the systems developed under this philosophy are still stuck in the development tar pit.

Underlying these unsuccessful management efforts is a set of management attitudes and beliefs—a set of myths—which do not reflect the true nature of computer systems and software and their development processes:

- Myth: Stating valid and complete requirements for a new computer system is a relatively simple task.
- Myth: Hardware and software can be purchased or developed separately and fitted together later, and still later, fitted into the administrative and procedural environment.
- Myth: Since software is somehow different from hardware, its acquisition must be managed differently.
- Myth: Management review mechanisms used in hardware development are superfluous for software (or are impossible to conduct).
- Myth: Many hardware inadequacies can easily be offset by simple software changes.
- Myth: Acquisition of software can be treated as a production-like process, similar to procurement of standard hardware.
- Myth: Software, once developed, hardly ever needs to be changed again.
- Myth: Support of software in operational systems is essentially the same as maintenance of the hardware.

Thus the problems in managing the development of computer systems are compounded by policy and organizational arrangements that reflect the above myths. Superimposed on this is a diffusion of responsibility and authority because of management's inadequate understanding of computers. There is a serious scarcity of personnel who are technically equipped and sufficiently experienced to make sound decisions in the management of computer system development.

BASIC PREMISES

Considerable information was collected during the cited computer resource management study on projects that have involved computer system development. This resulted in four basic premises that we believe to be fundamental to successful management of computer system acquisition.

System Approach

The phrase "computer system" usually refers to the combination of hardware and software needed to perform some task. In its broadest context, the phrase also includes hardware maintenance,

*R. Turn is now with TRW Defense and Space Systems Group, Redondo Beach, California.

Copyright © American Institute of Aeronautics and Astronautics, Inc., 1977. All rights reserved.

software support, documentation, data sources, personnel, administrative practices, and operational procedures. However, it is common to find both the acquisition and operation of these various parts managed separately with little or no attention to the overall integration problems.

If tradeoffs are to be made, a computer system development project must be managed as a total system, rather than as a series of separate parts. Although the parts may be different in some sense, the differences amplify rather than reduce the need to consider software and hardware as closely interrelated components of the same system, rather than atomistic entities to be managed separately and differently. It is especially important to address the hardware-software interaction during the initial phases of a project in which new software is being developed for a new hardware system.

Interactions between hardware and software are many; unless their development is closely integrated, they may each accumulate design characteristics that are inconsistent with the other, and these inconsistencies may not be visible until late in the development process. The necessary reconciliation can be very costly and time consuming.

Similar Management Procedures

An examination of computer systems in different functional areas (such as military command and control systems, on-line transaction-driven systems, management support systems, and computers embedded in industrial or military systems) has failed to reveal any justification for different management approaches. One argument that has been used in the past is that the difficulty of defining operational needs and functional specifications for a certain computer system is so great that an incremental* development approach is necessary. It is true that some systems can be specified with high confidence that subsequent events will not soon invalidate the details, and that in others there is a substantial risk that the specifications will need to be changed before the system development has been completed. It even may be necessary to perform operational tests of system components or prototypes before preparing complete functional specifications. However, these differences are not fundamental. A structured management doctrine applicable to well defined systems is equally applicable to a phased incremental development.

*The words evolutionary and incremental are used in a variety of meanings when applied to computer development. In the present study, we have adopted a precise set of definitions summarized as follows:

Evolutionary requirements are achieved by incremental development according to a phased program where each phase is a complete pass through an orderly process of development and produces an increment of capability.

The inappropriate phase "evolutionary development" can give a manager the impression that the process of development does not have to be managed or controlled, but can be allowed to muddle forward with a guarantee of successful outcome. Thus, we prefer not to associate "evolutionary" in any way with development, but confine it to a descriptor of system requirements.

Use of Development Management Methods

Acquiring computer systems is frequently treated as a simple production task, when in fact it is a development process which may demand sophisticated research and development management techniques. Both hardware and software components of a new computer-based system require highly disciplined development management because they share several characteristics:

- Both involve innovation, creativity, and ingenuity throughout the development stage. These aspects of the development process make precise scheduling, cost estimation, and performance prediction difficult.
- Both are subject to piece-part testing during the development phase and testing of the complete system to validate the peoper integration of its many parts and components.
- Both are vulnerable to tinkering and modification during development and in operation, so that configuration control becomes an important and essential function. This is especially true for software because of the ease of making seemingly simple changes without adequate management involvement and with unperceived implications for the future.

Software acquisition for a new computer system must be managed as a structured development process not only for the above reasons, but because of its special nature. For example, although the basic laws of physics clearly constrain the performance of hardware systems, and these constraints are widely recognized and understood, constraints on software performance are poorly defined and incompletely known. Many aspects of software design, performance prediction, and resource estimation are based only on experience with past projects of a similar nature, with the following results:

- Cost and schedule estimates may have large uncertainties.
- The correct design approach may not become apparent until after a substantial amount of preliminary design and development work. Prototypes, simulation, and even field tests may be required.
- The body of specifications, design standards, preferred practices, and documentation standards is much less well developed for software.

These characteristics raise special problems for software acquisition. Current management procedures are based on the premise that reasonably precise cost and schedule estimates can be made at the beginning of full-scale development and that there exists a body of specifications and design standards; in fact, however, the art of software development has not yet reached this degree of stability and predictability.

Because software has no significant production phase, its development cost is a significantly large fraction of its total life-cycle cost in comparison with hardware. This fact accentuates the need for both complete specification of functional requirements and thorough reviews of preliminary and detailed designs prior to initiation of full-scale software development.

Life-Cycle Management

Both hardware and software components of large computer systems require considerable support after delivery to the user. Hardware needs maintenance because it deteriorates over time. Software does not "wear out" in the usual meaning of the term (and, hence, the phrase "software maintenance" is a misnomer), but inevitably there are design oversights and errors not caught in testing. Furthermore, in the operational phase software invariably requires substantial amounts of support for technical improvement. As a consequence, support tends to be a major item in software life-cycle cost.

The mechanics of changing software are straightforward and simple, and the replication of changes in many copies of software is also straightforward. This has led to the myth that software is easy to change. What is missed is that software is an intricate fabric of interactions and mutual dependencies; changes can affect other parts of the program in hard to perceive ways and, more often than not, can introduce new problems and anomalies. Consequently, it is very difficult for anyone but the original developer to successfully make major software changes without a substantial investment in becoming familiar with the program. Ideally, the original developer should support the software throughout its operational life. Alternatively, the organization ultimately responsible for operational support must be thoroughly involved in the development, participate in the acceptance tests, and thoroughly understand the design details.

TOTAL SYSTEM MANAGEMENT APPROACH

In this section we will outline a management approach to the development of computer-based systems which we believe can avoid the development "tar pits." We will discuss the salient features of the necessary management policy, outline an orderly development process, and discuss the organizational aspects of its implementation.

Management Policy

Any organization with extensive commitments to the use of computer-based systems must adopt a policy at the highest management levels to fully support the development of these systems. This policy must prescribe a development process which conforms with the basic premises identified in the previous section:

- Adopts a total system approach in the acquisition of system hardware and software and in the integration with other systems.
- Applies uniformly (but on a scale commensurate with the nature and size) to all development efforts within the organization and addresses the total system life cycle.

A model of such a process and its implications to managing computer system acquisitions are discussed below. The management policy must also reflect other prerequisites to successful development management:

- Recognize that acquisition of a computer-based system, especially its software component, is a development process that must be managed as such, particularly with respect to management attitudes, procedures, and controls.
- Recognize the iterative nature of the development process.
- Recognize that the cost of rectifying design flaws discovered in the later phases of the development process is much greater than in earlier phases, and that it is important to place great emphasis on thorough reviews of the system specifications and design proposals.
- Recognize the importance of constructive confrontation in design reviews and the necessity of establishing review groups that are independent from end-users and developers.

Even if the organization adopts a policy along these lines, not all of its elements are likely to welcome the new controls placed on them. Therefore, it may be necessary to establish within the organization a special entity--a focal point for computer system development--which has the necessary expertise and is placed high enough within the organizational structure to successfully enforce the development policy and, if necessary, confront the existing power structure. An illustration of such a focal point is the recently established Assistant Chief of Staff for Communications and Computer Resources in the U.S. Air Force.

The Development Process

Acquisition of computer-based systems must be treated as an orderly process that encompasses the entire life cycle of the system, from the statement of a function to be automated to supporting the operational use of the system acquired for this purpose. Figure 1 depicts various phases of this process. Each phase is represented by a rectangle that also shows its most important subfunctions. Line coding is used to indicate the different organizations, or elements of an organization, that are principals in each phase; heavy solid lines indicate users, light double lines indicate independent review groups, and light single lines the computer system design and development group.

There must be sufficient organizational independence between these elements to encourage constructive criticism and to avoid compromises of the function of one element by the authority imposed by another. For example, slippage in development schedule must not be arbitrarily compensated for by shortening the testing time unless the end-user and all relevant management levels understand and accept the performance risk that will be incurred.

The iterative nature of the development process is underscored by the many feedback paths between functions and phases. If design reviews are established at appropriate points in the development process, and if it is understood by all involved that a review may result in repeating a previous part or parts of the development activity, then the process can proceed in a controlled and rational manner.

Finally, it is important to emphasize that the process applies equally well to all systems:

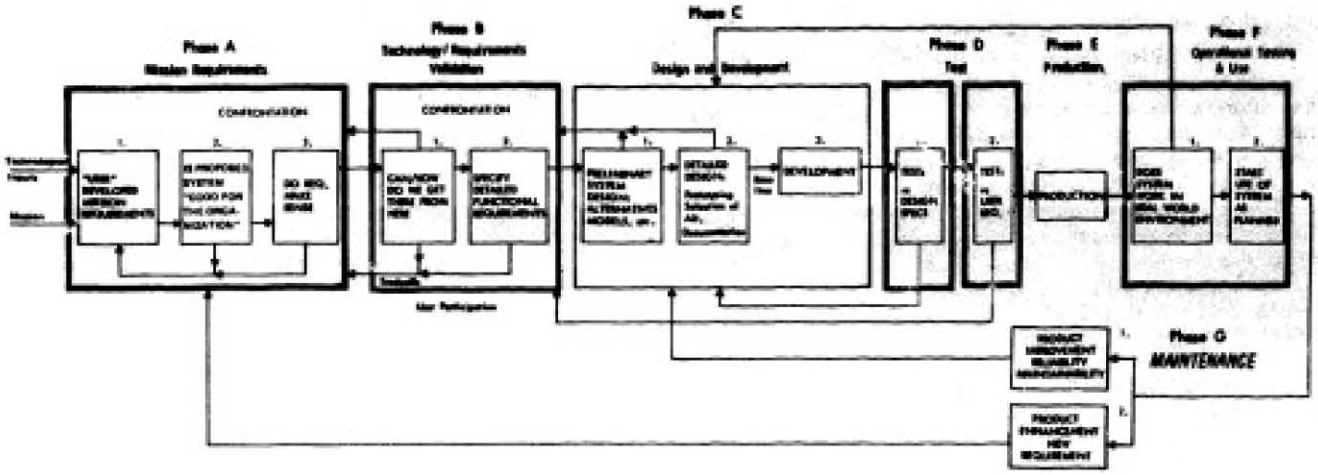


Figure 1. Computer System Development Life Cycle

to a turnkey system, to each phase of an incremental development of a system, or to each step forward in the phased progression of a system.

Mission Requirements Formulation. This phase (Box A in Fig. 1) is exceedingly important because it provides the performance inputs that drive the rest of the system development process. It is a relatively informal and sometimes lengthy activity which must be controlled by users rather than the developers. When the operational requirements are defined to the extent that development can be considered, they must be submitted to a process of review and confrontation (phases A2 and A3). In phase A2, the stated need is reviewed from a top level organizational point of view, looking at the functional rather than technical aspects of the requirements. Among the criteria that must be considered are:

- Does the organization require the proposed capability? Is it consistent with the long-range goals, policy, and planning of the organization?
- Are there existing systems within the organization that could be used to satisfy this need?

These questions serve a dual function of establishing the overall importance of the proposed system, as well as alerting those who may become involved in adapting the system to a standard organization-wide use.

Given a satisfactory review of the stated need, it is important to address the adequacy of the performance requirements that have been proposed (phase A3). It must be determined whether the requirements make sense in that they accurately represent what is needed. Among the questions to be asked in such a review should be: Is an automated system really necessary? Are the requirements stated clearly enough to be understood by the developers and by the users themselves? What impact will the proposed capabilities have on other functional areas?

After the above reviews have been performed by impartial groups and confrontation has taken place, the requirements can be documented and submitted to the Technology/Requirements Validation process (phase B). However, it is not implied that phase B's function will be performed only on a fully approved set of requirements. Even if requirements are too uncertain for formal approval so that further study by the use is required, some of the phase B processes can be helpful. The need for exploratory work during the requirements formulation process should be explicitly recognized and provided for in the management procedure.

Technology/Requirements Validation. The goal of this phase is validation of the technical and economic aspects of the requirements. This process should encourage confrontation between users and developers to ascertain the technical feasibility of implementing a system to satisfy the stated requirements. The user organization should be in control, but participation of an independent review group is required to identify and analyze technical problems and to perform tradeoffs. Satisfactory answers must be given to the following: What degree of technical risk or uncertainty is involved in meeting the requirements? Is any research indicated? Is the projected schedule appropriate? Are there any special technical aspects that require extraordinary management measures?

After completion of the technical review--and there may be several iterations of modifying functional requirements and evaluating the technical aspects of their implementation--the requirements and technical inputs are amalgamated into detailed functional requirements. These are again reviewed (in phase B2) by an independent group to determine whether:

- The detailed functional requirements specifically and accurately define the desired capabilities and the necessary resources, and whether there is an accurate match between the user's stated mission and the requested resources.
- Beneficial tradeoffs can be identified from an examination of the requirements to present to the user choices such as a less ambitious but less costly system, or a system that, while

providing less, can become operational at an earlier date

- Adequate provisions for technical management have been incorporated in the proposal to secure successful implementation and visibility to management.

A substantial amount of time and technical work, including the development and testing of a prototype system or a portion of one, may be needed to satisfactorily answer these questions.

The final products of this phase are a set of detailed specifications which have been subjected to thorough functional and technical reviews, and a management approach consistent with the technical characteristics of the project.

Design and Development. This phase should be performed by an organization independent of the user organization. This phase proceeds into the preliminary system design step (C1) in which analyses and tradeoffs are made to produce a set of design alternatives that are evaluated on the basis of various criteria such as performance, cost, and reliability. Important management questions in this phase include:

- Does the preliminary system development plan, especially those parts that include computer equipment and software provide adequate visibility to management?
- Is there a plan for proceeding through the development process in an orderly manner?
- Is there a plan for orderly transition to the user and to the support organization? Such plans are especially needed for software, for which specific provisions must be made during the design phase to allow subsequent testing, integration, and support.

Upon completion of a preliminary design review, the detailed design of the system (phase C2) commences. An iterative process is to be expected. The final design must also be subjected to a thorough and critical design review, it is important that the user participate in this review. Among the questions to be asked are:

- Is it clear that the correct design choice has been made?
- Have relevant hardware-software tradeoffs been adequately performed?
- In view of the technical insights now available, are the requirements properly stated? At this critical juncture in the development process, this fundamental question must be reviewed.
- Since more information is available now than when the estimates were first made, does the additional information indicate that new cost and time estimates should be made?
- Does the original design strategy still appear valid? Does the system development plan still seem appropriate?

The development of the system (phase C3) implements the detailed design in a form that can be tested and, in the case of hardware, passed into production. For software, the final development step consists of writing the computer code in the chosen programming language. Except for testing, this completes the software development process since no significant production step is involved other than making error-free copies and completing the documentation. Throughout the development phase, the project management must remain aware of considerations such as:

- Are there any technical problems to be solved before proceeding with development?
- Is development conforming to the program management plan?
- Are milestones, benchmarks, and other checkpoints being set?
- Are factors emerging that may require major revisions in the development process, the schedule, or the technical approach?

With the completion of the development step, the system is ready for a sequence of tests.

Testing. The testing phase comprises two distinct steps involving distinct organizations. The first step is not necessarily totally separate from the development phase--testing of subsystems can proceed as soon as their design and implementation are complete. Testing of the system performance against the design specifications (D1) must be performed by a technologically competent group independent of the system developer. The principal question is whether the system performs as specified in the detailed design specifications produced and accepted in phase C2.

The second step (D2) must be performed by the eventual users to test the system's functional performance--does it satisfy users' needs? It may turn out that even though the functional specifications are fulfilled, the system is not truly responsive to the user's requirements, necessitating a return to an earlier phase in the development process. For example, if the functional requirements are difficult to define, they may not be implemented exactly right at first. In other cases, the system capabilities may need to react to changes in high-level policies as well as to possible changes in operational environments. However, in any such instance the system may still be able to provide an initial increment of capability while development to meet the full needs continues. In any case, the development process will have produced a basic design and an initial increment of capability.

Consistent with the emphasis placed on independent review of each phase of the development process, a special review group must oversee every major testing activity, evaluate system module and integration tests, and assure that the tests specified in the management plan are conducted and their objectives met.

Production. In hardware systems, completion of the test step is followed by conventional factory production (phase E). For software, the production phase is minimal--the automated preparation of error-free copies by computer of the original programs for distribution to the users. The absence of a software production phase is one major distinction between hardware and software, especially in terms of funding. Nearly all of

the software acquisition cost consumes development funds, thereby putting additional strains on a traditionally tight budget category.

Operation and Support. The final test of the system's ability to satisfy the need defined by the user in phase A comes in the actual operational use of the system in the real world environment. At this point, the user has formally accepted the system from the developer, and the responsibility for support is transitioned to the user or to a supporting organization. Transitioning the system is an important step easily overlooked in the early phases of the development process, especially in embedded computer systems. Consequently, in numerous cases the necessary support equipment and software have not been ready when the system became operational.

The maintenance function (phase G) assures that the system remains operational despite the natural deterioration of the system's hardware. Since there is no intrinsic wear-out of computer software, software support mainly involves product improvement (increasing efficiency, reliability, or supportability), product enhancement (providing new functional capabilities), and correcting design oversights and errors. However, software changes can force a return of the system to earlier phases of the development process than of hardware changes. Managers who believe that software is easy to change are apt to ask for software improvements or enhancements much more casually than for hardware changes, not recognizing that their requests may require a repetition of the entire development process.

One consequence of the special nature of software support is that software is always at some level of continuing development; it therefore appears "never completed" to managers and auditors who believe the myth that "software can be produced as a turnkey system."

Organizational Implications

Although the process depicted in the life-cycle model is normally followed in the development of major systems, its application to computer-based systems has been sporadic. However, case histories show that skipping any phase in the orderly process schematized in Fig. 1 almost guarantees that the system in question will not meet performance, schedule, or cost expectations. It is essential that the development of computer resources be subjected to the discipline represented by this model, and it is necessary to apply vigorous and independent management review practices to assure that these vital procedures are adhered to.

Numerous organizational issues arise: Who should perform each function? Who should be responsible for the various reviews? Some organizations have combined the tasks of development and testing under common management; still others have split the design and development management to address hardware and software issues independently.

Furthermore, in some organizations computer resources are managed centrally from the organization's headquarters, in others they are managed in a decentralized fashion by functional elements of the organization. Whatever the structure, the success of the management approach we have described depends on the establishment of a focal point* for computer acquisition matters, who will provide technical expertise for conducting management reviews prescribed in the development process. Other changes in the organizational structure may be needed at the functional users' level, especially for requirements generation.

Focalization. The principal purpose of establishing a computer resource management focal point in an organization is to set up an organizational element that can assure the disciplined application of management reviews and controls throughout the development life cycle of a computer system. More specifically, the purposes of focalization are to:

- Develop and assure the application of uniform policy and management controls to all computer system acquisitions.
- Assure the presence of independent technical expertise at management decision points in the organization, and provide improved awareness of computer systems, especially their costs, to the top-level management.
- Assure the exploration of technical alternatives early in the requirements identification and validation process, and in early development phases, so that development risks can be identified early. From his vantage point of authority, and with visibility and technical expertise, the focal point can assure that the recommended confrontation in the development process will occur. This minimizes the risk that the organization will apply computer resources to unproductive goals.
- Provide a central point of contact within the organization to facilitate interaction between users, developers, and the computer community at large. Minimize counterproductive conflicts between different functional entities of the organization by possessing the authority to adjudicate any such disputes.
- Assure visibility to management of computer resources costs by collecting information on the financial aspects of resource acquisition and operation. Serve as a corporate memory of the difficulties experienced as well as successes achieved in computer system development.

To function effectively, the focal point must not serve as an initiator or participant in generating user's requirements, nor should this group be an advocate of any proposed computer system until after it has reached the full-scale development phase.

The position of the focal point within the management structure of the organization is crucial. In those organizations that presently experience difficulties with developing computer-based systems, or where the goal is to centralize a currently decentralized management of computer resources, it is essential to provide the focal point with a strong

*The focal point for computer resources management is an organizational entity represented in our discussion by its director.

starting position in the organization. For example, he could be made a vice president for computer resource management or an assistant to the president. Placing the focal point in an existing office (e.g., in the office of the comptroller, the traditional home for management of computer systems) is acceptable only in organizations in which computer systems are predominently used in that specific functional area.

The position of the focal point in the organizational structure is acceptable if he has:

- o Stature to enable productive confrontation of the existing power structure and to institute new policy.
- o Access to the top-level management in the organization and participation in the formal and informal decisionmaking activities of the organization (establishment of the focal point must transmit an unambiguous and strong signal that the top management is determined to solve its computer system development problems).

Furthermore, the focalization concept should be established not only at the highest management levels in the organization but extended to the lower-echelon organizational entities and functional areas (e.g., a computer resource single manager hierarchy in the U.S. Air Force).

Separation. It is important for effective development of computer systems that the organizational entity which develops the operational requirements for a computer-based system be organizationally separate from the design and development groups. The user must be intimately involved with requirements generation and must stay connected to the development process: Functional and technical compromises and adjustments are inevitable, specifications must be interpreted, and requirements may change sufficiently to justify changing the specifications. However, the user cannot be allowed to exploit his interaction to the detriment of the development process by interjecting a continuous stream of requirement changes as he conceives new ideas or refines his perception of needs. While he must be deeply involved in and even control the initial requirements generation phase during other phases he must interact only in a carefully controlled way under strict management control. Even in an incremental development approach to meet evolutionary requirements, each cycle of progress must be completed before expanded or revised requirements initiate the next cycle.

Such a separation of responsibilities and careful control of interactions must be exercised everywhere in the organization where the development process takes place. A user must be given the resources and the responsibility to define the systems that increase his effectiveness, but the development responsibility lies with other groups.

CONCLUDING REMARKS

Computer technology is essential in the operations of large organizations both in the government and in private business and industry. Past experiences of many of these organizations in developing computer-based systems have been disappointing--costs have exceeded estimates, schedules have shifted, performance has been unsatisfactory, and envisioned benefits have failed to materialize. Future computer applications will be even more extensive, complex, and critical than those whose development efforts have struggled in the metaphorical development tar pit. It is imperative for the successful development of such systems that the lessons of the past not only be learned but also put to use. We are convinced that computer systems can be developed as effectively as other complex systems, provided that (1) the unique nature of computers and computer software is recognized; (2) a development policy based on the application of an orderly development process is formulated; and (3) certain organizational entities are established to implement this policy and perform management reviews.

ACKNOWLEDGEMENTS

The authors gratefully acknowledge major contributions to this paper by their colleagues at The Rand Corporation, the directors of the Air Force Computer Resource Management study, S. M. Drezner, H. L. Shulman, and W. H. Ware, and the study team members G. K. Smith, R. S. Gaines, L. E. Catlett, R. A. Pyles, A. I. Robinson, D. K. Shelton, and H. J. Shukiar.

REFERENCES

1. F. P. Brooks, Jr., The Mythical Man-Month, Addison-Wesley Publishing Co., Reading, Mass., 1975.
2. S. M. Drezner, H. L. Shulman, and W. H. Ware, The Computer Resource Management Study: Executive Summary, R-1855-PR, The Rand Corporation, Santa Monica, Calif., September 1975.