MITRE-Bedford Division

MTR-8304

Improving the Acquisition of Software Systems

W. S. Attridge
G. R. Kinney
S. B. Lipner

27 March 1981

UNCLASSIFIED

mitre

MITRE Technical Report

MTR-8304

Improving the Acquisition of Software Systems

W. S. Attridge
O. R. Kinney
S. B. Lipner

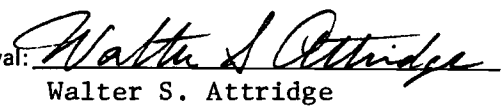
27 March 1981

CONTRACT SPONSOR	ESD
CONTRACT NO.	F19628-81-C-0001
PROJECT NO.	D11G
DEPT.	D-70

THE MITRE CORPORATION
BEDFORD, MASSACHUSETTS

Department Approval: 

Wesley S. Melahn

MITRE Project Approval: 

Walter S. Attridge

ABSTRACT

MITRE personnel with experiences in recent ESD software system acquisitions were surveyed for opinions on the practices employed in the acquisitions. The Review Team concluded that insufficient recognition is given to the developmental nature of software. Specific recommendations are advanced emphasizing the importance of prototypes, engaging with the contractor and insisting on software competence at high levels of program management.

ACKNOWLEDGEMENTS

The authors wish to acknowledge the contributions of John W. Shay, Wesley S. Melahn, Kenneth E. McVicar, and especially Alan J. Roberts who launched us on this brief study. Appreciation is extended to the several other MITRE opinion leaders who contributed their views based upon experiences with many ESD acquisition programs. Finally, thanks are due to Betty Aprile for her patience in casting the report into print.

FOREWORD

The principal objective of MITRE's work in support to the Air Force Electronic Systems Division (ESD) is to assist the client in the acquisition of command, control and communications systems. These are often computer-based information systems comprising data processing hardware and software. The uniqueness of the ESD systems invariably forces development rather than off-the-shelf procurements, especially for the software.

Historically development of software has been a complex undertaking and often a problem area; problems are further exacerbated when the development is acquired under government procurement regulations and practices. At ESD experiences have ranged from good to bad, as judged in terms of capabilities delivered matched against cost and schedule overruns. Sometimes such overruns or capability shortfalls should have been expected because of unrealistic initial goals. Sometimes even the most conservative plans have been missed due to poor actions by the development participants. Efforts are periodically made to try to learn from experience by recapitulating the good and bad practices. Thus we have summarized in this report the results of such a survey, which was precipitated by a recent cost and schedule overrun on one of ESD's programs.

The results of this survey point to the need for a creative acquisition strategy which recognizes and makes allowances for the imperfections in system acquisition policies and practices and the software development process. Let it be said directly: there are no panaceas. Rather there are several factors or principles, the combination of which when adopted should increase the chances of success. The failure to adopt any one of these factors in the acquisition strategy only makes the remaining factors that much more necessary. The aggregate objective and effect of these factors is to allow the government to understand the contractor's process and to build and maintain, throughout the contract period government confidence in his work effort and eventually the delivered product.

TABLE OF CONTENTS

TABLE OF CONTENTS - Concl.

LIST OF TABLES

SECTION I

INTRODUCTION

OBJECTIVES AND APPROACH

For many years ESD, with MITRE's support, has been acquiring software intensive C^3 systems. More often than not troubles are encountered during the course of the software acquisition. It is the purpose of this report to describe the results and conclusions of a review of recent software acquisitions and to document recommendations that may lead to an improved software acquisition process.

The software acquisition problems are typically manifested by:

- Cost escalation beyond the contract price, often with little advance warning;
- Schedule delays; and
- Delivered performance less than defined in the contract specifications.

In such circumstances it is evident that the acquisition program has "failed," but the question that cannot be so easily answered is "What should have been done differently to avert the poor outcome?"

In an attempt to answer this question the Review Team solicited the opinions of a large number of MITRE personnel knowledgeable about software system acquisition. All Technical Directors were interviewed briefly, and more detailed discussions were held with several Department Heads, Project Leaders and software opinion leaders. The one common thread of all these discussions has been that the management of software acquisition needs greater attention and much improvement. This finding is consistent with several other studies conducted in industry associations and government agencies of the same problem of software acquisition; this problem is not at all unique to ESD and its programs.

The present study focused on ESD programs and makes recommendations for actions to be taken by both ESD and MITRE in future acquisitions. The Review Team addressed the full range of activities from the formulation of the acquisition strategy through the development of the system, but concentrated mostly on

the actual developmental period between contract award and formal testing.

PLAN OF THE REPORT

Section II of this report provides some overall comments, based on the Review Team's interviews and observations, about the problems managing the acquisition of software-intensive systems. Sections III and IV identify "key factors" and "other factors" that the Review Team found to be important to successful acquisition of software-intensive systems. These factors are directly drawn from the Review Team's interviews and can be addressed (to a greater or lesser extent) by the managers of acquisition programs. The final section summarizes the team's conclusions and recommendations for further action. Although the opinions gathered are based upon actual experiences in various ESD programs, we feel it would serve no useful purpose in this report to make program-specific attributions.

SECTION II

MANAGING THE SOFTWARE ACQUISITION

INTRODUCTION

There are certain basic facts about any effort to build something, whether it be software or hardware, whether it be a computer or an airplane.

- Unless it has been done before, there is some element of risk. Unless there has been a similar model done before, there can be a large risk.
- In the course of building or developing something, there are bound to be unforeseeable problems requiring some change in plans and approach.
- Well understood and described objectives and methods for obtaining them, especially if based upon similar precedents will reduce but not eliminate risk.
- The successful program manager expects the unforeseen, stands ready to modify the efforts when required and minimizes the consequences of dead-end approaches.

For software acquisition programs such facts become even more significant, largely because the predictability of the implementation process is low or, said another way, there are innumerable possible implementations of a software system. Beyond that is the intangibility of software compared to hardware, again making it essential to keep the basic facts stated above in mind throughout the acquisition process.

MANAGEMENT STYLE

The acquisition of a software intensive system by ESD is managed by a System Program Office (SPO). SPO program management style has varied among the programs considered by the Review Team, ranging from the heavy involvement and active decision making control of a program where the contractor activity is collocated with ESD and MITRE to the distant posture with management direction limited largely to attendance at formal reviews. There are good reasons for the SPO to play an active role in the guidance of a contractor's

effort mainly deriving from the risky nature of most software development efforts as discussed above.

At the beginning of a program the major effort of the SPO is to describe in a specification the system to be acquired and in the Statement of Work (SOW) the work to be performed by the contractor. The proposal evaluation effort is an attempt to select an acceptable contractor who, through his proposal, gives assurance that he can accomplish the required work to build the system. This evaluation is the beginning of the process of establishing mutual and collective understanding of the program approach and system objectives. This understanding expands throughout the development program as more is learned about the system design and about the details of its operational performance. It is the nature of such development programs that problems will be encountered requiring redirection of effort. Ideally such redirection can be implemented at a low level as part of the daily routine of the software work, adequately controlled by the contractor's managers, and readily contained within the planned schedule and costs.

But when an accumulation of minor problems or a few major problems arise, the beginnings of schedule and cost impacts appear. It is here where SPO management presence must be applied. The SPO management must somehow be connected closely enough to the contractor's efforts so that events can be monitored and problems detected. The SPO, with MITRE assistance, may very well be able to help remedy technical problems. More commonly, help comes in the form of clarifying requirements. It is this kind of government-contractor collaboration that can best further the program's progress. Over the many months preceding contract award the government team has built up a significant base of knowledge about the program that must be readily available and carried forward into the contractor's development.

The Review Team recognizes a common concern of SPO managers that government involvement in areas that are clearly the contractor's responsibility can result in the risk of claims for extra charges against the government. The effective manager balances those kinds of risks against the risks of letting the contractor flounder and letting the program run into cost and schedule overruns.

A COOPERATIVE ENVIRONMENT

The Review Team concluded that the difference between success and failure in many of the system acquisitions reviewed could be traced to the management of the program. For an assertive management posture to be sustained by the ESD program manager, he

must have adequate ESD and MITRE technical support to help plan the overall acquisition strategy, to provide the technical system baseline at the time of source selection, to monitor the progress of the contractor and to provide recommendations for timely guidance and direction for the contractor.

The program manager must continually realize that the contractor's failure is the government's loss. Thus he has an important incentive to collaborate with the contractor from the outset; and, in turn, the importance of the ESD program manager gaining and maintaining valid confidence in the contractor's efforts cannot be overstated. Formality of reviews and reports is no substitute for (and in fact may hinder) good and cooperative working relationships. Upon the foundation of such basic attitudes a successful program can be launched.

SECTION III

KEY FACTORS IN SUCCESSFUL SOFTWARE SYSTEM ACQUISITION

The interviews conducted by the Software Acquisition Review Team convinced the team members that three key factors have a major impact on the success or failure of any software system acquisition. These factors -- prototype, engagement, and software competence -- are defined and discussed below.

In reviewing the history of a number of acquisition programs, it was apparent to the Review Team that there are no panaceas that guarantee the success of an acquisition program. Nonetheless, the three factors described in this section seemed to be highly correlated with overall program success. For this reason the Review Team believes that any program which fails to consider the material presented below is placing itself at risk of cost growth, schedule slip and program performance problems.

PROVIDE FOR A PROTOTYPE

It is the nature of requirements for ESD's C^3 programs to strive for a maximum amount of capability. These ambitious goals are tempered by analyses which factor in technical feasibility, cost and schedule such that the resulting program plan is judged to be manageable. Still, these systems are often complex and may explore new functions never before attempted, at least by the contractor involved, or the performance standards such as response time are much more demanding. In situations as these it is useful to build pilot models or prototypes of the parts of the system most crucial and risky.

Prototyping is a discovery process for system design and performance measured against requirements. A prototype effort will uncover risk areas and provide a sounder basis for cost estimating. In some instances the prototype will serve as the starting point for the FSED, but its value need not be predicated on only that possibility. Of course a prototype that does get implemented in a form that can be thought of as a "core capability" has the value of being step one in the incremental development of the target system. In such an instance, it incorporates the fundamental software architecture (e.g., operating system, data base management system, user-system interface), provides a realistic demonstration of its potential effectiveness with some application software, and permits early measurement and evaluation of typical functional performance.

The acquisition of a system prototype can proceed in a number of ways. In one case, a prototype for an Air Force system was "donated" by another government agency complete with hardware, system software, application software, and sketchy documentation. At the other extreme the development of the prototype was an initial phase in the system development contract. Both programs proceeded to be implemented with incremental expansions of capabilities. Their prototypes had the following attributes:

- Performed a significant fraction of the full system functions
- Ran on the full system hardware or a proper subset of that hardware
- Incorporated substantially all of the full system support software (operating system, display package, data base system, compilers)
- Provided a basis for testing, measurement, or prediction of the full system's performance.

The emphasis in any system prototype phase is on a demonstration of capability, rather than formality and documentation. It provides an opportunity for feedback to the user, "Does it perform the functions I expected?" and to the developer, "Does it work the way I intended?" Formal specifications and tests can (and should) be deferred until a subsequent full-scale development phase. As a practical matter, a successful prototype phase is closely tied to engagement which allows the substitution of government-contractor interaction for costly and time-consuming formal document preparation and review.

Inclusion of a prototype development phase and basing a system development on a government-furnished prototype are both legal and straightforward. While prototype development is likely to extend the planned schedule, the Review Team observed that the shorter schedules of programs that skip the prototype phase are not met. Competitive prototypes (not competitive design studies or concept definitions) are well precedented in aircraft development and can be useful as a way of assessing alternative approaches to a system that will enter production.

The Review Team speculated on the reasons why a system prototype phase is important to program success. These speculations centered on the developmental nature of new software systems -- especially large, complex new software systems to meet major DoD requirements.

Forcing such systems directly into the full-scale development process with its firm requirements, elaborate documentation, and formal interactions between government and contractor seems to overconstrain the development process. In particular, if it is accepted that a major software system is a developmental item, then the process of developing such a system from scratch should allow for relaxation of requirements, tradeoffs between components (especially hardware and software) and readjustment of cost and schedule projections. The full-scale development process is a more difficult environment in which to adapt to any of these changes.

The Review Team also observed two ineffective alternatives to a system prototype phase as discussed above. If prototype hardware and software are developed and discarded, the developmental effort is in fact restarted and a new prototype is required. The prototype hardware and software should be the basis for the full-scale development items and continuity should be provided in equipment and computer programs -- not in paper. The one program examined provides a poignant example of the cost of ignoring this lesson: little effort was expended to transfer the knowledge from extensive efforts by an advanced development contractor who had created a working model to the new FSED contractor. Some programs view a "concept definition" study phase as a cheaper alternative to a prototype development. It is not. The state of the art of analysis of software systems is such that a concept definition does not provide sufficient design and performance information to allow the initiation of full-scale development. The prototype experience is still required and will probably be acquired in the high-cost environment of a full-scale development program.

ENGAGE WITH THE CONTRACTOR

A frequent perception of the software system development process (by both government and contractor managers) is that the government should articulate its requirements, the contractor develop a system, and the government evaluate the finished product. In this scenario, there is little need for interaction between government and contractor teams during the development phase. Indeed, some forms of development contract may discourage such interaction by raising the fear of a contractor claim against the government for guidance or direction provided during the development process.

The Review Team's observation is that the development of a software-based system is almost never as "clean" a process as envisioned. Thus the contractor and government must interact both to clarify points that have been documented in advance and to respond to circumstances that have arisen in the process of

developing the system. Collocation of MITRE system engineers at the contractor's plant during system development ensured understanding of system requirements and design parameters and contributed to the success of several programs. This matter of a high level of engagement between government and contractor teams is crucial to the government's management of the development, as discussed in Section II. Doing initial (and critical) development in a prototype environment with reduced formal documentation requires a high level of engagement.

The Review Team noted instances of concern on the part of many government (and contractor) managers that a high level of engagement would result in redesign or overdesign of the system, excessive costs, and schedule delays. However, the major development problems noted by the team were characterized by a government refusal to engage the contractor. Instead the contractor proceeded with the development "on his own" to the point where he had accumulated major schedule slips and overruns. When the government became aware of the problems there was little remedial action that could be taken other than to accept slips and add to contract funds.

In contrast, management responded to a significant problem discovered in another ESD program in the prototype phase (inadequate memory) with a change of computers timely enough to allow the prototype phase to be completed on schedule. Government personnel, closely coupled to the contractor, understood the problem and its implications, and were able to react by supporting a machine change without initiating cries of program disaster.

Engagement requires only commitment by the government program manager to function as a manager and accept the consequences. The normal contracting mechanisms support engagement if the program manager does. The contractor will accede to a policy of engagement if the government wishes to have one. The contractor may warn of potential cost and schedule impacts -- but the potential is present in any case; and the government, by electing to avoid engagement, cuts itself off from the possibility of managing that potential.

APPLY SOFTWARE COMPETENT MANAGEMENT

The final key factor identified by the Review Team is the matter of software competence in development program management. The team noted a number of occasions where contractor program managers were characterized as hardware engineers and lacking understanding of software development and the sorts of problems that can arise in a software system. While competent software development groups (and software managers) may have been present, they were unable to

communicate effectively to the overall program managers to provide effective input to system decisions or to have their (software) difficulties assessed in a realistic way.

Two kinds of difficulties can arise when a program manager does not understand software development problems and issues. First, the program manager can "shove" program problems into the software without realizing their impact. Second, the program manager can fail to realize that software difficulties are major -- or indeed that he has software difficulties at all. When the problem is discovered it may be too late to recover. If the government is operating at a high level of engagement (and has competent software people) it can alert the contractor program manager to the problem. If not, all may be surprised together. On one program which had a contractor program manager who had no understanding of software, and where there was little or no engagement even though there was software competent government team management available, the contractor program manager did not realize that a major schedule slip and cost overrun attributable to software problems had occurred until the originally scheduled date for completion of Phase A testing had passed.

Assurance of software competence on the contractor's part should be supported by the contractor's management proposal. The key elements are the proposed project organization and the backgrounds and qualifications of the individuals who will fill that organization. While a contractor may change organization and personnel after award, the proposal at least provides some indication of the contractor's views on organization and staffing. If these are not to the government's liking, they can be reflected as negative factors in proposal evaluation or corrected through the interchange of fact-finding and negotiation.

SECTION IV

OTHER FACTORS IN SUCCESSFUL SOFTWARE SYSTEM ACQUISITION

In its examination of software acquisition programs, the Software Acquisition Review Team identified a number of other factors related to program success. The team concluded that these factors are important to program completion, but even if the related guidance is ignored they can be overcome in a properly managed program. In a program where there are major management problems (Sections II and III) these factors can, in effect, administer the coup de grace.

These other subordinate factors address the issues of program funding, production options, contractor commitment to plans, contractor competence, tool development and system software development. In each case a brief narrative is provided, defining the preferred approach and problems that can arise if that approach is ignored. In many programs, the "subordinate factor" problems are unavoidable, but the Review Team believes it is wise for managers to be aware of them. A matrix showing the incidence of these factors in the programs that the team examined is presented in Table III-1.

INITIAL UNDERFUNDING

A number of the MITRE personnel that the team interviewed made the point that insufficient funding has a major effect on the ability of the government to acquire a software system successfully. The theory expressed in a number of forms runs as follows: The contractor agrees to develop a system for a contract (target) price less than the real cost as manifested by the government's Independent Cost Estimate (ICE). To avoid early indications of disaster, the contractor initially manages at the target cost level, meeting his milestones but including in his plans insufficient resources for the code and test phases of the development. When these phases are reached, the contractor is in a posture of continually economizing (since the funds shortfall is now obvious) by cutting down on the measures that would produce reliable and adequately tested programs. This economizing in turn leads to a phase where major portions of the development must be corrected and reintegrated, rather than coming together smoothly the first time round. Thus the initial underfunding leads to a total program cost much greater than the initial ICE.

It is clear that the measure to address the funding problem is to avoid contracting for a target price less than the ICE. The four-step procurement process with its price negotiation favors selection of the low bid price. Other regulations, seemingly ignored, militate against inadequate initial funding.

The Review Team believes that initial inadequate funding creates special problems for the government manager. He must work closely with the contractor (engagement) to achieve a sound understanding of the required funding. He must use the early work of the contractor not only to gauge technical progress but to accumulate evidence for a program recosting. His reasonable options are to manage his program to the existing funding level by reducing requirements or to seek appropriate funding relief if basic system requirements cannot be met within the contract award. The Review Team observed that to proceed with a contract that is known to be less than reasonably funded without taking remedial action is certain to produce failure in the form of even larger funding problems later.

CONTRACTOR COMPETENCE

It should go without saying that an incompetent contractor team will probably be unable to develop a complex software system successfully. The one contractor characterized as incompetent to the Review Team was described as having had a good proposal written by its corporate headquarters (not by the division doing the project). When it came time to do the job, the contractor proved incapable of understanding the proposal or the job, and the contract was eventually terminated.

Careful review of proposals and examination of contractor performance history have been advocated as ways of avoiding an incompetent contractor. It is not clear that either is practical. It was clear that in the case of alleged incompetent contractor, the government's failure to manage the program made a bad situation worse. Thus the Review Team concludes that contractor competence is an important factor to be considered, but that attention to the key factors identified in Section III is both more effective and more practical in dealing with any contractor.

CONTRACTOR COMMITMENT TO PLANS

The emphasis on software quality and development techniques in recent years has resulted in government requirements and standards for a number of documents. Key among these are software development and test plans. The Review Team encountered a significant number of

cases where contractors had prepared acceptable development or test plans -- then ignored them as the program progressed. Thus the Review Team concluded that having contractor commitment to plans may be at least as important as the exact content of the plans.

The team tentatively reached two conclusions about contractor commitment to plans: (1) The contractor should prepare development and test plans that he believes are appropriate to the system and program. If the contractor has corporate standards or approaches, he should use them -- not newly imposed government standards. The government should, of course, review the contractor's plans and approaches for adequacy; and (2) The contractor's development and test plans should be made contractually binding. As with the other "subordinate factors", the government's attention to management and engagement is key to assuring contractor commitment to development and test plans, and to a successful program. A corollary conclusion is that the government should avoid forcing an arbitrary government development plan upon a contractor if the contractor is unfamiliar with that approach.

TOOL DEVELOPMENT

In one development program considered by the Review Team the contractor committed himself to developing a new softare development environment before he could develop the required system. The tool development took longer, was more risky than projected and had an adverse effect on system development and testing, contributing to schedule slips and cost overruns.

The Review Team takes the view that it is a rare tool indeed that must be developed before a system can be completed. Support libraries and compilers abound in 1981; some are not ideal and may not comply with government requirements and standards, but almost without exception they work better and sooner than anything that could be built for a specific program. Standards and procedures can supplement an imperfect but working tool for months while the perfect tool is being debugged. Tool development, being "in series" with and a prerequisite to the system development is an especially risky area. The current emphasis on standard higher-order languages can lead to a new cycle of tool development and delay a generation of systems. The government would be better off to use what it has or especially what the contractor has.

SYSTEM SOFTWARE DEVELOPMENT

The arguments presented above on software tools apply as well to system software (operating systems, data base systems, graphics, communications). However new system software may in fact be needed and unavailable, so the strictures against its development cannot be as severe. Nonetheless, the government would be well advised to focus on hardware that is supported by system software, rather than unsupported hardware. Several programs examined failed to observe this important factor.

PRODUCTION OPTION

The Review Team encountered programs where a priced production option had been negotiated along with the system development contract. The production phase was fixed price but the development was carried out as a cost-plus contract. As the contractor proceeded with the (already troubled) development, he saw inflation erode the adequacy of the production contract price. As a result he was willing to do an arbitrarily large amount of development (at the government's expense) to reduce the price of building the production systems. Steps were only taken to limit the development effort when it became apparent that the government did not have adequate money to fund the ongoing development of the system.

The lesson on production options are that they should be used with care, and the government should consider carefully the real effects of its contract and incentive structure. In a system where production costs are projected at ten or more times development costs, a priced production option may be desirable for the government. In programs where development and production costs are roughly comparable, it may be all too easy for the contractor to make a tradeoff that is not to the government's benefit. Until the system development is substantially complete, it may be unreasonable to fix a realistic and fair price for the production units.

SECTION V

CONCLUSIONS AND RECOMMENDATIONS

CONCLUSIONS

The Review Team's major conclusion is that insufficient recognition is given to the developmental nature of software in the management of C^3 system acquisitions. In recent years it is true that ESD and MITRE have placed increasing importance and emphasis on software as a potential problem area in C^3 system acquisitions; but it is less obvious that the program participants -- the government teams and the contractor -- have gained sufficient appreciation for the kinds of unpredictable risks often encountered in software development. The several existing production control techniques and management methods contained in regulations inadequately recognize the software development process. They make the invalid assumption that creation of software is a serial process which begins with well-founded requirements. Only infrequently can this be the case, as with the creation of a well-precedented system, such as some air defense programs. Normally, despite large efforts to create a quality system specification, the specification is found when system design is undertaken to be imperfect, ambiguous, internally inconsistent or too ambitious. In such instances the specification should be used as a guide for the development, and as a design progresses a feedback process must take place to clarify and perhaps reduce the intent of the system objectives.

A second conclusion is that there have been in several programs a tendency for adversarial relationships to be built between the government and the contractor. This is partially caused by the misunderstanding about software development, as stated above. It is also the result of government's holding the contractor accountable to the terms of the contract and an unwillingness to participate actively to help solve problems for fear that claims may result. Thus there has been a reluctance for the government and contractor to work collaboratively to address problems in a timely manner and adjust goals before much time and money has been has been spent on the wrong goals.

A third conclusion is that prototypes have been shown to be an effective means to resolve early the uncertainties of software systems. Prototypes have been completely new developments in themselves or have been adopted from some other similar systems. A decision to begin the system development with a prototype is evidence of appreciation of the developmental nature of software and

permits investigation of risk areas unconstrained by firm requirements and documentation which are so formally required in the full-scale development process.

RECOMMENDATIONS

Earlier we have discussed three key factors (1-3) which are especially important to program success and six other factors (4-9) which can significantly reduce some of the risks and problems encountered in such acquisitions. These are summarized below as recommendations of the Review Team.

Key

1. Incorporate a prototype phase in the development of software intensive systems

2. Manage the software acquisition using a high level of engagement between government and contractor

3. Assure the incorporation of software competence at a high level in the government and contractor teams

Subordinate

4. Avoid underfunding the development program

5. Assure that the contractor program team has a reasonable level of software competence

6. Assure that the contractor is committed to his development plans, techniques and standards

7. Avoid developing new tools as part of the development of a software intensive system

8. Avoid developing new system software as part of the development of a software intensive system

9. Do not acquire priced production options before system development is substantially complete

A cooperative effort of all parties and the knowledge and experience they possess must be brought to bear on the problems encountered in software acquisitions. For its part, the government team and especially the management must be competent to represent the system objectives and to understand the meanings and

implications of the software design as it emerges. Perhaps one of the greatest values of a good manager in any endeavor is his ability to recognize and confront shortcomings in the process under his control. Accomplished in a timely manner, remedying shortcomings can spell the difference between success and failure; and action by the contractor to effect the remedy may require support by the government, including rescoping and refunding of the program.

BUILDING AN ACQUISITION STRATEGY

The government's management plan should reflect all of these considerations. It should carefully identify the respective responsibilities of the program participants, based upon reasonable assessments of their expected capabilities as well as their proper roles.

A recommended initial cut at assigning such responsibilities is shown in Table V-1. It assumes MITRE will have the SE/TD role. It applies to the initial prototyping effort as well as the incremental add-ons leading to full operational capability. Note that the responsibility assignments reflect the intended cooperative effort among the SPO, MITRE and the contractor. But note also that the quantitative level of effort for each assignment is not implied (except as N is distinguished). The relative and absolute staffing levels must be determined by the program management based upon the specific circumstances and constraints surrounding the program.

The acquisition strategy must be thought of as a living plan that can change as the program proceeds. For instance, a decline in the government's confidence in the contractor may augur for a higher level of engagement or a reduction in goals for the prototype or increment currently being pursued. As most military men understand, a strategy must be managed.

Table V-1. Responsibilities

	SPO	MITRE	CONTRACTOR
PREPARE SPECIFICATIONS	J	J	N
EXPLAIN REQ'T ↺ Iterate "Daily"	J	J	S
EXPLAIN DESIGN ↺	N	S	P
TRADEOFF REQ'TS/DESIGN	J	J	S
BASELINE CONFIG CONTROL	P	S	S
DEVELOP CODE	N	N	P
MONITOR CODE DEVEL AND CHECKOUT	N	P	S
REDIRECT CODE DEVEL	N	S	P
PROVIDE TEST PLAN/PROCEDURES	A	S	P
CONDUCT DEMOS/TESTS (PQT,FQT)	A	S	P
MONITOR/EVALUATE TESTS	S	P	S
INITIATE ECP/CCN	J	J	S
MONITOR COSTS	J	S	J

J - JOINT PRINCIPAL
P - PRINCIPAL
S - SUPPORT
A - APPROVAL
N - NONE

DISTRIBUTION LIST

A-10

R. R. Everett
N. Waks
C. A. Zraket

D-10

E. L. Key
K. E. McVicar

D-11

J. J. Croke
C. A. Fowler
J. H. Monahan
A. J. Roberts

D-14

D. S. Alberts

D-40

B. M. Horowitz
F. R. Murphy
J. H. Phillips
C. M. Sheehan

D-41

T. Jarvis
G. J. Koehr
D. L. Martell

D-42

J. F. Jacobs

D-43

L. Elias
R. Platcow

D-44

R. W. Dix
T. A. Mackey
L. M. Thomas

D-45

R. C. Navarro

D-50

L. S. Billig
D. W. Kelliher
S. J. Pomponi

D-51

N. W. Briggs
G. A. Fagan

D-52

K. C. Taylor

D-54

O. F. A. Arnold
K. C. Heyda

D-55

F. D. O'Connor
M. G. Vacherot

D-56

G. A. Klein

D-57

P. N. Sinesi

DISTRIBUTION LIST - Cont.

D-60

C. E. Ellingson
J. W. Shay

D-61

N. E. Bolen
W. F. Lynch
A. R. Petrino
J. Shapiro

D-62

L. R. Herlacher
J. C. Naylor

D-63

R. S. Nielsen

D-64

V. A. DeMarines
M. L. Zeichner

D-65

D. D. Neuman

D-66

H. A. Bayard
W. Zimmer

D-70

W. S. Attridge (15)
E. K. Gates
O. R. Kinney (5)
E. L. Lafferty
W. S. Melahn
J. G. Wohl

D-71

J. B. Glore
R. L. Hamilton
M. Hazle
D. W. Lambert
L. J. LaPadula
A. D. McKersie
O. Shapiro

D-72

J. B. Campbell
D. A. MacQueen

D-73

C. J. Carter
J. A. Clapp
T. L. Connors
G. D. Knapp
J. J. Wandling

D-74

E. S. Herndon
J. F. Spitzer

D-75

E. H. Bensley
E. L. Burke
D. E. Kane
C. W. McClure
R. S. Popp

D-76

D. I. Buckley
C. K. Rowe

DISTRIBUTION LIST - Cont.

D-80

R. W. Jacobus
G. P. Langelier
S. J. Rabinowitz

D-81

R. A. Kingsbury
S. M. Newman
F. E. Patrick

D-82

R. D. Haggarty

D-83

W. A. Cleven
H. B. Goldbert
W. F. Higgens
R. F. Shackford

D-84

W. J. Canty
M. F. Owens

D-85

E. T. Gallivan

D-86

J. F. Sullivan

D-90

R. G. Grandy
L. R. Jeffery

D-91

E. J. Ferrari

D-92

J. F. Mercurio

D-93

E. D. Lundberg

D-95

H. T. Cervantes

D-96

J. F. McGinn
G. R. Strohmeyer

D-97

R. Alan

W-20

L. L. Holmes

W-30

R. P. Granato
J. S. Quilty

W-31

J. K. Summers

W-36

A. E. Budd

DISTRIBUTION LIST - Concl.

W-40

H. J. Kirshner
J. P. Locher

W-70

J. Dominitz