FINAL REPORT OF THE SOFTWARE ACQUISITION AND DEVELOPMENT WORKING GROUP

ADA 094 218

(Chairman, Mr. Victor E. Jones)

JULY 1980

PREPARED FOR THE
ASSISTANT SECRETARY OF DEFENSE
FOR
COMMUNICATIONS, COMMAND,
CONTROL, AND INTELLIGENCE

FINAL REPORT OF THE SOFTWARE ACQUISITION AND DEVELOPMENT WORKING GROUP

(Chairman, Mr. Victor E. Jones)

JULY 1980

PREPARED FOR THE
ASSISTANT SECRETARY OF DEFENSE
FOR
COMMUNICATIONS, COMMAND,
CONTROL, AND INTELLIGENCE

FOREWORD

The Assistant Secretary of Defense for Communications, Command, Control, and Intelligence established the Software Acquisition and Development Working Group (SADWG) in February 1979 to address the difficult problem of controlling software cost growth. The specific objectives of the working group included determining the efficacy and cost effectiveness of current software acquisition and development practices within the Intelligence Community, and ascertaining areas which could benefit from better management controls. This report presents the findings and recommendations of the working group.

Victor E. Jones
Chairman, SADWG

ACKNOWLEDGEMENTS

The Assistant Secretary of Defense for Communications, Command, Control, and Intelligence, Dr. Gerald P. Dineen, and the Deputy Assistant Secretary of Defense for Intelligence Systems, Dr. James H. Babcock, under whose auspices the SADWG operated, wish to extend their thanks for a job well done to the following working group members whose time and dedication made this effort possible:

Office of the Secretary of Defense	-	Mr. V. E. Jones, SADWG Chairman
Defense Intelligence Agency	-	Mr. Roy Cox
National Security Agency	-	Mr. Kermith Speierman
Central Intelligence Agency	-	Mr. Richmond Wise
Intelligence Community Staff	-	Mr. David Perlstein
U. S. Army	-	Colonel Jerry Timlan
U. S. Navy	-	Captain (Mac) McCutchan
U. S. Air Force	-	LTC Bill Gray
Rome Air Development Center	-	Dr. Gerry Plante

Special thanks are also extended to the following organizations whose detailed presentations to and in-depth discussions with the SADWG were instrumental to the preparation of this report:

BOEING AIRCRAFT	LOGICON
COMPUTER SCIENCES CORPORATION	MARTIN MARIETTA
CONTROL DATA CORPORATION	NETWORK ANALYSIS CORPORATION
FORD AEROSPACE	PLANNING RESEARCH CORPORATION
GENERAL ELECTRIC	RCA
GTE SYLVANIA	SOFTECH
HRB SINGER	SOFTWARE ENTERPRISES CORPORATION
HUGHES AIRCRAFT	SYSTEMS DEVELOPMENT CORPORATION
IBM	TRW SYSTEMS GROUP
LOCKHEED MISSILE & SPACE COMPANY	UNIVAC

Finally, we are indebted for the services of INCO, INC., and in particular Warren Polk and Barry Clapsaddle, for their splendid work in the preparation of this report.

TABLE OF CONTENTS

SECTION I

INTRODUCTION

A. BACKGROUND

The Assistant Secretary of Defense for Communications, Command, Control, and Intelligence established the Software Acquisition and Development Working Group in February 1979 to address the difficult problem of controlling software cost growth--growth which is approaching 70% of total C3I system costs.

To find out why these costs are so high and escalating at an ever-increasing rate, the SADWG set out to assess the effectiveness of current regulations and guidance relative to software acquisition and development, recommend approaches to improving the overall software acquisition and development process, and identify problem areas requiring top-level attention. The SADWG concentrated its efforts on the software development aspects of C3I systems. Specifically, the SADWG:

- Invited software development corporations doing business with the Intelligence Community to present their views on the software acquisition and development process, and associated problems.
- Studied actual case histories of software development projects ranging in value from $100,000 to millions of dollars.

The SADWG combined these inputs with their own in-depth experiences in C3I software acquisition and development and, after performing extensive analysis, detailed a series of conclusions which resulted in recommendations for improving the acquisition and development process.

As noted, one of the major inputs to the SADWG's collection, collation, and assessment of data on the software acquisition and development process was obtained from detailed presentations and in-depth discussions with software

SECTION I

INTRODUCTION

A. BACKGROUND

The Assistant Secretary of Defense for Communications, Command, Control, and Intelligence established the Software Acquisition and Development Working Group in February 1979 to address the difficult problem of controlling software cost growth--growth which is approaching 70% of total C3I system costs.

To find out why these costs are so high and escalating at an ever-increasing rate, the SADWG set out to assess the effectiveness of current regulations and guidance relative to software acquisition and development, recommend approaches to improving the overall software acquisition and development process, and identify problem areas requiring top-level attention. The SADWG concentrated its efforts on the software development aspects of C3I systems. Specifically, the SADWG:

- Invited software development corporations doing business with the Intelligence Community to present their views on the software acquisition and development process, and associated problems.
- Studied actual case histories of software development projects ranging in value from $100,000 to millions of dollars.

The SADWG combined these inputs with their own in-depth experiences in C3I software acquisition and development and, after performing extensive analysis, detailed a series of conclusions which resulted in recommendations for improving the acquisition and development process.

As noted, one of the major inputs to the SADWG's collection, collation, and assessment of data on the software acquisition and development process was obtained from detailed presentations and in-depth discussions with software

industry representatives. Specifically, SADWG invited 20 software development corporations doing business with the Intelligence Community to present their views on the overall effects of current policies and practices. In response to this invitation, the corporations addressed the following problems as they relate to C3I systems:

- Documentation Standards
- Requirements Definition
- Software Uniqueness
- Software Development Management
- Software Lifecycle Cost Estimating
- Hardware Constraints
- Contract Types
- Security Constraints

These problem areas received the top-level attention of the corporations. Each corporate group consisted of both a corporate representative and experts in the areas being addressed. These corporations spent many thousands of hours in preparing presentations for and working with the SADWG and, as a result, the SADWG collected and analyzed more than 3,000 pages of transcripts. This information was synthesized into Section II, Industry Comments. It would be safe to say that these comments represent a good cross-section of the software expertise in this country today. The comments have not been quantified or qualified in any way, and every attempt has been made to preserve the intent and accuracy of industry views. In addition, the participating corporations have reviewed a draft version of this report, and their review comments have been incorporated in the final report. All of the participating corporations feel the report accurately reflects their views on the software acquisition and

development process. A summary version of industry comments is contained in Appendix A. This summary has been designed to allow readers easy access to the major points presented.

Another major input to the SADWG was obtained from the investigation of case histories of actual software development projects. These case histories vividly illustrate the effect of problems on the current software acquisition and development process, and demonstrate the need for better software management at all levels of government. A summary and detailed discussions of the case histories are presented in Section III of this report.

In the course of their analysis efforts, the SADWG studied a candidate list of software development "commandments," which they felt merited inclusion in this report (Appendix B). Although not all-inclusive, the list nonetheless presents a reasonable set of "dos" and "don'ts" for software acquisition and development. In addition, a list of measurable milestones for use in software development is included as Appendix C.

B. CONCLUSIONS

It is common knowledge that software development projects rarely meet cost-benefits originally projected, usually cost more than expected, and are usually late. In addition, the software delivered seldom meets user requirements, oftentimes is not usable, or requires extensive rework. SADWG's analysis has shown that all facets of the software acquisition and development process need varying degrees of improvement. The analysis has also shown that software acquisition and development problems are not insurmountable, and that the remedy for most of these problems is better management at all levels of government. This remedy includes training software managers to effectively control programs, and providing managers with the proper tools to monitor the

development process. In addition, there is a need to improve cost and schedule estimating at all phases of project development, improve the productivity of software development projects, and bring the maintainability of software systems within reasonable cost limits.

The following conclusions reached by SADWG illustrate the depth of these problems:

1. There are a multiplicity of standards within the government which cause inefficiency and confusion. Currently, standards are not precise enough to eliminate misunderstanding between contractors and the government. The level of detail is often open to interpretation.

2. Projects, and subsequently contracts, often get started with inadequate planning. Also, because of ambiguous or vague requirements, there is often a lack of understanding between the government and contractor as to what is to be delivered.

3. Software development projects are being conducted with a lack of good management practices (i.e., poorly trained managers, inadequate record keeping, insufficient management tools, and misdirection of emphasis at the various development stages).

4. The government inadvertently impacts costs and schedules by specifying hardware for a particular development before knowing whether the hardware will meet the processing and performance requirements of the proposed system. It has been shown to be very expensive to "shoe-horn" software into minimally acceptable hardware configurations, particularly since hardware is less expensive than software.

5. There is often a mismatch between the contract type and the complexity of the work to be performed. The government has awarded complex

development efforts with fixed-price or heavy penalty contracts, placing both the government and the contractor in high risk situations unnecessarily. On the one hand, the government is hesitant to make decisions on this type of contract because of potential "out-of-scope" work and subsequent cost impacts. On the other hand, the contractor adopts high risk development approaches in an attempt to meet costs and schedules which, more often than not, fail.

6. Security requirements impact software development costs because work cannot begin on a project until the required personnel have been cleared by the agency for whom the work is to be performed. There are built-in delays because interagency transfers of security billets take an inordinate amount of time, and there are no interagency agreements on standards for the investigation or authorization of security clearances. This impact is rarely considered by the government and contractors in estimating costs and schedules. These costs are difficult to estimate because delays for clearances depend on the individual under consideration and the agencies involved.

7. Software development is unique. The software industry is the only industry required to build usable products right the first time without benefit of intermediate development stages such as prototyping. Although prototyping is an accepted practice in other less complex industries, it is not in software development. Without prototyping or some sort of intermediate development stage, risk factors are significantly increased.

8. Because the government must estimate system costs often years in advance of the actual procurement, the estimates are usually wrong. And because contractors must bid on systems before they are designed, their estimates are also usually incorrect. Neither the government nor contractors have adequate means to estimate lifecycle costs with any reasonable degree of

accuracy. The current state-of-the-art in lifecycle cost estimating is grossly inadequate.

9. Everyone agrees that software development productivity must be increased on the part of both the government and contractors. One method of increasing productivity is through the use of available productivity tools. However, because productivity tools are not well understood and difficult to use, instead of enforcing their use, the government only pays them lip service. At present, there is no easy method to quantify the value of productivity tools. Such a method needs to be developed.

10. In addition, the SADWG concluded that because C3I systems are unique, they are affected by a unique set of acquisition and development problems. Specifically, the following factors, in addition to those described above, have direct bearing on how these systems are managed and implemented:

a. "Finished" requirements do not exist with military software systems because requirements are constantly changing, and the systems, as a result, are never "complete."

b. C3I systems generally have a development cycle of more than five years before becoming operational.

Due to the unique nature of C3I systems (generally one of a kind and software dominant), "package" procurements, which include data collection and analysis through testing and maintenance of the system, should be avoided. In a "package" software system, problems in one phase of development often have a rippling effect in subsequent phases of development, impacting completion of original requirements. Rather than contracting for the entire system, it would be better to break the acquisition process into multi-phases. In a multi-phase strategy there are several distinct phases, each with its own confidence level

relative to cost and technical risk. The acquisition of C3I systems lends itself well to this type of procurement.

C. RECOMMENDATIONS

Based on the conclusions presented earlier, the SADWG has identified recommendations that will improve the overall software acquisition and development process. It is anticipated that from these recommendations a list of action items will be formulated and forwarded to the Assistant Secretary of Defense for Communications, Command, Control, and Intelligence.

The following recommendations have been listed in order of relative importance for implementation purposes only.

1. Multi-phased contracting for software development should be encouraged on large, complex, multi-year development programs.

- Suggested phases for this type of contracting include:

  Phase 0 - Concept Definition; Time and Material or Cost-Plus-Fixed-Fee contracts

  Phase 1 - Requirements Analysis, Definition, and Preliminary Design; Cost-Plus-Fixed-Fee contracts

  Phase 2 (To Phase n) - Implementation, Maintenance, and Enhancements; Firm-Fixed-Price contracts.

2. More effort should be placed on concept formulation and development of adequate requirements definition.

- The government should understand what the system is to do before embarking on design and code of a system.

- Users should become involved in this stage.

- Schedule pressures and obligating funds within fiscal constraints should not preclude adequate requirements definition.

- Methodologies for requirements and design traceability should be firmly established, and Preliminary Design Review (PDR) should not be passed without one. This eliminates "code searching for design and design searching for requirements" syndrome.

3. The government should re-examine its approach to software development, and encourage the use of risk-reduction techniques.

- Prototyping high risk segments should be encouraged, and the resulting software should not be deliverable.

- The use of available automated project management aids should be encouraged.

- More emphasis should be placed on configuration management and quality assurance.

- The government should encourage the development of tools to detect errors as early as possible, and thus minimize the rippling effect of undetected errors.

4. The government should attempt to reduce the multiplicity of standards which exist among services/agencies.

- If more than one standard is required, the reasons for it must be well established.

- Each agency/service should participate in streamlining the number of standards.

- Once standards are established, a single focal point should be responsible for changes or deviations.

5. The government should investigate, and invest in the development of, adequate software lifecycle cost estimating techniques.

- Although lifecycle cost models are still in the infancy stage of development, the government should nonetheless encourage the use of these models as one input to better cost estimating.

- Research should be encouraged to define software metrics more apropriate to lifecycle cost estimating than the current metric (i.e., number of instructions).

- The government should encourage contractors to be consistent in reporting pertinent cost data.

- The government should establish guidelines for the standard definition of metric terms (i.e., complexity, productivity).

6. The government should establish education and training courses and guidelines for program managers.

- These courses should reflect software engineering material on a continuous basis. The courses should also inform program managers how to prepare adequate cost and schedules for their projects with full appreciation of trade-offs available to them in their planning.

- Program managers should be encouraged to maintain compre hensive historical data on projects under their purview.

- Guidelines for evaluating and determining the adequacy of requirements, design, program specification, implementation, testing, etc., should be instituted.

7. The government should encourage the inclusion of independent software validation and verification (V&V) as part of the system specification. Optimally, V&V should be accomplished by someone other than prime contractors.

- By including V&V in the system specification, the amount of V&V necessary will be delineated, thus avoiding overkill.

- Independent V&V ensures impartiality, and should be used in each contract phase.

8. The government should encourage the use of independent cost estimating during concept development.

- Whenever possible, models and simulations should be used to determine cost and concept/design feasibility. Based on preliminary cost analysis, a design-to-cost approach should be encouraged.

9. The government should allow contractors the flexibility of offering alternative hardware configurations for software development projects when

these configurations can be proven to be more cost-effective.

- Specific hardware specifications should not be mandated in a Request for Proposals (RFP).

- Guidelines for cost effective hardware trade-offs should be established.

- The government should encourage the use of "computational plenty" in considering development approaches.

10. To circumvent costly and time-consuming security constraints, the government should encourage the following practices:

- Clearly designate position functions which require clearances, rather than requiring clearances for all project personnel.

- Invest in unclassified hardware test beds and test data sets (many unclassified projects require clearances only for access).

- Separate classified work from unclassified work.

- The majority of software developments do not involve security considerations, hence the government's blanket security demands should be scrutinized.

SECTION II

INDUSTRY COMMENTS

A. INTRODUCTION

This section contains comments presented by industry to the SADWG. These comments have not been quantified or qualified in any way, and every attempt has been made to preserve the intent and accuracy of industry views.

B. DOCUMENTATION STANDARDS

1. Virtually every government agency handles the development of software using different standards despite the fact that regulations such as DoD 5000.29 exist.

2. Standards are not precise enough to eliminate misunderstandings between contractors and the government; for example, the level of detail in documentation is often a question of interpretation.

3. The vast number of different standards and guidelines decreases the value of standards. It is impossible to be familiar with all of them.

4. Each agency/service (and smaller activities in many instances) implement DoD standards in their own way. This results in confusion for contractors, and probably decreases the synergism possible throughout all DoD.

Perhaps the most prevalent difficulty with the software acquisition and development process seen by the software industry representatives is the lack of a consistent software standard not only among the various government agencies, but also often within a single agency. The current multiplicity of standards promotes gross inefficiency and adds significantly to the cost of software.

Current standards are generally inadequate. This has caused many of the agencies/services to conduct their own studies to improve them. Many of the industry representatives indicated that they have been involved, in one way or another, in these previous studies. As a result of all this activity, industry

feels nothing has really happened except that a number of different standards have been developed by the various agencies and services, and each of these standards addresses the particular concerns the agency/service had at the time of the study. It was generally believed that having each agency/service develop its own standards is not the most cost-effective way to do things.

However, adopting a single standard for all services and agencies is not necessarily the solution to this problem. For although some standard must be implemented, it must be flexible enough to conform to the wide range of program types and sizes. Or, if a variety of standards are used, then they must be similar in scope and function in order to allow contractors and government personnel to move from one to another without traumatic, devastating results to both the cost and schedules of individual projects.

Concomitant with the recommendations for streamlining sets of software documentation standards, is the need for uniform definitions of software elements. Although it's only a matter of semantics, this deficiency can nonetheless cause a great deal of confusion. And confusion, once again, leads to cost and schedule overruns or, even worse, to incorrect design and development.

If definitions for such things as a unit, program, module, routine, task, element, and subsystem are standardized, then many of the problems associated with the use of inconsistent terms will be solved. However, as with documentation and software standards, if a standard is imposed, it is useless if not enforced -- enforced for both vendors and government personnel.

In addition, standards do not adequately address the question of the level of detail required. Oftentimes, these same standards require too much production of little used documentation with no agreement between government

and industry on the right amount.

Industry is also concerned that the government, in attempting to correct the standards problem, may impose overly restrictive development policies. Each contractor has its own development process with personnel trained in that process. Therefore, the government should concentrate on what is to be delivered instead of the actual development process, because retraining personnel to follow one specific process over another is costly.

C. REQUIREMENTS DEFINITION

1. Requirements documents are usually too ambiguous or vague and subsequent documents produced from these requirements are inadequate.

2. Although technical people and system users may agree on specific requirements, technical people find it difficult to describe the system to be delivered in terms of requirements and therefore exclude the user in the process of translating requirements to the final system. This often leads to user dissatisfaction with the delivered system because trade-offs were made which the user either did not under or, because he was not part of the process, did not want.

3. Requirements are constantly changing and often force overruns; management doesn't know how to handle this problem in relation to modifying original budget and schedules. Not all modifications to original budgets and schedules are overruns but are the result of honest requirement changes.

4. "Finished" requirements do not exist with military software systems because they are constantly changing; military software systems are never "complete."

5. System developments are too long; C3I systems generally have a development cycle of up to 14 years before a system is operational. Other systems take 5-6 years. When things take this long, you forget what you started out to do -- few people remember the initial requirements and the cost-benefits to be derived. Because these development cycles are so long, requirements, more often than not, are obsolete before the system is completed.

In the definition phase, there is often a lack of clearly defined requirements; vague requirements fail to fulfill users' expectations or do so only at an exorbitant price. Industry representatives agreed that the content and quality of requirements documents should be improved. Currently these documents are too ambiguous or vague and do not convey an adequate understanding of what is to be provided. Part of the problem stems from a lack of understanding of what is a requirements document. For example, requirements should be stated in as much detail as possible and should be organized as to which are necessary and which ones are simply desirable. In practice, everybody wants everything. However, if all requirements have equal weight, developers don't have much leeway in designing systems within reasonable cost.

Conversely, requirements can be too specific and, therefore, box designers into techniques and methods incapable of performing the required functions. Requirements, if they are to accomplish their purpose, must be clear and concise, and relate what the exact need is, not how it should be fulfilled.

It is generally believed and accepted that if a good requirements document is not available, then the program should be stopped and one should be written. As already mentioned, what constitutes a good document needs careful review. A general rule for their preparation states that requirements should be decomposed until the next level of detail adds nothing to the acquisition phase currently in progress.

However, C3I systems are unique, and it is difficult to write adequate requirements for the following reasons:

a. People involved don't know what the requirements are except in broad general terms.

b. C3I requirements for ADP systems support are constantly changing, and therefore it is difficult to write a finished set of requirements.

c. C3I systems take many years to develop and changes are inevitable.

Once the actual implementation of a project is undertaken, software industry representatives feel that the biggest problem, endemic to virtually all C3I systems, is that requirements continually change. However, stopping changes is not the solution to this problem. For if it is mandated that requirements cannot be changed, the end-products may be useless to users. Changes are often simply the result of design; that is, as design progresses, parameters are enlarged or shrunk and requirements as a result are changed.

Changes can, however, be accomplished in an orderly manner and their impact minimized. Also, many changes can be foreseen before design if stringent system requirements reviews are conducted. By employing baselines for the requirements, if changes are necessary then a formal change-control mechanism can be used. Software industry representatives feel that if these recommendations are imposed and enforced, changes to requirements will not necessarily result in cost overruns and schedule slippages.

In dealing with government procurements, actual users and customers are rarely the same organizations. For example in a typical procurement, a user outlines a requirement and turns it over to a particular procurement agency for action. The requirement is then translated into an RFP, through various channels, and a procurement subsequently is made. From the moment the requirement statement leaves the user's hands, the details are open to subjective interpretation. And in not a few cases, misinterpretation has become catastrophic, resulting minimally in dissatisfaction, delays, and changes.

Several solutions which might help circumvent these difficulties include obtaining the user's approval at the system requirements reviews, guaranteeing user approval with purchase description, and keeping a user representative involved throughout the development cycle. Indeed, a majority of the software industry representatives concluded that it is essential for the ultimate user to maintain constant communications with both the procuring agent and the contractor during all stages of the acquisition process. This ensures user satisfaction and protects contractors from being caught in an awkward disagreement between user and procuring agency.

D. SOFTWARE UNIQUENESS

1. The government lacks the understanding that software configures the whole system much like mortar between bricks, and is much more expensive than hardware.

2. Because software developments produce only paper from the requirements phase through the coding phase, which is often 2-3 years in process, software projects are difficult to manage. This problem is compounded by the government's lack of adequate tools to manage software developments. Because of ill-management, deficiencies in the software system often remain undetected until the system is built.

3. More than 60% of lifecycle costs are spent after the system is built, not in building it. These costs are attributable to errors introduced in the requirements, design, and development phases that remain undetected until the system test phase.

4. The software industry is required to develop error-free systems on a one-time basis. This is almost an impossible task. Other industries such as construction, computer hardware, tank building, and ship building develop prototypes. Prototyping is an accepted practice in these industries and hardware development. Software, however, which is often far more complex, is rarely prototyped and, if costed in a competitive bid, that bid is unlikely to win.

5. Because management fails to appreciate the importance of adequate testing, design problems receive priority and testing is kept to a minimum with resulting consequences.

6. Hardware is developed at the manufacturer's site in a friendly environment while more complex software is usually developed at the customer's site in an often pressure environment.

7. The government does not take full advantage of software transportability where applicable; hence the same software is developed over and over again. Unlike hardware, software costs are confined to the initial investment plus maintenance.

As a further recommendation for designing software systems, industry representatives identified the concept of prototyping; that is, building an actual model of the proposed system to verify design. Prototype construction is different than simulation. In prototyping, an algorithm is actually coded and tested. A simulation only simulates an algorithm. Software prototype construction is not an accepted practice at this time and, if bid in response to an RFP, would not stand up under competitive pressures. There is a need to make prototype construction of software systems more cost-beneficial -- currently it is not.

However, industry presented some strong arguments in favor of prototyping software systems. It was estimated that better than 60% of the lifecycle costs of major systems is attributable to errors introduced in the requirements, design, and development phases. Errors result in redundant effort. The earlier the error is introduced (i.e., requirements phase), the more costly the error is to fix due to the rippling effect. Industry suggested that the government put emphasis on risk-reduction efforts. Such efforts would detect and fix errors in the earliest phases of a program. Everyone agreed that software development is currently a high risk area. In the government's

situation, risk is the likelihood of cost exceeding original estimates. In industry's situation, risk is related to profits and, if not controlled, will result in loss of a business. One such risk-reduction effort is prototype construction of software systems.

Prototyping has, for a long time, been an accepted practice in the design of hardware. However, few people recognize the importance of prototyping in the design of software. And yet, by all accounts, software development is far more difficult and expensive than hardware. The problem is largely one of changing accepted practices.

Not all software systems, however, require prototyping; only those which are extremely complex and involve new technology. Software industry representatives agree that although prototyping is expensive, in certain instances it is well worth the expenditure. However, all also agree that software produced for a prototype should not be delivered to the government, for software prototypes mean very little if not seen in light of their development purposes.

According to the software industry representatives, test plans are often validated too late in the development cycle. By postponing the test plan approval, the nature of the test can be affected, and schedule slippages can occur. For example, if the acceptance test plan is not completed until the end of the development cycle, then often the test will focus not on the original requirements but only on the subsequent design. Also, in obtaining approval of the test plan from the government, precious time can be lost and time, at this point in the project lifecycle, is an extremely critical factor.

To speed up test preparation and validation, software industry representatives have suggested that at least a first draft of the test plan be

completed for Preliminary Design Review (PDR). The test plan can, of course, be updated as the project progresses. But by having a draft already prepared, time will be saved and the integrity of the test will remain intact.

Although significant strides have been made in software technology, similar accomplishments are not evident in software productivity. Software industry representatives have been attempting, through a variety of means, to close the gap between technology and productivity. Several areas currently being investigated include the use of the programmer's workbench; automated design aides; automated documentation tools; top-down methodologies; and programming design languages. These methodologies, tools, and languages, although still relatively new, have shown great potential for increasing productivity.

Even with advanced techniques, software development is still treated by many as a magical process. This is, of course, a transient problem. For, as more personnel with software development training move into senior management positions, software will cease being seen as a magical process and become viewed more as a science.

Also, there is a lack of visible milestones in software development. Therefore, software industry representatives have proposed that other types of milestones be imposed; that is, measurements (Appendix B) instead of actual component pieces should be monitored, making the process more visible.

As previously discussed, software industry representatives feel that C3I systems are different from other types of software developments. Specifically, C3I software systems usually are large and complex; involve high technology; are turnkey; demand low error rates; and have the user and buyer associated with the requirements. These characteristics make C3I systems unique, and

software industry representatives believe special measures must be taken in the software management, acquisition, and development process to handle this uniqueness. The recommendations presented by the software industry representatives are a major step towards confronting problems associated with the current process.

Other industry comments related to software are presented below:

a. Software is not a visible product; hardware is.

Individual hardware components are no more visible to the end user than are individual program instructions. Only the outer package is visible -- the hardware box or the software capabilities. We must come to understand that unexplained and perhaps seemingly excessive costs for "invisible" products are not limited to software. (What are the cost components of a Rolls Royce? The product is visible, but why it is so much more expensive than a Chevrolet?) In fact, an experienced software cost estimator could probably better deduce the development cost of a system by analyzing its programs than an automotive engineer could deduce the cost of a Rolls Royce by taking it apart.

b. Once hardware is fielded it is correct and never changes (except for corrective or preventive maintenance); software bugs are continually being corrected.

If the statement were true for hardware, Detroit would never recall cars and computer configurations would never change. The statement for software is marginally true -- most large systems do contain bugs. However, many function as reliably as hardware -- man-rated systems and those having high consumer visibility, for example. Many software bugs are only of an annoying, rather than catastrophic, nature, but we tend to blow them out of proportion because we feel that software is easier to fix than hardware. Hardware systems contain the same types of annoyance bugs as do software

systems--the windows in your car leak when it rains and Master Charge has been dunning you to pay a credit balance already paid.

c. Software never "rusts" (once it's correct, it stays right); hardware wears out.

The statement is true for hardware. Software systems do degrade with time, particularly as data bases gradually become corrupt and deteriorate. (One may argue that software permitting data base corruption is not correct, but software is rarely completely correct anyway, and could theoretically be rated in terms of amount of data lost for each system failure, for example, just as hardware is rated in terms of life expectancy under various operating conditions.)

d. Software is more difficult to configuration manage than hardware because it's easily duplicated and modified.

This is true, but the problem is one of approach and degree rather than kind. Users of fielded hardware seldom modify it themselves, but everybody changes a software package they receive. Enforcing central maintenance eliminates unmanageable fielded software variability. Even so, the easy modification of software encourages the use of different versions for the same purpose. But this only changes the degree of configuration management required, not the kind. Buy a new car and watch how the dealer has to manage the configuration of what options are required by others, what options are included in others, and what options are excluded by others.

e. Personnel shortfalls influence software quality and development.

In the area of personnel requirements, a majority of the software industry representatives expressed concern about software personnel shortfalls. In some cases, the shortfalls, particularly in the California area, are quite serious. The software industry is not immune to current economic constraints.

Specifically, the voluntary 7% wage guideline has had an extremely adverse effect on personnel requirements. For example, if a contractor is prohibited from raising salaries more than 7%, software personnel will quit and go to another company which can hire them at a higher than 7% level. This constraint is, of course, temporary. Also, colleges and universities continue to turn out higher numbers of trained computer personnel each year which may alleviate this shortfall in the future. But until then, the industry is faced with severe shortages nationwide, provoking highly competitive hiring practices among contractors -- hiring practices nurtured by the 7% guideline.

E. SOFTWARE DEVELOPMENT MANAGEMENT

1. Government ADP program managers are not trained to manage ADP development programs.

2. Government personnel are often unable to evaluate a good design and relate that design to requirements due to lack of technical depth and management aids.

3. Many government managers often fail to make decisions or consistently change their decisions, and thus adversely impact contractor productivity.

4. Government procuring agencies are inadequately staffed (or lack technical support) to manage software development efforts.

5. Few records are kept concerning the history of past programs that can be used on future programs--"lessons learned." There are no guidelines for recording this data at present, with the result that information learned from past experience is lost.

6. Government agencies spend the majority of development funds on design, code, test, and maintenance, and usually spend little money on verification and validation, configuration management, quality assurance, documentation, etc.

7. There are no adequate tools for tracing requirements through the development lifecycle, and no easy-to-use process to determine the degree to which designs meet stated requirements.

8. The government attempts to foreshorten schedules unreasonably, making on-time accomplishment of a project highly unlikely, and is often unaware of the risk introduced as a result of those actions.

9. Industry does not have, nor will the government allow (mostly because of security), integration, testing, or configuration management groups across projects. Each project is treated as a separate entity. The government's reasoning is that each project is different and, therefore, an overall group would be inefficient.

10. Competitive procurements encourage contractors to offer maximum technical effort for the dollars; however, by doing so, contractors tend to skimp on supporting services such as management, configuration management, quality assurance, etc., because the government minimizes the importance of these efforts.

As an outgrowth of the need for standards for software documentation, development, and definition of terms, it is readily apparent that a need exists for training all personnel involved in the software acquisition and development process. It was learned through the briefings given by the software industry representatives that virtually all of the contractors have undertaken some form of training not only for program managers but also for programmers and senior level management.

Yet no comparable approach to training has been implemented on the part of the government. Training is perhaps the most important aspect to reorganizing the software acquisition and development process. For if program personnel are well versed in the nuances of the process, a great deal of the confusion, inefficiency, and inconsistency prevalent in software acquisition and development will be alleviated.

The phase between Critical Design Review (CDR) and acceptance test is usually the longest time period of the development cycle; it is also the phase which has the fewest number of formal reviews. Software industry

representatives believe this lack of review during the crucial development cycle results in significant problems. For example, without reviews or even baseline points, little visibility of the development is possible. And when development takes place out of sight, the end-product usually does not meet the user's requirements. Also, schedules cannot be monitored if no reviews are conducted, and the customer is unable to measure the contractor's performance.

Several recommendations have been made by software industry representatives to bridge the gap between CDR and acceptance test. First of all, review points should be implemented during the development cycle and customer representatives should be encouraged to attend. By using hierarchical implementations, both contractors and the government can measure performance against established criteria; that is, by using structured walk-throughs, integration ready reviews, and subsystem tests, work already accomplished and work to be done can be identified.

Concomitant with the idea of hierarchical implementation is incremental acceptance. Instead of waiting until the end of the development cycle to test everything at once, parts should be tested as they are completed. In this manner, if a problem is detected at an incremental test, usually it can be solved without greatly impacting cost and schedule. However, if a component is found to be defective during acceptance test, the entire project would likely be adversely affected.

Problems in the management process are further compounded by the demand to adhere to often unrealistic development schedules. These demands not only allow insufficient time to solidify requirements but also insufficient time for the actual design to be completed at a detailed level. Thus, coding often begins even before the design has been approved. To alleviate these problems,

it has been recommended that schedules should be driven based on historical data and similar sized programs. If schedules are to be defined in this manner, the necessity for having an experiential or historical data base is even more evident.

However, the problem of unrealistic schedules cannot be assigned completely to the government. For contractors, in RFPs, readily concur with government plans in attempts to win contracts. It isn't until contracts have been awarded and development undertaken that the unreality of a schedule becomes publicly known. And by then, unfortunately, it is often too late to amend.

Schedules, if they are to perform their required functions, must not be prepared based on what the user or procuring agency believes or wants it to be, but instead on a realistic evaluation of the current effort in view of previous undertakings of similar dimensions; historical data or past experiences are essential ingredients. If through investigation and the development of a lifecycle cost model it is ascertained that a proposed schedule cannot be met, then this information can be made known prior to a point of no return, and alternate schedules can be implemented.

This problem is further compounded when the RFPs are disseminated because contractors are expected, if they hope to win the contract, to write proposals detailing schedules, manpower, and methodologies within a certain, often inadequate, timeframe. It is highly unlikely, without advance information, that contractors, in what is essentially an overnight proposal, can accommodate sufficient analysis, evaluation, and trade-offs. Within these time and informational limitations the result is often a proposal, as one contractor explicitly stated, steeped in a little more than plain science fiction.

Indeed, the plans set forth in the best proposals rarely bear the slightest resemblence to the finished product.

To alleviate this situation, software industry representatives have recommended submitting the RFP to a non-bidding contractor or independent cost and planning estimator. This contractor would then prepare a proposal as if he were attempting to win the contract. However, since the contractor is ineligible to actually compete, his cost estimates, time schedules, and technical discussions and approaches will perhaps be more realistic, having been prepared from a different perspective than those of competing contractors. If the resulting proposal is within the time, budget, and technical factors previously ascertained by the government, then the non-bidder's proposal can be used to compare competing proposals and actual contract performance after the proposal period has ended.

F. SOFTWARE LIFECYCLE COST ESTIMATING

1. Government, and to a large extent industry, has little understanding of how to estimate software development cost and schedules with any quantifiable degree of confidence.

2. The current state-of-the-art in cost estimating has as its basis the number of instructions and cost per instruction. Cost per instruction varies with system complexity and ranges from $3.00 per instruction to $250.00 per instruction, and is by itself a poor measure. It is difficult to estimate the number of instructions prior to design. Estimating "horror stories" are well known but government and contractors have not become any better at it.

3. Available cost estimating models are rarely used by the government because of their cost and government and contractor's inability to qualify output.

4. Standards do not adequately address the lifecycle; for example, can software maintainability be quantified and placed in a standard?

5. The current state-of-the-art in cost estimating is more art than science. A need exists for a set of cost estimating metrics that are consistent with the level of detail or phase in which the project is currently involved. Software cost estimating techniques, although primitive at this time, have been developed for use in cost estimating the implementation phase (phase following design). Little or nothing has been done for the other phases.

6. The further a system is from the implementation phase, the higher the risk of cost estimating. When doing cost estimating, the uncertainty of risk is a function of how far away the system is from being implemented. Currently, budget estimating is done years in advance of system implementation.

7. There is a lack of understanding of how to estimate software costs (cost estimating accuracy is directly proportional to an understanding of the problem). Accurate cost estimation requires good requirements analysis and design -- both are currently lacking.

8. Cost cannot be accurately estimated until a design is complete. In today's environment, neither the government nor the contractor can reasonably estimate the cost of a system prior to design.

9. Congress establishes program development budgets without the knowledge that program costs cannot be accurately determined until after design. In addition, the estimates they receive do not include accurate estimates of the total lifecycle costs, with the net effect that programs meeting requirements appear to be overruns, when in fact the case might be that the programs are meeting "realistic" budgets.

Program budget estimating is grossly inadequate. Budget figures are estimated years before the true software development costs are known. Industry representatives agree that the further one is from the implementation of a system the more uncertain one is of a cost estimate. They seem to agree that an accurate cost estimate cannot be developed until after the design of a system is complete, has been validated, and critical design areas are prototyped.

However, this means that in most development programs accurate cost estimates cannot be developed until more than three years after the original estimates were made. Industry representatives also feel that the budget estimates fail to include all the factors that can impact cost and schedules (i.e., security constraints) and rarely encompass total lifecycle cost (i.e., maintenance).

A key factor in preparing cost estimates for budgets and subsequently for contract support, and one which is still in the infancy stage of development, is software cost estimating aids. With these aids, the government and contractors will be able to develop more adequate cost, schedule, and quality tradeoffs, and hopefully come up with more reliable front-end estimations. The software cost estimating aids, if carefully prepared, can also provide better management tools for in-process predictions and control. RCA's Price-S, the Putnam cost estimating model, and variations on both of these have been the primary aids assessed by the software industry representatives.

The current state-of-the-art in cost estimating models is predicting the number of instructions for a particular system and estimating factors such as complexity, productivity, computer resources available, experience with developing similar systems, etc. This estimating cannot be done with any degree of confidence. It was recognized that there is a need to define a set of software metrics that could form the basis for better cost estimating. It was recommended that the government start initiatives in this important area.

By underestimating required products and legislating unrealistic schedules, both cost and productivity suffer. It is to these two areas, cost and productivity, that current software cost estimate models are aimed. The lifecycle costs of software are divided into three periods or phases of the

total acquisition and development process: (1) operation which includes reliability, correctness, efficiency, integrity, and usability; (2) revision which includes maintainability, flexibility, and testability; and (3) transition which includes portability, reusability, and interoperability.

However, all software lifecycle models are subject to a plethora of difficulties. For example, although there is a high correlation between the number of instructions and the final cost of a product, it is extremely difficult to ascertain the precise number of instructions at the outset of a program. Also, few opportunities exist to examine the conditions which affect a program before making estimates. In the collection of cost data, there is little uniformity with the result being that the cost data collected is often incomplete. Another problem which adversely impacts the accuracy of software lifecycle models is that little attention is given to the time span of a proposed program and a clear understanding of what is maintenance and what are enhancements.

Although it is difficult to isolate and quantitatively measure the impact of problems which influence software cost, these factors must be identified if aids like a software lifecycle model is to perform its required functions. It must be noted that factors affecting lifecycle cost models are never constant from one program to another.

Although currently available cost estimating models are admittedly imperfect, it was generally suggested that the government should use them as one input to developing cost estimates. However, caution was expressed in this recommendation.

Estimates of 60% to 80% of lifecycle cost were presented as being spent after the system is built, not in building it. These cost are due to errors in

the specification of requirements, deficiencies in design, and/or errors in coding. The earlier the error is introduced the more costly it is to fix due to the rippling effect. Since pertinent data is not kept on software development programs, and that which is kept is suspect, it is difficult to determine exactly what is happening.

As a starting point for employing a software lifecycle cost model, the importance of maintaining an historical data base or library is readily apparent. For example, if a contractor has information in his possession which reflects previous in-depth program experience, this information may be put to use as input for deriving current and future lifecycle costs. Also, in preparing a lifecycle cost model, separate estimates of all software must be obtained. These results can then be documented and examined in accordance with such things as instruction count, similarity/transportability, and development time. Using these variables, several models can be run and the results audited and compared to previous estimates.

G. HARDWARE CONSTRAINTS

1. The use of government-specified hardware, in some instances, unnecessarily constrains contractors and definitely impacts cost, which at present is not fully understood. This requirement also eliminates contractors' proposals for possibly better and more cost-effective hardware/software combinations.

2. The government often uses standardized hardware without consideration of cost impacts on software development. Any hardware deficiencies/limitations must be made up by the software regardless of the fact that software is generally much more expensive than hardware. Sometimes physical limitations prevent software developers from building systems to meet stated requirements, or from building systems economically.

3. GAO contracts for hardware to be bought, often years in advance of actual use, regardless of what kind of software/application it will be used for; thus failing to fully exploit new technologies as they become available.

4. Use should be made of the inexpensive availability of "computational plenty" to build systems; it is very expensive to "shoe-horn" software into minimally acceptable hardware configurations, particularly since hardware is less expensive than software development.

Current GAO hardware procurement practices, software industry representatives feel, unnecessarily constrain software developers. Hardware and software are purchased separately, sometimes years apart. Since hardware is usually procured first, unmindful of its eventual use, software developers are forced to fit the hardware available. Since the cost of software is generally much greater than that of hardware, software costs are increased even more when hardware is dictated. The problem is further compounded because in these mass GAO buys, new technologies are rarely considered.

With the rapid rise in the cost of software and the advancements in technology, software industry representatives recommend that hardware be procured to fit specific software. Also, a software contractor may be able to offer a more cost-effective hardware/software combination for a particular application. Software industry representatives believe these recommendations would not only reduce costs, but also update procurement procedures to keep pace with current situations.

H. CONTRACT TYPES

1. Often contract types do not fit the C3I environment. For example, it is not possible for the government to develop the detailed specifications required for firm-fixed-price contracts in an environment of constantly changing requirements.

2. The government needs to rethink the process of procuring systems in "packages." "Package" systems include data collection and analysis through testing and maintenance of the system. In a "package" system, problems in one phase of development have a rippling effect in subsequent phases

of development, causing these subsequent phases to be short-changed, thus impacting the completion of original requirements.

3. Contracts do not permit the construction of test tools because of prohibitive cost. Software tools, because they interface deeply into the code, are single-project oriented, and if they cannot be used on more than one development, the government doesn't want to pay for them.

4. Software is never truly free of discrepancies and, consequently, clauses that say all discrepancy reports will be cleared up before acceptance are not reasonable. Such clauses should read that all critical discrepancy reports should be corrected before delivery, but some reasonable number of minor discrepancies may remain.

In the acquisition stage of the overall process, many of the software industry representatives made recommendations concerning the types of contracts that should be employed on various developments. Specifically, it has been suggested that Fixed Price contracts should not be used on development programs. Indeed, several contractors asserted that C3I systems, because of their uniqueness, are not conducive to Fixed Price contracts at all. Often in performing Fixed Price contracts, the contractor necessarily focuses on the cost of the development, while the customer, quite rightly, is more concerned with the progress of the development. This division of primary interests results in products limited to strict interpretations of the specifications, and products that do not change with changes in requirements.

In developing C3I systems, software industry representatives feel that cost-type contracts should be used. Only when the job is clearly production work should Fixed Price contracts be employed; that is, when the work is low risk, the requirements are firm, and no R&D is involved. Some of the software industry representatives futher recommended, for the development of C3I systems, that cost plus incentive contracts be used, and that the contractor be

involved in determining exactly what the incentive should be. Thus, the contractors will become directly responsible for the terms of contracts and perhaps have a greater vested interest in the contract's successful completion.

However, one facet of the cost-type contract is of concern to software industry representatives. Specifically, the current practice of negotiating Best and Final on a cost-type contract among a number of contractors often leads to significant cost and schedule overruns. When contractors are solicited to submit Best and Final proposals, they must reevaluate their initial cost and attempt to further reduce it. However, the margin for reevaluation is often extremely narrow, and the final costs are, therefore, unchecked by any cost-realism criteria. Proposals rarely capture precise cost data. Combine this with the narrowing process prevalent in Best and Final negotiations and the results can be catastrophic to the project. To circumvent these difficulties, software industry representatives have recommended that, first, a winning contractor should be selected and then negotiations for Best and Final costs should be conducted. If done in this manner, the contractor will be more reticent about trimming costs below the level required to successfully complete the contract.

I. SECURITY CONSTRAINTS

1. Restrictive security requirements impact software development costs but are rarely considered in cost estimating and schedules. There is also a lack of consistency in government clearance procedures which results in unnecessary delays in transferring contractors' clearances among agencies/services.

2. The shortage of cleared personnel makes them valuable commodities, and as such they are susceptible to lucrative offers from competitors. When these personnel leave, valuable expertise is removed from projects and productivity suffers.

3. While awaiting clearances or billets, personnel are not productive.

4. The government is not flexible in moving people off of projects. Many projects require that the work be done at a government site and clearences are, therefore, necessary. Also, because of security requirements, personnel are assigned to a project for its duration. When the project is over, they don't want to go to another classified project for fear of being "pigeon-holed" again.

As a further inhibition to the timely and cost-effective completion of contracts, software industry representatives identified problems with the current personnel clearance process. Specifically, there are insufficient intra-agency agreements on standards for the investigation or authorization of security clearances. Consequently, when a vendor is awarded a contract, work cannot begin until the required personnel have been cleared by that particular agency despite the fact they may have been cleared by another. This can be costly for the contractor and time-consuming for the government. This is extremely frustrating when special-access billets are necessary and only a certain, usually insufficient, number exist. When the project is over, the billets are once again lost and the contractor must repeat the entire process for subsequent efforts.

In examining these security requirements problems, software industry representatives suggested that the process can be substantially expedited by questioning the classification of certain aspects of projects. For example, if a system is to handle special access information, is it absolutely necessary to require special access billets for personnel designing the system? All too often special access clearances are required needlessly, causing long delays before work can begin. Furthermore, adding an unnecessary requirement for special access clearances compounds the problem of locating qualified personnel in an already desperate shortage of experienced people.

SECTION III

CASE HISTORIES

A. SUMMARY OF CASE HISTORIES

The case histories contained in this section are representative of the widespread problems associated with software development projects. SADWG selected these particular case histories, which range in value from $100,000 to millions of dollars, to emphasize that virtually identical problems occur regardless of program size. Two of the case histories were extracted from a GAO report studied by the SADWG.1/

B. OVERALL EFFECTS

1. Programs were cancelled after expenditures of millions of dollars with little usable results.

2. Programs ran out of money before work was completed causing more dollars to be budgeted than originally planned.

3. User requirements were not satisfied at all, partially satisfied and/or systems delivered late, thus impacting important operational production.

4. The government and contractors wasted a great deal of manpower and years in developing systems that were either marginally effective or, in some cases, didn't work at all.

1/ Contracting for Computer Software Development -- Services Problems Require Management Attention to Avoid Wasting Additional Millions, a Report to the Congress Prepared by U.S. GAO, Nov. 9, 1979.

C. DOCUMENTATION STANDARDS

1. Because documentation standards were deficient, the scope and content of some technical documents became points of contention between the government and contractors.

2. The documentation standards specified at contract award were replaced with a more comprehensive standard calling for greater detail. However, the contractor had already prepared documentation under the old standard, and thus was required to redo documentation. This double standard resulted in a six month delay and an increase in cost, in addition to causing other phases of the program to slip.

3. Government and contractor personnel were unable to reach a common understanding of the scope of the design plan, causing major program problems.

4. The government, because of schedule pressures, eliminated documentation essential to successful software development. In some instances, the government incorrectly scheduled the delivery of documentation.

D. REQUIREMENTS DEFINITION

1. Requirements were too general and were therefore open to subjective interpretation.

2. The government failed to acknowledge that design requirements were inadequate. Well defined requirements did not become available until one year after contract award.

3. No clear understanding of the required capabilities existed between the government and contractor before design and implementation started.

E. SOFTWARE UNIQUENESS

1. Neither the government nor the contractor were able to estimate with any quantifiable degree of accuracy the amount of software (lines of code) to be developed.

2. Government managers and contractors often underestimated levels of development complexity with the result that both cost and schedules were impacted.

3. Difficulties were experienced in measuring progress in software development and assessing the status of the development.

F. SOFTWARE DEVELOPMENT MANAGEMENT

1. Government personnel, although technically competent, were inadequately trained, ill prepared, and understaffed to manage complex software developments.

2. No aids or tools were available to assist government personnel in managing software developments.

3. Unrealistic government schedules for the development of operational systems caused pressures and compressed activity to the extent that many essential management and technical reviews and validation steps were eliminated.

4. Both government and contractor management were inadequate for the complexity of the programs involved. Often, only a single design review was scheduled for the life of a lengthy contract.

5. To meet government specified (unrealistic) schedules, high risk approaches to development were initiated. These approaches subsequently failed because neither the government nor the contractor knew how to manage them or comprehend the level of risk.

6. Government managers were unable to track programs and as a result always found themselves in reaction modes (reacting to adverse situations which have already occurred) rather than in planning nodes (making decisions before adverse situations occur and thus avoiding them). However, few, if any, tools or aids were available to assist government managers. In one instance, the government was totally unaware of the level of complexity required for system integration, and a preliminary design, subsequently, described as "woefully inadequate," was produced. This situation caused major problems for the entire program.

7. As a result of grossly underestimated costs, the government, at contractor selection time, changed the primary evaluation criterion from technical excellence to cost. The project failed.

8. At one point in a contract, open action items totaled over 100, with approximately 25% six months old or more.

G. SOFTWARE LIFECYCLE COST ESTIMATING

1. Both the government and contractor grossly underestimated cost and schedules. In addition, many cost factors were either inadequately considered or, in some instances, totally ignored.

2. A clear understanding of the GFE required for the program did not exist between the government and the contractor. It was also unclear whether the contractor was capable of actually completing a successful system.

3. Lifecycle costs were based to some extent on estimated number of instructions which, in all cases, was estimated extremely low. For example, in one case, instructions were estimated at 30,000 lines; the last estimate before contract termination was 275,000 lines of code. In another case, for a single component of the system, the estimate was 17,000 -- 200,000 lines of code were eventually generated.

4. The software cost for one project began at $7M. Within seven months, the cost was at $9.3M and still rising -- a $2.3M overrun.

H. HARDWARE CONSTRAINTS

1. Because the government specified hardware was inadequate for the proposed system, the software had to be modified to compensate for hardware deficiencies. This situation resulted in substantial increases in software development costs and schedule delays.

2. Rigid government hardware specifications contributed significantly to the degree of complexity required to successfully complete the project (i.e., 96K core memory was provided, 416K core memory was required).

3. The government had an inadequate understanding of the cost of hardware versus software and, therefore, failed to make cost-effective design trade-off decisions. Software development costs much more than hardware.

I. CONTRACT TYPES

1. Because the government and contractor inadequately assessed the risk involved in the project, an inappropriate type of contract was selected for the work. As a result, the government and contractor based subsequent decisions on a faulty foundation and the project failed.

2. On one project, the selection of a Fixed Price Incentive type contract should not have been made because the risk level was too great. Problems resulting from this poor selection of contract type caused major problems for both the government and the contractor.

3. On another project, the selection of a Cost Plus Incentive Fee contract with schedule slip incentive (which was extremely steep) caused excessive schedule pressures. To meet these pressures, the contractor adopted a high risk development approach (software development parallelism), and the project failed.

4. In the Project Management Plan, the government set hard dates for certain milestones (i.e., IOC), predicated on a specific contract award date. However, the contract award date was five months late and the milestone dates were not changed. Although it was impossible to meet the milestone dates, substantial pressure resulted from efforts to meet them. B

5. Budget related fiscal year pressures caused the government, in some instances, to begin projects without proper planning. As a result, both the contractor and the government started off on the wrong foot.

J. SECURITY CONSTRAINTS

1. The impact of security requirements on cost, schedules, and quality of products was not clearly understood by the government and contractors.

2. Billet transfers between agencies took substantially longer than expected.

3. A sufficient number of SI cleared personnel with the required technical backgrounds were not supplied to the project at appropriate times.

4. Because background investigations took longer than anticipated, critical personnel were not available when needed.

CASE HISTORY I

A. PROJECT SUMMARY

The government contracted for the design and implementation of a large data processing system that would automate field station handling of time-sensitive data to include identification, selection, extraction, preparation, and dissemination of product and technical information to tactical consumers. The contract was awarded in February 1976 and terminated in October 1978, after an expenditure of almost $13M. The original contract price was $9.2M. The hardware developed and integrated by the contractor operated properly and could be used by other government agencies. There was little, if any, usable software delivered.

B. CONTRACT PERFORMANCE

The government awarded the contract in February 1976, exactly one year after work began in developing the Functional Requirements for the proposed system. The budget of $10.9M for contractor assistance was approved in June 1975. The government admits that they were under heavy time constraints, and because of this pressure produced an inadequate Functional System Description (FD) that was used as input to the subsequent Purchase Description (PD) document. Likewise, a System Acquisition Plan (SAP) was not prepared.

However, in June of 1975, a draft PD of the system was given to seven prospective bidders; the completed PD was distributed in August 1975 with the RFP. Four responses to the RFP were initially evaluated as follows:

BIDDER	COST $(M)	SCHEDULE	TECHNICAL
Contractor A	9.9	June 1977	Unacceptable
Contractor B	13.7	August 1977	Acceptable
Contractor C	26.6	September 1977	Acceptable
Contractor D	27.8	April 1978	Unacceptable

The government had questions concerning proposed technical solutions of the offerees, and after these questions were clarified felt that Contractor A could be made technically acceptable through negotiations. Contractor A was awarded the contract because the government felt that the difference in Contractor B's technical approach was not sufficient to justify the added investment.

The source selection panel showed concerns, which were ignored, in the following areas:

- Contractor A's technical proposal lacked detail concerning hardware implementation.
- Contractor A's technical proposal lacked detail in the software approach to the problem solution.
- Although Contractor A proposed a schedule to match the government's request, government personnel estimated that Contractor A would be six months behind in meeting it.
- The Evaluation Panel was concerned that Contractor A was "buying in" to the particular business area.

The government, in selecting Contractor A, had certain problems, inconsistencies, and pressures. For example, the source selection was inconsistent with the selection criteria. The RFP stated that the award of contract would be based on the best overall evaluated proposal consistent with the evaluation factors listed, and the contractor must be rated "acceptable." Cost was the least important of the evaluation factors stated in the RFP. However, the government changed its selection criteria and omitted certain procurement steps as illustrated by the following:

1. The contractor chosen (Contractor A) was initially evaluated as unacceptable. Contractor A's offer was the least desirable technically.

2. Before the selection process was completed, cost (price) replaced technical excellence as the prime selection criterion. However, no change was made in the RFP as to priority of criteria, and the bidders were not notified. On the basis of the new criterion, Contractor A was selected because Contractor B and Contractor C's technical superiority were not commensurate with the difference in cost.

3. Even though this was Contractor A's first effort in the particular application area, a formal pre-award survey was not accomplished.

In addition, the government had a budget problem. Their Independent Government Cost Estimate (IGCE) of $10.9M was grossly inadequate. It was based on earlier inapplicable estimates. The IGCE cost estimate favored Contractor A, because although the government believed Contractor A's ($9.9M) estimate was low, it was closer to the IGCE cost estimate than other bidders. It should be noted that the estimate was developed prior to PD preparation and was not updated.

A Fixed Price with Incentives (FPI) type contract was chosen by the government because the technical risk in developing and integrating the software was considered low. The incentives were negative incentives concerning the schedule requirements. This was done despite a slip of five months in the Contract Award Date. The award date was important to the government because of specified "fixed" milestone dates. For instance, the integration and test completion and the IOC milestones did not change from those originally indicated in the Project Management Plan schedule even though the contract was awarded later than anticipated.

There were also conflicts in the scope of the effort (lines of code to be written), hardware constraints (noted by all offerees except Contractor A), and confusion as to the amount of change to existing software which was not cleared up prior to start of contract.

At the time the contract was terminated, the government had estimated that an additional $12M would be necessary to complete the development.

C. FINDINGS

The findings presented below are the major reasons the government terminated the contract.

1. The Purchase Description (PD) was deficient and caused problems on the contract. The following critical issues were found to be contributing factors:

a. The document was drafted without benefit of supporting documents such as a concept of operations. Therefore, it was difficult for the contractor's software designers to understand the operational mission.

b. Additional requirements were added during the PD preparation which were never a part of the functional system description.

c. A very tight schedule did not provide the opportunity to review and validate the PD before issuance.

d. The PD contained rigid hardware specifications coupled with general system performance specifications, producing inconsistencies and contradictions.

e. System hardware selected was a complicated amalgamation of two minicomputers (PDP-11's) operating in a dual mode, a different minicomputer (TI 960) driving a display and an in-house developed

microprocessor. The system used four different programming languages. This level of complexity, and its impact on cost and schedules, was not well understood.

2. The magnitude of the required software development effort was underestimated by both the government and the contractor, causing schedule delays and severe cost growth on the contract. The government and the contractor were unable to estimate the size of the effort with any degree of confidence, as can be seen from the following factors:

a. There were inconsistencies and confusion in the number of lines of code required. At contract award, the contractor estimated 30,000 lines of code, and the government believed that the contractor was estimating 60,000 lines of code. However, the government had independently estimated that 110,000 lines of code needed to be developed. Prior to terminating the contract, the contractor estimated that 275,000 lines of code would have to be written and integrated with 265,000 previously available lines of code, for a total software package of 540,000 lines of code.

b. The contractor underestimated the modifications required to available software (operating system for two directly coupled mainframes, Data Base Management System, and Terminal Interface Program). The need to modify this software was known to both the government and the contractor prior to contract award; the contractor believed he only had to make minor modifications to the software. "Minor" was never clarified. However, the contractor was required to make major modifications, and expend resources not originally planned. In addition, because of these major modifications the

software was now hand-tailored and non-standard, and the effect of the increase on lifecycle cost was not considered.

c. Premature hardware specification by the government contributed to increasing the difficulty of software development. For example, although a 96K core memory was initially considered necessary, six months into the contract, the contractor determined that 96K was not enough and a change order for 320K of additional core was executed. During the competitive phase, other contractors had claimed that 96K was insufficient. The issue was left unresolved.

3. An unrealistic schedule produced schedule pressures that caused higher risk approaches to be taken. The related factors are as follows:

a. An overriding issue was the insistence by the government to acquire operational capability as soon as possible. This schedule pressure caused elimination of required documentation such as the Concept of Operations Plan, the System Acquisition Plan, and a fully developed Functional System Description. In addition, because of this pressure, normal checks and balances such as a Preliminary Design Review were omitted, and the PD analysis and validation was deficient.

b. Prior to award, the acquisition team recognized that a 4-6 month schedule slip could be expected of the contractor. Three of the four bidders (the winning contractor was the exception) expressed disagreement with the specified government schedule (despite very strong statements in the RFP of the government's desire to stick to the schedule). However, despite available contradictory information, the government contracted according to its original schedule.

c. Because of the tight schedule, the contractor developed a high risk approach of software parallelism to meet the schedule. This risk was improperly assessed.

4. The development effort was performed using a Fixed Price Incentive Fee (FPIF) type contract -- the wrong type of contract for the effort for the following reasons:

a. Substantial software development was involved.

b. Significant and difficult system integration was required.

c. The PD contained many general requirements that proved to be subjective or open to much interpretation.

5. The contractor's management was inadequate during the critical first year of the contract. Even though the contractor believed he had an excellent system of internal management reviews, he acknowledged that such was not the case on this contract during the initial stages. The contractor and the FDR team stated that the contract, during the first ten months, was managed in a "cost-plus" mode. The contractor made a major mistake in accepting the contract as a FPIF; the risk was too great. The contractor also had other management problems, as can be seen from the following:

a. The contractor did not question or challenge any part of the PD prior to award, because he thought by doing so he would lose the contract. As a result, the contractor made too many incorrect assumptions.

b. The contractor believed the government had accomplished a system engineering design, but did not check or validate this design.

c. The contractor had considerable difficulty understanding the government's direction and guidance on the status of certain contract amendments.

d. Because of perceived clearance problems, the contractor believed he did not have the traditional program control mechanism to provide necessary contractor management overview.

6. Government contract management functions were lacking. For example, only one design review was scheduled throughout the life of the contract. In addition, a major milestone (Software Design Plan approval) was missed in June 1976, yet the management attention of both the government and the contractor was not focused on it until November 1976. It was also unclear whether or not the monthly reports provided by the contractor were used in any meaningful way by the government in the early stages of the contract. It became evident that the internal organization reporting was deficient, and hastily and irregularly prepared during the critical first year of the contract. Other management deficiencies noted are as follows:

a. Acquisition team manning apparently was insufficient to provide the necessary technical and contractual direction. At one point, open action items totaled over 100, with approximately 25% six months old or over.

b. The involvement of the Contracting Officer was minimal.

c. There was a lot of internal pressure not to make any changes in the PD because of the fixed price contract. The government was concerned about "change-in-scope" claims. The contractor stated that this resulted in his inability to receive needed guidance on inconsistencies in the PD.

d. Although it was a fixed price environment, the contractor asserted that the government was performing micromanagement to the detriment of the contract.

e. The contractor estimated a loss at completion of $7M on a $9.2M contract in March 1977 (contract was terminated in October 1978), and yet there was no strengthening nor augmenting of the government acquisition team.

7. The government management of the effort was deficient and caused confusion. The Program Management Plan attempted to overlay a program management organization on existing line organizations. The resulting management was a mixture of matrix and line management styles without a clear definition of which took precedence. The degree of control was constantly challenged. Other management problems identified were:

a. The acquisition team was technically competent; however, they were admittedly untrained, unprepared, and understaffed for an effort of this type. Upper management involvement was minimal during the critical first year of the effort.

b. Separating fiscal control and internal reporting responsibility from the acquisition authority produced built-in controversy.

c. Although senior management was kept aware of the status of the program on a regular and candid basis, little action was taken to address problems.

8. Government management of the software problem and use of software expertise was not well exercised. The System Acquisition Team did not have representatives from the software organization integrated with the team; they

were simply called upon to help. Members of the acquisition team, while having varying amounts of applications programming experience, were limited in their system software experience. The contractor believed that the government acquisition team did not perceive the size of the problem, and, as the magnitude of the software job became apparent, the government took no extraordinary steps toward enhancing its software expertise on the team. Furthermore, nine months into the contract, the contractor performed a number of system studies which produced a fairly consistent answer on the number of lines of code required (270,000); yet again, the government did not exercise any augmentation on the software team. The project had reached the point where very little progress was being made relative to expenditure of resources.

9. Software documentation standards were deficient, contributing to the continuous controversy of the adequacy of the contractor's System Design Plan. The contractor and government could not reach a common understanding of the scope of the Design Plan, which resulted in significant problems. The documentation standard specified, LYN-6-70021A dated 10 December 1971, was much less comprehensive than other standards available at the time (e.g., MIL-STD-490 and DoD Instruction 4120.7). In addition, the Final Computer Software Design Plan was not scheduled for delivery until the actual system was delivered -- far too late in the project.

10. The contractor had problems in obtaining a sufficient number of cleared personnel to work on the project. He failed to account for even normal clearance lead-times in agreeing to a software design plan 90 days after contract award. Therefore because of clearance problems, an adequate base of critically needed expertise was unavailable, causing the contractor's performance during the early steps to be subpar. Although the contractor

believed that the clearance processing time delineated below would be followed, he had no substantial basis for this belief:

a. 120-150 days for persons who had no active SI clearance or existing EBI.

b. 45-60 days for people identified to be expedited.

c. 60 days for those who had an existing EBI but no SI access.

The misunderstanding was further compounded in obtaining clearances for personnel identified to be expedited because:

a. The contractor requested that 35% of his personnel clearances be expedited, vice an expected norm of 10%.

b. There was general increase in the average processing time experienced at DIS for BI's.

In addition, clearances for the contractor's employees indoctrinated SI on other agency's billets took an exceptionally long time to be granted by the contracting agency.

CASE HISTORY II

A. PROJECT SUMMARY

The government contracted for the design and implementation of a large, real-time data processing system characterized by extremely high data processing input/output rates. The data was to be processed and reports sent to consumers within one hour after receipt of input by the automated system.

In April 1973, the government signed a multi-phase, CPIF contract for $9.1M. The contract overran by $15M and was late by about 3 years.

B. CONTRACT PERFORMANCE

The project began in September 1972. In April 1973, an RFP study phase was submitted to three contractors, with the government only committed to doing the study phase. At the conclusion of the study phase, the government evaluated the results of the three contractors and selected one to continue through design and implementation of the proposed system.

The study contractors responded with their interpretations of the problem, design and code of critical algorithms, hardware configurations, and cost and schedule estimates. The study contractor selected to continue was evaluated on its past performance and management techniques. In retrospect the government did not pay enough attention to evaluating the extent to which the contractor was performing testing of software at the various development phases and overall development methodology.

Also, the government continued gathering additional problem definition data during the study phase, but elected not to present it to the study contractors. This decision resulted in the hardware being selected before this additional problem definition data was factored into the program. The hardware

resource margin was later discovered to be not enough to accommodate the additional load. Therefore, the system design underwent changes during the course of the contract, causing software problems and, subsequently, cost increases. For example, it was essential to develop a dynamic file management subsystem which had not been planned earlier. Also, there were many modifications made to the operating system as a result of the additional information and lack of original technical understanding.

The original estimate for software was about $7 million. At the preliminary design review in November 1973, the contractor estimated a $2.3 million overrun in software which had already occurred by that date. This generated considerable controversy. The surprise to the government was that all three study contractors estimated approximately the same amount for software development, and all three were incorrect (low). New specifications were presented in May 1973, causing increased costs.

A problem arose regarding what to do with contractor personnel in phased projects during the periods when government management is deciding how to proceed. In this instance it was felt that the contractor's study team was small enough to retain and was able to proceed with work. In fact the team, starting at 12 in number, expanded to 50 during this period.

The documentation standards being used were inadequate. Therefore, in January 1974, a military standard specification was injected which was very useful in that it caused a point-by-point examination and established testabilty of the system. This caused a slippage in the critical design review to the summer of 1974.

The project had heavy penalties on schedules. The contractor pressured the work force by expanding parallel activity, having smaller increments of inspection, etc. In January 1975, the contractor projected an overrun, and the government made a schedule relief offer which was turned down.

After a more vigorous review of the remaining work, the project was rescheduled for another 6 months and delivery was set for November 1975. Another overrun occurred. The project manager felt that adequate cost and schedule information was received, but that techniques for inspecting this information to detect problem areas were weak.

C. FINDINGS

The following points with regard to this case should be noted:

1. It was not clear until well into the program that the government had the project under control.

2. The government and the contractor assumed that the problem was state-of-the-art, as were the individual algorithms. However, system integration vastly exceeded the state-of-the-art because of precision and speed requirements.

3. The project was budget driven because beginning the contract during the fiscal year was a driving force. The government felt that a more orderly acquisition, unconstrained by fiscal year constraints would have generated much more realistic target costs and eliminated some of the confusion and misunderstandings that occurred at the start of project.

4. Although the government had clues of trouble early in the project, the contractor did not acknowledge this trouble. The government believed that the contractor recognized a messy situation, but assumed that plasticity of schedules, costs, etc., would allow latitude to resolve them. The contractor

claimed that they did not believe that the schedule and cost were elastic. The significant problems were: performing out-of-scope work on the promise of the government's technical office to make the contract right after-the-fact, and a significant underestimation of proposal time for the technical tasks.

5. Inadequate design requirements were a critical issue. Well-defined requirements were not available until a year after contract start.

6. On the subject of incentive contracting, there were several critical issues:

a. The pressure of incentive was great enough for the contractor to agree to a price which was less than their estimate of the work.

b. On the subject of schedule-slip incentive, the contractor agency must believe in its commitments. Yet, in this instance, for a contract costing $100,000 per week, it was set up on the premise that one week's schedule was worth $500,000 in overrun.

c. The schedule-slip incentive was so steep that it forced big decisions to be made, rather then allowing for a series of smaller ones.

d. The incentive fee structure constrained the contractor on decisions for manpower loading.

7. The biggest flaw in the contract was pricing. The size of the technical job was inaccurately estimated. Sizing was influenced by previous experience on batch jobs, even though this system called for on-line transaction processing activity. Although the entire system was estimated at 55,000 instructions, one component, estimated at 17,000 instructions, took 200,000 instructions.

8. There was too much inexperience, especially on the government side, on incentive structures. The contractor also erred in applying incentives based on batch projects on a transaction-processing project (which was new to them in this instance).

9. No adequate definition of the problem was made. Management techniques were not vigorously applied until April 1975.

10. There were no government audits or walk-throughs exercised.

CASE HISTORY III

A. PROJECT SUMMARY

In this case, the contract was for the development of a major subsystem of a large automated system designed to maintain and retrieve engineering data and related information to support the agency's engineering and procurement function. The subsystem was designed to provide data to aid engineers in reconstructing the original configuration of a major part of hardware assembly before modifications were introduced. The original cost and time of the subsystem contract were to be $93,039 and 28 months. Contract modifications and cost overruns increased the final cost to $123,726 in 28 months.

B. CONTRACT PERFORMANCE

The contractor was to develop a formalized plan along with detailed procedures for implementation, as well as computer programs needed to establish the subsystem. The subsystem was to be developed as a modification to an earlier contract. The subsystem contract did not define standards or criteria for measuring product quality. Timetables were provided within the contract for completion of individual requirements. Although no standard test data package existed for acceptance testing, the contractor was required to demonstrate the ability of the computer programs to operate and to correct or replace any unsatisfactory work.

The software printed three reports. Agency officials stated that requirements were inadequately defined for one of the reports. A contract modification was required at a cost of $7,902 to correct this deficiency. About 18 months into the contract period, a second modification was negotiated which cost $19,044. Of this amount, about $10,000 was for correcting a deficiency and the rest for adding another function. An agency programmer

stated that before the outputs could be used he had to write auxiliary computer programs whose output supplemented the information shown on the contractually-required reports.

Generally the contractor did not meet specified completion dates. The agency delayed payments to the contractor for failure to meet the schedule. About $3,741 was awarded to the contractor for cost overruns, bringing the total amount paid over the original cost to $21,643, excluding the cost of added functions.

System documentation seemed to be adequate, but the computer programs themselves were not properly documented.

C. FINDINGS

The agency did not perform, or contract for, adequate system analysis work. This is indicated by the fact that specifications were not adequately defined to assure that the software would have the necessary capabilities. Quality assurance and testing procedures in the contract were inadequate to assure that the software would meet user needs without modification by agency programmers. Documentation standards should have made program documentation mandatory.

CASE HISTORY IV

A. PROJECT SUMMARY

The agency, which used non-uniform accounting systems, contracted for the design and development of a centralized accounting system to increase responsiveness, timeliness, accuracy, and to overcome inefficiencies in the operation of 10 accounting offices within the agency. The cost and time spent were estimated to be $958,682.40 and 27 months. After 30 months the system was only about one-fourth complete, and the agency cancelled the contract. Although the system was not complete, the agency paid about $981,200.

B. CONTRACT PERFORMANCE

At the time the contract was let, the agency had no formal design or specification documents for the contractor to work from. The agency had collected a list of concepts and standards which supposedly were the basis of a conceptual design. The Cost Plus Fixed Fee contract obligated the contractor to deliver a workable accounting system.

The contract called for three development phases. Each covered the development of a major accounting subsystem, with the first phase to include the overall system design. Each phase was further divided into conceptual design, detail design, and implementation. The agency was to approve each phase at completion. Under the terms of the contract, the contractor was to develop a project control plan for such items as progress and cost reporting, documentation review, and acceptance testing. The contractor was responsible for formulating the criteria by which the agency would judge his performance. The contract called for the contractor to submit proposed changes along with the reasons for them to the agency for approval. The agency reserved the right to make modifications it considered necessary to ensure that the system fit its

needs. System documentation requirements were fairly detailed, but guidance on program documentation only referred the contractor to agency standards. No subcontractors were involved.

In the first phase, the contractor was to develop a general design of the overall system and also the design for one major subsystem. In the development of this phase the contractor stated that he encountered two problems -- (1) the agency staff generally resisted the new system and (2) virtually none of the existing accounting processes and procedures, which the new system was to automate, was documented. When he submitted his report on the first phase, the contractor assumed he could immediately start on the next phase, but agency review of the report took about 250 staff-days. The contractor said that agency delays and the low level of agency participation together added about 350 staff-days.

As the contractor entered the second part of the first phase, he still encountered problems he blamed on (1) the poor quality of agency review and agency staff participation, (2) agency indecision, and (3) agency changes in direction. The contractor felt the changes deviated from earlier agreements and that some of them could not be made. Some products were submitted for agency approval three or four times. Disagreements arose over the amount of documentation necessary, and the lack of existing agency procedures continued to be a problem.

The contractor contended that the agency insisted on a system that was not needs-oriented but one which was designed to satisfy many individual preferences. To illustrate his point, he compared the excessive number of management reports asked for in this system (188) to the maximum number of reports (44) called for in four other agencies' accounting systems. The agency

director of systems admitted the general lack of direction and specifics in the contract and stated that more definitive planning and guidance were needed to let the contractor know what was expected.

To determine what, where, and how data would be stored and retrieved, the contractor asked the agency to specify (1) the computer hardware to be used, (2) the data base management system (DBMS) 1/ to be used, and (3) system requirements, such as output reports and transaction coding. The agency took about six months to provide guidance in these areas, and during that time, the contractor proceeded to design conventional file structures and processing routines.

When the agency finally decided on the DBMS, the design had to be substantially reworked. The contractor said that even small changes generated extensive reviews to determine all other areas which were affected and required change. Of the six major reasons given by the contractor for overruns, three dealt with changes that were constantly being made to both the requirements and the operational environment and the impact those changes had on system development.

1/A Data Base Management System (DBMS) is a computer software package which can facilitate the management, manipulation, and control of data.

Agency officials maintained that the contractor's report reflected a general lack of understanding of the job to be done, that deliverables were inadequate, and that agency documentation standards were not followed. Conversely, the contractor stated that agency staff assigned to work on the subject did not understand the agency, its mission, or its needs.

After about 2-1/2 years, the agency had paid the contractor over $981,000. The system was estimated to be only about one-fourth complete, and the time frame had exceeded the original estimate by several months. At this point the agency terminated the contract. The contracting officer said that the agency's counsel had informally advised him that a precisely defined set of requirements was never incorporated in the contract. Therefore, since neither party could define the product, in their unofficial opinion, the agency could not force the contractor to finish the system for the maximum cost allowed by the contract.

C. FINDINGS

Too many factors were left to be subjectively determined outside the provisions of the contract. This condition is evidenced by such things as arbitrary changes, disagreements on various subjects, and the agency's admissions that more specific requirements should have been given to the contractor. These problems could have been avoided even if the exact characteristics of the needed software were not known at the time the contract was let.

First, user requirements, performance specifications, quality control procedures, and documentation items required could have been specified in the contract to establish a framework at the outset. Second, the agency should have required the overall system design to be defined in a first phase to the point that its adequacy could have been determined, approved, and frozen to

allow stable and systematic development before committing itself to the rest of the contract.

Other factors which contributed to the failure of the contract included the agency's failure to:

- Establish firm, realistic requirements and fix them.
- Render timely decisions and timely review of products.
- Promptly carry out responsibilities so that software development was not delayed and so that the way was left clear to invoke contract penalties against the contractor if he failed to perform.
- Maintain adequate monitoring and tracking procedures which would have avoided allowing the entire original contract amount to be spent in the first of three development phases.
- Define the user requirements served by the existing accounting system.
- Create an environment which enhanced chances for success, including consensus on agency needs, proper orientation of agency staff, and provision of a strong focal point of qualified agency staff to work with the contractor.

APPENDIX A

SUMMARY OF INDUSTRY COMMENTS

This appendix contains a summary of comments presented by industry to the SADWG. These comments have not been quantified or qualified in any way, and every attempt has been made to preserve the intent and accuracy of industry views. For presentation purposes these comments are organized as follows:

- Overall effects of current policies and practices
- Problems

A. OVERALL EFFECTS OF CURRENT POLICIES AND PRACTICES

- Very few military software systems have been delivered on original schedule, within original budget, meeting original requirements, and achieving original cost-benefits.
- Every agency/service, in spite of the fact that there are a number of policy directives like DoD 5000.29, establishes their own. There is general misunderstanding by the agencies/services of the intent of standards directives like DoD 5000.29.
- Competitive procurements encourage contractors to offer maximum technical effort for the dollars, with the result that contractors skimp on providing important supporting services such as management, configuration control, and quality assurance. There are no guidelines as to acceptable support requirements.
- Industry is discouraged by lack of results from the studies conducted by the agencies/services concerning improvements to ADP development practices. Industry has little confidence in government doing anything because a number of studies have been conducted and nothing has happened.
- Budget estimating for development systems is grossly inadequate. Budget estimates are prepared long before it is determined what is to be developed and the scope of development.

- Current state-of-the-art in cost estimating is more art than science, for unless one can measure, quantify, and process something, it isn't a scientific process, and this we cannot do as yet.

- Restrictive security requirements impact software development costs, but are rarely taken into consideration.

- The further one is from the implementation phase, the higher the risk of cost estimating. When doing cost estimating, the uncertainty of risk is a function of how far away one is from being able to implement a system. Currently, budget estimating is done years in advance of system implementation.

- Government program managers are either not trained at all, or inadequately trained to manage ADP development programs.

- Every agency/service does its own development to its own schedules with little or no regard to transporting software to other agencies/services with similar needs. Software solving similar problems is developed over and over again.

- Government is loathe to establish IV&V on programs unless they are large and complex; thus, because of the lack of adequate definitions for "large and complex," many systems are developed without an adequate level of IV&V. If a contractor bids an adequate level of IV&V in a competitive bid, the chances are good he will not win.

- The software industry is required to develop error-free systems on a one-time basis. This is almost an impossible task. Other industries such as construction, computer hardware, tank building and ship building develop prototypes. Although prototyping is an accepted practice in these industries and in hardware development, software, which is often far more complex, is rarely prototyped and, if costed in a competitive bid, that bid is unlikely to win.

- Proposals are usually "science fiction" written to win contracts. Because requirements are constantly changing or ill-defined, the contractor very seldom delivers what was originally proposed.

- The government contracts for computers, in a mass buy, irrespective of what kind of application they will be used for. Any hardware deficiencies/limitations must be made up by the software regardless of the fact that software is much more expensive than hardware. Sometimes physical

limitations prevent software developers from building systems to meet stated requirements, or from building systems economically.

- o A clear understanding of what is to be developed at the time of budget estimating does not now exist. Cost cannot be accurately estimated until after a design is complete. Currently, the government has no procedures to adjust budget estimates based on later findings, with the net effect that programs meeting requirements appear to be overruns when in fact the case might be that the programs are meeting "realistic" budgets.

- o Government does not want to pay for configuration management/control, which is the one management technique/tool that would provide the most improvement to software development.

- o The government does not take full advantage of software transportability where applicable, hence the same software is developed over and over again. Unlike hardware, software costs are confined to the initial investment plus maintenance.

B. PROBLEMS

1. Documentation Standards

   - o Virtually every government agency handles the development of software using different standards despite the fact that regulations such as DoD 5000.29 exist.

   - o Standards are not precise enough to eliminate misunderstandings between contractors and the government; for example, the level of detail in documentation is often a question of interpretation.

   - o The vast number of different standards and guidelines decreases the value of standards. It is impossible to be familiar with all of them.

   - o Each agency/service (and smaller activities in many instances) implement DoD standards in their own way. This results in confusion for contractors, and probably decreases the synergism possible throughout all DoD.

2. Requirements Definition

- o Requirements documents are usually too ambiguous or vague, and subsequent documents produced from these requirements are inadequate.

- o Although technical people and system users may agree on specific requirements, technical people find it difficult to describe the system to be delivered in terms of requirements and therefore exclude the user in the process of translating requirements to final system. This often leads to user dissatisfaction with to delivered system because trade-offs were made which the user either did not understand or, because he was not part of the process, did not want.

- o Requirements are constantly changing and often force overruns; management doesn't know how to handle this problem in relation to modifying original budget and schedules. Not all modifications to original budgets and schedules are overruns but are the result of honest requirement changes.

- o "Finished" requirements do not exist with military software systems because they are constantly changing; military software systems are never "complete."

- o System developments are too long; C3I systems generally have a development cycle of up to 14 years before a system is operational. Other systems take 5-6 years. When things take this long, you forget what you started out to do. Very few people remember the initial requirements and the cost-benefits to be derived. Because these development cycles are so long, more often than not, requirements are obsolete before the system is completed.

3. Software Uniqueness

- o The government lacks the understanding that software configures the whole system much like mortar between bricks, and is much more expensive than hardware.

- o Because software developments produce only paper from the requirements phase through the coding phase, which is often 2-3 years in process, software projects are difficult to manage. This problem is compounded by the government's lack of adequate tools to manage software developments. Because of ill-management, deficiencies in the software system often remain undetected until the system is built.

- More than 60% of lifecycle costs are spent after the system is built, not in building it. These costs are attributable to errors introduced in the requirements, design, and development phases that remain undetected until the system test phase.

- Although prototyping is an accepted practice in hardware development, software is rarely prototyped and, if costed in a competitive bid, that bid is unlikely to win.

- Because management fails to appreciate the importance of adequate testing, design problems receive priority and testing is kept to a minimum with resulting consequences. Hardware is developed at the manufacturer's site in a friendly environment while more complex software is usually developed at the customer's site in an often pressure environment.

4. Software Development Management

   - Government ADP program managers are not trained to manage ADP development programs.

   - Government personnel are often unable to evaluate a good design and relate that design to requirements due to lack of technical depth and management aids.

   - Many government managers often fail to make decisions or consistently change their decisions, and thus adversely impact contractor productivity.

   - Government procuring agencies are inadequately staffed (or lack technical support) to manage software development efforts.

   - Few records are kept concerning the history of past programs that can be used on future programs--"lessons learned." There are no guidelines for recording this data at present, with the result that information learned from past experience is lost.

   - Government agencies spend the majority of development funds on design, code, test, and maintenance, and usually spend little money on verification and validation, configuration management, quality assurance, documentation, etc.

- There are no adequate tools for tracing requirements through the development lifecycle, and no easy-to-use process to determine the degree to which designs meet stated requirements.

- The government attempts to foreshorten schedules unreasonably, making on-time accomplishment of a project highly unlikely, and is often unaware of the risk introduced as a result of those actions.

- Industry does not have, nor will the government allow (mostly because of security), integration, testing, or configuration management groups across projects. Each project is treated as a separate entity. The government's reasoning is that each project is different and, therefore, an overall group would be inefficient.

- Competitive procurements encourage contractors to offer maximum technical effort for the dollars; however, by doing so, contractors tend to skimp on supporting services such as management, configuration management, quality assurance, etc., because the government minimizes the importance of these efforts.

5. Software Lifecycle Cost Estimating

   - Government, and to a large extent industry, has little understanding of how to estimate software development cost and schedules with any quantifiable degree of confidence.

   - The current state-of-the-art in cost estimating has as its basis the number of instructions and cost per instruction. Cost per instruction varies with system complexity and ranges from $3.00 per instruction to $250.00 per instruction, and is by itself a poor measure. It is difficult to estimate the number of instructions prior to design. Estimating "horror stories" are well known but government and contractors have not become any better at it.

   - Available cost estimating models are rarely used by the government because of their cost and government and contractor's inability to qualify output.

   - Standards do not adequately address the lifecycle; for example, can software maintainability be quantified and placed in a standard?

- The current state-of-the-art in cost estimating is more art than science. A need exists for a set of cost estimating metrics that are consistent with the level of detail or phase in which the project is currently involved. Software cost estimating techniques, although primitive at this time, have been developed for use in cost estimating the implementation phase (phase following design). Little or nothing has been done for the other phases.

- The further away from the implementation phase, the higher the risk of cost estimating (the uncertainty of risk is a function of how far away the system is from being implemented).

- There is a lack of understanding of how to estimate software costs (cost estimating accuracy is directly proportional to an understanding of the problem). Accurate cost estimation requires good requirements analysis and design both of which are currently lacking.

- Cost cannot be accurately estimated until a design is complete. In today's environment, neither the government nor the contractor can reasonably estimate the cost of a system prior to design.

- Congress establishes program development budgets without the knowledge that program costs cannot be accurately determined until after design and, in addition, that the estimates they receive do not include accurate estimates of the total lifecycle costs.

6. Hardware Constraints

- The use of government-specified hardware, in some instances, unnecessarily constrains contractors and definitely impacts cost, which at present is not fully understood. This requirement also eliminates contractors' proposals for possibly better and more cost-effective hardware/software combinations.

- The government often uses standardized hardware and programming languages without consideration of cost impacts on software development.

- GAO contracts for hardware to be bought, often years in advance of actual use, regardless of what kind of software/application it will be used for; thus failing to fully exploit new technologies as they become available.

- Use should be made of the inexpensive availability of "computational plenty" to build systems; it is very expensive to "shoe-horn" software into minimally acceptable hardware configurations, particularly since hardware is less expensive than software development.

7. Contract Types

- Often contract types do not fit the C3I environment. For example, it is not possible for the government to develop the detailed specifications required for firm-fixed-price contracts in an environment of constantly changing requirements.

- The government needs to rethink the process of procuring systems in "packages." "Package" systems include data collection and analysis through testing and maintenance of the system. In a "package" system, problems in one phase of development have a rippling effect in subsequent phases of development, causing these subsequent phases to be short-changed, thus impacting the completion of original requirements.

- Contracts do not permit the construction of test tools because of prohibitive cost. Software tools, because they interface deeply into the code, are single-project oriented, and if they cannot be used on more than one development, the government doesn't want to pay for them.

8. Security Constraints

- Restrictive security requirements impact software development costs but are rarely considered in cost estimating and schedules. There is also a lack of consistency in government clearance procedures which results in unnecessary delays in transferring contractors' clearances among agencies/services.

- The shortage of cleared personnel makes them valuable commodities, and as such they are susceptible to lucrative offers from competitors. When these personnel leave, valuable expertise is removed from projects and productivity suffers.

- While awaiting clearances or billets, personnel are not productive.

- The government is not flexible in moving people off of projects. Many projects require that the work be done at a government site and clearences are, therefore, necessary. Also, because of security requirements, personnel are assigned to a project for the duration. When the project is over, they don't want to go to another classified project for fear of being "pigeon-holed" again.

APPENDIX B

CANDIDATE LIST OF SOFTWARE DEVELOPMENT COMMANDMENTS

1. Stop putting design in the computer program (CP) requirements specification.

2. Stop applying preliminary design review (PDR) standards to the CP requirements specification.

3. Stop planning to enter full-scale engineering development (FSED) with the intent of not completing the specification of CP requirements until the time of preliminary design review.

4. Stop contracting for the parallel generation of CP design and completion of CP requirements specification.

5. Stop permitting contractors to design before successful system design review (SDR).

6. Stop letting FSED contracts with unbaselined CP requirements specifications (CP allocated baselines not established).

7. Stop letting combined definition and development contracts which do not require baselining (either by the government or contractor) of CP requirements specifications before starting design.

8. Stop permitting contractors to code before successful critical design review (CDR) with the exception of prototype code.

9. Stop letting FSED contracts which call for delivery of the Program Design Specification (PDS), but not for the Program Description Document (PDD) and Data Base Document (DBD), thinking that the PDS contains all the CP design.

10. Stop letting combined definition and development contracts which do not call for delivery of CP requirements specifications until the end of the development phase.

11. Stop issuing new or revised software acquisition/maintenance policy, directives, standards, regulations and instructions by DoD or its components, without first having them independently validated by two organizations which do not develop or maintain software, and without having areas in controversy or conflict with existing standards, etc., identified explained and justified.

12. Start issuing changes to resolve controversies, conflicts and errors in existing software directives, standards, regulations and instructions.

13. Start spending more effort in software management training (particularly for software contracting), to apply what is already known and to preclude application of misinformation.

14. Start requiring, and paying for, software cost, error and similar data from contractors, prepared to government standards (to be developed), to support improved software research, prediction and management.

15. Start requiring, and paying for, current cost and schedule estimates to complete development, based on data available at key development milestones, e.g., at CDRs.

16. Consider funding parallel development contractors through, e.g., CDR, with remaining development awarded to the one with "better" design, documentation, projected costs/schedules, etc.

17. Stop issuing RFPs or contracts for software which have not been reviewed by someone with training and understanding covering problem areas such as those identified above.

APPENDIX C

MEASURABLE MILESTONES

1. Requirements Specification Complete

2. Systems Requirements Review Committee
   Requirements Specification Signoff
   Systems/Subsystem Specification Complete

   Preliminary Design Review (PDR)
   Systems/Subsystem Specification Signoff
   Data Base Specification Complete

   Critical Design Review (CDR)

   CDR Rework Complete

   For each program:

   Unit Development Folder (UDF) Initiated
   Internal Program Specification Complete
   Design (Program Specification) Signoff
   Internal Design Walk Through/Signoff
   Code Initiated

   Code Complete (Error Free Compile)

   Internal Code Walk Through/Signoff

   Unit Test Plan Design

   Unit Test Plan Signoff

   Unit Testing Started (Run Against Test Plan)

   Debug (Recycle Until Successful Test Execution)

   Unit Test Execution/Signoff

   Program Maintenance Manual (PMM) Documentation Complete

   PMM Signoff

   Program Specification Updates Complete (As Built Documentation)

   Unit Development Complete/Integration Ready Review.