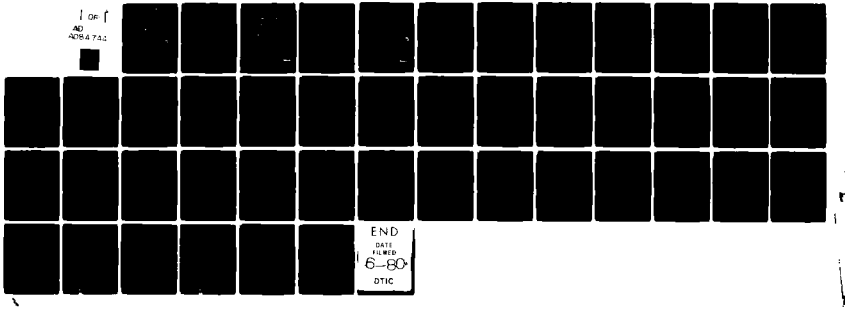
AD-A084 744 RAND CORP SANTA MONICA CA F/G 5/1
SOFTWARE REQUIREMENTS FOR EMBEDDED COMPUTERS: A PRELIMINARY REP--ETC(U)
MAR 80 S GLASEMAN, M DAVIS F46920-77-C-0023
UNCLASSIFIED RAND/R-2567-AF NL



UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER: R-2567-AF	2. GOVT ACCESSION NO.: AD-A084 744
3. RECIPIENT'S CATALOG NUMBER: AD-A08	
4. TITLE (and Subtitle): Software Requirements for Embedded Computers: A Preliminary Report	5. TYPE OF REPORT & PERIOD COVERED: Interim
	6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s): S. Glaseman and M. Davis	8. CONTRACT OR GRANT NUMBER(s): F49620-77-C-0023
9. PERFORMING ORGANIZATION NAME AND ADDRESS: The Rand Corporation, 1700 Main Street, Santa Monica, California 90401	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS: 1146
11. CONTROLLING OFFICE NAME AND ADDRESS: Requirements, Programs & Studies Group (AF/RDQM), Ofc, DCS/R&D & Acquisition, Hq USAF Washington, D. C. 20330	12. REPORT DATE: March 1980
	13. NUMBER OF PAGES: 34
14. MONITORING AGENCY NAME & ADDRESS (if different from Controlling Office)	15. SECURITY CLASS. (of this report): UNCLASSIFIED
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

LEVEL II

16. DISTRIBUTION STATEMENT (of this Report)

Approved for Public Release; Distribution Unlimited

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

No Restrictions

DTIC ELECTE MAY 28 1980 D

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Computers
Computer Systems Programs
Aircraft Equipment
Electronics
Command and Control
Aircraft Defense

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

See Reverse Side

DD FORM 1 JAN 73 1473

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

ADA084744

Results of a study of Air Force procedures for formulating and communicating software requirements and their effects on software acquisition for embedded computers. Conceptual models are used to describe the software acquisition management activities of the Air Force and the software development activities of prime contractors. It is found that, compared to contractors, the Air Force gives relatively little attention to software during preprogram decisionmaking. Moreover, the nature of software acquisition management problems is changing, and current management techniques are applied to embedded software with little regard for the comparative maturity of different application areas. At a more detailed level, the type of software expertise required for effective acquisition management has become at least as important as the numbers of specialists, documentation produced in response to current military standards is of questionable utility to any audience, the review structure is becoming increasingly inadequate for effective management, and separate system definition contracts appear to be desirable whatever the type of system being considered. Promising areas for further research are identified. (JDL)

R-2567-AF
March 1980

Software Requirements for Embedded Computers: A Preliminary Report

Steven Glaseman, Malcolm Davis

A Project AIR FORCE report
prepared for the
United States Air Force

Rand
SANTA MONICA, CA. 90406

This research reported here was sponsored by the Directorate of Operational Requirements, Deputy Chief of Staff/Research, Development, and Acquisition, Hq USAF, under Contract F49620-77-C-0025.

Library of Congress Cataloging in Publication Data

Glaseman, S
Software requirements for embedded computers.

([Report] - The Rand Corporation ; R-2567-AF)
1. Aeronautics, Military--Automation. 2. Aeronautics, Military--Data processing. 3. Programming (Electronic computers) I. Davis, Malcolm, 1921- joint author. II. United States. Air Force. III. Title. IV. Series: Rand Corporation. Rand report ; R-2567-AF.
AS36.R3 R-2567 [UG1420] 081s [623'.028'5425]
ISBN 0-8330-0230-9 80-14948

The Rand Publications Series: The Report is the principal publication documenting and transmitting Rand's major research findings and final research results. The Rand Note reports other outputs of sponsored research for general distribution. Publications of The Rand Corporation do not necessarily reflect the opinions or policies of the sponsors of Rand research.

Published by The Rand Corporation

(12

R-2567-AF
March 1980

Software Requirements for Embedded Computers: A Preliminary Report

Steven Glaseman, Malcolm Davis

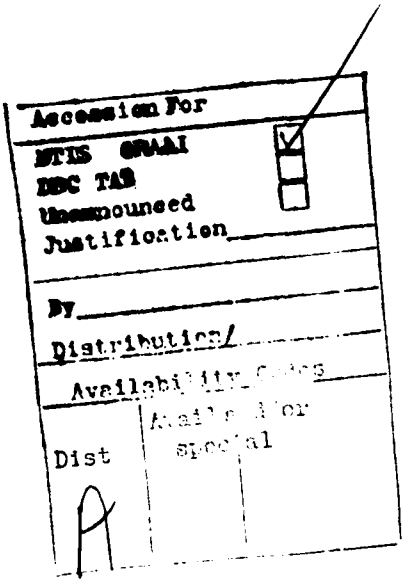
A Project AIR FORCE report
prepared for the
United States Air Force

Rand
SANTA MONICA, CA. 90406

PREFACE

Under the Project AIR FORCE study "Computer Resources Requirements Management," Rand is examining a number of problems associated with managing the acquisition and support of software for computers "embedded in" (i.e., an integral part of) major defense systems, such as aircraft or missile weapon systems. The management of software requirements for such computers is a major problem area that affects mission performance as well as cost and schedule variables. This report documents the initial results of work aimed at learning how the software acquisition process is influenced by existing techniques for generating and communicating software requirements.

Although preliminary, these results should be useful to Air Force and other agencies responsible for, or interested in, the application of software technology to specific mission areas.



SUMMARY

This report describes the results of an initial year-long study of current Air Force procedures for formulating and communicating software requirements and their effects on software acquisition for embedded computers.

An "embedded" computer is defined as an integral component of a larger defense system whose major functions go beyond data processing. Occasionally such computers are physically embedded in the systems they support (e.g., on-board computers in aircraft or missile weapon systems), but physical proximity is not necessary to the definition. Many command and control systems employ computers considered to be embedded in the functional sense, although they are physically separated from other system components.

The results presented here are based on an examination of eight major acquisition programs,[1] drawn primarily from the Air Force's Aeronautical Systems Division (ASD) and Electronics Systems Division (ESD), each containing a substantial embedded computer component. For each program, we interviewed personnel from the System Program Office (SPO) and from the prime contractor; wherever possible, these were people who were or had been directly associated with software acquisition.

Our initial hypothesis was that some of the performance, cost, and schedule problems often associated with embedded computers could be traced to specific inadequacies in the formulation and communication of software requirements. We believed that some problems could be eliminated, or their impacts lessened, by introducing more discipline into software requirements management.

It soon became clear that requirements management activities could not be studied out of context. Accordingly, we enlarged the scope of our study to encompass the embedded software development life cycle. We concluded early in the project that embedded software acquisition and support problems are driven not by requirements problems alone, but by several acquisition management anomalies that combine in various ways to cause those problems. Thus our focus shifted from requirements management to software development management.[2]

We describe, with the aid of two conceptual models, the observed software development activities of the Air Force and of the prime contractor. In view of the number of systems examined, the current models must be considered preliminary; however, they are useful in two ways. First, they present top-level flow diagrams of the software acquisition process. Each node represents an activity that can be separately expanded, and the resultant second-level diagrams provide a more detailed view of each activity, and of the overall process. Thus, when further refined, the models will serve as a framework for understanding the processes, resources, and costs associated with embedded software acquisition.

[1] F-4G (Wild Weasel) Update; F-15; F-16; B52G/H OAS Upgrade; TACC AUTO; COBRA DANE; COBRA JUDY; and one Navy program, the F/A-18. The authors are also familiar, from earlier work, with the World Wide Military Command and Control System (WWMCCS) and with the 427M program.

[2] Many of the terms used in this area are inadequately defined and inconsistently used, which results in needless confusion. To provide a firm basis for the discussion in this study and to encourage dialogue on the important problem of language, we suggest some definitions of commonly used terms in the body of this report.

Second, these models form the basis for a set of observations on current software acquisition practices. Two related assumptions appear to be implicit in these practices:

1. Software is infinitely flexible and need not be treated as an explicit variable during program-level decisionmaking.
2. The software development process can easily accommodate changes resulting from technological opportunity and the gradually increasing precision with which mission needs are understood.

However, the observations presented in this report lead us to conclude that these assumptions are unwarranted. Our observations, which are based not only on the models but also on in-depth discussions with program office and contractor personnel, and with colleagues both at Rand and elsewhere, are summarized below:

1. In the programs we studied, the Air Force gave relatively little explicit attention to software during preprogram decisionmaking. In general, it was not until the program office began preparing a development Request for Proposal (RFP) that explicit planning for software became evident.
2. We see a distinct change in the nature of software acquisition management problems now facing the Air Force. Program offices understand the importance of, and the requirements for, more effective software management. Problems of understanding are being replaced by problems of implementation. Schedule, budget, and resource pressures often combine to hinder the application of software management lessons learned over the last decade.
3. Program office management techniques are applied to embedded software applications with little regard for variations in the comparative maturity of different application areas. These differences bear on the concreteness of system and software requirements and on the experience of the contractors, and they are striking enough to suggest the need for tailored management techniques.

These general observations are both rooted in and exemplified by the following four specific observations:

1. The *type* of software expertise required by a program office has become at least as important as the *number* of people required, and existing formal Air Force training programs are not producing the most important types. Briefly, what is needed are individuals with sufficient training and experience to bring the most relevant software technologies to bear on those mission functions most amenable to them. This requires people with substantial knowledge in at least two areas, and the ability to see the best bridges between those areas.
2. Current techniques for producing software documentation are based on the assumption that the developer can satisfy not only his own information requirements, but those of the support and training organizations as well. This appears to be an unwarranted assumption. Our impression is that the Air Force is paying dearly for documentation that is of questionable utility to any audience.

3. The current review structure is growing increasingly inadequate for the timely and effective management of software development. In theory, a limited number of formal reviews augmented by informal but more frequent technical interchange meetings seems a reasonable approach. In practice, however, growing complexity and limited software resources argue for increased formality and for careful tailoring of reviews to specific development environments.
4. The use of a separate system definition (Phase-0) contract seems to pay dividends for software development regardless of the type of system under consideration. Without a separate effort, the major portion of system definition is done under the pressures of a full-scale development environment. This can lead to software design anomalies that, once discovered, result in expensive and complicated correction. These kinds of problems appeared less frequently in programs that used Phase-0 contracts.

Although more work is needed before strong recommendations can be made, our initial observations do raise specific issues that bear on acquisition management policy and indicate the following promising areas for further research:

1. *Information*

   - Identification of software-relevant information that is both available (from defense contractors, commercial R&D institutions, universities, and other sectors) and useful at various stages of the system acquisiton process.
   - Identification of the minimum essential information requirements of the various groups (e.g., development, support, training, operations) involved with embedded software applications.
   - Development of procedures for accessing such information, and for its effective application in various Air Force acquisition environments.

2. *Personnel*

   - Techniques for improved recognition of, and program office access to, operational personnel with particularly appropriate skills and experience in applying software technology to specific mission areas.
   - Development and evaluation of incentives for attracting skilled personnel to operational assignments that produce a doubly expert resource.

3. *Structure*

   - Development of an analytic framework for a priori identification of software applications that would benefit most from tailored software review and documentation procedures.
   - Development of appropriately tailored software review and documentation procedures, and of methods for implementing them in various Air Force acquisition environments.
   - Development of an improved environment for technical cooperation between individual program offices and the Air Force Avionics Laboratory with regard to embedded software R&D.

As this work moves forward, we expect a narrowing of focus and more detailed attention to issues that represent the greatest potential policy leverage for embedded software acquisition management.

ACKNOWLEDGMENTS

The authors wish to acknowledge the contributions of Robert Reinstedt and William Faught to the research reported here. Thanks are also due Stephen M. Drezner, head of Rand's Operations and Readiness Improvements Program, and to our colleagues R. Stockton Gaines, Lt. Col. Edward Puscher (USAF), Anthony Robinson, Hyman Shulman, Giles Smith, and Willis Ware, for periodic review of the progress of this work, and for suggestions that helped to keep it on track. Finally, thanks are due to Monti Callero and Kenneth Marks, whose reviews added to the substance and clarity of this report.

CONTENTS

ACRONYMS

AAPC	Armament and Avionics Planning Conference
AFR	Air Force Regulation
AFSC	Air Force Systems Command
ASD	Aeronautical Systems Division
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CPC	Computer Program Component
CPCI	Computer Program Configuration Item
CRISP	Computer Resources Integrated Support Plan
CRWG	Computer Resources Working Group
DAR	Data Automation Requirement
DCP	Decision Coordinating Paper
DSARC	Defense System Acquisition Review Council
ESD	Electronics Systems Division
FSD	Full-Scale Development
HOL	Higher-Order Language
MENS	Mission Element Needs Statement
MIL-STD	Military Standard
OMB	Office of Management and Budget
OSD	Office of the Secretary of Defense
PDR	Preliminary Design Review
PMD	Program Management Directive
PMP	Program Management Plan
RDT&E	Research, Development, Test and Engineering
RFP	Request for Proposal
ROC	Required Operational Capability
SAC	Strategic Air Command
SON	Statement of Operational Need
SOW	Statement of Work
SPO	System Program Office
TAC	Tactical Air Command

I. INTRODUCTION

This report describes work undertaken to help the Air Force more effectively manage software acquisition for computers embedded in defense systems. Such systems are growing rapidly both in number and in complexity as computer subsystems take on increasingly important roles and acquisition problems have increasingly severe impacts on mission performance.

An "embedded" computer is defined as an integral component of a larger defense system whose major functions go beyond data processing. Occasionally such computers are physically embedded in the systems they support (e.g., on-board computers in aircraft or missile weapon systems), but physical proximity is not necessary to the definition. Many command and control systems employ computers considered to be embedded in the functional sense, although they are physically separated from other system components.

Earlier studies have described a number of important software problems that accompany this growing committment to embedded computer systems [1,2,3]. A major issue has been the frequent failure of the first products of software development efforts to meet the user's operational needs. Cost growth and schedule slips also often share the spotlight with user dissatisfaction. Moreover, for large systems, dissatisfaction with initial software products contributes to a growing software maintenance problem. (A recently initiated Rand study is addressing this third issue.)

One of the most consistent study results has been the identification of software requirements management as a major problem area. It may be that as embedded software becomes more complex, current techniques for generating and specifying its requirements are becoming less adequate. This report examines these techniques and assesses their overall effects on system acquisition.

Another important issue is the careless use of language in this area. The term "requirement" has yet to be satisfactorily defined; it means different things to different people. Moreover, "requirement" is often substituted for "specification" or is combined with it to produce the equally ambiguous "requirements specification." Appending "software" to any of these terms does not clarify matters.

This is not a semantic issue. The difficulty reflects confusion not only about what requirements are, but also about their roles in the embedded software development process and, most importantly, about the management of that process. To provide a firm base for the discussion in this document, and to encourage more dialogue on this important problem, we suggest the following definitions:

Operational Need or Mission Need: The capability to perform one or more mission tasks or functions required to achieve mission or mission area objectives. The operational need is expressed in terms of task and functional capabilities (what must be done and how well), not in terms of specific hardware or software system characteristics.[3]

[3] Source: AFR 57-1, Attachment 1, 12 June 1979.

Requirements: A set of characteristics that, taken together, represent one approach to satisfying an operational need.[4] Characteristics are grouped according to function, performance, operation, test, safety, and other categories.

Software Requirements: That subset of requirements to be operationalized as one or more cooperating computer programs.

Specifications: A set of documents prepared to provide a standard record of requirements evolution and to support decisionmaking about the design, implementation, and testing of defense systems.[5]

Software Specifications: That documentation subset concerned with recording and supporting decisionmaking about system requirements satisfied via computer programming.

Our initial hypothesis was that some of the performance, cost, and schedule problems often associated with embedded computers could be traced to specific inadequacies in the formulation and communication of software requirements. We believed that some problems could be eliminated, or their impacts lessened, by introducing more discipline into software requirements management.

The most frequently cited requirements problem is the need to respond to late-appearing changes in the requirements baseline. That baseline is established very early and usually represents a preliminary and incomplete view of user needs. All parties to a major acquisition expect changes to the baseline as more is learned about those needs. Changes are also driven by evolving threat scenarios and by technical advances. One of the primary tasks of program management is to ensure that such changes are accurately reflected in the requirements that the contractor must satisfy. Since changes occur throughout the software development process, every step along the way can involve assessing and responding to new or changed requirements. Thus, requirements management is at the heart of and inextricably tied to the entire software development process.

In its management of this process, the Air Force employs techniques that evolved during a time when hardware was the principal element of concern. There is a general intuition that these techniques are hardware-biased and therefore ill-adapted to the task of managing software development. Without taking sides, we note the absence of empirical evidence for this claim—evidence based, for example, on an examination of important differences between hardware and software development.

These considerations, then, led us to question the relevance of a requirements study that ignored the broader context of acquisition management. We concluded that the symptoms usually attributed to requirements problems per se reflect instead more general problems with the application of available management tools to the software development process. Thus our focus shifted from requirements management to software development management, and project objectives were broadened to include additional factors and to trace their impacts on the development process.

[4] Any approach incorporates compromises driven by constraints on available technologies, budget, and other resources. Thus, a given set of requirements frequently satisfies a constrained version of the original mission need.

[5] Mission needs, and therefore requirements, typically change during the acquisition period. Since contracts are based on specifications, one of a System Program Office's (SPO's) most important tasks is to ensure that specifications accurately reflect such changes. In the extreme, it is possible for a contractor to satisfy all specified requirements and receive full compensation for a delivered system that satisfies no mission needs.

THE PRESENT STUDY

We have limited our attention to software development activities that precede actual code development. It is important to note the difficulty of identifying the starting point of this process. Development programs emerge gradually, at first, from ongoing background activities carried on by both the Air Force and contractor communities. These early activities have some formative influence on future software development in that they result in an initial set of system-level requirements. However, recent software requirements research has been characterized by a lack of attention to this situation. Most efforts have focused on systems that manipulate requirements which are already in the form of specifications and have ignored that part of the process concerned with initial formulation [4-8]. Therefore, an important subgoal of this study is to describe the activities that characterize this crucial early period and to examine their influence on later events.

Our approach has been to examine the early software development activities of several recent acquisition programs, primarily from the Air Force's Aeronautical Systems Division (ASD) and Electronic Systems Division (ESD). We selected five aircraft weapon systems and three command and control systems as appropriate subjects.[6]

We focused our attention on operational avionics software for the aircraft programs, and on mission software for the command and control programs. This constraint reflects an arbitrary narrowing of scope and implies no ranking of the relative importance of mission versus other software.

For each program, our major concern was to examine the software development process in the light of requirements decisionmaking. The processes for all programs were then compared with one another to highlight similarities and differences and to explore the utility of a single aggregated conceptual model of the decisionmaking process.

It is important to note that system and program management information was made available to us on the condition that we would report only aggregate results. In particular, we agreed not to discuss the details of any single program or to encourage direct or implicit associations of specific programs with any issues we might discuss. Thus our results are presented here without the system-specific information upon which they are based. That information can be made available subject to a determination that disclosure would not violate our agreement.

Beyond developing models of the processes we observed, we have made some preliminary observations based on our understanding of these processes and have identified a number of issues we hope to address in future work.

Perhaps the most important observation is that software, despite its mission-critical nature, is not routinely treated as a major decision variable from the earliest stages of the acquisition process. In the remainder of this Introduction, we discuss this problem in some detail, because of its importance and because it sets the stage for the material that follows.

[6] F-4G (Wild Weasel) Update; F-15; F-16; B52G/H OAS Upgrade; TACC AUTO; COBRA DANE; COBRA JUDY; and one Navy program, the F/A-18. The authors are also familiar, from earlier work, with the World Wide Military Command and Control System (WWMCCS) and with the 427M program.

SOFTWARE AS A DECISION VARIABLE

As the Air Force goes through the process leading to an acquisition decision, R&D activities within the contractor community intensify and focus on areas related to perceived Air Force interests. These separate and parallel activities are coordinated by frequent communications between the two communities. Although the discussions often address technical issues, software, apparently because it is viewed as infinitely flexible, is rarely a subject of detailed concern.

The Air Force begins to focus explicit attention on embedded software at about the time a SPO is formed. By that time many important decisions about software function, performance, and design have been made implicitly, as the indirect consequences of earlier hardware-oriented decisions. This is not a surprising observation, considering the significant methodological and experiential hardware bias accumulated over the last three decades of defense system acquisition. Without implying that hardware acquisition is a solved problem, we note that there is nowhere near the same tool kit available for embedded software.

Software has only recently—but very rapidly—emerged as the other half of the hardware-software dyad that now characterizes weapon systems. At first, software was treated as a low-level development detail, to be provided by the contractor whenever necessary. The gradual introduction of software technology led to the belief that there was nothing very special about software acquisition management techniques, so software decisions were driven at first by a priori hardware decisions.

Today, however, we know that embedded software is pivotal to the operational utility of entire weapon systems. As such, its conceptual, design, and developmental aspects must be seen as decision variables, influencing the fundamental nature of entire systems. Existing Air Force acquisition management techniques, although they are improving, do not yet reflect the importance of embedded software. While much attention is now given to software development, many software decisions are still made indirectly. One effect is a reduction of the software design and implementation flexibility available during full-scale development.

But software flexibility is of critical importance as the pace of technology increases and as embedded computers are used to support more complex mission needs. Cost alone dictates that systems be designed for extended operational life. Over this period system requirements *will* change and technology *will* offer new opportunities. Without explicit attention at the earliest decisionmaking stages to a software architecture that can accommodate the user's evolving operational environment, his needs will simply not be met. Systems that require extensive and continuing software "maintenance" from their date of delivery are a costly consequence of ignoring the need for early software planning.

A related implication, yet to be validated, is that some of the basic hardware/software tradeoff issues may be decided in the absence of software expertise. If this is true, it could easily lead to poorly informed decisions about system design alternatives that are absolutely fundamental to a future system's operational utility, responsiveness to change, and maintainability. This could happen not only because of failure to appreciate the full range of software's technical influence on a system, but also through the effects of uncontrollable organizational and political variables. The latter are particularly relevant to major acquisitions like those we examined.

These considerations led us to develop separate models for program office and contractor activities. We wanted to understand how each group dealt with software questions during early acquisition decisionmaking. Individually, the models highlight procedural differences between the two groups; taken together, they embody the requirements management process for embedded software during the period we studied.

We found both processes to be heavily influenced by the concept of requirements, yet neither model contains an identifiable requirements phase. It appears that in practice the influence of requirements not only pervades all phases of software development, it begins much earlier than previously thought. It is useful to visualize a continuum, along which requirements are both represented and managed at various levels of detail.

This view suggests that many of the basic decisions made prior to establishing a SPO, and prior to initial Request for Proposal (RFP) development, have important consequences for software requirements. For example, references to system size, weight, and power can determine at least the broad outlines of computational subsystems. They represent constraints on important parameters, and therefore on the technologies to be considered for the computer hardware; hence they may limit available alternatives for both hardware and software architectures.

Some fair sense of functional requirements—at the major subsystem level—can be said to exist prior to formal program initiation. The outlines of embedded software requirements take shape concurrently with those for other major subsystems; they are refined along with, not extracted from, an evolving system specification.

If software directions are in fact set far in advance of formal program initiation, the deferring of explicit management attention to software until preliminary Part I software specifications are in preparation—a frequent procedure, based on our observations—is clearly inappropriate.

Existing management guidance, primarily AFR 800-14 [9], establishes the major subsystem aspect of embedded software and gives overt recognition to the need for applying software expertise at the Program Management Directive (PMD) level. Nevertheless, measurable resources are apparently first devoted to software issues as a function of ensuring that the development RFP embodies the major provisions of AFR 800-14. Such deferral is not observed in the various hardware areas, where considerable technical information is available and under consideration long before the development RFP is prepared.

Whatever the reasons for this different treatment, the impact is that embedded software is not effectively represented during system concept formulation. Thus, many critical software issues are decided by implication, as a by-product of attention to other aspects of the emerging program. By the time expert attention is focused on software, much of the flexibility correctly associated with it has been lost because of constraints imposed earlier. This observation is confirmed by the often disproportionate cost and schedule impacts of seemingly minor development-phase software changes. More flexibility is assumed than is actually available during the development cycle, and even changes attempted "early" in that cycle often have surprisingly complex ramifications. An obvious conclusion is that the Air Force must get into the software business earlier. This is not a new suggestion, but we suspect that it has been difficult to implement because of incomplete understanding of the situation.

Before discussing these ideas in greater detail, we shall describe our preliminary models of the development process in Sec. II. Section III discusses our observations on resources, documentation, reviews, and contracting as they apply to embedded software acquisition management. Section IV then presents our current views of the policy issues that bear on those observations and indicates the work still required to develop and justify specific recommendations.

II. THE SOFTWARE DEVELOPMENT PROCESS

We shall describe the software development process in terms of two conceptual models, one for the activities undertaken by the Air Force and one for those undertaken by a prime contractor. The models are based on eight acquisition programs, each containing a substantial embedded computer component, all begun at different times and each now in different stages of maturity.

Before we introduce the models, we must state several caveats. First, the models represent an aggregate of the key activities undertaken by the program offices and the contractors we interviewed. The level of detail and representation were chosen to enable an easy understanding of how these activities move through time, and how they influence each other and the product. Moreover, the models present observed practice and incorporate lessons learned from past experience. Differences among the individual programs relate primarily to resources applied, not to procedural or operational matters.

Second, only a limited period of time is represented. Software life cycle activities beyond the critical design reviews are not included in the current models.

Third, flow diagrams suggest a discreteness that is not observed in the real world, where activities overlap enough to make their boundaries hard to discern and not all activities are equally important. The current models do not reflect these facts.

Finally, no attempt is made here to identify or comment on individual or combined problem areas. Our observations on these matters are deferred until the next section.

PROCESS DESCRIPTIONS

The Air Force

The process shown in Fig. 1 (p. 29) is an aggregation of observed Air Force behavior across the programs we examined. It represents the top-level decision-making activities common to most of the programs, from the earliest conception of a program up to the Critical Design Review (CDR) that precedes computer-program coding release.

Pre-RFP Period. The need for and outline of a new defense system emerge gradually from a background of continuous mission-area analyses, planning studies, and R&D efforts by both military and industrial organizations. Sometimes the surfacing of a new technological capability motivates interest in its mission-enhancing potential; at other times a known and worsening functional shortfall eventually generates enough concern to support informal but focused examination of solution alternatives. As Air Force interest grows, a corresponding growth in interest is seen in a larger community, including the relevant industrial contractors. Over time, possible solutions are informally discussed among many elements of the Air Force, and between the Air Force and the industrial community.

Eventually, ad hoc study teams are formed from many elements of the Air Force, including operational, system planning, and management personnel from the Air Staff, major operational commands, AFSC, and AFLC. It is significant that hardware engineering skills predominate in the backgrounds of most of the personnel involved. Early team formations often fail to include adequate software expertise, even in programs containing a substantial embedded computer component. And, although both formal and informal relations are maintained with the industrial sector, there appears to be little tendency to tap its software resources during these early system-level studies.

The initiation of formal acquisition proceedings usually comes from an operational command—TAC, SAC, etc.—in the form of a document describing and justifying the need. Most of the programs we observed predate OMB circular A-109 [10] and its implementing Air Force Regulation 800-2 and were formally initiated via the submission of a Required Operational Capability (ROC) [11] or, in the case of systems procured under the 300-series Air Force regulations, a Data Automation Requirement (DAR) [12]. Under the then-existing guidance, the ROC or DAR often suggested ways to achieve the required operational capability and included descriptions of specific items of equipment and software. The new guidance represents a substantial departure in that the ROC has been replaced by the Statement of Operational Need (SON) [13], which must explain the need in mission or capability terms, explicitly avoiding discussion of specific solutions. Among other objectives, this change was intended to encourage innovation and exploration of a wider set of solution alternatives than had become customary.

The next formal step is a review conducted by Hq USAF and coordinated with all relevant operating commands. If a proposed solution involves a major acquisition,[7] approval is needed from the Office of the Secretary of Defense (OSD) before the program can be formally continued. This approval is Milestone 0 in the Defense System Acquisition Review Council (DSARC) process.

Many non-technical factors affect review outcomes and influence subsequent program guidance: international trade agreements, treaties, political pressures, media pressures, lobbies, economic factors, defense posture, and national policy, among others. The review must also reconcile any proposed acquisition with other DoD capabilities, resources, and priorities; consequently, some cost estimations and tradeoff analyses must be available early in the process. The output of this stage is the granting of authority for Hq USAF/RD to prepare a PMD specifying how AFSC should proceed with the acquisition program. Prior to the publication of OMB-A-109, a PMD usually directed the establishment of a SPO or pre-SPO with authorization to prepare a plan for system development. Under the new guidance, the SPO or pre-SPO is required to investigate and evaluate alternative system concepts that might satisfy the approved need, and to submit the findings of this investigation to the DoD for approval prior to proceeding with the system development plan.

It is not uncommon at this stage to establish a steering group to guide the evaluation process. The new guidance requires that every resource available (laboratories, industry, universities, etc.) be employed in developing a comprehensive picture for the DoD review. Contracts may be let to relevant contractors to

[7] A major system acquisition is considered to be one with an anticipated cost of $75 million in RDT&E or $300 million in production [14].

assist the Air Force in evaluating alternative system concepts. In fact, informal cooperation between the Air Force and these contractors is common at some level throughout the process thus far described. The outputs of this stage are recommendations by the SPO that can be formulated into a Decision Coordinating Paper (DCP) for DoD approval at Milestone I of the DSARC process.

After approval, the SPO's major tasks are to prepare a Program Management Plan (PMP) and to develop an initial system specification. How well such tasks are done depends on a number of things, including (1) how well-defined the problem areas are, (2) the kinds of resources available to the SPO, (3) the guidance received from the steering group, (4) the experience of the contractor community, and (5) schedule and budget pressures. Initial system-level studies provide some insights for the allocation of system requirements to software, and these are now included in the documentation prepared by the SPO as it prepares for the DSARC II review. However, even though AFR 800-14 suggests consideration of a large number of software-related items, in most of the programs we investigated there was no evidence that SPO attention was focused on substantive software issues at this point.

RFP to FSD Award. An important decision at this stage is whether to write the initial RFP for full-scale development (FSD) or to contract separately for system definition only (Phase 0), and use the Phase-0 output to drive a subsequent FSD decision. Phase-0 contracts may be awarded to more than one competitor so that the Air Force can consider the merits of alternative approaches.

The remainder of this section deals with the procedures for awarding an FSD contract.

The RFP elements most relevant to software are the Statement of Work (SOW) and the Contract Data Requirements List (CDRL). Although general guidance for their preparation is found in AFR 800-14, knowledge of the software development process is needed to translate this guidance into a good development RFP. For the programs we examined, the software portions of both RFP and contract were contained in relatively few pages and dealt with the use of a higher-order programming language (HOL) whenever applicable, the use of top-down structured design and programming concepts, early attention to interface requirements, and other very general aspects of software development.

Proposal evaluation is a formal effort supported by analyses of the technological and economic tradeoffs among competing proposals. Other inputs to the process include Congressional influence and local, national, and international political and economic considerations. It is noteworthy that in the programs we investigated, the contractor's past experience or demonstrated capability in computer or software technology was not a significant factor in the selection process.

The next major step for the Air Force is to negotiate the FSD contract. Although we have not examined this activity in detail, it seems obvious that better contracts will be struck where knowledge is more complete. This is particularly true for embedded software, since it is the newest and least well-understood component of system acquisition. In terms of maturity of ideas and utility of documentation, those programs that contracted separately for system definition appeared to have more complete knowledge going into contract negotiations than did other programs.

FSD Award to Critical Design Review. After the development contract is signed, the SPO enters a phase aimed at monitoring the progress of the contractor

and participating in various aspects of problem resolution, test planning, etc. The performance of this job often requires the formation of special monitoring groups, each responsible for certain aspects of the development process. One specialized group, the Computer Resources Working Group (CRWG), interacts with contractors to coordinate computer subsystem development activities, to assure compliance with AFR 800-14, and to develop and maintain the Computer Resources Integrated Support Plan (CRISP). This is often the first appearance of significant software expertise in the Air Force process. Most of the programs we examined had a CRWG; very few had formed any other group that dealt exclusively or even primarily with software development management.

Since the initial system specification does not delineate every requirement in detail, constant interaction is needed between the contractor, the SPO, and the user, especially in the early phases of development, to resolve functional issues so that the contractor can develop and evaluate increasingly detailed software specifications.

The contractor's initial approach to software development—the Computer Program Configuration Item (CPCI)-level design—is usually reviewed along with other draft documentation presented at the Preliminary Design Review (PDR). Although the PDR is required by MIL-STD 1521, the form it takes and its value vary greatly from program to program, depending on the software expertise available and on budget and schedule pressures. The PDR is an opportunity for the Air Force to detect any misunderstanding in the contractor's interpretation of the system-level specification and to reevaluate his ability to produce a satisfactory final product. The review is intended to identify any discrepancies before detailed software design begins.

Once all critical software issues raised in the PDR are resolved, the software development specifications are finalized and the contractor can officially proceed with the detailed software design. The results of this process are documented in increasingly explicit software product specifications. In Volume I of the product specification, each CPCI is broken down into a number of computer program components (CPC). The contractor specifies the exact details of each group of programs that comprise a CPC so that coding can proceed. However, the SPO must first assemble the talent needed to conduct a software CDR and to approve or take issue with the contractor's designs. As with a PDR, the value of the CDR depends on the software resources available to the SPO. Moreover, to the extent that more sophisticated expertise is needed to adequately review the technical detail available at the CDR, software resources are more critical at this stage than at the earlier PDR.

Once the SPO and the user approve the detailed design represented in the draft software product specifications, the contractor can proceed with code development and testing.

The Prime Contractor

The Pre-RFP Period. The pre-RFP process is shown in Fig. 2 (p.31). Most of the contractors we interviewed maintain a group whose primary task is to pursue technological advances that are applicable to one or more of the contractor's interest areas. These groups are permanent entities and are staffed by personnel highly qualified in most of the appropriate disciplines; an increasing number are formally trained in computer science. In the main, these are R&D groups, concerned with

technical feasibility and supported by laboratory facilities. Their tasks include providing technical information to support various positions taken by management in the contractor's ongoing dialogue with the Air Force.

When this dialogue makes it sufficiently clear that the Air Force is moving toward an acquisition decision, appropriate members of the advanced technology group meet with technical management to consider the broad outlines of the anticipated program. Their objective is to develop technical information for input to a decision on corporate participation should a development RFP be issued. Attention centers on system-level operational and structural characteristics such as weight, power, and complexity, and on developing rough estimates of the costs involved.

Discussions started in these meetings continue in small teams that break out of the original group. The teams focus special attention on major subsystems. For example, an aircraft contractor develops teams to examine airframe, propulsion, and avionics subsystems. For command and control systems, teams might form for communications, data processing, and display subsystems. Each team attempts to further define its subsystem by considering the latest technologies, developing and experimenting with models, and making initial contacts with potential subcontractors. In this way broad initial software designs are developed and evaluated, known constraints are assessed for their impacts, and information is gathered for use in allocating subsystem functions to hardware and software.

Attention to software is limited, at first, to these allocative issues. Gradually, as past experience and new information are factored in, alternative designs for each major software component are reviewed and coordinated among teams, and consideration shifts to issues of programming language, program size, interface characteristics, and specific functionality. Each component is also associated with a list of functional requirements as they emerge from the continuous tradeoff analyses that characterize this period. Thus, software requirements at the subsystem level are available in some form quite early.

The teams come together often during this period to summarize and coordinate their individual activities. This process usually identifies problem areas and raises new issues, some of which are returned to the teams for resolution. The cycling between team activity and joint review continues, in some cases over one or more years; it runs parallel to and shares information with the Air Force's efforts to come to an acquisition decision. In some cases coordination is formalized by technology-demonstration or prototype development contracts.

Thus, when a development RFP becomes available, the contractor is usually prepared to respond in considerable detail. For software, informal team documentation is, at some point, collected, reviewed for consistency, and incorporated into what looks very much like a set of draft software development specifications. These typically identify a programming language, require the use of top-down techniques, and present a philosophy of programming dealing with modularity, data use, etc. They can also include early lists of functional requirements at the major subsystem level, interface requirements for functions within a given subsystem, subsystem interface diagrams, and even top-level flowcharts for some functions. In some cases, early drafts of a computer program development plan, interface control documents, and preliminary software product specifications are also available.

RFP to FSD Contract. The initial RFP can be for a Phase-0 contract, or for an FSD contract. If, as is normally the case, an FSD contract is anticipated, the

contractor typically forms a proposal-writing group. The technical part of the proposal is then written by small teams, each addressing a specific functional area, such as the target acquisition task for a command and control radar. These teams are drawn from the contractor's preliminary system definition efforts (described above) and are supported by whatever level of analysis has been achieved in each area.

Each team produces what might be called an area specification. These are eventually reviewed for technical accuracy and consistency and for their responsiveness to all requirements described in the preliminary system specification. The resultant volumes are then submitted to the Air Force as part of a formal proposal.

If a Phase-0 contract precedes the award of a development contract, a different path is followed. In this case the system definition efforts already under way are expanded; more people become involved and additional teams are formed to deal in greater detail with the separate functional areas of each subsystem. Thus, for avionics, the software design for fire-control processing is elaborated by subfunction. For example, air-to-air combat functional software is considered in terms of separate algorithms for the air-to-air gunnery and air-to-air missile subfunctions.

Each specialized group carries out some mix of technical assessment, modeling and experimentation, subcontractor interaction, and documentation. In this way subsystem design alternatives are refined, evaluated, and formalized as CPCIs. For software, design focus moves gradually from the CPCI level to the CPC level. In a command and control radar, for example, once the general outlines of the target acquisition CPCI are agreed upon, detailed consideration is given to software for the detection, recognition, and identification CPCs that define it. As this process continues, the work of each team is subject to periodic reviews by other teams, and by the contractor's technical management. Historically, the Air Force has at least maintained visibility, if not participation, in these efforts.

Phase-0 contract deliverables typically include the same kinds of software documentation described earlier. However, their contents reflect the additional analyses carried out over the time frame of the Phase-0 contract; for example, a Phase-0 contract results in more complete and more detailed drafts of Volume I software product specifications.

At this point, the Air Force evaluates the performance of each Phase-0 competitor and either directly awards the FSD contract or issues an RFP. In the cases we examined, FSD contracts were awarded directly on the basis of Phase-0 results.

FSD Contract to CDR. When the FSD award is preceded by and based on a Phase-0 contract, the winning contractor usually knows enough about the software design to begin immediate preparations for a PDR. If no Phase-0 contract was used, the contractor must first expand both his personnel resources and his analyses—in essence acquiring, early in the FSD contract, the results of a Phase-0 contract.

Preparations for the software PDR include internal reviews of functional and module interface designs, preliminary test plans, and the contractor's current strategies for software development. Available documentation is finalized and may be submitted to the Air Force for review prior to the formal PDR. Included are preliminary CPCI development specifications, draft interface specifications, draft CPCI product specifications, the computer program development plan, and other documentation. Senior technical and administrative managers may conduct system-level design walk-throughs; for software, this implies reviewing plans for eventual functional integration of individual CPCIs. The contractor may also hold PDR

rehearsals, conducting dry runs of the various briefings to be presented to the Air Force review team.

Conceivably, a contractor may fail to satisfy the review team. In that case, he must resolve the problems leading to the failure and schedule another PDR. None of the cases we examined had this problem; the issues raised were less serious and were resolved and re-reviewed by the Air Force at frequent technical-interchange meetings between the PDR and the CDR.

In theory, detailed software design follows PDR approval of the overall software design contained in the development specifications. Approval is based on satisfying the Air Force that the overall functions, performance, interfaces, and structure envisioned for each CPCI are compatible with one another and are traceable to approved system-level requirements. Each CPCI is then broken down into its component software modules, and detailed design proceeds. In practice, detailed software design begins much earlier, before the PDR, when, for example, the avionics teams begin to specialize in specific areas, e.g., navigation or radar.

The emerging design details of each separately programmable CPC are documented in the software product specification. The contents of this document are of primary concern at the CDR. Therefore, as the CPC designs are refined and documented, the contractor begins to prepare for that review.

Problems discovered during CDR are resolved by the design teams for the affected CPCs, with the Air Force monitoring through technical-exchange meetings. If no significant problems are found with the detailed software design, with its traceability to the approved CPCI development specifications, or with the available software product specifications, the contractor begins program development.

MODEL UTILITY

Embedded computers are being used in increasingly specialized roles, and new applications appear frequently. There is a growing need to augment general management guidelines with specific techniques tailored to individual application areas. One such area is avionics, where the Air Force has been working to develop a detailed software investment strategy.

An important component of this effort is the development and use of enhanced modeling capabilities. The 2nd USAF Avionics Planning Conference recognized this by producing a Software Modeling Roadmap [15], which calls attention to the need for detailed modeling of the processes, resources, and costs associated with avionics software acquisition and support.

The models presented here are an initial contribution to the goals of Path III of the modeling roadmap discussed on p. 4-247 of Ref. 15. They are designed as top-level flow diagrams for the software acquisition process. Each node represents an activity that can be separately expanded; the resultant set of second-level diagrams presents a more detailed view of the overall process. This technique is a standard systems engineering tool and rarely proceeds beyond two levels before resource bottlenecks and cost drivers become apparent.[8]

[8] The Software Cost Estimating Working Group at ESD (ESD/SCEWG) is funding a study by the MITRE Corporation, one objective of which is the development of a model of the software acquisition process. That model is similar in concept to those presented here, but it continues the tradition of combining government and contractor activities in one model. This effort is described in more detail in Ref. 16.

A top-level model has the advantage of being able to serve as the root for several different expansions, each one tailored to a specific area, such as avionics or command and control. The resultant models provide a framework for understanding the processes, resources, and costs uniquely associated with the selected area.

III. OBSERVATIONS ON THE PROCESS

This section presents a number of observations based on the models just introduced and on discussions with program office and contractor personnel and with colleagues at Rand and elsewhere. The first three observations are broader than the rest in the sense that they seem to have roots in several more specific areas. Although more work is necessary to analyze and refine them, we believe that these issues must be understood and dealt with before improvements can be made in the management of embedded software acquisition.

GENERAL OBSERVATIONS

Initiation of Concern for Software

Direct comparison of Figs. 1 and 2 shows that the contractor community begins to deal with software questions earlier than does the Air Force; this trend was confirmed in the programs we examined. As shown in Fig. 2, a contractor's decision to participate early in a budding acquisition focuses certain resources on software development alternatives for that particular program, as well as keeping him generally up to date on the latest advances in software technology.

The Air Force seems to place comparatively less emphasis on software during this period. Their attention is focused on system-level alternatives that might reduce operational shortfalls and, in particular, on the gross functional and performance characteristics of each alternative. Questions dealing with size, weight, speed, and overall cost dominate these initial discussions.

The centrality of software and its role as a major cost driver is well represented in the area of guidance. Air Force Regulation 800-14 encourages explicit concern for software, starting with the contents of the PMD. But observed behavior seems to reflect an assumption that the major characteristic of software is its extreme flexibility. And since flexibility is seen as relevant primarily to development decisions, most of the planning for software management is usually deferred until a development decision has been made. By then, however, earlier decisions have made it difficult to deviate from the existing rather detailed conception of software's roles.

But changes are always necessary during FSD, for several reasons, including the following:

1. Requirements change as understanding of operational or mission needs evolves.
2. New requirements are made feasible or attractive by technological advance.
3. Changes are required to correct unanticipated problems that result from prematurely embracing seemingly broad system outlines, which, in fact, have quite explicit consequences for software design.

Besides fostering its own specific technical problems, the third item complicates responses to the first two. Some variables, like software modularity, interface design, and, in some cases, even processor selection, are preordained or otherwise constrained before development-time design analyses are accomplished. In later responses to new or changed requirements, developers tend to leave these decisions intact, fearing the impact of changes on budget and schedule. Instead, the changes are made at lower levels, usually by complicating the software. As is well known, the more complicated the software, the less reliable it is and the harder it is to test and maintain. Thus, budget and schedule are affected precisely because of poorly considered efforts to avoid such effects. To the extent that the chosen actions fail to respond adequately to new requirements, the system's value to the user is adversely affected.

The most reasonable explanation for the casual treatment of software prior to issuing a development RFP is that some of the people involved at that level do not yet fully appreciate the increased complexity software implies for the decisions they make. Ten years ago this complexity was understood by very few in the Air Force. Today we observe it as generally understood at the program office level, and as making inroads at higher levels. Thus there is gradual progress, but it remains true that at the level where the earliest and most fundamental program decisions are made, software gets short shrift, and this sets the stage for later problems.

The solution is not necessarily to speed up the process by which software knowledge percolates through the Air Force; the process is not well understood in the first place, and we probably couldn't change it if we did understand it. However, it might be possible to inject appropriate expertise when and where it is most needed. (We will say more about that expertise in Sec. IV.)

Changes in the Nature of the Problems

While the Air Force has significantly improved its understanding of software acquisition management, the limited improvements in program outcomes over the same period are testimony to the difference between understanding a thing and doing something about it. Long-standing resource problems, the "start from scratch" character of most program offices, and the limited transfer of experience from one program to another are all major impediments to lasting improvements. As a result, the Air Force cannot yet fully apply to individual cases what it has learned collectively about effective software acquisition management.

In the late 1960s and early 1970s the Air Force treated embedded software as a given. Program-level decisionmaking revolved around system hardware, and it was the contractor's job to deliver hardware that worked. The fact that development of useful hardware was beginning to involve conceptualizing, implementing, and testing computer programs was of little interest until something went wrong enough to affect costs and schedule milestones. The Air Force's acquisition management activities changed very little at the time embedded computers were first added to the inventory.

Some of the programs we examined were initiated during this early period. The software problems encountered in those programs support the above characterization. For example, the fact that software could—indeed, must—be engineered was not initially understood by the Air Force or by its contractors. Because of this, software submitted to formal system testing often wouldn't fit the mission comput-

er, or couldn't be tested, or failed to interface appropriately with other systems, or, when integrated, formed entities too complex or expensive to support.

Later acquisition programs give clear evidence that the need for software engineering is now understood and that when the right resources are available, it can be effectively applied. Resources are critical to most of the software problems encountered by these more recent programs. For example, funding cuts (or increases, for that matter) can induce schedule pressures that work against good engineering practice.

Thus, we see a distinct change in the nature of problems having to do with managing software acquisition. The program offices we contacted now seem to have a good grasp of the importance of, and the requirements for, more effective management. But resource problems—primarily attracting, training, and keeping talented people—have so far diluted the benefits available from improved understanding. This is not a new problem; it is an extension of long-standing resource problems into a relatively new area. The nature and magnitude of this extension have yet to be defined, and they deserve focused attention.

There is some evidence that improvements might be possible without waiting for Air Force-wide solutions to these general resource problems. Specifically, a "think small" approach might provide some near-term benefits. For example, software problems on a number of large programs have been quickly and effectively turned around by the efforts of single individuals. What are the characteristics of such people, and how can their skills be accessed? These questions, and the other problems mentioned above, are discussed in more detail below.

Appreciation of Functional Area Differences

We were struck by the sameness of the Air Force's software management approach in the two functional areas we examined.[9] Program office activities are based on guidance, formal reviews, informal discussions and reviews, and documentation requirements. These tools are applied to software development management in ways that do not account well for important differences among functional areas.

The current approach can be generally characterized as a partnership between a program office and a prime contractor, wherein the latter has the major responsibility for design and implementation and the program office monitors progress and tries to assure tracking of system requirements with evolving mission needs. Both share responsibility for testing and for identifying and resolving problems.

Currently, this approach appears to work better for avionics software than it does for some kinds of command and control software. We can identify two areas where differences between these applications suggest the need for corresponding differences in management approach. The areas are briefly described below; we have not analyzed them extensively.

1. The application of computer technology to avionics tasks is reasonably well understood. Most avionics functions are conceptually well defined, and software problems are rarely based on a failure to understand the tasks at hand. Rather, many of the problems in this area now deal with controlling its rapid growth by the development and use of standards, and with Air Force-contractor management coordination issues [18,19].

[9] This problem also appears in somewhat different form in Ref. 17.

By contrast, we don't know a great deal about the application of computers to some command and control tasks. In the first place, the term encompasses a wide range of system types, some well understood, others not understood at all. The software problems encountered by the command and control systems in this study, and in others with which we are familiar, often reflect this lack of understanding. It appears as a lack of concreteness in software requirements: Programs get into trouble when progress comes to depend on developing software to support functions that have never been clearly defined. Ultimately, the press for completion that is characteristic of acquisition programs can result in software that fails to meet the user's overall expectations and that is so badly engineered as to severely limit the system's future responsiveness to change.

2. The aircraft industry is dominated by several large contractors, each of whom has experience in producing well-defined avionics products. Typically, a major part of the software used in avionics is produced by these prime contractors, who also monitor the production and integration of software embedded in subcontracted items. The prime contractor's product orientation and his continuity of experience ease the task of coordinating Air Force and contractor management techniques and encourage a certain degree of confidence in his ability to perform well, even with limited technical guidance by the program office.

The same product orientation and continuity of experience are not yet characteristic of software development for command and control. Even where there is product orientation, as with the two systems examined in this study, the entire software package is often developed by a software subcontractor who has little knowledge or experience in the overall application area.

The situation is worse when the system and its software are fragmented into major subsystems, each the responsibility of a different associate prime contractor. Although one contractor is usually responsible for the integration task, none may have a total product orientation or much experience with developing pieces of command and control systems, for example, let alone complete systems. Especially in the early stages of such programs, the burden is on the Air Force program office to determine that appropriate and coordinated progress is being made. Here, however, there is little basis for trusting to contractor experience, or to the contractor's grasp of the product as a whole, to offset any program office technical limitations.

To summarize, the important differences between the two acquisition environments we studied can be categorized according to the following topic areas:

1. The overall understanding of the tasks.
2. The perception of a coherent product by the prime contractor.
3. The experience of the contractor.
4. The concreteness of requirements.

We argue that differences in these areas are fundamental and that they suggest a need for correspondingly different management approaches. We must add that this does not imply an infinite number and variety of management approaches. Developing an overall software management framework that can be tailored to a variety of undertakings seems a reasonable topic for further research.

SPECIFIC OBSERVATIONS

Program Office Software Expertise

It appears to be common knowledge that program offices would do a better job of software development management if they could attract more and better-trained "software people." However, the attributes implied by the latter term are seldom made clear, and specific training requirements are not well defined.

Our observation is that the *kind* of software expertise required has become at least as important as the *number* of people required, and that existing formal Air Force training programs are not producing the most needed kinds of software expertise. A predominantly real-time, interactive avionics suite is very unlike a base payroll system in either development or operation. The software talents required in one area contribute little to success in the other.

We mentioned earlier that some of the programs we examined avoided major software problems largely by the efforts of certain individuals. In these cases, the training, experience, and initiative of such individuals were dominant factors in the success of the programs. The particular attributes of these key people need to be examined in greater detail, but we can provide a general characterization. Such an individual has at least an undergraduate degree in computer science, EE, or mathematics (the latter two being substitutes for the first before it became widely available), has spent some time as a programmer or systems analyst, and remains interested and current in at least one subdiscipline of computer science. In addition, he or she has substantial experience in the application of computer technology to at least one mission area, such as phased-array command and control radars.

What we have described is a person who is expert in at least two areas: a technical discipline and a functional application. This is exactly the kind of expertise that is needed at the pre-Milestone-0 stage represented on the Air Force model in Fig. 1. As computer technology provides the Air Force with more opportunities to enhance mission performance, the ability to discern how best to apply technological tools to mission requirements is becoming increasingly important. The Air Force must find a way to generate this kind of ability as a natural product of its personnel policies. The development and evaluation of alternatives for doing so is a potentially fruitful area for further research.

Software Documentation

Software documentation for major programs is produced by the developer and reviewed by the Air Force. It is assumed that the products will satisfy the information requirements of three different audiences: the development teams, those responsible for the support mission, and the training organizations. Our discussions have led us to believe that this assumption is unwarranted and that, in fact, neither those producing the documentation nor those reviewing it know enough about the specific information needs of the intended audiences. Indeed, we encountered cases where the contractor produced two largely separate documentation products—one for delivery to the Air Force, and one for his own internal use. In these cases, it may be concluded that the military standard documentation requirements [20] are inadequate.

Most of those with whom we spoke agreed with the need for standards in software documentation. However, they questioned whether the current standards encourage the hoped-for results. One source suggested that the main benefit documentation standards should provide is the assurance to program offices that contractors have addressed all relevant software issues during the development period. And those who advocate conducting an independent software verification and validation effort are interested in exactly that issue. However, standards must also lead to technical documentation that is of use to operational and maintenance personnel, and this requirement often seems to be unmet. Most of those interviewed felt that the Air Force was paying more for documentation (in direct and indirect costs) than is warranted by its ultimate utility.

We suggest that the Air Force should determine the minimum operationally essential information requirements of all the groups mentioned above and examine current documentation standards in light of those requirements. However, those performing this task must remember that existing standards are enforced across a range of application areas whose software documentation requirements differ. Thus, extra interim design documentation might be required from programs that push software technology into relatively more uncertain areas. This would guarantee additional cost, but it might, over the long run, help to reduce some kinds of late-discovery and very expensive software problems. Moreover, the same review might lead to a reduction in documentation requirements and costs in more mature application areas.

The Utility of the Review Structure

In most of the cases we examined, the software PDRs and CDRs seemed to be more useful to the contractor than to the program office. Contractors tended to structure software development tasks around these reviews and to prepare carefully for them. Upcoming reviews provide incentives for internal review and coordination of software-relevant activities progressing on a number of fronts. To the extent that these internal reviews lead to early discovery of and remedies for errors, the results are beneficial to the program office and to the end user.

However, both program office and contractor personnel agree that the Air Force review team is generally unlikely to discover software problems the contractor might have missed. Only two of the eight program offices we visited felt at all confident of their ability to mount a technically competent software design review. The remainder appeared to view management competence as a proxy for technical competence, the argument being that if the contractor's efforts seem well planned and managed, then at least some confidence in his eventual product is warranted.

This is not to fault program office personnel. We have already noted some of the serious personnel problems they face, and while these problems have existed for a long time, effective reviews have been carried out despite their limiting influence. Two activities appear to contribute to more effective software reviews: (1) taking time to develop and implement a formal review plan, and (2) gaining access to software expertise. Thoroughgoing written document reviews carried out before a site visit and the orchestration of contractor presentations according to the needs and capabilities of the review team should be made the rule instead of the exception.

As for accessibility of software expertise, program office review teams frequently augment their ranks with representatives from the using and support organizations. Where items other than computer software are concerned, this action can reasonably be expected to enhance the team's effectiveness. But the effectiveness of additional expertise in the software area is diluted by opportunities for the introduction of initially invisible errors in software design and implementation and by the fact that system operators do not necessarily need or have software expertise.

Two of the three AFSC product divisions have formal and continuing relationships with non-Air Force organizations that can provide technical, scientific, and managerial support when needed: ESD has MITRE Corporation, and the Space Service Division relies on Aerospace Corporation. ASD has no such outside support organization and relies instead on internal resources, such as Avionics Laboratory personnel, and various external sources of expertise. However, there has been evidence of conflict between individual program offices and the Laboratory, and the detailed expertise needed at the program office level is often unavailable in the R&D community. Thus, ASD must rely more heavily than the others on the competence and initiative of their prime contractors. This is not necessarily a disadvantage, but it is reasonable to think that ASD interests would be better served by an organization dedicated to providing the needed support services.

Finally, review planning and the use of non-organic expertise should be sensitive to the fact that software is applied to different functional areas with varying effectiveness. One of our general observations was that little attention is given to this fact. Thus, an area with well-defined and clearly understood tasks might be more confidently reviewed because the development process reflects this order. However, automation programs for other tasks have more uncertain outcomes. In those cases, more and/or different kinds of formal reviews may be necessary to assure, as much as is possible, a favorable software outcome. We suggest that the Air Force act to determine the nature and magnitude of such functional area differences and to assess their implications for software development management.

The Proposal Generation Process

Development contracts are sometimes awarded on the basis of overly ambitious or poorly considered technical proposals. This is not a new problem, but it is particularly important in systems with embedded computers, because of the high costs associated with software changes during the development process.

As we stated earlier, such problems are apparently less frequent when system definition contracts are employed. As the contractor model shows, receipt of an FSD RFP leads to the formation of proposal-writing teams. These are usually made up of the contractor's advanced technology personnel, and the resultant area specifications often reflect their bias toward the most advanced technically feasible approaches to system design. This has led to trouble in some programs where those responsible for developing the proposed software do not share the proposal writer's confidence that a successful laboratory demonstration constitutes practical feasibility.

The problem is both enabled and aggravated when, as is often the case, a relatively short time is available for proposal development. It is enabled because in the absence of a system development contract, those in the advanced technology

group are the only ones with sufficient knowledge to write a proposal in a short time. The problem is aggravated because to do so, they must rely on existing analyses which, having been accomplished in a predominantly R&D milieu, often give little weight to practical issues.

Where system definition contracts have been used, or where FSD proposals have been produced over a longer period, these problems appear to be less frequent. Thus, we conclude that the Air Force should consider more widespread use of system definition, or Phase-0, contracts.

Taken together, the foregoing observations suggest that, given its pivotal roles in both program management and system performance, software should be considered an important program-level decision variable. Deferring attention to software issues until development time and expecting flexibility in both software and the development process itself to cover any resultant anomalies is to ignore a decade of experience in embedded software acquisition. The fact that we observed exactly this behavior in most programs we studied suggests that the lessons learned over that decade have not been collected and made available in a way that encourages their application. That is a task requiring immediate attention.

IV. POLICY DIRECTIONS AND FUTURE WORK

POLICY DIRECTIONS

Although more work is required before strong recommendations can be made, we have discussed a number of observations that raise specific policy issues, and we have indicated fruitful directions for continued research. These issues are discussed below from the viewpoint of four specific questions. Finally, we shall summarize our current impressions of the most appropriate directions for continued research.

1. *Should software receive the same attention given to the more familiar hardware variables during program-level decisionmaking?*

Early and formal analysis of the broad software options visible prior to RFP development is one approach to improving the acquisition process. Is an off-the-shelf approach really sensible, given what's on the shelf? Can all envisioned processing really be accomplished with one mainframe? Where is software technology going in areas that will be relevant within the life cycle of the program under consideration? What system/software designs will allow maximum advantage to be taken of that progress when it is time? Some of the information needed to answer such questions may be available earlier than is now assumed. Even when certain information is initially unavailable, that fact would become known at a point in time when efforts to obtain the needed information will be least impacted by budget and schedule pressures.

A basic problem is that of creating incentives for treating software as a decision variable from the outset. One solution might be to require explicit attention to computational issues in the earliest decision support documents. For example, the role of a Mission Element Needs Statement (MENS) includes establishing constraints on the search for alternative responses to a mission deficiency. This involves identifying the "limits on resource investment to be made" [21], and since software is the major component of investment in any computational subsystem, even a rough estimate of its potential scope might contribute to the utility of the MENS.[10] It might also be expected to facilitate continued attention from that point on—as is suggested in AFR 800-14 with respect to the PMD and the subsequent PMPs—to variations in those estimates resulting from (1) purposeful changes to the evolving software picture and (2) the indirect consequences of decisions in other areas.

Our models suggest that useful information is sometimes available from contractors early in the decisionmaking process. Their efforts to develop a corporate response to impending RFPs involve making rough resource estimates based on analyses of preliminary design alternatives. However, further work will be needed to establish exactly what software information is available from contractors and other sources that would be useful in the Air Force's program-level deliberations,

[10] We use the MENS only as an example; there is no recommendation at this point to focus on it as the best carrier for additional software information.

and to work out the procedural issues associated with making such information available.

2. *Should earlier attention to software include identification of the specific types of software expertise required by a future program office and attention to matching existing resources to those requirements?*

In spite of the shortage of software personnel, program managers who went out of their way to acquire needed talent were usually at least partially successful. This implies, first, that more talent exists than is routinely assumed to be available to a program office and, second, that it is difficult to recognize.

Some talent exists at the operational level in individuals who routinely apply their training and growing experience to a variety of Air Force functional missions. We suggest that they are difficult to recognize because they may not be classified or thought of as software or even computer experts. Yet their responsibilities may involve computer programming and systems analytic tasks in specific mission areas. In some cases, these people have college degrees in computer science or in related fields, and they represent an experienced pool of software and functional-area talent. Program office personnel who successfully acquired needed software expertise despite the shortage often did so on the basis of personal familiarity with exceptional individuals from this pool.

It is important to emphasize that "exceptional" in this context connotes not only superior, but also particularly appropriate, expertise. The point is that program offices for major weapon systems have less need for "software people" in general than they have for the appropriate combination of software expertise and functional application experience that is often found at the operational level.

Successful acquisition of weapon system software could become dependent on the continued and accelerated development of this newly important resource. This does not necessarily call for a solution to the more general personnel training and retention problems now facing the Air Force. Rather, the objective is to recognize and better utilize an existing resource. Whether and how to do so without major changes to current Air Force personnel policies are still very difficult questions.

3. *Should the Air Force make more regular and widespread use of system definition (or Phase-0) contracts?*

The activities carried out under a Phase-0 contract are also accomplished in the absence of such a contract. The difference is that, without it, the work is done under the budget and schedule pressures of the FSD contract, by a group whose bias is to leave prior decisions intact. For software, the result can be the introduction of design anomalies—for example, specifying faster processing throughput on a mission computer than is found to be required by an interfacing device. Efforts to meet such requirements lead to more complex code than is necessary, and this, in turn, leads to cost growth, lower reliability for the user, and maintenance problems.

Although our sample was limited, those programs that contracted for system definition separately from development seemed to experience fewer software problems than the others. Moreover, the extra time involved was seen by Air Force and contractor alike more as a benefit than as a built-in delay. The extra time given to system definition resulted in more complete analyses, improved documentation, and a lessening of some problems caused by the advanced technology bias (because

more and different people were involved). Development was started with increased overall confidence, and software efforts gained from an improved understanding of the operational mission and requirements picture at the time development began.

Thus, on balance, there appear to be important benefits in at least one major cost area associated with the use of Phase-0 contracts.

4. *Should the review structure and documentation requirements for embedded computer software be examined with a view toward tailoring their use to the maturity of the application area?*

The current software development review process relies heavily on an assumption that the prime contractor is expert in the development of a particular kind of system. For a large number of system types, this assumption is a fair one. For certain systems, however, particularly those in which the functions to which embedded computers are applied are not well understood, the assumption appears unwarranted.

As computers are applied to increasingly complex defense systems, we will probably continue to encounter problems based on trying to program what we think we understand, but don't. In such cases the earlier the existence and extent of this problem are discovered, the better. Our analysis of a number of embedded software acquisition efforts leads us to argue that some of their problems might have been discovered earlier under a more appropriate review structure.

One of the programs we examined could have benefited from a User Requirements Review, wherein, for example, operational personnel critique the translation of their stated needs into the contents of a software Part I specification. In fact, this has been suggested before [22], but it has not, to our knowledge, been formally tested. Another program might have used a System Integration Review because it had major problems in that area. We don't suggest that these, or any other specific reviews, be added across the board, nor can we argue for a wholesale revision to the review structure. But it seems apparent to us that as software is applied to increasingly complex tasks, the practice of lumping concern for all potential problem areas into the preliminary and critical design reviews is becoming less and less effective.

The need is for early recognition of problems that lead to increased software uncertainties as development proceeds, and for procedures—contingency plans, in a sense—to enable selective focus on potential sources of trouble.

A similar argument can be made for software documentation: Different development environments require different information. This point has been adequately discussed earlier; here we simply note its close relationship to the review issue, and we suggest both as fruitful topics for further research.

FUTURE WORK

We have identified a number of areas where more focused research seems indicated. These areas are outlined below.

1. *Information*

   - Identification of software-relevant information that is both available (from defense contractors, commercial R&D institutions, universities, and

other sectors) and useful at various stages of the system acquisition process.

- Identification of the minimum essential information requirements of the various groups (e.g., development, support, training, operations) involved with embedded software applications.
- Development of procedures for accessing such information, and for its effective application in various Air Force acquisition environments.

2. *Personnel*

- Techniques for improved recognition of, and program office access to, operational personnel with particularly appropriate skills and experience in applying software technology to specific mission areas.
- Development and evaluation of incentives for attracting personnel to operational assignments that produce a doubly expert resource.

3. *Structure*

- Development of an analytic framework for a priori identification of software applications that would benefit most from tailored software review and documentation procedures.
- Development of appropriately tailored software review and documentation procedures, and of methods for implementing them in various Air Force acquisition environments.
- Development of an improved environment for technical cooperation between system program offices and the Air Force Avionics Laboratory with regard to embedded software R&D.

EPILOG

The 1979 Armament and Avionics Planning Conference (AAPC) was held while the draft version of this document was being circulated for comments. Most of the material presented here was discussed with members of the Acquisition Management Policy subpanel at that conference. Those discussions provided insights that helped to crystallize our results around the notion of an acquisition framework consisting primarily of the various studies, reviews, audits, and documentation standards generally associated with acquisition programs.

One task of a program office is to guide system acquisition efforts so as to satisfy requirements that exist when the system is delivered. The tools currently available to a program office often fail to provide the visibility and control needed for this job. This seems especially true where embedded computers are involved. The improvements these computers bring to system performance, operational simplicity, and maintenance techniques carry the price tag of increased complexity in computational subsystems. The software component of these subsystems is the major source of such complexity.

Because of their differing roles in the introduction of computers to defense systems, the Air Force has lagged the contractor community in perceiving and dealing with this complexity. A fundamental issue, harmonious with our own results and given concrete recognition in the report of the AAPC [23], is the decreas-

ing relevance to modern systems of the Air Force's current hardware-biased acquisition management framework.

Not all software acquisitions should be managed alike. Where uncertainty is low and believable baselines can be established, existing acquisition management procedures may be adequate. More frequently, uncertainty is high and initial baselines are fictional. In such cases, there is seldom either a structural or experiential basis for planning an effective management strategy. Moreover, management techniques must be sensitive to certain categorical differences among software applications. For example, software development for an avionics suite can be carried out under circumstances quite different from those applicable to the development of some command and control software.

There is also wide agreement that, among other discrepancies, current documentation products are inadequate for the development process. Frequent waivers on contract-deliverable documents, conflicts over rights to contractors' internal documentation, and uneven attention to pre- and post-review reporting reflect confusion about who needs what kind of information. And the review structure itself seems increasingly deficient as a means of illuminating software problems in time for cost-effective treatment.

These observations deal with fundamental concepts: estimation, visibility, information, structure, etc. They are applied in varying degree, and with varying importance, to all management tasks. After more than a decade, the Air Force is still in a cut-and-paste mode in applying these concepts to software acquisition. Their basic management tools and the framework within which they are applied are long overdue for reexamination. The results presented in this report argue for this, and their agreement with the recommendations of the Acquisition Policy group at the recent AAPC reflects significant Air Force support for such a study.

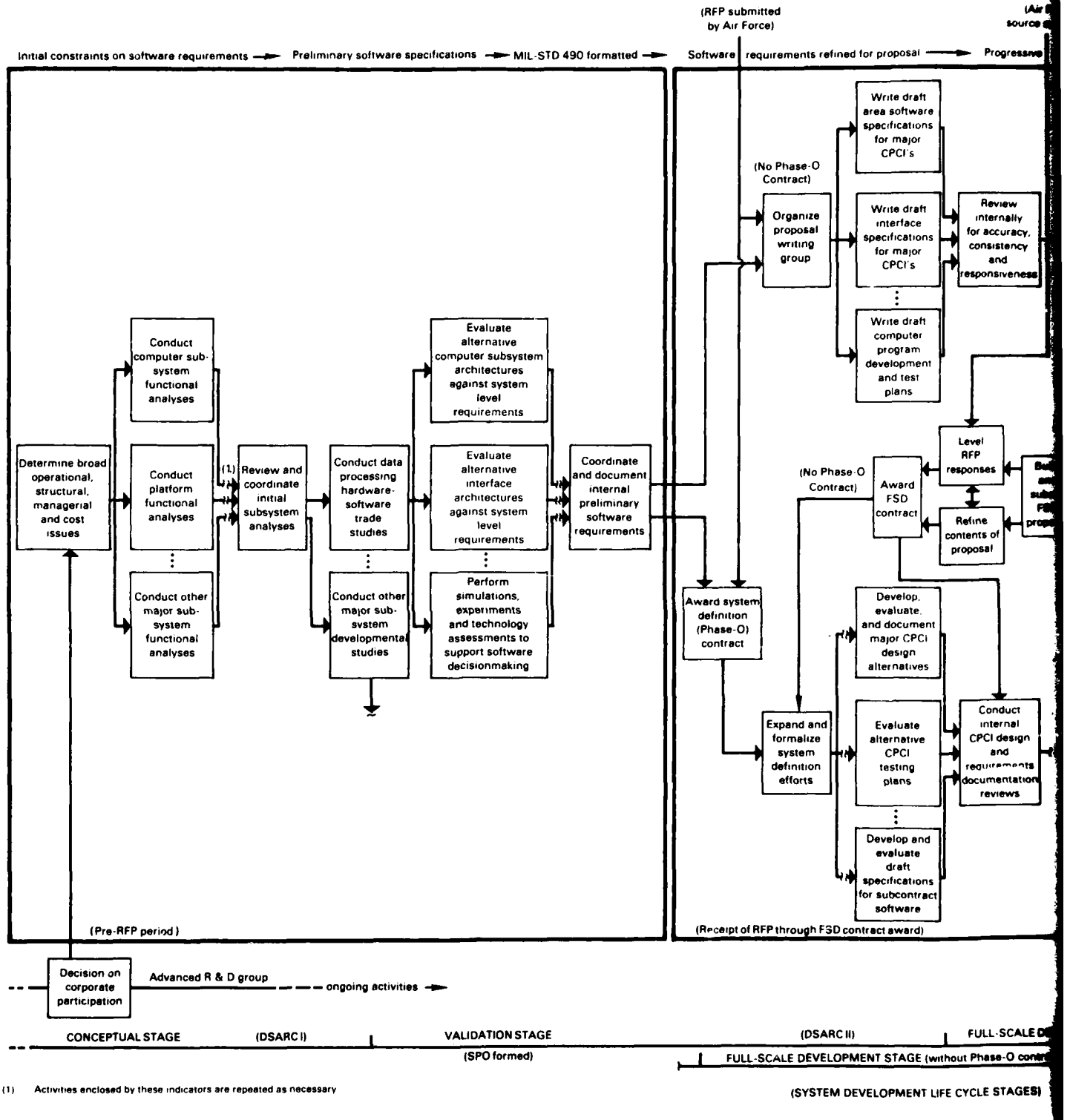
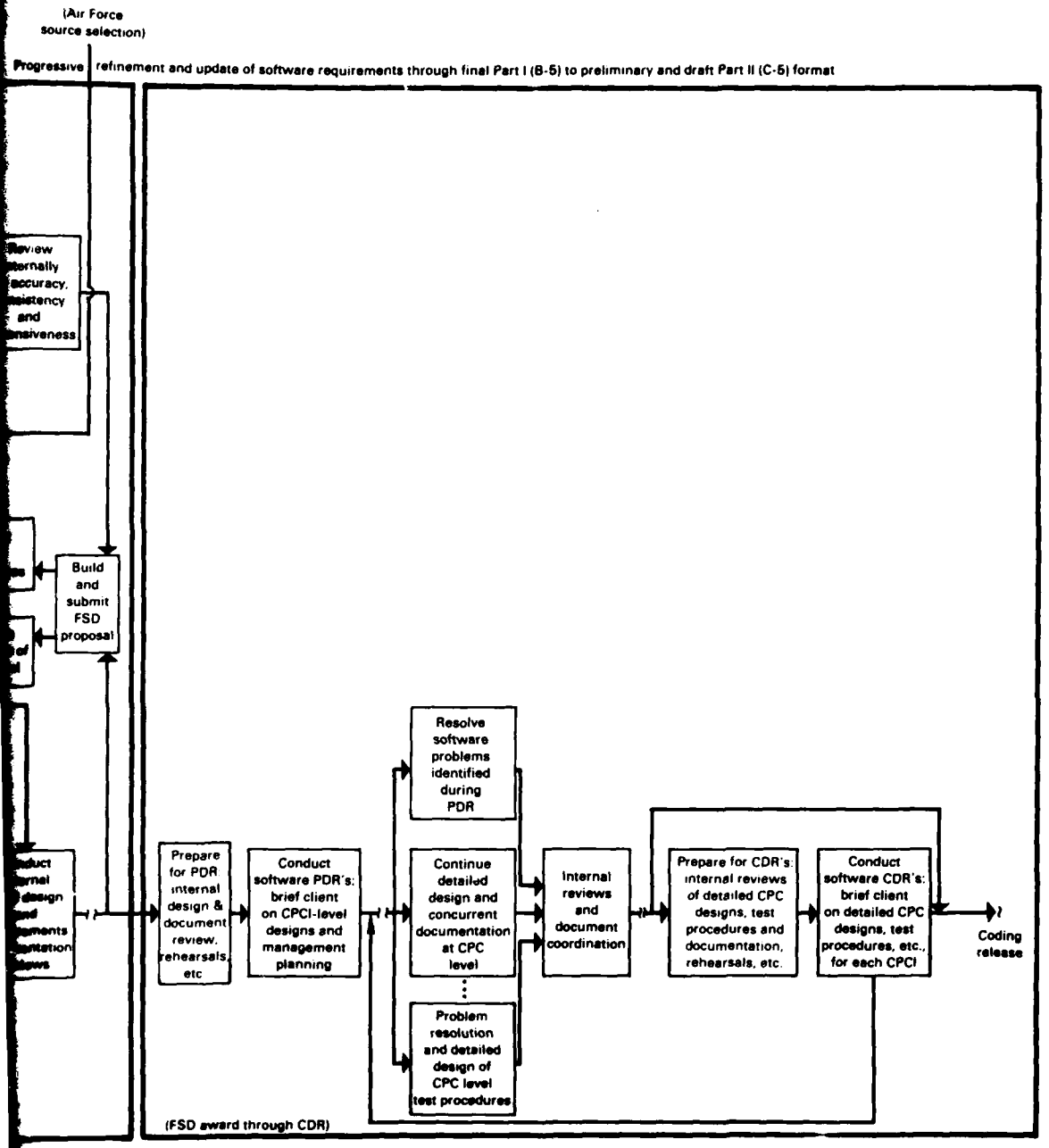


Fig. 1—Aggregate model: Air Force embedded software a

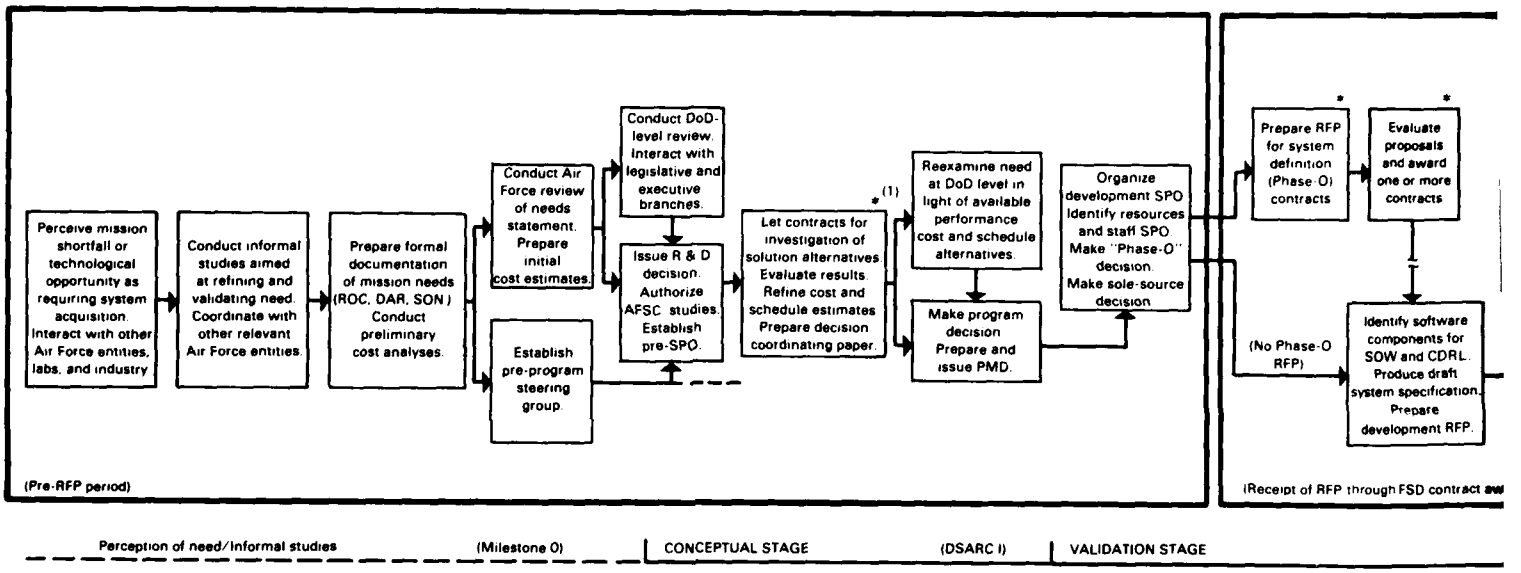


CALE DEVELOPMENT STAGE (with Phase-O contract)

-O contract)

AGES)

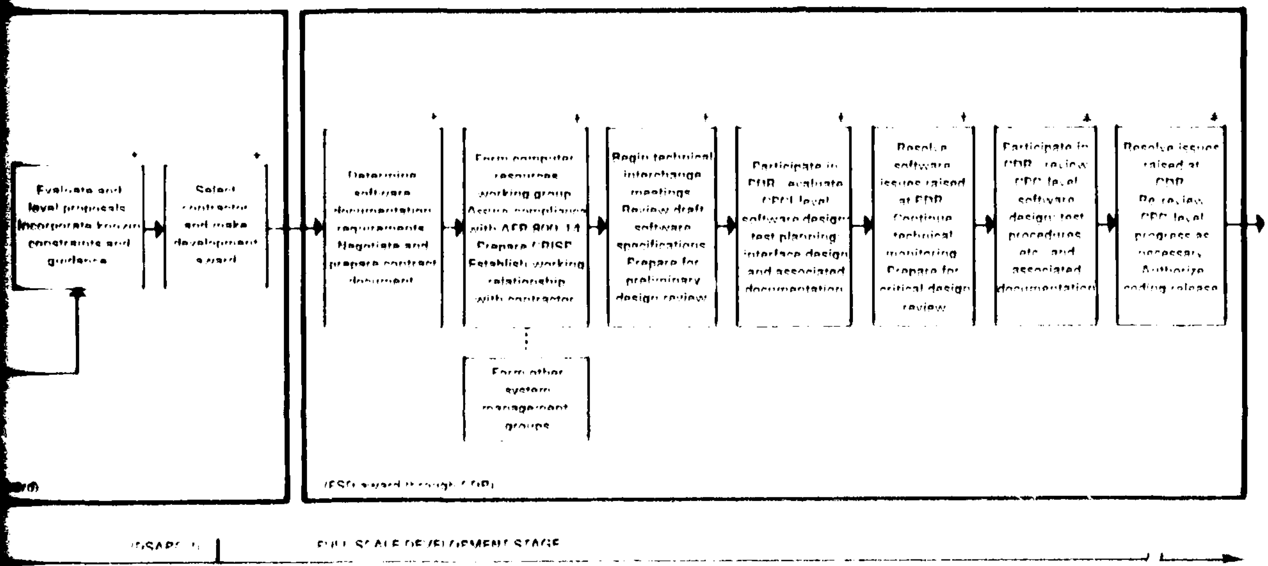
re acquisition management process



(1) * denotes predominantly formal Air Force-Contractor interaction.
Note that most activities, marked or not, involve informal interaction.

(SYSTEM DEVELOPMENT LIFE CYCLE STAG

Fig. 2—Prime contractor software deve



es)

lopment process

REFERENCES

1. *DoD Weapons System Software Management Study*, The Johns Hopkins University Applied Physics Laboratory, Baltimore, May 1975.
2. *DoD Weapons System Software Acquisition and Management Study*, MITRE Corporation, May 1975.
3. Drezner, S. M., et al., *The Computer Resources Management Study*, The Rand Corporation, R-1855/1-PR, April 1976.
4. Teichroew, D., and E. A. Hershey, III, "PSL/PSA: A Computer Aided Technique for Structured Documentation and Analysis of Information Processing Systems," in *IEEE Trans. Software Engineering*, Vol. SE-3, No. 1, January 1977.
5. Bell, T. E., D. C. Bixler, and M. E. Dyer, "An Extendable Approach to Computer-Aided Software Requirements Engineering," in *IEEE Trans. Software Engineering*, Vol. SE-3, January 1977, pp. 49-60.
6. Davis, A. M., and W. J. Rataj, "Requirements Language Processing for the Effective Testing of Real-Time Systems," in *Proc. Software Quality Assurance Workshop*, San Diego, November 15-17, 1978.
7. Martin, W. A., and M. Bosyj, "Requirements Derivation in Automatic Programming," in *Proc. MRI Symposium on Computer Software Engineering*, April 1976.
8. Pierce, R. A., "A Requirements Tracing Tool," in *Proc. Software Quality Assurance Workshop*, San Diego, November 15-17, 1978.
9. *Management of Computer Resources in Systems*, Air Force Regulation 800-14, Vols. I and II, Department of the Air Force, September 1975.
10. OMB Circular No. A-109, *Major Systems Acquisitions*, Office of Management and Budget, April 5, 1976.
11. *Policies, Responsibilities, and Procedures for Obtaining New and Improved Operational Capabilities*, Air Force Regulation 57-1, August 1971.
12. *Management of Automatic Data Processing Systems*, Air Force Regulation 300-2, February 1975.
13. *Operational Requirements: Statement of Operational Need (SON)*, Air Force Regulation 57-1, June 1979.
14. Department of Defense Directive 5000.1, *Major System Acquisitions*, January 18, 1977.
15. Minutes of the Modeling Subpanel, Second Annual Avionics Planning Conference, Colorado Springs, Colorado, October 31-November 7, 1978.
16. Glore, J. B., "ESD Software Cost Prediction Aids Project: Interim Results and Plans," The MITRE Corporation, Working Paper 22029, November 17, 1978.
17. *Software Engineering and Management Plan*, Vol. II, Air Force Systems Command, AFSC/XRF, March 1976.
18. *Air Force Policy on Avionics Acquisition and Support*, Air Force Regulation 800-28, Department of the Air Force, September 1978.
19. Ziernicki, R. S., "Avionics: The Road Ahead," *Air Force Magazine*, July 1979, pp. 67-75.

20. *Specification Practices*, Military Standard (MIL-STD) 490, October 1968.
21. Department of Defense Directive 5000.2, *Major System Acquisition Process*, January 18, 1977.
22. *Management Guide to Avionics Software Acquisition, Vol. II: Software Acquisition Process*, Logicon Corporation, ASD-TR-76-11, June 1976.
23. Minutes of the Acquisition Management Policy Subpanel, Armament and Avionics Planning Conference, Nellis AFB, Nevada, October 15-19, 1979.