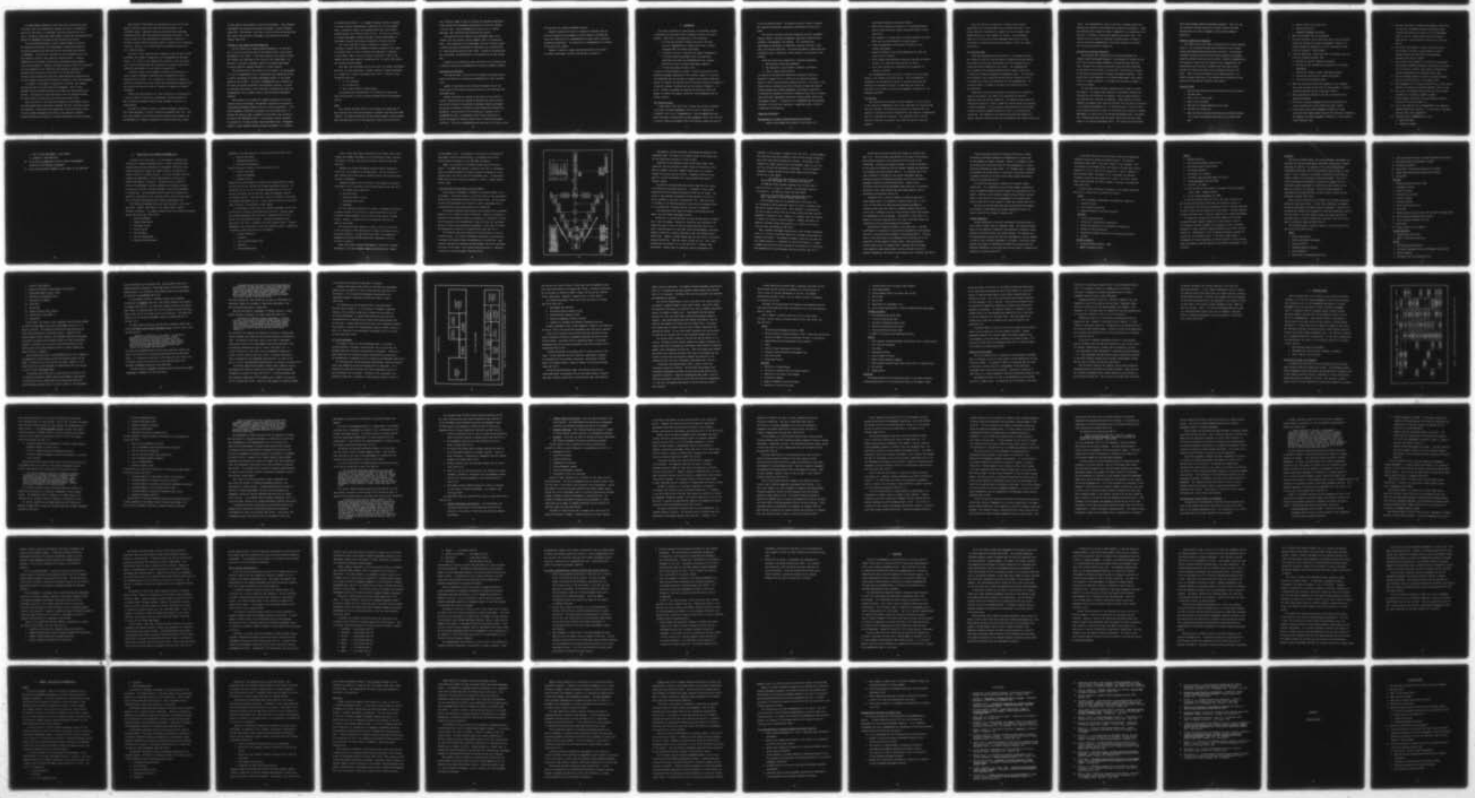
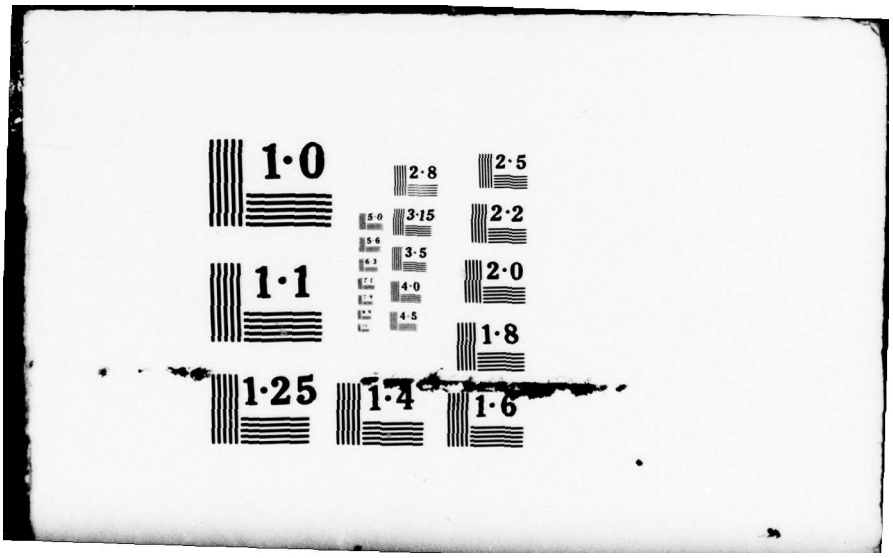
AD-A065 879 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
A STUDY OF EMBEDDED COMPUTER SYSTEM SOFTWARE ACQUISITION MANAGE--ETC(U)
SEP 78 G M BARBEE

UNCLASSIFIED AFIT/GSM/SM/78S-1 NL

1 OF 2
ADA
065879




1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
5·0
5·6

AFIT/GSM/SM/78S-1

LEVEL II

AD A065879

DDC FILE COPY

A STUDY OF EMBEDDED COMPUTER SYSTEM SOFTWARE ACQUISITION MANAGEMENT AND RECOMMENDATIONS TO IMPROVE DEVELOPMENT VISIBILITY

THESIS

AFIT/GSM/SM/78S-1 Gary M. Barbee
Capt USAF

Approved for public release; distribution unlimited

(14) AFIT/GSM/SM/78S-1

(6)

A STUDY OF EMBEDDED COMPUTER SYSTEM SOFTWARE ACQUISITION MANAGEMENT AND RECOMMENDATIONS TO IMPROVE DEVELOPMENT VISIBILITY.

(9) Master's ~~THESIS~~,

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

in Partial Fulfillment of the

Requirements for the Degree of

Master of Science

(12) 107p.

by

(10) Gary M. Barbee, B.S.E.E.

Capt USAF

Graduate Systems Management

(11) September 1978

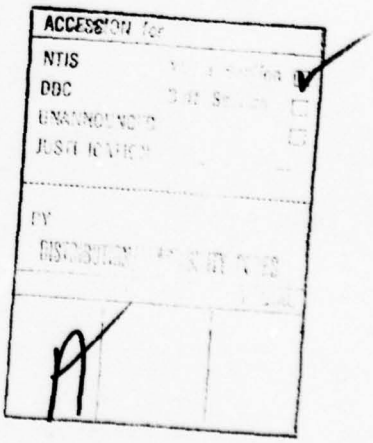
Approved for public release; distribution unlimited

Preface

This thesis is an attempt to document Embedded Computer System (ECS) software development problems and recommend solutions and improvements to some of those problems. This research may also be of interest to readers desiring an introduction to the DOD software development process and the accompanying guidance and regulations. I accept full responsibility for any errors contained herein.

I would like to gratefully acknowledge the advice and assistance of my thesis advisor, Professor Charles W. McNichols and my reader Professor Saul Young. Very special thanks go to my wife, Shirley, and children, Michelle and Jeffrey for their help and understanding.

Gary M. Barbee



Contents

List of Figures

Abstract

The United States Air Force is the largest user of computers in the world and a major portion of that information processing capability is comprised of digital avionics computers. This thesis describes some of the major problems of acquiring Embedded Computer System (ECS) software for avionics systems. A description of the DOD avionics software acquisition process is included for background information as well as a discussion of the applicable guidance, policies, and regulations.

Recommendations to improve software acquisition were derived from literature research, refined by interviews with practicing software engineers and managers, and presented as a product of this thesis. The interviews were conducted with software acquisition personnel at the Aeronautical Systems Division of Air Force Systems Command at Wright-Patterson AFB, Ohio. A major conclusion of this thesis is that the development of a computer software management discipline is both necessary and feasible.

I. AVIONIC SOFTWARE ACQUISITION PROBLEMS

Introduction

Avionic software development and procurement has been an area that has largely escaped normal managerial controls. Frequently, actual software costs exceed the initial budget by 100% and the time to reach operational status is often twice as long as scheduled (Ref 29:138).

It is difficult to estimate the effort required to produce software, especially avionic software. The software development process is not well understood and the numerous factors affecting the development leave decision makers with a limited ability to effectively monitor and direct the process.

Prior to WWII, the standard practice was to develop and acquire most system components and sometimes complete subsystems separately from their use in the total weapons system. Design and integration approaches of most weapons systems were stable enough to permit components and entire subsystems to be integrated for the first time only after each was separately completed. The Air Force was literally buying major weapons systems in bits and pieces rather than as a total functioning system (Ref 32:73).

Following WWII, the impact of advanced technology required a new approach to system development that designed a component from its inception to integrate efficiently into the total system. This called for greater controls over the complex parallel developments to coordinate schedules, functions, physical characteristics, etc. (Ref 32:73).

Methods were developed within the Air Force to control the technical and managerial functions of both contractor and in-house development.

This resulted in a profileration of staffs and multiple levels of review in both industry and the Air Force (Ref 32:75).

The need to improve avionic system software acquisition has been made apparent by the succession of cost overruns and defective systems that have drawn sharp criticism to one or more programs in recent years. The clutter of programs and problems has made it difficult to understand or grapple with the underlying causes of software acquisition difficulties or their solutions. Well known major systems with these characteristics have been the B-1, C-5, F-15, F-111, etc.

Each of these weapons systems contains a computer system as an integral part of its identity and function. This has become so common that the term Embedded Computer System (ECS) has evolved to describe the computer hardware and software that is buried within a major weapons system. This type of ECS is physically incorporated into a larger, generally mobile, electromechanical system whose primary function is not data processing (Ref 30:159). Generally, the DOD specifies the weapons system requirements and the developing contractor determines whether a computer system must be integrated into the weapons system to satisfy those design requirements.

Until recently, the majority of managers and contracting personnel were content to treat software merely as data (Ref 17:1). These managers were primarily concerned with the weapons system meeting overall design specifications, not how a black box subsystem worked inside the total system. Documentation of software of this type was sometimes given a low priority because of its expense, and the idea of buying it later if necessary was prevalent. This created a situation where a major weapons system was acquired without knowing completely what was inside it or how to maintain or modify it. In some cases, when the software documentation had to be

bought, it did not exist and the individual people who developed it were no longer available to provide it. The "data" therefore had to be completely reconstructed at great expense.

Many studies have been made to find ways of improving software acquisition. A major idea of these studies has been to increase management visibility of software development. The importance of this visibility is supported by the DOD Weapon Systems Software Management Program designating software management visibility as one of its four main objectives (Ref 14:3). The lack of software management visibility was also identified as a major DOD problem in a Johns Hopkins University Applied Physics Laboratory (APL) study entitled DOD Weapon Systems Software Management (Ref 19:2-4).

Software is the least visible and tangible of any aircraft subsystem and is therefore the least understood. System Program Office (SPO) management seems to have a problem working with and understanding this nebulous area because management is generally far removed from its development. Unlike the manager, who sometimes fears software, the engineer who deals with it on a day to day basis refers to it as if it were a physical unit, not as the concept which it really is.

The development, application, utilization, and management of software for avionic systems is one of the most critical problems in implementing digital avionics. This is important because digital avionics is apparently the airborne information processing tool of the future.

This statement has great significance to the DOD as evidenced by the large software expenditures to date. Jacques Gansler, Deputy Assistant Secretary of Defense for Material Acquisition, has stated that there are at least 115 different defense systems that utilize ECS, approximately one-half of which are now in service. He also stated that the DOD is spending more than $3 billion annually for software (Ref 16:41-43) (Ref 34:5).

This has created great interest in the cost of software because some people still consider software to be merely data required to utilize hardware.

The relative cost importance of software versus hardware is depicted by the graph in Figure 1. This graph was on the cover of the Defense Management Journal, October 1975 (Ref 18:3).

Defense systems costs have been rising at a rate approximately five times the national inflation rate over the last twenty years. (Ref 4:3-4). Lack of contract costing visibility further complicates the cost problem because software in the past was never classified as a separate contract line item. Software was usually embedded in the total weapons system cost and could not be tracked separately. Management did not place enough

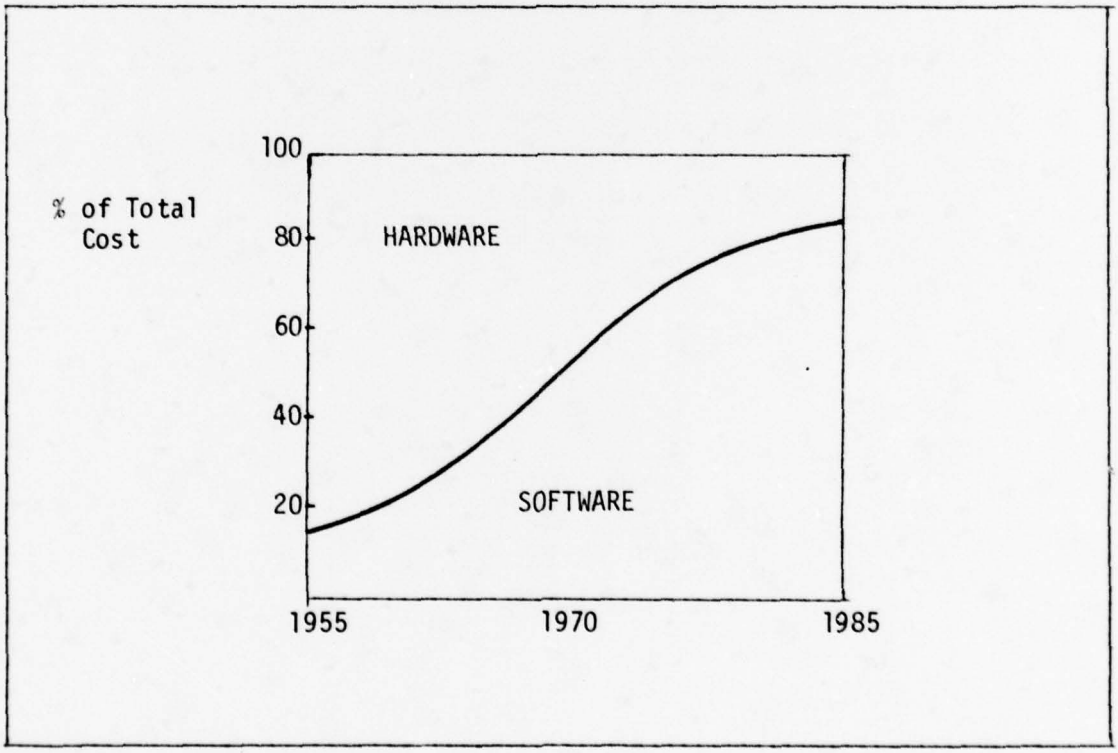


Figure 1. Hardware versus Software Cost Trends (Ref 18:3).

emphasis on development of computer resources until after it was designated a problem area, and achieving visibility after the fact was nearly impossible.

The indirect costs of software are even greater than the direct costs because software is generally on the critical path in overall system development. That is to say that software schedule slippages usually translate directly into total system schedule slippages (Ref 39:7).

Billions of dollars are being spent annually for embedded computer systems (hardware and software) by the DOD. However, a search of the literature uncovers no formal design method to insure that a given hardware/software mechanization is near optimum for either a general or specific application (Ref 6:98).

Some Major Software Problems: (Ref 31:39):

1. Faulty and incomplete communication of user requirements.
2. Unrealistic cost estimates caused by insufficient visibility and control. (There appears to be no reliable techniques for estimating development effort.)
3. Unrealistic time schedules caused by the same lack of visibility and control.
4. High software failure rates.
5. Incomplete and insufficient specifications.

Software visibility has been heralded as the key to solving the software development problem. The major reason has been the lack of importance associated with software development in relation to the overall weapons system. This lack of attention amplifies errors made early in a program such as inadequate requirements definition and incomplete integration of hardware and software requirements. The lack of development status monitoring during development aggravates the problem (Ref 14:i).

An understandable eagerness to avoid past errors and minimize future criticisms leads to the bureaucratic tendency to bring all information to the top for decisions or to leave major decisions and data at too low a level. The DOD has shown wide swings between "centralized" and "decentralized" patterns of management philosophy on decision making (Ref 32:87).

The current DOD philosophy on decision making attempts to maximize the advantages and minimize the disadvantages of centralized control by selectively decentralizing some decisions and encouraging participation in management. This policy has increased Air Force responsibility for its own programs by giving it more authority to make decisions. Finding a compromise position between centralization and decentralization appears to be the best approach. However, philosophy and policy need to be reinforced by clear statements on the limitations and placement of authority and responsibility within the Air Force and all of the DOD (Ref 32:87). Currently, policy making and acquisition program monitoring responsibilities are split between the business and technical functions of high level management.

In the DOD, a large number of regulations, directives, and standards have been written for systems acquisition management. Most of these documents were not designed originally for software but were modified after the fact. The majority of these publications are still hardware oriented and conflict with each other and with current policy (Ref 35:29).

Upon initiation of a new acquisition program, the procurement function must begin employing the contractual techniques and tools required by policy and regulation. These policies and regulations, which were usually written for more orthodox procurement applications, have themselves frequently created difficulties when applied to advanced technology acquisition programs.

When technical requirements and considerations occur first and the business function second, the acquisition process operates in an information vacuum. Important issues go unresolved and are sometimes determined without inputs from affected organizational functions. Those issues include roles and relationships of government to industry in defining a system, method of approach, degree of technical risk, scheduling, priorities, and cost, all of which should be considered from the start by all affected parties.

With only a single organized effort underway to meet the need, system performance and schedule slippage have to be accommodated by additional funding. As a result of this monopoly-like situation, costly and burdensome controls and regulations must be applied to a greater extent than in competitive procurements to assure public accountability.

In addition, there are no standards to measure the efficiency of a single software development undertaking and no competition to aid in choosing the best system. This, coupled with a contractor's tendency to promise the customer what he wants rather than innovating and demonstrating new products that were not asked for, decreases the emphasis on product improvement.

Another recurring problem area is that contractors are overoptimistic in their estimates of system cost, performance, and delivery date and make contractual commitments based on those estimates in order to win program awards.

The need for software visibility to combat development problems has been stated repeatedly in current literature and speeches (Ref 14:1). However, many attempts at corrective action to date have been piecemeal and counterproductive, leading to regulations to explain regulations, people

to check people, and procedures to facilitate procedures. These patchwork improvements only aggravate the underlying problem in avionic software acquisition. That problem is the lack of visibility over the key decisions that control the definition, development, and acquisition of avionic software (Ref 32:70).

Statement of the Problem and Study Objectives

The ECS software for a weapon system development is not the major part of the total system but must be considered critical to the overall performance of the system. If the total system will not function without the software, the importance of this factor is well established. The question is, how does the manager supervise ECS software development to acquire effective software delivered in a timely fashion?

This question leads to the subject of this research: the development of a set of recommendations that if implemented would improve the ability to monitor and manage ECS software development progress in the United States Air Force (USAF). The initial recommendations are a synthesis of the ideas and concepts contained in the available literature. A series of interviews with experts in the field helps consolidate and refine the recommendations into a product usable to the ECS software development community.

There appears to be no generally accepted guidelines for planning software acquisition programs, and therefore, provisions for monitoring and managing development progress often are initially overlooked. If the proper management information requirements are not defined early in a program, the necessary data is generally not available later to monitor and control the development effort. This prevents accurate evaluation of contractor or in-house progress toward development goals. Management, therefore, cannot determine whether software development is on schedule

or whether more attention, i.e., manpower and money, should be expended to prevent software from becoming a troublesome critical path program delay. According to Brooks, any increased effort must occur before a problem develops to be effective (Ref 7:44-52). The lack of visibility of software development progress precludes accurate establishment of development milestones and measuring completion of those milestones.

This forces the USAF to accept a contractor's forecast of a "reasonable" schedule and incremental estimates of degree of task completion because there is no way to question the contractor's evaluation. In other words, there is no way to refute a contractor's reply of 90% complete and two months ahead of schedule when he is actually 50% complete and two months behind schedule.

What then, does the manager need to know about the software development? Basically, the same information is needed to manage a software development as is needed for a hardware development (Ref 14:4-5). Primarily, these information needs are:

1. Is it on schedule?
2. Is it within cost?
3. Does it meet technical specifications?

The problem being researched here is to find ways to ensure that these three questions are more accurately answered during the development process.

Scope

This research has been limited to ECS software even though some of the discussions and conclusions may apply to Automatic Data Processing (ADP) software. The primary orientation has been toward computer systems acquired under the 800 series of Air Force regulations rather than the 300 series.

This limitation tended to focus on airborne and spaceborne Operational Flight Program (OFP) development because most Air Force ECS software is of this type. These developments also tend to be of a smaller magnitude, thus simplifying data collection and analysis.

The research was also limited to the development phase of the software life cycle because, unlike hardware, there is no production phase. After completion of the development, the final software program is merely duplicated, distributed for operational use, and controlled as a Time Controlled Technical Order (TCTO). Therefore, the operational use, maintenance, and modification of ECS software has not been covered here.

Software cost estimating has been covered only as was necessary and no detailed inclusion was attempted due to extensive treatment elsewhere.

Organization of the Thesis

The study was done in the form of two separate but related investigations with Chapter II discussing the methodology for those investigations.

Chapter III describes the ECS software development process and relates much of the relevant literature directly to the traditional DOD development cycle.

Chapter IV contains the results of the portion of the search of current literature that was intended to determine the state-of-the-art and current thinking on the problems of managing software development. Discussions of potential solutions to these problems were digested and documented. The end result of this section was a tentative set of recommendations that, if implemented, would improve the ability to monitor and manage ECS software acquisition by increasing development visibility. This set of recommendations was then used to initiate a series

of interviews with software development experts.

Chapter V presents the results of subjective interviews with ten Air Force software project managers and other experts to validate and revise the tentative set of recommendations to reflect their collective thinking and experiences. The revised set of recommendations is included at the end of this chapter.

Chapter VI contains a summary and some conclusions with respect to software development visibility and how best to achieve it.

II. METHODOLOGY

This thesis represents an investigation of the problems involved in developing one particular class of computer software, ECS avionic software. Specifically, the objectives of this study were:

1. To review the available literature and derive from that a set of recommendations to improve the ability to monitor and manage USAF ECS software development.
2. To conduct a series of interviews with people knowledgeable in the ECS software development field, who would help consolidate and refine the recommendations into a product beneficial to the ECS software development community.
3. To learn more about the thesis topic.

In pursuit of these objectives, Chapter I discusses the nature of ECS software development problems while this chapter, Chapter II, describes the methodology that was followed to accomplish the research objectives. Chapter III presents and analyzes the ECS acquisition process and Chapter IV surveys the relevant literature that was not covered in Chapters I, II, or III. Chapter V documents and analyzes the interview results and Chapter VI contains the summary, conclusion, and recommended areas for further research.

The Literature Search

A major goal of this thesis was to review the available literature in the area of software management visibility and to combine that information into a set of recommendations. Those recommendations were points the author believed would increase management control and visibility of avionic software development efforts and therefore improve the quality

of the ECS software product. An extensive search of current literature was conducted and problems, experiences, and possible solutions were noted.

The relevant literature consisted of magazine articles, guidebooks, research reports, conference proceedings, regulations, specifications, standards, correspondence, and speeches. This information was supplemented by attendance at conferences, seminars, briefings, interviews, and personal experience. The increasing number of publications and conferences reveal a substantial current interest in the subject of ECS software.

There were three basic categories of literature encountered:

1. DOD sponsored studies and guidebooks.
2. DOD regulations, specifications, standards, and manuals.
3. Thesis, reports and periodicals.

The majority of the published research was sponsored by DOD and is presented in Chapter III as it relates to the acquisition process because it deals with acquisition guidance and policy. The survey of the state-of-the-art software acquisition literature included the areas of software engineering, software management, and software contracting. It provided the information necessary to determine those areas of software acquisition where improvements would most increase product quality and developmental control. A tentative set of recommendations was generated from which to initiate the interviews, a summarized version of which is included here for discussion.

Summarized Preliminary

Recommendations to Improve Software Acquisition Visibility

1. Require using command participation in and input to all

requirements definitions and design reviews.

2. Require that software be included in all System Requirements Analysis (SRA) during the Concept Formulation Phase of system acquisition.
3. Move software to a higher level in the Work Breakdown Structure (WBS) and revise MIL-STD-881A to include software.
4. Establish measurable and achievable milestones for each software development.
5. Emphasize software in the Program Management Plan (PMP) and greater use of the CRISP.
6. Define support and operational software as separate deliverable contract line items with configuration item status.
7. Ensure that one person is accountable and responsible for software in the SPO.

The recommendations were the author's estimate of the most needed changes in the software acquisition process. Each recommendation, if implemented, was believed by the author to increase the quality and timeliness of information necessary to understand and control the acquisition cycle either by affecting the management information or the management personnel.

The Interview

The interviews were designed to compile feedback as to the accuracy of the tentative recommendations and to test additional conclusions derived from the literature. One interview per subject was conducted with the consensus of subjective results integrated into the tentative recommendations after all interviews were conducted. The subjective results include additions, deletions, corrections, and overall opinions of interviews subjects.

Since the interview is subjective in nature and not readily quantifiable, the interview format was not formally pretested. There was no statistical requirement for sample size, and ten subjects were deemed sufficient to give feedback for analysis. An approximately equal mix of civilian and military government experts were utilized. The answers to interview questions were assumed to reflect the subjects true beliefs.

The Interview Format

After the recommendations were formalized, a structured interview was created to guide the interview subjects through the required material. Each subject was asked the same questions with the responses and any ensuing discussion documented by this author. The recommendations were included in the structured interview which covered a wide range of subjective software development and management issues. The interviews were designed to test conclusions derived from the literature and also to evaluate the accuracy and completeness of the recommendations. The interview format is included at the end of this chapter and will be discussed here.

Because the interviews were primarily the expression of subjective views by the subjects, the format first established the subject's experience level and general credibility. The experience and credibility factors were necessary in order to judge the relative weight and usefulness of subject responses. This was done by asking questions about the number of years and type of computer experience as well as current grade and current job. The remainder of the interview involved both general and specific questions concerning technical and managerial ECS software acquisition

issues. The recommendations as well as two sets of software acquisition problems were separately rank ordered and discussed by the subjects. Issues were addressed that allowed for either a managerial or an engineering viewpoint and a difference of opinion was apparent along those lines. The results of the interviews are discussed in greater detail in Chapter V and the Preliminary Recommendations are further discussed at the end of the literature discussion in Chapter IV.

Description of Interview Population

The interviews were conducted with ten software engineers and computer program development managers of the Aeronautical Systems Division of Air Force Systems Command (ASD/AFSC) at Wright-Patterson AFB, Ohio. Their individual and collective experience levels have been determined by the author to be sufficient to justify their classification as experts in the practical applications of software development techniques. These experts have actual experience in the software development field ranging from 5 to 18 years, therefore, are assumed to be qualified to speak on visibility problems.

The interviews were personally conducted by this author in private environments. The subjects were separately interviewed to maintain the individuality and originality of responses. Each was chosen because of his experience level and type of background. Some subjects were senior engineers and managers while some were Captains with five years of experience in the software development area. An interesting point is the fact that a junior Captain with only six years in the service and five years in software development is a senior man in the software development field. The general lack of experienced people made the junior Captain the Functional Group Leader of his software development office. The subjects were distributed

about evenly between managers and managing engineers. That is to say the shortage of software acquisition personnel requires that some engineers must also serve as managers, usually without adequate preparation.

Methods of Analysis and Comparison

This thesis did not lend itself to quantitative or totally objective analysis. The literature provided expert opinions on how to analyze software development problems and possible solutions. The interviews documented the opinions of practicing software engineers and managers concerning software visibility and the author's tentative recommendations. The synthesis of ideas from the literature into recommendations, the interpretation and inclusion of feedback from interviews, and the derivation of thesis conclusions are the subjective efforts of the author. While striving for objectivity and unbiased analysis, the author's personal experience in the software area influences the interpretation of interview and literature data sources.

Interview Format

1. Have you been associated with the acquisition of ECS software?
2. How many years?
3. What is your current grade?
4. Where are you assigned?
5. What type of software experience do you have?
6. Describe your current job.
7. What order would you place the following ECS software problems in to reflect the greatest difficulty to your organization? Why?

a. Dynamic state of the technical art.

b. Contracting policies.

c. Inadequate management techniques.

8. What other problems have you encountered?

9. From your experience, do you agree that some of the ECS software acquisition problems are caused by management's inability to develop appropriate techniques as fast as the technical state-of-the-art advances? Please comment.

10. From your experience, in what rank order of importance would you place the following problems? Why?

    a. Defining the specific software requirements.

    b. Defining and then implementing milestones for ECS software development.

    c. Tracking the software system's development progress.

    d. Defining and specifying the software end product.

    e. Verification and Validation (V&V).

11. Would you say that ECS acquisition managers are well prepared and trained or would you say that, for new personnel, a learn-by-experience education system is employed? Explain.

12. What experience and training do you feel are required for an adequate background?

13. Do you feel that good management practices and expertise are usually available but are not effectively used? Explain.

14. Do you believe that useful management information is often unavailable when needed because practices for evaluation, formatting, and feedback of software management information is inconsistent or loosely defined? Why?

15. From your experience, do software requirements, definitions, risk analysis, development planning, preliminary design interface definitions occur during Full Scale Development (FSD) or earlier? Should software design and analysis begin earlier in the acquisition process than it does now? Explain.

16. Do you feel that hardware is usually initiated so early that software is forced to accept changes to relieve hardware difficulties even without the appropriate engineering and design? Explain.

17. Do you believe that software is so different from hardware, that hardware management approaches, techniques, and procedures will not work for software? What aspects of hardware and software development can be considered alike? Why?

18. Can most hardware problems be solved by changing software? What are the implications? Is this good or bad?

19. Does management of ECS software acquisition use a total systems approach for hardware and software combined? Should it now?

20. Do you feel that hardware design drives and limits software alternatives? Should more tradeoffs be made?

21. Should software be designed first and hardware designed or acquired off the shelf to match it?

22. Look at the separate list of "Recommendations for Improving Software Acquisition Visibility." In what order of importance would you place these suggestions?

23. Concerning these recommendations, do you:

    a. Completely agree.

    b. Completely disagree.

c. Feel it needs improvements - what changes?

d. Incomplete - what additions?

24. Do you have any suggestions on how to improve the management visibility of ECS software development?

25. Do you have any general comments on the subject of the interview?

III. DESCRIPTION OF THE SOFTWARE DEVELOPMENT CYCLE

A primary aim of this thesis is to find means of increasing the visibility of software development efforts. Better visibility would provide greater warning indications of development difficulties, help prevent catastrophic schedule and budget overruns, and improve the technical quality of software products. To discuss software visibility problems and improvements, a basic understanding of the process of software development is necessary. The process must be studied in terms of the development functions performed, the resources required to perform the function, and the environment in which the development is accomplished. Each development function varies with program size, complexity, and degree of risk, and these factors along with many others must be considered when analyzing the software development cycle.

The software development process has been described in great detail many times. However, each author on the subject seems to utilize different terminology. For example, mangold defines the software development process with seven steps (Ref 20:2-8):

1. System requirements
2. Software requirements
3. Preliminary design
4. Detailed design
5. Code and debug
6. Test and preoperations
7. Operations and maintenance

Etheredge on the other hand uses a three step description (Ref 15:21):

1. Analysis and design
2. Implementation and test
3. Delivery and maintenance

Wolverton proposed a similar classification (Ref 38:13):

1. Analysis and design
2. Coding and debugging
3. Checkout and test

Wolverton also derived emperical evidence that followed what he labeled the 40-20-40 rule for allocation of software development resources. He stated that 40% of the cost of software development was for step one, analysis and design; 20% for step two, coding and debugging; and 40% for checkout and test. These are only a few of the methods of describing the software development process: a more complete discussion of the applicable literature is reserved for a later chapter.

For the purposes of this thesis the software development process will be discussed in relation to DOD acquisition activities. Watson's model for software development (Ref 37:5-55) is formulated within the five phases of the DOD acquisition model. The normal DOD weapon system acquisition life cycle is defined in Air Force Regulation 800-2, "Program Management," and in more detail in Air Force Regulation 800-3, "Engineering for Defense Systems," and consists of five phases:

1. Concept Formulation
2. Validation
3. Full Scale Development (FSD)
4. Production
5. Operation/Maintenance

Figure 3 shows these phases combined with the primary tasks of both software and hardware development, the related design reviews, configuration audits, baselines, and the four milestones defined by DOD 5000.1 (Ref 30:5).

Hardware and software development programs progress through the system life cycle primarily in the same manner. The big difference is that software acquisition needs no production phase, while going directly from FSD to deployment.

Software development after definition of the hardware and software requirements can be classified in the following general tasks (Ref 30:5):

1. Preliminary Design (Analysis)
2. Detailed Design
3. Coding and Subunit Testing
4. Integration and Testing
5. Deployment

Air Force Regulation (AFR) 800-14, Volume II (Paragraph 2-8) defines a computer program life cycle that is separate and distinct from the traditional DOD development life cycle that includes phases 1 through 4 above. However, it defines step 5 as Installation and step 6 as Operation and Support.

The reviews and audits depicted in Figure 2 (Ref 30:7) are based on the requirements of MIL-STD-1521A with hardware and software usually considered separately. The MIL-STD-483 principles of configuration management and the MIL-STD-490 specification requirements are also reflected in Figure 2.

Watson also relates software development to the parallel hardware development as the total Embedded Computer System progresses through

the development cycle. This approach is very useful for analyzing the development visibility problem because it illuminates the critical decision points encountered in the DOD DSARC environment.

DSARC is an acronym for the Defense Systems Acquisition Review Council. The DSARC body examines a program at the end of a development phase to determine whether more resources should be expended on it and thereby allow it to pass to the next development phase. An unfavorable DSARC decision either cancels the program or leaves it stagnant pending further study.

Traditional DOD ECS Acquisition Life Cycle Phases

A new system is developed in response to a perceived change in the environment. The change could be in a military threat or new technological advances that significantly modify military capabilities. Active systems may even need replacement, but the point is that a requirement must be recognized before a new system concept can be formalized.

The Concept Formulation phase analyzes the perceived need to determine whether or not it should be firmly established. Studies are conducted to determine if the proposed systems are economically or technically feasible and if production can be accomplished in time to satisfy the requirement. During this phase, some exploratory development is often done to estimate the technological feasibility of producing the system (Ref 25:11).

The Validation phase was previously called the project or contract definition phase. The system's performance requirements are defined and a minimum of preliminary design and engineering is accomplished. Major technical approaches are analyzed and some hardware may even be developed. The result of this phase is the contract definition which is required to initiate Full Scale Development (FSD) (Ref 25:20).

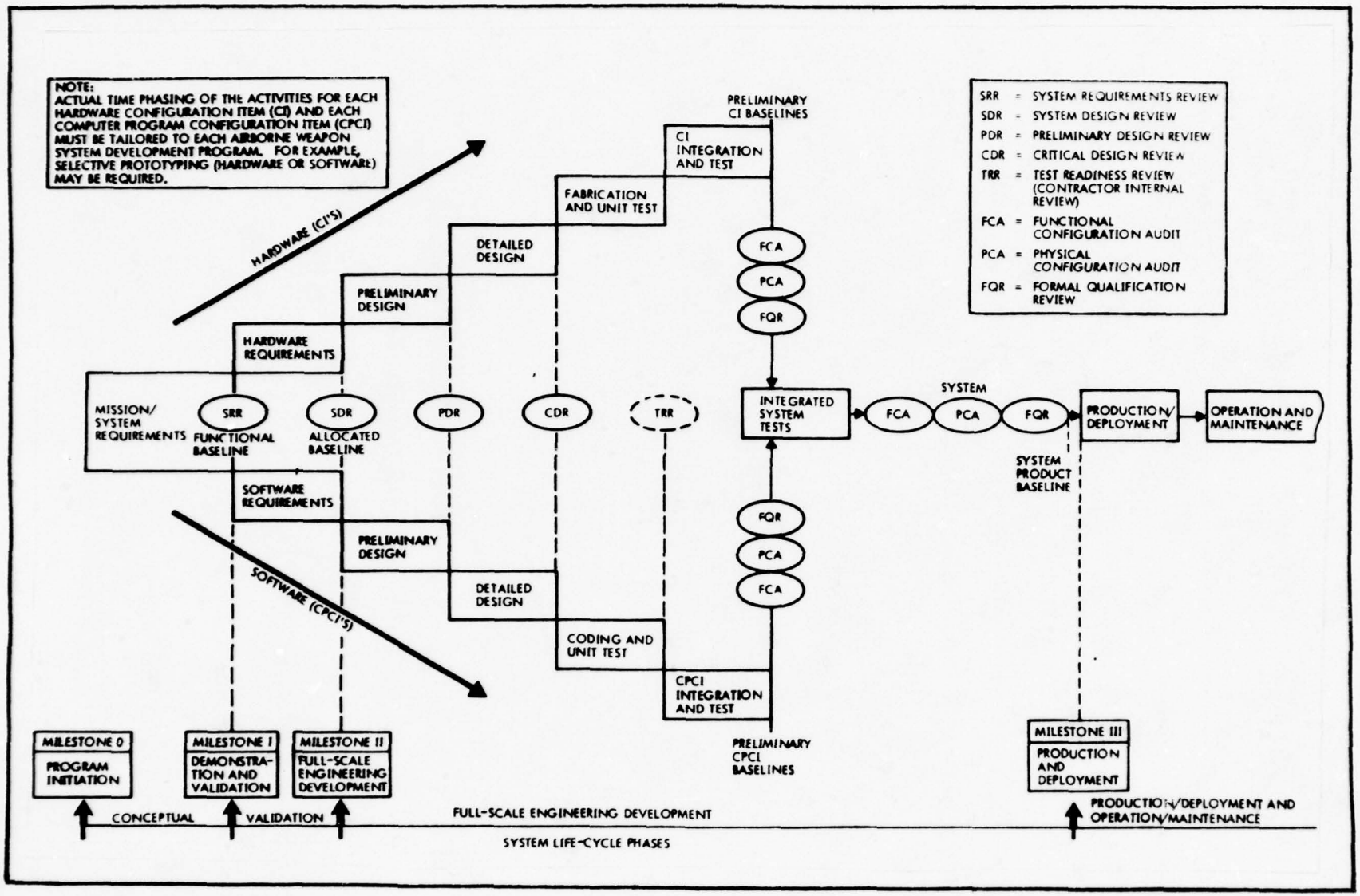


Figure 2. Idealized System Life Cycle (Ref 30:7)

FSD generally includes the design, prototyping and testing of the completed system. This phase is of primary concern to this thesis and will be extensively discussed in a later section.

Production of the completed system can mean many things. Mass production lines are required to produce fleets of aircraft but a few hours of computer time could "produce" enough copies of a software program to issue one copy per aircraft. Acceptance testing can last weeks for aircraft and seconds for a single punched tape copy of a computer program.

The operation and maintenance phase begins when the first system is delivered and considered functional. A statement is then issued announcing the Initial Operating Capability (IOC) for the system. The system is then operated, maintained, and even modified to utilize it over an average 10 to 20 years of operational life. When the system is no longer a cost effective method of satisfying its assigned mission, it is considered for retirement and the life cycle is completed.

A few additional terms should be touched on here before delving deeper into the software development process.

Reliability is one term that does not mean the same thing to software that it does to hardware. Generally, reliability can be defined as the probability that an item will function within specified limits for at least a specified period of time under specified environmental conditions. Hardware reliability is usually described as mean time between failure (MTBF) rates. However, the MTBF concept does not lend itself to describing software reliability. There are several reasons for this. One is that software does not normally fail in the sense that it suddenly stops functioning. Software does not experience physical degradation and

therefore, is not subject to sudden failure (Ref 37:21). Unlike hardware, the older more widely used software is more reliable because through use most errors have been discovered and corrected. The problem is that software will almost always do what the programmer coded it to do, and still may not meet the performance requirements. The Joint Logistics Commanders Software Reliability Work Group (SRWG) defined software reliability as (Ref 36:89):

> " . . . the probability that software will satisfy stated operational requirements for a specified time interval. . ."

The SRWG definition considers software to be reliable even if there are errors, so long as it performs its operational functions satisfactorily. The SRWG also states that (Ref 36:90):

> "There are no quantifiable means at present which can be used to guarantee or measure software reliability."

Unlike hardware there are no imperfections or variations that are introduced by making additional copies of a piece of software other than easily checked copying errors. While hardware is constrained by the laws of physics and can fail due to heat, gravity and other physical phenomenon, software will not. However, if a computer fails the software it contains will cease to function properly. Software interfaces are generally conceptual rather than physical, e.g., there are no easy-to-visualize wires and connectors (Ref 36:47).

When a hardware development manager moves to the software development field, he is forced to reevaluate some of his attitudes and approaches to problems and concepts. The concept of hardware reliability just does not fit software because it is impossible to prove that software is reliable. A manager can only hope to get a feeling for the probability of reliability from documentation of the development and testing process (Ref 37:21).

The DOD has historically treated ECS software as technical data (Ref 17:1). This has caused some problems in the areas of Verification and Validation because of the varied and conflicting uses of the terminology. Validation of computer programs is defined by AFR 800-14 as the process of determining that the computer programs were developed in accordance with specifications (Ref 2). In a different use of the term, Validation is the phase of the project development when the preliminary design and engineering concepts are verified and definite management planning is performed (Ref 25:20). Another type of validation occurs during the development phase just prior to production. Validation at this time certifies that the system complies with its performance specifications (Ref 37:21).

Validation also is used with respect to technical data such as repair manuals when it is prepared as Technical Orders (TO). A TO is a set of instructions for operating and maintaining equipment or performing other tasks that require a standard procedure (Ref 30:15). In this case, validation consists of the contractor proving the accuracy and completeness of the TO document. Verification in this sense is an actual Air Force user checking the TO for clear, sufficient content and compatibility with existing equipment and procedures.

Verification of computer software, on the other hand, is defined by AFR 800-14 as the process of assuring that performance of the required functions in the specified environment is satisfactory (Ref 2). Satisfactory is a key word because of the impossibility of checking all of the potentially infinite number of logical paths. There are many more distinct paths to test in software than in hardware and the software errors generally come without advance warning, provide no period of graceful degradation, and usually do not announce their occurrence (Ref 36:47).

The best possible condition of software verification is merely satisfactory performance combined with documentation to assure that the development was properly executed. However, all hardware circuits can be physically tested and except for future degradation, it can be said that performance has been proven. If a hardware fault is detected and repaired, the system usually is restored to its previous condition. A software correction on the other hand, always changes the previous state of the system (Ref 36:91).

The flexibility provided by software also introduces another problem area. It is sometimes too easy to make changes, and modifications are sometimes made too casually without full consideration of the impacts. Making a correct change is generally very difficult and even harder to test because of the interdependency of the software parts. The main point here is that software is never 100% reliable (Ref 37:22).

Now that the required terms have been discussed and the framework constructed for understanding the software development process, the process steps will be reviewed in greater detail.

Concept Formulation

Concept Formulation is basically the method of deciding whether or not to commit further funds to a proposed system. The formulation phase weeds out the technically impractical and economically unfeasible system proposals and further defines the selected systems. The proposals may be rejected because they do not meet user requirements or are not feasible in terms of cost, technical performance, or schedule. During this phase, analyses and studies are performed to document the necessary data and pertinent information to allow high level decision makers to determine the necessity of a project (Ref 25:11).

To facilitate these high level decisions, reports are prepared and submitted stating the options and relevant factors. The Decision Coordinating Paper (DCP) is one type of report in this category. DCP's are required for each DSARC review, are limited to 20 pages each, and are labeled DCP I, DCP II, and DCP III respectively. A DCP is required by DOD Instruction 5000.2, Paragraph IV.A.2, to document the important information concerning the system and its status. The factors covered would include the need, the threat, concept, milestones, and unresolved issues (Ref 25:9).

The major inputs, functions, and outputs of the Concept Formulation phase are as follows (Ref 27:12-20) (Ref 37:24-25):

INPUTS

1. User Requirements (Requirement for Operational Capability)
2. Planning Criteria
3. Cost Estimation Approaches
4. Required Resources vs Available Resources

FUNCTIONS

1. Initial System Definition
2. Evaluation of Technological Alternatives
3. Studies and Comparison of Cost/Benefit of Alternatives
4. Selection of Best Alternative
5. Engineering Refinement and Draft of Functional Specification
6. Planning

REVIEWS and AUDITS

1. System Requirements Review - (SRR)
2. DSARC I(Program Decision)

OUTPUTS

1. Program Description
2. Draft System Performance Specification
3. Preliminary Resource Requirements
4. Preliminary Schedules
5. Preliminary Cost Estimates
6. Preliminary Subsystem Requirements Allocation
7. Program Management Directive (PMD)
8. Draft DCP I for DSARC I
9. Draft Validation Phase Request for Proposal (RFP) and Statement of Work (SOW) - Software Sections
10. Initial Program Management Plan (PMP)

It must be recognized here that this is the ideal situation, the way the acquisition process should work, not necessarily the way it is. Some factors that should be considered as early as Concept Formulation in actual practice are not covered until later phases. The most important of these overlooked items is planning, especially for software development.

It should also be noted that the primary emphasis of this phase is toward the major weapons systems such as the aircraft or missile. The subsystems including computers are reviewed only from a functional standpoint to answer questions regarding subjects such as their technical feasibility. Determination of how to approach performance problems are reserved for the validation phase. If the major system concept is recommended for further expenditure of funds and passes to the Validation phase, the subsystem technical approach to performance requirements is then addressed.

Validation

During the Validation phase, the system performance requirements are allocated into subsystem performance requirements and interfaces between subsystems are defined. The summation of the subsystem performance requirements should then satisfy the performance specified for the total system. First, an analysis is performed to evaluate the technical and economic aspects of the preliminary system requirements. The user's mission and operating environment is analyzed and compared with the preliminary system requirements to determine any deficiencies and the degree of technical risk involved. The total system performance requirements and system definition are then revised to reflect the new performance concepts (Ref 25:20).

After the total system has been reviewed, the individual subsystems undergo the same type scrutiny. At that time, operational, performance, and design requirements and specifications are generated. Planning, cost estimating and scheduling are conducted for each major subsystem as well as the total system. For major weapons systems, the largest part of the validation phase is usually conducted by a contractor with the Air Force merely reviewing and approving (Ref 37:29-31).

The Validation phase is primarily made up of the following (Ref 27), (Ref 37:30-31) and (Ref 30:5-67):

INPUTS

1. Mission Requirements
2. Mission Operational Environment
3. Planning Criteria
4. Subsystem Operating Concepts
5. Cost Estimations
6. Description of Program Characteristics

7. Draft System Specification or System Performance Specification
8. Preliminary Resource Requirements Estimate
9. Preliminary Schedules
10. Revised DCP I and Possible Redirection from DSARC I
11. Budget Authorization/Program Authorization (BA/PA)
12. Revised PMD

FUNCTIONS

1. System Requirements Analysis (SRA)
2. Development Planning
3. Contract Planning
4. Detailed Cost Estimating
5. Trade Off Studies
6. Interface Planning
7. Revise System Specification
8. Revise PMP
9. Review Validation Phase Contract RFP and SOW for Software Items
10. Draft Software Development Specification
11. Draft Preliminary Software Test Plan
12. Draft CRISP
13. Prepare Draft DCP II for DSARC II

REVIEWS AND AUDITS

1. System Design Review (SDR)
2. DSARC II (Ratification Decision)

OUTPUTS

1. Revised System Specification
2. Preliminary Subsystem Design and Development Specification
3. Revised Schedules
4. Preliminary Test and Integration Plans

5. Trade Off Study Reports
6. Detailed Cost/Benefit Study Reports on Alternatives
7. System Requirements Analysis (SRA)
8. Operation and Development Plans
9. Detailed Cost Estimates
10. Draft CRISP
11. RFP and SOW
12. Signed Validation Phase Contract
13. Revised DCP II from DSARC II
14. Revised PMP

At this point, the reader is again reminded of the difference between the ideal ECS development cycle and the development process as it is actually practiced. The preceding discussion has been concerning the ideal Concept Formulation and Validation phases. Compared to the extensive paper studies, analyses, and sometimes exploratory development of hardware, Validation phase software studies are almost non-existent. Software is analyzed only to the point of deciding whether the total system should utilize some software and then to roughly estimate the amount and cost of software.

Both the Rand Study (Ref 13) and AFSCM/AFLCM 375-7 (Ref 3) emphasize the generation of specifications during the validation phase with the DSARC II milestone as the final event in the phase. DSARC II is the decision point as to which programs are funded further and thus proceed to the Full Scale Development phase.

The question that arises here is, who should do the analysis that is required to establish software design and performance requirements? The abstract nature of software must be considered when making this decision because software logic is so much a product of the programmer's

creative imagination and individual mind. Because people look at and approach problems differently it would seem logical to have the software design analysis, cost estimation, and scheduling done by the same people who would do the programming (Ref 37:35).

Dr. Barry Boehm, Director of Software Research and Technology at TRW wrote in Datamation, May 1973, that software emphasis was secondary to hardware (Ref 5:48-59). He stated that software decisions and requirements should be completed before the critical hardware decisions have been made and that 35% of all software costs are for analysis. This means that almost 35% of the software development should be accomplished prior to FSD (Ref 5:57).

The Johns Hopkins University Applied Physics Laboratory (APL) study titled, DOD Weapon System Software Management Study, stated in June of 1975 that (Ref 18:2-3),

> "Despite the implications in the DSARC II review that an adequate design and costing basis must exist, current directives are vague on the formal requirements for the validation phase of the acquisition process. Many software cost overruns that stem from vague and inconsistent requirements could be eliminated by more thorough analysis and reviews of requirements specifications."

The study also recommended that directives and regulations require this analysis and definition of software be performed during the Validation phase. It is important to note that the APL study must have found a lack of software analysis and definition in the Validation phase or the preceding recommendation would not have been made.

The Joint Logistics Commanders Software Reliability Work Group (SRWG) recommended in November 1975 that (Ref 35:91),

> "Comprehensive policies should be developed and emphasized to ensure that the same attention is given to software requirements analysis, planning and design as hardware during the conceptual and validation phases of system development. Such policies should ensure that software is addressed in ROC's, SOR's, and DCP's and all other appropriate planning documents and enforced through system design reviews."

The three acronyms ROC, SOR, and DCP are acronyms for Requirement for Operational Capability, Statement of Operational Requirements, and Decision Coordinating Papers, respectively.

DOD Directive 5000.29, "Management of Computer Resources in Major Defense Systems," dated April 26, 1976, stated that (Ref 12:2),

> "Validation of computer resource requirements, including software, risk analyses, planning preliminary design, security where applicable, interface control, and integration methodology definition will be conducted during the Concept Formulation and Program Validation Phases of Defense System Development, prior to Defense Systems Acquisition Review Council (DSARC) II."

This directive also required that defense ECS resources, including both computer hardware and software be specified and treated as configuration items. Configuration item status places a high level of management attention and control on ECS development efforts which previously could be designated configuration items or critical items (Ref 12:2). A critical item was on a much lower level of management attention and control, and was therefore much less visible. A primary difference is that a configuration item development specification is an input to FSD while a critical item development specification is a product of FSD (Ref 37:37).

In addition, DOD Directive 5000.29 requires that a computer resource development plan (CPDP) be written prior to DSARC II and be maintained throughout the system life cycle (Ref 12:2-3). The above mentioned items were normally accomplished for ECS hardware prior to DSARC II but usually not for software (Ref 37:38). These are a few examples of important changes

in the policies and directives that apply to software.

Hardware and software tasks during the Validation and Development phases do not run parallel in spite of new directives and policies to the contrary. This non parallel development characteristic was supported by Watson's study which provides the graph of Figure 3 (Ref 37:39).

The termination of the Validation Phase is the Ratification Decision known as DSARC II. This milestone is intended to judge the adequacy of the resultant system and to reassess the continuance of the system development. An adverse Ratification Decision would end the program while a favorable decision would allow program passage into the FSD phase. Program termination could result from inadequate Validation phase products, discovery of insurmountable technical problems, excessive costs, or a reduction in the operational requirements that first called for the system (Ref 25:35).

Full Scale Development

The purpose of the Full Scale Development phase is to produce a working prototype of the major system and then test to verify that the prototype meets the functional and performance requirements. During this phase technicians are trained to operate and maintain the system while the documentation is generated for use in production and deployment (Ref 25:36).

The FSD objectives include completing system design, resolving important issues, and completely testing the prototype and its subsystems. It is interesting to note that FSD yields the initial operational versions of the ECS software, not a prototype. Prototype in this case means a pre-production system that is identical to the production system in form, fit

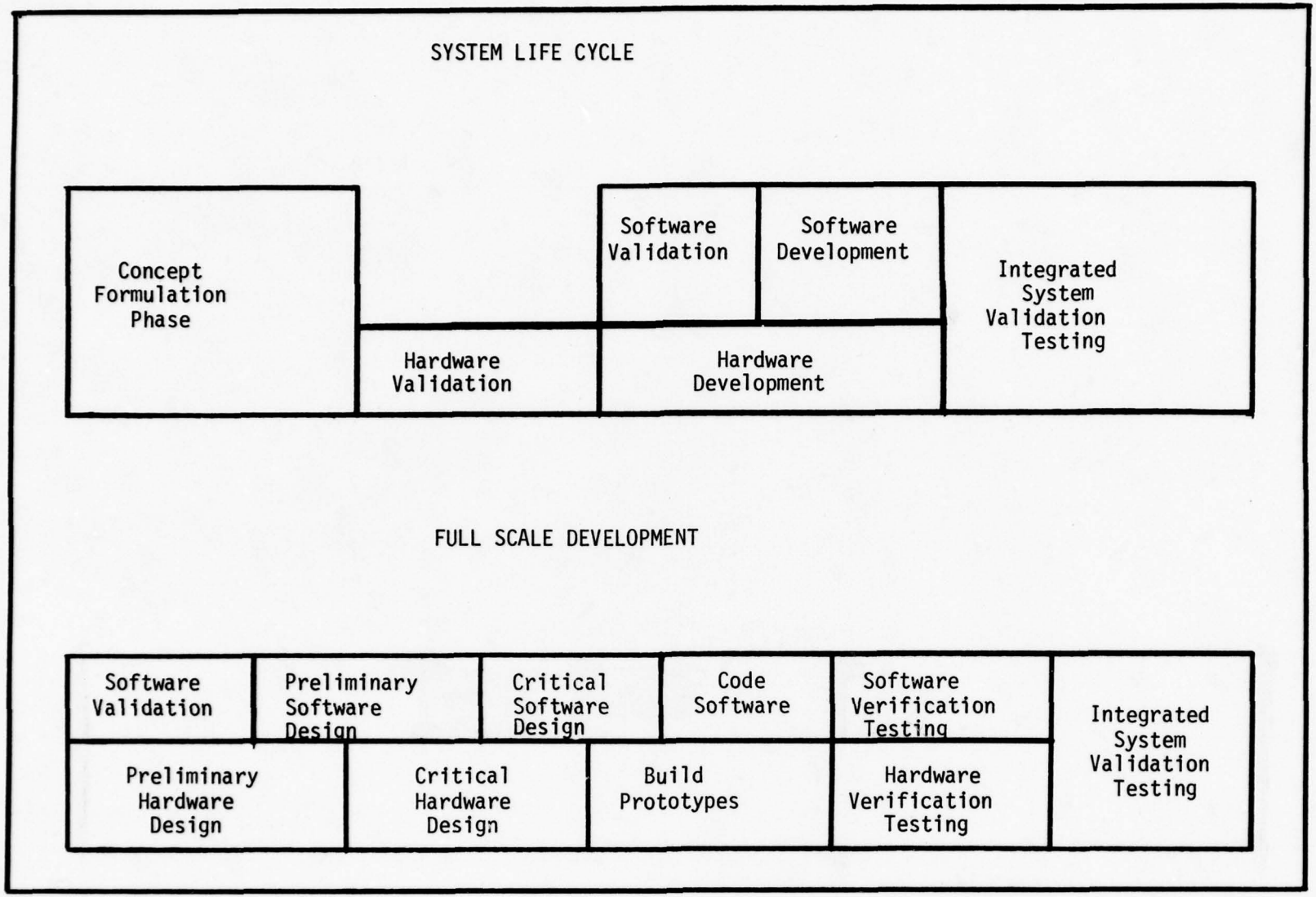


Figure 3. Task Relationships During Fall Scale Development (Ref 37:48)

and function, but usually differs in other ways such as breadboard versus fully qualified printed circuit boards (Ref 25:36). Favorable ratification by the Secretary of Defense at DSARC II begins FSD but may also redirect certain system goals, schedules, allowable costs, or other factors.

The software development itself can be classified into five stages (Ref 37:41) (Ref 30:7):

1. Development Tool Building
2. Preliminary Design (Leading to PDR)
3. Detailed Design (Leading to CDR)
4. Coding and Debugging (Software Testing)
5. Complete ECS (Hardware and Software) Integration and Test

Software development tools include assemblers, compilers and simulators, to mention a few of the specialized programming tools that may be required.

Trade offs and analyses are utilized during the preliminary design stage to determine alternative programming approaches and then to select the best method. Functional flows are generated, memory is allocated, programming tasks are allocated and generally high level software work is accomplished (Ref 20:143).

A formal preliminary Design Review (PDR) is held at the end of this stage. The PDR evaluates the basic design for completeness, adequacy, and compatability thru briefings, discussions, and analyses to decide whether the software design is ready to progress to the detailed design stage (Ref 33:37).

During the detailed design stage, those design activities are accomplished which are necessary prior to the actual coding of software. High level software system flows are finalized and lower level detailed

subunit flows are generated. The Computer Program Development Specification (Part I) is reviewed and the draft Computer Program Product Specification (Part II) is generated. Preliminary test plans and procedures are finalized and submitted for approval.

The Critical Design Review is held at the end of this stage to ensure the design is complete enough to start actual coding. During this coding the functional flow charts are converted into lines of software instructions usually on a module to module basis. These modules are then checked manually for errors and then further checked with software diagnostic programs. When enough modules have been checked individually, they are compiled and assembed into larger units of software code. The process goes through many iterations until the total software program has been compiled and tested. At this point formal integration and testing begins by combining the software with all the associated hardware (Ref 33:41).

The testing stage is made up of Verification and Validation (V & V) testing. Validation testing reviews hardware and software separately and compares them with the appropriate requirements specifications to determine whether those requirements have been met. After the hardware and software have passed their separate validation tests they are integrated and submitted for Verification testing which compares the operation of the combined ECS with the user's performance requirements (Ref 37:41-47).

A Functional Configuration Audit (FCA) is used to verify the successful conclusion of testing. The FCA checks and documents that performance requirements have been satisfied and thereby qualifies the system for production. A Physical Configuration Audit (PCA) is held to identify the products and then to verify that the technical documentation is a realistic and complete description of the FCA qualified product (Ref 33:45-51).

A Formal Qualification Review (FQR) is sometimes held after the PCA. When possible the FQR and FCA are combined, but if the FCA does not fully satisfy all testing requirements of the Part I Development Specification the FQR is held. For this reason, the FQR is considered an extension of the FCA.

The DSARC III milestone makes the Production and Deployment decision that ends the FSD phase and allows the program to pass into the Production Phase or cancels it.

FSD of software is generally made up of the following inputs, functions, reviews, audits and outputs (Ref 30:7-63), (Ref 25:44), and (Ref 37:41-44):

INPUTS

1. Revised Program Management Directive (PMD)
2. Revised Decision Coordinating Paper II (DCP II explaining Ratification Decision of DSARC II and any redirection of goals or constraints)
3. Budget Authorization/Program Authorization (BA/PA)
4. CRISP
5. Computer Program Development Plan (CPDP)
6. Computer Program Configuration Management Plan
7. Draft RFP and SOW
8. System Specification

FUNCTIONS

1. Definition of Inputs/Outputs
2. Allocation of Software Tasks to Software Subunits
3. Generation of Functional Flow Diagrams
4. Allocation of Memory
5. Update of Schedules and Cost Estimates
6. Generation of Initial Test Plans

7. Interface Definition (including signal formats)

8. Timing Requirements

9. Revise Software Portion of Contract RFP and SOW

10. Revise CRISP

11. Revise CPDP

12. Review Need for Independent V & V

13. Revise System Specification to Reflect Changed Software Requirements

REVIEWS and AUDITS

1. Preliminary Design Review (PDR)

2. Critical Design Review (CDR)

3. Functional Configuration Audit (FCA)

4. Physical Configuration Audit (PCA)

5. Formal Qualification Review (FQR)

6. DSARC III (Production and Deployment Decision)

OUTPUTS

1. Final Computer Program Development Specification (Part I Specification)

2. Test Plans and Procedures

3. Flow Charts

4. Input/Output Formats

5. Source Program (listings)

6. Object Program (machine language)

7. Draft Computer Program Product Specification (Part II Specification)

8. RFP and SOW

9. Signed Contract

Production

The primary objective of the Production phase is to produce and deploy in good working order all of the planned duplicates of the weapons system.

During this phase, the version of the software program qualified during FSD is duplicated and accepted by the Air Force usually in a very short process of copying punched tapes or other machine readable media and checking for errors. This trivial duplication and acceptance operation is often completed in only a few days. During FSD the acceptance tests are specified to demonstrate compliance by hardware and software with production requirements. The computer hardware is produced in a more orthodox manner with items manufactured sequentially over a long period of time and accepted on an individual basis. When the hardware and software have passed their acceptance tests they are passed to the operational user.

A favorable DSARC III decision initiates this phase and may redirect factors such as quantities, cost and schedule thresholds. Program Management Responsibility Transfer (PMRT) from Systems Command to the using command, on the other hand, terminates the Production Phase. The occurrence of PMRT begins the Deployment Phase which is otherwise known as the Operation and Maintenance Phase.

Operation and Maintenance

The first production unit is deployed to the operational environment where it undergoes a series of tests known as Initial Operational Test and Evaluation (IOT&E). The purpose of IOT&E is to guarantee that the system as delivered, satisfies the user's requirements in the operational environment under real mission conditions. Satisfactory completion of IOT&E results in a formal statement of the Initial Operating Capability (IOC).

The operation and use of an ECS is the performance of the assigned mission on a regular basis. The effective use of the system is the prime

objective of this phase and requires that it be maintained efficiently until it is replaced, retired, or consumed by war or accident. A discussion of hardware and software maintenance must be treated separately because of their unique differences.

Hardware mechanically wears out, electronic components fail, and degradation occurs with operating time and usage. Components change electrical characteristics and need adjustment and alignment. Test equipment, TO's, and spare parts are required to bring the equipment back to the physical condition it was in prior to the failure.

Software does not fail in the same way as hardware and does not degrade with use. Software is consistent and will perform in a given scenario, the same way every time. Software errors are constant and will not go away without program changes. Software failures during operational use are merely the recognition of an error that had been in the program all along (Ref 37:54).

In the case of software, maintenance consists of investigating possible software errors and devising corrections or ways to work around the problem. Modification of software would be changing it to meet altered operational system requirements, or performing requested improvements. Both maintenance and modification require retesting and checkout of the entire computer program and are actually redevelopments through the FSD steps of design, code and test (Ref 25:45).

Level-of-effort contracts are primarily used for both maintenance and modification and too often are informally managed. These after the fact software development efforts should, however, be contractually as well defined as during FSD. The Air Force on the other hand, may decide

to maintain and modify the software internally rather than with contractor personnel. If this "organic maintenance" option is chosen, the necessary specialized support tools and software should be acquired and developed during FSD. Too often these support tools were developed by the contractor during FSD but the government did not acquire "rights" to these items in the original contract Statement of Work (SOW) and does not get them. Later, if an organic maintenance decision is made by a tardy CRISP these special tools such as assemblers, compilers, editors, simulators and emulators must be purchased at a significant cost increase (Ref 25:46).

IV. LITERATURE SEARCH

One of the major goals of this thesis was to review the available literature in the area of software management visibility and to combine that information into a set of recommended improvements to increase management visibility and control of avionic software development. There is a substantial body of reputable sources, largely from DOD sponsored research, that provide a solid foundation on which to build the set of recommendations. These sources contain a large amount of factual material and expert opinion on most aspects of system software development. A large portion of the literature reviewed for this thesis was referenced previously during discussions of both software acquisition problems and the DOD software development cycle. The remaining literature that is deemed relevant will be covered now. For the sake of organization and understanding, the results of the literature search will be treated in three sections:

1. DOD Sponsored Studies and Guidebooks
2. DOD Regulations, Specifications, Standards, and Manuals
3. Thesis, Reports, and Periodicals

DOD Sponsored Studies and Guidebooks

To complete the discussion of the ECS software acquisition process, some coverage of specific problems must be made. The "DOD Weapon Systems Software Management Study" performed by the Applied Physics Laboratory (APL) of Johns Hopkins University offers a "road-map" of critical problem areas encountered during the software life cycle. This road-map is shown in Figure 4 (Ref 19:1-3) and identifies 55 interrelated categories of problems.

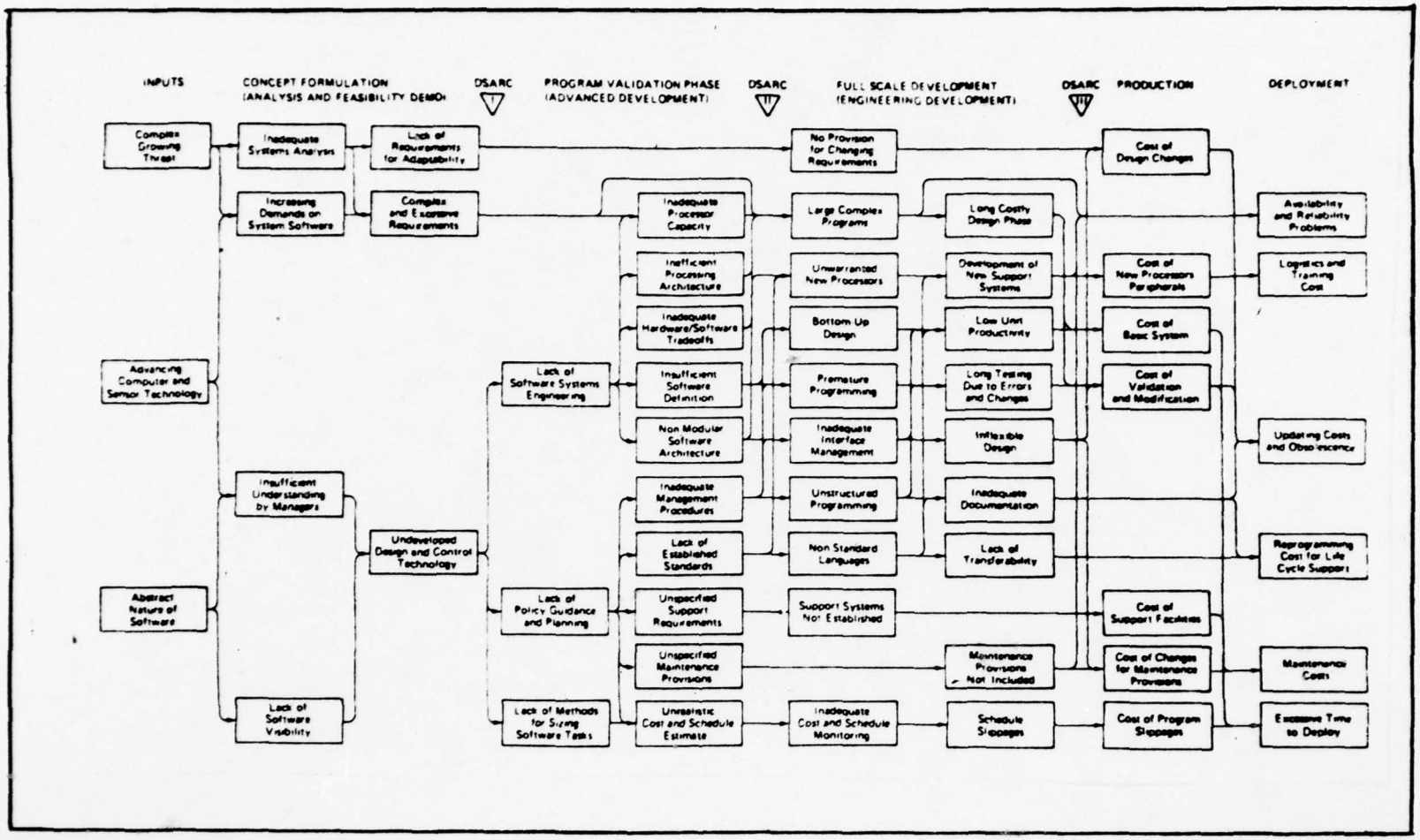


Figure 4. Interrelation of Software Acquisition Study Findings (Ref 19:1-3)

Each acquisition phase has its own distinct problems and each problem has a cascading effect on later problems. Most of the literature encountered deals with the interrelationship of these problems and recommendations for their solution. In the Defense Management Journal, October 1975, Deputy Assistant Secretary of Defense Jacques S. Gansler summarized the problems listed in the APL study by stating that the most critical ECS software acquisition problems are (Ref 17:1):

1. Increasing weapon systems dependence on software without adequate management methods to control costs.
2. A lack of software Research and Development programs.
3. A need for improved management expertise and attention in relation to ECS software.

Much has been written about software acquisition problems and the lack of management ability to control or even monitor development progress. The APL study expresses this concern well by stating (Ref 19:2-4),

> "The lack of software visibility, as compared to hardware, in the acquisition of major subsystems is generally agreed to contribute to the fact that it is not well managed. This acquisition management problem, in turn, results in numerous sins of omission throughout the development process that result in unrealistic cost and schedule estimates, inadequate configuration management, and related problems."

The reviewed literature reveals the feeling that only recently has software been considered as an important subcomponent of major weapons systems. The heightened status of software is no doubt related to the increased cost of software relative to hardware as evidenced by Figure 1.

Mr. Gansler, speaking as the former Deputy Assistant Secretary of Defense in January 1978, listed the five major observable software management problems as (Ref 34:6):

1. Excessive Development Costs
2. Excessive Maintenance Costs
3. Schedule Slippages and Delays
4. Excessive Errors/Faults
5. Duplication and Lack of Standardization

He then itemized in simple form what he considered to be the underlying causes (Ref 34:6):

1. Lack of early management visibility
2. Lack of management discipline (professional expertise)
3. Lack of life cycle perspective
4. Lack of sufficient control over expenditures
5. Lack of hardware/software tradeoffs
6. Lack of standardization
7. Software treated as data

In conclusion, he cited the following areas for improving the DOD software acquisition process (Ref 34:9):

1. Increased emphasis on requirements analysis and validation
2. Increased emphasis on resource acquisition planning
3. Increased emphasis on software configuration management and control
4. Increased emphasis on cost estimation and sizing
5. Increased emphasis on personnel development and training
6. Improved procurement practices

Malcolm Davis of the Rand Corporation made several pertinent points in a paper submitted to the Software Reliability Work Group (SRWG) of the Joint Logistics Commanders Electronics Systems Reliability Workshop:

"Much of the approval chain in computer systems (both procurement and R&D) is made up of people who are not up to speed in contemporary computer business and are not sufficiently supported by people who are (Ref 36:39) (Ref 1:85)."

"There is an insufficient number of skilled software workers in research, technique development and practice (Ref 36:40) (Ref 1:92)."

The development of software personnel with the expertise to satisfy DOD requirements is frequently mentioned as a high priority action item. The recommendations of the SRWG devoted 8 of its 45 points to the acquisition and development of software professionals within the DOD. It pointed out that the application of engineering disciplines to the decision and management of software resources is emerging as a systematic and useful activity. Because of its infancy, however, those software development practitioners are largely self-taught, with varied experience levels and backgrounds, and practice very little uniformity of approach (Ref 35:100). This software discipline must be formally established with a scientific basis and then included in educational programs to furnish the necessary trained people.

Also, there is no written guidance or body of knowledge that facilitates the transfer of software development expertise or the exploitation of lessons learned. As people become technical experts in the area of software development, they are too often promoted into management, taking with them the knowledge they have built up through trial and error. In the dynamic area of software development, this is a costly oversight, because of its extreme knowledge orientation (Ref 35:30).

A professional profile of what makes a good software engineer and manager must be defined and classified by education and experience requirements for levels of proficiency (Ref 35:101). Career paths with accompanying career incentives must also be developed to foster the

development and retention of professional ECS software engineers and managers.

This lack of knowledgeable people is a common theme in the software literature that is compounded by a general lack of understanding of the software development process. The abstract nature of software and the relatively undeveloped methodologies for system engineering of ECS systems makes it very difficult for program management to properly direct the software acquisition process (Ref 19:2-18).

The DOD has received much unfavorable attention in recent years over the excessive costs of defense weapons systems. These excessive costs have been in part due to the mismanagement of programs and cost overruns with software getting its share of the infamy.

Dr. Currie, the Director of Defense Research and Engineering wrote to the Service Secretaries in March 1974 (Ref 10),

> "It is becoming increasingly apparent that an area which could provide us with substantial payoff for successful R&D investment is that of computer software. In 1955, our computer system costs were about 85% hardware and 15% software. A Rand forecast predicts that by 1985 we will be spending 5% of our ADP budget for hardware and 95% for software. The Assistant Secretary of Defense (Comptroller) and I agree that our software development strategy deserves a careful review based upon this trend."

This need to improve software acquisition moved the OSD to go one step further, and wrote to the Assistant Service Secretaries again in December 1974 notifying them of a two phase plan that was being implemented.

> "The sharply rising costs of software programs in the weapon system acquisition process, with respect to acquisition procedures, development and maintenance of such software, and the increasing importance of the software role in the overall mission effectiveness of major DOD weapon systems constitute serious technical and management problems that must be solved if we are to have the weapon systems that are needed for our national security. To find solutions to these problems, we are initiating a two phase study program which will require the joint involvement of the OSD staff and the Services."(Ref 23) (Ref 19:R5-R6).

The two phase study referred to above consisted primarily of the APL study cited previously and a MITRE Corporation study referred to as the "DOD Weapon System Software Acquisition and Management Study." The MITRE results were reported to the DOD in June of 1975 and are summarized below as "Major Observations" (Ref 26), and (Ref 39:9-10).

1. A major contributing factor is lack of discipline and engineering rigor consistently applied to software acquisition (Ref 26:xi).
2. Good management practices are often available but are not always followed (Ref 26:xii).
3. The acquisition process does not always recognize the need for early and complete emphasis on software resources. There are major differences in application of management practices between hardware and software (Ref 26:xii).
4. Software indirect costs are often much greater than the direct costs (Ref 26:xii).
5. There is a lack of consistent practices for feedback of software management information. Meaningful cost and management information for most system developments is not readily available (Ref 26:xii).
6. Many weapon systems software problems are similar to problems in other types of software development, e.g. Automatic Data Processing (ADP).

The same MITRE study also identified four areas as "High Payoff Areas" (Ref 26:2-13):

1. Software Performance Specifications - The establishment and consistent application of sound engineering principles and practices to the process of specifying and validating software requirements.

2. Software Acquisition Planning - Early and complete software life cycle planning. The establishment and application of management practices and strategies designed specifically for software.
3. Software Technology - High leverage technology progress is needed to further improve software practices and development techniques.
4. Personnel - Provisions are needed to develop and retain experienced DOD software management and software engineering personnel.

The APL study arrived at separate but similar conclusions and centered its effort on seven categories of recommendations (Ref 19:2-1),

1. Management Policies
2. Acquisition Planning
3. Systems Engineering
4. Implementation Procedures
5. Program Management Support
6. Acquisition Management Standards
7. Development of Tools and Techniques

Several of these categories are of interest to this thesis and will be discussed here. The area of management policies includes several issues, the first which is the analysis and validation of ECS requirements. Initial requirements are often excessively ambitious and require changes largely because the initial requirements were not critically analyzed and validated through a program of advanced development or system definition. The problem of frequent requirements changes during development is further complicated by the impact of changing technology and the difficulty of obtaining usable cost data (Ref 28:144).

The DSARC II review implies that an adequate basis must exist for design and costing of software. Current directives are vague, however,

on the formal requirements for the validation phase of the acquisition process. Software cost overruns arising from vague and inconsistent requirements definition could often be eliminated by more complete analyses and reviews of requirements specifications prior to FSD (Ref 19:2-3).

Another area of potential management policy improvement that would increase visibility and understanding of major ECS software components would be to place them on a par with hardware components. Policies should consider software a Configuration Item to be delivered on a schedule, controlled and reviewed, rather than view it as an item of data. This will require a change in software policy documents and in the Armed Services Procurement Regulations (ASPR) (Ref 19:6-9).

A particular document that requires modification is Appendix B of MIL-STD-881A. The appendices describe the levels of the Work Breakdown Structures (WBS) for seven types of systems such as missiles, aircraft and electronics. Computer software is addressed only as an item under electronics systems along with sensors, communications, data displays and auxiliary equipment. ECS software should at least be placed equal to major subsystems and not relegated to a subhead under electronics systems.

The WBS for software is of particular importance to allocation of software costs which has been a continuing problem area. If software is not sufficiently high in the WBS, costs cannot be controlled or tracked by management information systems. This type of allocation of costs to the work unit is required to understand how resources are expended and achieve visibility into the development process (Ref 19:6-10).

The area of acquisition planning brings up two recommendations for improved software visibility. The abstract nature of software makes the measurement of development progress very difficult. Therefore, it is

important to formalize the steps in design implementation and test with policy guidelines. The lack of formalized steps leads to difficulties in interface management and to the late discovery of inadequate requirements definitions or design errors resulting in schedule slippages and increased costs (Ref 19:2-6).

The requirements for FSD milestones should be more clearly defined to guide software acquisition through the proper sequence of analysis, design, development, integration, and test. The milestone definitions should include criteria that would be used to demonstrate that each milestone had been achieved (Ref 19:6-13).

The actual definitions of the milestones and the level of design control required should be allowed to vary from system to system. Variations would allow flexibility to adapt the guidelines to the unique requirements of each developing system. There are a number of management and documentation standards which refer to development milestones. None of these standards, however, define a clear set that is well suited to software acquisition (Ref 19:6-14).

The Air Force Source Selection Document (SSD) Exhibit 61-47B was issued in April 1966 and supposedly superseded by MIL-STD-1483. It is still used, however, as a basic reference on computer program development milestones apparently because no other document covers the software acquisition steps nearly as well. This exhibit should be updated to provide an acceptable basis for current procurement activities by including test events and by including document and event related milestones. These milestones should be established in the Request for Proposal (RFP) and would provide a foundation for program planning and evaluation of proposals. Also, each milestone should require a specific deliverable product.

The B-1 program included SSD-61-47B in the development contracts because current Acquisition Management regulations, such as MIL-STD-490, MIL-STD-483, and Air Force Regulation 800-14, Volume II cite milestones but do not clearly define the required work or the products to be delivered (Ref 19:2-6).

The System Development Corporation (SDC) has emphasized another area of acquisition planning which needs increased attention. The Computer System Resource Development Plan (CSRDP) is the most important single management document, according to SDC, who recommends that it be specifically required for major software development efforts. The CSRDP is generated to ensure that software development is well organized and managed, and that all requirements are correctly defined as understood prior to FSD. The development plan explains the contractor's approach to engineering and management issues and therefore could aid in selecting the proper development contractor (Ref 19:6-17).

The Air Force has come closest to requiring a CSRDP by detailing requirements for a Computer Program Development Plan (CPDP) in AFR 800-14, Volume II. The CPDP covers such factors as organization, management controls, design, test, milestones, status monitoring, support, documentation, and engineering practices (Ref 19:2-7).

The lack of application of systems engineering methodology to the development of software leads to a number of major problems. Computer systems are too often considered as hardware alone with software design impacts addressed only after hardware design is final. This failure to consider the total system, both hardware and software together, has caused many costly design mistakes (Ref 36:40). During the program validation,

tradeoff analyses must be performed for hardware versus software approaches to design issues with appropriate decisions made at SDR and DSARC II (Ref 19:16-21). The final formulation of software requirements should occur during the advanced development phase. For many systems this phase has concentrated on design and demonstration of hardware subsystems or components, it being assumed that demonstration of software design at this stage is not necessary. If these elements are not given full management attention from the outset, penalties appear almost certain in terms of increased costs, delayed delivery, and compromised performance (Ref 6:98).

The large variation in the requirements and organizations of development programs demands a great degree of flexibility in the application of management standards and procedures. It is not desirable to develop standards and directives to govern all aspects of software acquisition. Program managers must be allowed to direct their programs in the most appropriate and cost-effective manner that is consistent with DOD Directive 5000.1 and the unique requirements of the particular program. However, the abstract nature of software discussed earlier and the lack of strong systems engineering methodology requires that the SPO managers have access to an organized body of knowledge to guide them. The Air Force program to develop software acquisition management guidebooks is a result of the APL study recommendation and was implemented by the DOD Weapon System Steering Committee (Ref 19:6-49).

In 1974 the Assistant Secretary of Defense (Installations and Logistics) and the Joint Logistics Commanders of all services established in the Weapon System Steering Committee a joint committee to attack ECS software acquisition problems. The committee issued a "Capstone Directive" which stated policies and principles for future software management directives (Ref 35:105). This

directive was the first step in a massive overhaul of directives, regulations and standards dealing with ECS software. The overhaul was structured to correct inconsistencies and to coordinate future publications concerning ECS software (Ref 37:9).

The directive also committed the DOD to:

> "Prepare and maintain guidelines, checklists, handbooks, and examples covering development, acquisition, operation and support (of ECS software)" (Ref 35:106).

This was interpreted to be a set of guidebooks to lead the software practitioner and the project managers. The MITRE study proposed a similar series of guidebooks, which are starting to appear. Electronics Systems Division (ESD) and Aeronautical Systems Division (ASD) of Air Force Systems Command are each publishing guidebooks.

The ASD versions are called Software Acquisition Engineering (SAE) Guidebooks for Airborne Systems. Many of the SAE books have already been published and appear to go a long way toward supplying the software development engineer and manager with usable acquisition information. The ESD guidebooks are referred to as the Software Acquisition Management (SAM) series and are primarily designed for SPO personnel in the software acquisition management area. All of these documents are considered "living" documents in that they will be revised as needed to reflect the true state of affairs and to provide the type of current information the users want. These guidebooks appear in this author's opinion to be the best attempt to date to provide for visibility into the software acquisition process. These guidebooks aid the engineer and manager by helping him ask for the right information at the right time and by helping him require that activities be accomplished in a timely and properly sequenced manner. This approach allows the manager to apply only the guidance that is appropriate to his program

and yet requires the necessary rigor and discipline to improve software quality. This type of systems methodology provides for effective management control of software development.

Several models and techniques have been developed to estimate software acquisition efforts and costs. One of the more promising is the RCA Price model which was analyzed by Schneider in his AFIT thesis entitled, "A Preliminary Calibration of the RCA Price Software Cost Estimation Model." The Army has developed a macro-model for estimating the manpower and time required for getting a software program operational. This system gives a manager the data he needs in terms he can use and understand.

Acquisition management standards have been recommended to establish a common set of software development requirements and criteria to be applied across all the services. This becomes more complex when considering the recent efforts of all services to increase the uniformity and control of their software development programs by issuing new standards and directives. The proliferation of new regulations contains only one comprehensive document, AFR 800-14, that deals with software acquisition management. This regulation should be the basis for any tri-service document covering the required software acquisition procedures and approaches (Ref 19:6-48). That brings us to Section II of this Chapter, DOD Regulations, Specifications and Standards.

DOD Regulations, Specifications, and Standards

The majority of the regulations, specifications and standards that apply to ECS software acquisition were covered in Chapter III as they apply to the software development cycle. The coverage here is to clarify and further describe the most important of those documents.

The SRWG, "Chairman's Report to the Joint Logistic Commanders," further supports the premise of a lack of managerial understanding of software by stating that,

> "We have generated in the DOD, a large number of regulations, directives, and military standards for systems acquisition management. The vast majority of the procedures outlined in these documents are not tailored for software. Software considerations have been added to some of them after the fact, but they are still really hardware oriented. The result is that they conflict with each other, use non-standard terminology... We will have to rewrite all of those regulations, military standards and directives so that they are consistent with policy and with each other (Ref 35:29)."

The most important regulation affecting ECS software acquisition is AFR 800-14, Volume II, entitled "Acquisition and Support of Computer Resources in Systems." This regulation along with Volume I, and AFSC Supplement 1 to Volume I, provide guidance for planning, development acquisition, use, and support of computer resources in defense systems. Computer systems are the only commonly used components of Air Force weapon systems whose development is addressed in a separate regulation. This is partially because computer systems, especially software, are usually minor portions of a total weapon system in terms of expended resources, and therefore, too frequently receives insufficient management attention. Also, computer technology is new and not well understood and is too often on the critical path of procurement efforts (Ref 30:17).

Volume I of AFR 800-14 provides the basic management policies for the acquisition and support of computer systems while Volume II presents the concepts and procedures necessary to implement the policies of Volume I. Volume II defines four major plans and directives (Ref 30:23):

1. Program Management Directive (PMD) - "The official HQ USAF Management directive used to provide guidance to the implementing and participating commands (AFR 800-2).

2. Program Management Plan (PMP) - "The document developed and issued by the Program Manager which shows the integrated time-phased tasks and resources required to complete the task specified in the PMD (AFR 800-2).
3. Computer Resources Integrated Support Plan (CRISP) - "The CRISP identifies organizational relationships and responsibilities for the management and technical support of computer resources (AFR 800-14, Volume II).
4. Computer Program Development Plan (CPDP) - "The CPDP identifies the actions needed to develop and deliver computer program configuration items and necessary support resources (AFR 800-14, Volume II).

Air Force Regulation 800-2, entitled "Acquisition Management - Program Management," is closely related to AFR 800-14. This regulation applies to all Air Force acquisition programs identified by DOD as major defense systems. That is to say, the total system is guided by AFR 800-2 and the computer portion is guided by AFR 800-14.

DOD Directive 5000.1, 18 January 1977, entitled "Major System Acquisition," is the basis for acquisition of major defense systems, and is implemented via AFR 800-2.

DOD Directive 5000.2, 18 January 1977, entitled "Major System Acquisition Process" defines the policies and procedures used by the DOD in the decision-making processes of acquiring major defense systems. This directive supplements DOD 5000.1 and establishes the Defense Systems Acquisition Review Council (DSARC) charter and is a key directive for acquisition of embedded computer systems.

DOD Directive 5000.29, 26 April 1976, entitled "Management of Computer Resources in Major Defense Systems," is used for management and control of

computer resources during the development, acquisition, deployment, and support of major defense systems. It addresses milestone definition, requirements validation, risk analysis, deliverable software and also charters the DOD management Steering Committee for Embedded Computer Resources. The Air Force implements this directive via AFR 800-14, Volume II.

Air Force Regulation 800-3, 1 June 1976, entitled "Engineering for Defense Systems," helps define the engineering effort that will be applied phase-by-phase throughout the acquisition life cycle. This AFR describes the policies, principles, concepts and techniques required for the efficient planning and control of the technical development program and implements AFR 800-2.

MIL-STD-483(USAF), 31 December 1970, entitled "Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs" applies configuration management requirements and baseline specifications to development contracts. This standard specifies the use of the reviews and audits that are detailed in MIL-STD-1521A as well as Engineering Change Proposal (ECP) guidelines. Appendix VI addresses Computer Program Configuration Item (CPCI) Specifications such as the Part I (development) and Part II (product) Specifications as a supplement to MIL-STD-490.

Overall, this standard specifically addresses requirements of software not found in MIL-STD-480, -481, and -490, such as:

1. Preparation instructions for computer program specifications.
2. Computer program specification and support documentation maintenance.
3. Computer program formatting and change processing.
4. Computer program configuration audit objectives.

MIL-STD-490, 30 October 1968, entitled "Specification Practices," establishes the format and technical content of specifications that are unique to a certain project. It defines uniform practices for specification preparation and ensures the inclusion of essential requirements. Two specification types, B5 and C5, apply directly to software. Appendix VI details the requirements for specification type B5, the Computer Program Development Specification (PART I). Appendix XIII details the requirements for specification type C5, the Computer Program Product Specification (Part II). This standard applies to all services and a revision is in process. The revision is intended to resolve discrepancies that exist between MIL-STD-490 and 483.

MIL-STD-881A, 25 April 1975, "Work Breakdown Structures for Defense Material Items," details the preparation and utilization of Work Breakdown Structures (WBS). Computer software currently appears at WBS level three in the ground support subsystem appendix. However, the appendices covering aircraft, missile, and space systems are currently deficient with respect to identifying on-board software as a level three WBS element. Until this standard is revised to include ECS hardware and software at level three USAF SPO's must improvise their own level three WBS element to guarantee visibility of ECS software development.

MIL-STD-1521A, 1 June 1976, "Technical Reviews and Audits for Systems, Equipments, and Computer Programs," defines the requirements for conducting the following seven milestone events; Systems Requirements Review (SRR), System Design Review (SDR), Preliminary Design Review (PDR), Critical Design Review (CDR), Functional Configuration Audit (FCA), Physical Configuration Audit (PCA), and Formal Qualification Review (FQR). These reviews and audits are used by contracting agencies throughout the acquisition life cycle to

monitor program efforts to ensure contractual requirements are being satisfied. The standard identifies responsibilities and outlines the minimum information requirements. The chronological relationship of the reviews to other program activities is also established and explained.

Thesis, Reports and Periodicals

As was the case with the previous section of this chapter, most of the relevant literature in this category was discussed in Chapter III as it related to the software development cycle. Many authors have added to our understanding of the software development and acquisition process, however, and a few of these contributions will be discussed now.

In reviewing numerous articles, abstracts and research papers, the literature was generally divided into two categories. The first consisted of authors investigating and describing their approach to some technical problem. The research problem areas in this category can generally be summarized by Figure 4. This category is not the primary concern of this thesis and will not be covered further.

The remaining literature primarily deals with various approaches to computer software acquisition management problems and solutions. A partial summary of these articles can be found below and in the Bibliography. The majority of this literature describes the software development process in isolation, unrelated to computer hardware or other weapon system development activities.

Zabriskie, in a paper entitled "Development of Weapon Systems Computer Programs," described the software acquisition process isolated from other development activities by using thirteen steps (Ref 39). He detailed each stage of the development process but did not relate the parallel hardware development activities. Etheredge (Ref 15) and Wolverton (Ref 38) each used

different three step descriptions as discussed in Chapter III, but neither related their steps to the computer hardware development process or to the DOD weapons systems acquisition phases of Concept Formulation, Validation, FSD, Production, Operation and Maintenance.

Nelson, however, described a six-step acquisition process for Automatic Data Processing (ADP) software, which is covered by the 375 series of regulations. This type software is not embedded in another system, but is a complete unit in itself, such as a personnel or finance software program. In his report entitled, "Management Handbook for the Estimation of Computer Programming Costs (Ref 27)," he related the six steps to the DOD acquisition process, for ADP software. This does not directly correspond to ECS software development and does not relate the parallel computer hardware development.

Watson was a notable exception and was quoted extensively in this thesis. His thesis entitled, "Acquisition of Embedded Computer Software: A Descriptive Model (Ref 37)," used three steps to describe the ECS software development process and related the parallel computer hardware and software development activities and referenced these to the five traditional DOD acquisition phases.

In summary, the software acquisition process was described by the literature with from three to thirteen separate and distinct phases. Some of the more significant versions are listed by author and number of phases:

1. Etheredge --- three phases (Ref 15)
2. Wolverton --- three phases (Ref 38)
3. Driscoll --- three phases (Ref 14)
4. Watson --- three phases (Ref 37)
5. Merwin --- four phases (Ref 24)
6. Capps --- five phases (Ref 9)
7. Nelson --- six phases (Ref 27)

8. Mangold --- seven phases (Ref 20)

9. Mathis and Willmorth --- nine phases (Ref 22)

10. Bucciarelli --- eleven phases (Ref 8)

11. Zabriskie --- thirteen phases (Ref 39)

These authors all basically describe the same process but label and group activities differently. The obviously emerging and most recent pattern appears to be the three-phase approach, with the most usable version being Watson's effort. It relates most of the relevant activities to the DOD acquisition process making his description more easily understood by personnel inexperienced in working with software.

Driscoll stated that errors such as poor or delayed definition of system requirements and incomplete integration of hardware and software requirements have been amplified by a lack of managerial attention to software in the past. He also stated that a lack of ability to measure software development progress and inadequate numbers of qualified personnel are conditions that add to the problem.

The emphasis in Driscoll's report as well as most others was to increase the emphasis on early long term planning for software development. This could be done by raising software to a higher level in the WBS, requiring software be considered in total System Requirements Analyses (SRA), or requiring using command participation in requirements definitions and design reviews. These are only a few of the most important specific improvements that could be made but the ultimate improvement would be for management to raise software out of the category of "data" and require that plans for its development are on a priority level with computer hardware.

The recommendations that follow are a direct result of the problems and proposed solutions documented in the available, relevant literature. These

recommendations summarize this author's perception of the most needed changes to improve ECS software acquisition visibility. These recommendations were also the basis for initiating interviews with software development experts to derive a validated set of recommendations that if implemented would benefit the software development community.

Preliminary Recommendations To Improve Software Acquisition Visibility

1. Require using command participation in and input to all requirements, definitions and design reviews. No party can judge whether system design and performance requirements meet the user's needs as well as the user himself. Too often, the user generates a Requirement for Operational Capability (ROC) and then functionally steps out of the development picture until evaluation of the completed project. The using command representatives must have authority to speak for the command in important decisions.
2. Require that software be included in all System Requirements Analysis (SRA) during the concept formulation phase of system acquisition. This necessitates the consideration of software as part of the total system from the initial planning stages onward. It also forces early analysis of software requirements to allocate what portion of the total system requirements software must perform.
3. Move software to a higher level in the Work Breakdown Structure (WBS) to force its removal from the category of mere data. Placing software at the third level of the WBS would require detailed planning for and consideration of software from the beginning of the system development process. To do this, MIL-STD-881A on the WBS system would have to be revised to include software.

4. Establish measurable and achievable milestones for each software development. This would require increased USAF guidance but not regulation of what should be accomplished and reviewed at each milestone event. The Navy has issued beneficial and detailed documents in this area. Incremental milestones such as Critical Design Reviews (CDR) for individual Computer Program Components (CPC) would encourage review to a level of detail not possible with a single CDR for the entire program.
5. Place more emphasis on software in the Program Management Plan (PMP) and more widely disseminate the information contained in the Computer Resources Integrated Support Plan (CRISP). This would focus more program management attention on what tasks software is required to perform and how those tasks are to be accomplished.

The PMP is the program manager's Bible and he must have intimate knowledge of what it requires him to do. Emphasizing software in the PMP gets high level attention and prevents software from being ignored. The CRISP documents how software is to be developed, controlled, and maintained for a particular project and any policy or decision concerning software should be documented there.

6. Define support and operational software as separate deliverable contract line items to provide increased visibility into the development process and to provide a separate software cost breakdown. In the past, the cost of software was generally hidden inside a contract for a complete computer system and could not be tracked separately. This made analysis and comparison of software costs almost impossible. The software should also be

designated a configuration item when it will be transferred to other commands in order to control changes and the versions being used.

7. Ensure that one person is accountable and responsible for software in the System Program Office (SPO). This requires more technically qualified people and better education and training programs. It must be pointed out that there are no formal training or orientation programs for new software managers that the author has been able to discover.

V. INTERVIEWS

The set of recommendations resulting from the literature search in Chapter IV was used as a starting point for interviewing representatives of Air Force software management offices. These interviews were used to subjectively evaluate and refine the recommendations by critique and comparison with the collective experience of the interview subjects. The various backgrounds and areas of expertise of the subjects established a data base of expert opinions concerning the proposed recommendations. This base was of considerable help in refining and validating the set originally developed from the literature.

The interviews were conducted through a structured format (Appendix A) that first investigated the experience and knowledge level of the participants. This information was used to gauge the usefulness of the responses received. Next, several types of software problems were noted and discussed. Then, different types of solutions were recommended by this author and evaluated by the participants. The list of specific recommendations for improved ECS software acquisition was then reviewed and discussed. Finally, general comments on the thesis topic were solicited from the subjects to record any suggestions or opinions not noted in the formatted interview. The subjects were selected because of their experience levels and their varied backgrounds. Appendix C provides a list of those who were interviewed.

It must be noted that the limited time and scope of the research did not permit interviews with more than a limited number of personnel at a U.S. Air Force Research and Development (R&D) product division. It is believed, however, that the results obtained are representative and sufficient to support the recommendations made in this report.

All of the subjects agreed that management of ECS software acquisition is a major problem for the USAF and the DOD. The responses emphasized different problems, however, depending upon the tasks and responsibilities of the participant. Program managers, for example, were highly concerned with measuring the percentage of software completion, while engineers were more concerned about technical requirements defintitions and methods to verify and validate that the end product was what it was originally specified to be. Configuration managers, on the other hand, tended to be more concerned with describing, documenting and controlling changes of the software end product. This configuration control allows the user to know exactly what he has and allows him to modify and update the ECS in-house after deployment.

Of particular interest to this thesis is the fact that all responses except one listed inadequate management techniques as the greatest current ECS software acquisition problem in his organization. The one differing response noted past difficulties with contracting policies and related that this was also a management problem. The dynamic technical nature of computer hardware and software was not considered a significant problem by any participant.

When asked what other problems had been encountered, almost across the board, a lack of early high level planning for software development was cited. Most subjects expressed the opinion that software is not considered early enough in the development effort to have an impact on design issues. Apparently software personnel are often not even assigned to a SPO until after some irreversible major decisions on ECS development have been made. This imposes constraints on the computer hardware and software design before qualified ECS personnel have even considered the options.

In defense of the SPO and the R&D community, it must be stated that adequate numbers of qualified software people usually are not available. Some offices expressed a desire for people of any background who could be trained in software management. Currently, the software development area is so understaffed that many participants believe the government cannot match the inflated industrial pay scale. One respondent lamented that there are not enough government engineers to supervise the technical development of software and there are even less managers to monitor the contracting and managerial side of software acquisition. The result is that engineers or even programmers are expected to manage software acquisition and administer contracts in addition to their technical responsibilities. Expressed another way, the proper management techniques appear to exist but they have not been consistently applied because the management role in software has not been emphasized. There just are not enough people to properly perform the software acquisition management function and the engineer usually does both jobs on several programs at once.

Another problem uncovered in the interviews and akin to the lack of qualified people is the lack of training for the new people that are available. Generally, there are few instructors available because they cannot be removed from development tasks long enough to train new personnel. Until recently, a new hire was given a few weeks of self-study in dry regulations, a few days talking with an experienced "software man," and then assigned to his own program to learn-by-experience. This leads to some costly mistakes and was believed to have contributed to the bad track record of software development.

Recent efforts in ASD, as well as all of DOD, have attempted to better acclimate new people, first to the DOD acquisition environment and then to the unique features of software development. Most interviews surfaced opinions as to the inadequacy of training. Some, however, noted the trend toward increased orientation through hands on in-house laboratories. The System Engineering Avionics Facility (SEAFAC) allows new hires to join a cadre of engineers and managers who are performing small, low pressure, low risk development tasks in the Avionics Engineering Directorate (ENA) of ASD. These tasks are chosen primarily to provide a learning process prior to assignment of personnel to SPO's. Another improvement is the increased retention of experienced people in the home office for reference while the engineers assigned to SPO, are encouraged to return to them for guidance. The general feeling of the interview subjects was that there is not enough training but the situation is improving and that some learn-by-experience training is required anyway.

Concerning the software development process itself, the consensus was that more effort should be expended early to alleviate major problems. The idea of earlier and more extensive planning has already been introduced. This would include setting up better management information systems to get the right data at the right time to illuminate issues and decision options. This increased planning would provide better milestone definition and scheduling which are primary requirements for improved software development efforts.

The definition of milestones requires a detailed knowledge of the software development task. A recurring and associated issue described in the interviews concerns the inadequate and often delayed definition of software requirements. Apparently, specified system performance requirements

are allocated to the computer hardware early in a program with hardware design analysis and validation occurring before software tasks have been defined. Usually, software requirements are not completely defined until after the other weapon system components are designed and under FSD contract. This allows very little flexibility to choose the best ECS design options because the contraints already exist and software is usually considered the easiest method to provide the functions that other systems find difficult to perform.

That leads to another often mentioned problem, inadequate systems analysis and tradeoff studies. It seems when a development problem is encountered, the automatic response is "fix it in software." Hardware design is considered to be inflexible compared to software's inherrent ease of modification and, therefore, some design options are often not considered. The interview subjects generally felt that design changes made with software sometimes cost more, over the total system life cycle, than if made in hardware. The total system impact should be considered because a change affects many factors and tradeoffs should be made to get the best decision mix on those factors.

Participants also expressed concern over using command participation in the definition and review of system performance requirements. Too often it seems, the user does not actively follow the development process and, therefore, does not notice the system design gradually straying away from the original requirements. It is a costly mistake for engineers and managers to develop a system to satisfy what they understand to be the users' needs only to discover after completion that the user needed something entirely different. This happens because the user does not stay involved with the development process and many interviewers believed this should be required to a greater extent.

The idea of raising ECS software development to a higher level in the WBS received mixed reviews. Generally, the managers agree with the idea because they perceive an increased visibility into the development process, and engineers primarily disliked the recommendation because of the added workload and the increased cost. The literature supports the concept because of the detailed planning, scheduling and cost information it would require to describe and estimate sub-units of software work for the WBS.

With only minor exceptions, there was agreement that the same management approaches and techniques should be applied to computer software and hardware. The prevailing opinion was that applying the same techniques across the board would increase the visibility and attention given to software and would therefore improve the development effort and the product quality.

All of the interview subjects felt that their level of knowledge and expertise as well as the software community as a whole had increased significantly over the last few years. They also expressed the opinion that a learning process was occurring, that software management ability was slowly improving, but that many problems still remained.

VI. SUMMARY, CONCLUSIONS AND RECOMMENDATIONS

Summary

This thesis documents a study of ECS software development and the derivation of recommended improvements to current software acquisition management approaches. It has been estimated that the DOD currently spends more than $3 billion annually on software, with entire weapons systems depending on ECS software for successful operation. The ECS software for a weapons system is not the major part of the total system but must be considered critical to the overall performance of the system. Because of this virtual explosion in the use of computer resources, and specifically ECS software in weapons systems, software development problems and costs have received significant recent attention.

Increased visibility into the software development process has been proposed repeatedly in current literature and speeches as the key to solving the software development problem. A primary aim of this thesis, therefore, is to find means of increasing the visibility of software development efforts. Better visibility would provide greater warning indications of development difficulties, help prevent catastrophic schedule and budget overruns, and improve the technical quality of software products.

To discuss software visibility problems and improvements, a basic understanding of the software development process is necessary. This thesis employed a model of the software development process that was defined with respect to the traditional DOD weapon system acquisition life cycle. The cycle's five phases are:

1. Concept Formulation
2. Validation
3. Full Scale Development (FSD)

4. Production

5. Operation/Maintenance

A new system is developed in response to a perceived change in the environment. The change could be in a military threat or new technological advances that significantly modify military capabilities. Active systems may even need replacement but the point is that a requirement must be recognized before a new system concept can be formalized.

The Concept Forumulation phase analyzes the perceived need to determine whether or not it should be firmly established. Studies are conducted to determine if the proposed systems are economically or technically feasible and if production can be accomplished in time to satisfy the requirement. During this phase, some exploratory development is often done to estimate the technological feasibility of producing the system (Ref 25:11).

The Validation phase was previously called the project or contract definition phase. The system's performance requirements are defined and a minimum of preliminary design and engineering is accomplished. Major technical approaches are analyzed and some hardware may even be developed. The result of this phase is the contract definition which is required to initiate Full Scale Development (FSD) (Ref 25:20).

FSD generally includes the design, prototyping and testing of the completed system. This phase is of primary concern to this thesis and was further categorized into the following general tasks:

1. Preliminary Design (Analysis)

2. Detailed Design

3. Coding and Subunit Testing

4. Integration and Testing

5. Deployment

Production of the completed system can mean many things. Mass production lines are required to produce fleets of aircraft but a few hours of computer time could "produce" enough copies of a software program to issue one copy per aircraft. Acceptance testing can last weeks for aircraft and seconds for a single punched tape copy of a computer program.

The operation and maintenance phase begins when the first system is delivered and considered functional. A statement is then issued announcing the Initial Operating Capability (IOC) for the system. The system is then operated, maintained, and even modified to utilize it over an average 10 to 20 years of operational life. When the system is no longer a cost effective method of satisfying its assigned mission it is considered for retirement and the life cycle is completed.

Because of the increasing importance of software to the total system performance, the problems of cost, schedule, and technical performance of software have become critical weapon system development considerations. Some of the most important causative factors associated with these software development problems are:

1. Faulty and incomplete communication of user requirements.
2. Unrealistic cost estimates caused by insufficient visibility and control.
3. Unrealistic time schedules caused by the same lack of visibility and control.
4. High software failure rates.
5. Incomplete and insufficient specifications.

Computer hardware and software both suffer from these problems, however, software is newer and in the past displayed unrecognized criticality that made it more prone to these type problems. There appear to be solutions to

most software development problems if the development manager will only recognize the potential for them and take the necessary steps early enough to prevent them. The recommendations to prevent these type problems are presented in the Conclusion.

Conclusion

Software acquisition managers often complain of a lack of visibility in software development efforts. In this case, visibility means the ability of someone not directly involved with the actual development to know just how well a software development is progressing. This lack could reflect an inability to measure software development progress and status or it could reflect the lack of importance assigned to ECS software in the past in relation to the overall weapons system. This lack of attention amplifies errors made early in a program such as inadequate requirements definition and incomplete integration and allocation of hardware and software requirements. The best opportunity for improving software development efforts would be to concentrate on the concept formulation and program validation phases of the acquisition life cycle, which is pre-DSARC II, where the greatest leverage exists.

Most subjects interviewed in association with this thesis believed that a large portion of all software problems could be minimized if more software design and requirements analysis was done prior to FSD. These analyses should include requirements allocations to software, feasibility studies, hardware vs. software tradeoff studies, generation of the Computer Program Development Plan (CPDP), preliminary Part I Development Specification, Configuration Management Plan, and risk analysis to name only the major pre-FSD studies and plans.

Proper objectives, schedules, plans and milestones, must be established early enough for them to be usable during the system development process. This emphasis on planning should be proportional to its importance to the total system rather than its relative cost or work level. A net reduction of the time and cost of going from a ROC to an effective, operational system could be accomplished, not by shortening and expending less for every phase of activity, but by spending more time and money during early development tasks that will produce net savings in important areas that follow. In short, if there is any one rule to follow, it is to spend as much time and effort as possible planning at the beginning of a software development, as detailed as possible, exactly what the desired end product should be and how to develop it.

ECS software does not now receive the same degree of management control and attention that is given to weapon system computer hardware. In the future, it must receive this level of interest. There are a number of ways this should be done. One major approach would be to elevate software out of the data category and into the deliverable contract line item classification. This would also require that software be controlled as a configuration item, thereby giving it added visibility. Placing software at a higher level in the WBS and including software in the SRA would also eliminate many development problems by requiring more detailed task definition and work estimation.

More emphasis should be placed on a simplified but flexible decision making process that places greater reliance on sound judgment and less on regulations and complicated contract clauses. Contracting should be used as a tool of software development, not as a substitute for good management of acquisition programs.

Weapon systems software by its nature does not fit previously defined procurement categories. Software is not physical equipment, nor is it data. Therefore, attempts to define and address software in existing terms often causes confusion and frequently subjects it to inappropriate regulations by ill-qualified weapons system management personnel. Increased management focus and better communication are needed to assure that all levels of management are knowledgeable in software development and acquisition so as to better control software acquisition efforts.

The ideal acquisition structure does not eliminate the need for competent personnel to exercise sound judgment. It only highlights the fundamental decision points that must be dealt with as a system moves through the acquisition process. It also identifies the kind and quality of information that should be available when each decision is made.

Policy guidelines should be set whereby experienced personnel may exercise judgment in selectively applying detailed contracting regulations. Contracting methods and procedures have been used as remedies for acquisition problems found in past programs. This has stimulated a large growth in contracting regulations that have been applied to most programs, whether appropriate or not.

Excessively detailed guidance and requirements to use ineffective contract procedures have often been an impediment to major software acquisitions. In this area, there is a great need for personnel to have adequate management authority to adapt, modify, innovate, and be held responsible for actions taken.

Some success has been attained by having the SPO retain direct control and responsibility for defining and developing a software subsystem through a competent program staff, giving itself flexibility to change system characteristics and performance requirements.

Although these avionic software programs warrant special controls and organizational visibility, overreliance should not be placed on complicated regulations and contractual clauses. Better assurance of program success could probably be attained from proper contractor selection and the involvement of a strong, technically competent program management office coupled with a good test and evaluation capability.

In the DOD, a large number of regulations, directives and standards have been written for systems acquisition management. Most of these documents were not designed originally for software but were modified after the fact. The majority of these publications are still hardware oriented and conflict with each other and with current policy. Patchwork improvements only aggrevate the underlying problem, which is the lack of visibility over the key decisions that control the definition, development, and acquisition of avionic software.

There are a few published examples of important changes in the policies and directives that apply to software but these changes are minor compared to what is needed. Hardware and software tasks during the Validation and Development phases do not run parallel in spite of new directives and policies to the contrary. This non-parallel development characteristic was supported by the interviews discussed in Chapter V and by Watson's thesis.

The last major area that requires management emphasis is the lack of sufficient qualified software development personnel. Increased education of both military and civilian DOD resources is required to combat the high turnover of personnel. The DOD corporate knowledge has not been documented, allowing invaluable experience and lessons learned to be irretrievably lost. The problem arises from personnel leaving Government or leaving the software career fields. This underscores the need to develop new and capable software

managers as well as sufficient career incentives and controls to retain them.

Based on a review of previous studies and the collaboration of practicing software development personnel, this author believes that the primary factor contributing to ECS software development problems is the lack of consistently applied engineering discipline and sound management practices. No single correction will provide the required discipline and rigor to every facet of the software acquisition process.

The following are the Final Recommendations of this thesis. The final version consists of the Preliminary Recommendations revised to reflect the interview results. In essence, the Preliminary Recommendations were validated, however, the concensus of interviewed experts was to change the order of importance and to add additional points.

Final Recommendations to Improve Software Acquisition Visibility

1. Place software development on a level of importance equal to hardware development.
2. Require using command participation in and input to all requirements definitions and design reviews.
3. Require that software be included in all system requirements analysis during concept formulation.
4. Earlier emphasis on software planning by addressing software in the PMP, greater use of the CRISP, and increased attention to the required management information.
5. Increase efforts to train and retain qualified software engineers and managers.
6. Generally, apply the same management approaches and techniques to the development of both computer hardware and software.

7. Move software to higher level in the work breakdown structure and revise MIL-STD-881A to include software.
8. Establish measurable and achievable milestones for each software development effort.
9. Define support and operational software as separate deliverable contract line items with configuration item status.
10. Ensure that one person is accountable and responsible for software in the SPO.

Recommendations and Areas for Further Study

This thesis concludes that the preceding"Final Recommendations to Improve Software Acquisition Visibility" are valid and would be advantageous to the software development community. It is, therefore, recommended that these recommendations be implemented to foster more effective software acquisition discipline and practice.

It is also recommended that the following areas be studied further:

1. Methods for measuring software development progress and the possible application of C/SCSC.
2. The creation of intergovernment cataloging and exchange of government owned computer hardware and software resources.
3. The creation of a USAF or DOD technical school for orientation of new software development personnel.
4. Methods for increasing the satisfaction, retention, and indentification of software development personnel.

Bibliography

1. Aeronautical System Software Workshops. Proceedings, Aeronautical Systems Division, Wright-Patterson AFB, Ohio: 2-4 April 1974.

2. AFR 800-14. Management of Computer Resources in Systems. Washington: Department of the Air Force, September 1975.

3. AFSCM/AFLCM 375-7. Configuration Management for Systems Equipment, Munitions, and Computer Program. Wright-Patterson AFB: July 1971.

4. Air Force Logistics Command. Project Pacer Flash. Volume I: Executive Summary and Final Report. Wright-Patterson AFB, Ohio: 28 September 1973.

5. Boehm, Barry W. "Software and Its Impact: A Quantitative Assessment," Datamation, XIX (5) May 1973.

6. Breneman, H. M. "Software Design for Hardware Interaction on Real-Time Military Systems." Proceedings, IEEE Computer Society International Conference, Washington, DC, 7-10 September 1976. New York: IEEE, 1976.

7. Brooks, Frederick P. "The Mythical Man-Month," Datamation, Volume XXI, Number 12, December 1974.

8. Buciarelli, Marco A. "Technical Performance Measurement for Computer Software Development Programs," a study project to the Defense Systems Management School, Fort Belvoir, Virginia, May 1974.

9. Capps, Larry R. "Software Management and the Testing of Weapons Systems that contain an ECS." A study project presented to the Defense Systems Management School, Fort Belvoir, Virginia, November 1975.

10. Currie, Malcolm R. Memorandum to Assistant Secretaries of Military Departments (R&D). Washington, DC: 20 March 1974.

11. DeRoze, Barry C. "An Introspective Analysis of DOD Weapon System Software Management," Defense Management Journal 14(4): October 1975.

12. DOD Directive 5000.29. Management of Computer Resources in Major Defense Systems. Washington: Assistant Secretary of Defense (I&L), 26 April 1976.

13. Dredner, Stephen M. and Shulman, Hyman. Computer Resource Management Study; Executive Summary. Santa Monica, California; The Rand Corporation, September 1975.

14. Driscoll, A. J. "Software Visibility for the Program Manager," a study project presented to the Defense Systems Management School, Fort Belvoir, Virginia, May 1974.

15. Etheridge, Boyd, Major, USAF "Computer Software Management from the Point of View of the System's Manager," Air Command and Staff College research study, Air University, Maxwell AFB, Alabama, May 1974 (AD920559).

16. Gansler, Jacques S., "Software Improvement Plan Pressed," Aviation Week and Space Technology, 104(14), 5 April 1976.

17. Gansler, Jacques S., "Comment" Defense Management Journal 14(4): October 1975.

18. Government Report. "Space and Missile Systems Organization, Air Force Systems Command, Information Processing/Data Automation Implications of the Air Force Command and Control Requirements in the 1980's (CCIP-85), Volume IV, Technology Trends: Software," October 1973.

19. Johns Hopkins University Applied Physics Laboratory. DOD Weapon Systems Software Management Study. Washington, DC: Office of the Assistant Secretary of Defense (I&L). AD-ADZZ-160. June 1975.

20. Mangold, Eldon R. "Software Management Visibility. Proceedings of the Aeronautical Systems Software Workshop, Dayton, Ohio. April 1974.

21. Managing the Development of Weapons System Software. Conference Proceedings, Maxwell AFB, Alabama: 12-13 May 1976. (AD-B013-011).

22. Mathis, N. S. "Software Milestone Measurement Study," Systems Development Corporation, Santa Monica, California, November 1973. (LD-37329A).

23. McClary, T. E. Joint Memorandum from OSD/DDR&E, OSD/I&L, and OSD/ Comptroller to the Assistant Service Secretaries for R&D and I&L; Subject: Management of Weapon System Software, 3 December 1974.

24. Merwin, Richard E., "Software Management Through Product Control." A paper presented to the Conference on Managing the Development of Weapons Systems Software, held at the Air Command and Staff College, Maxwell AFB, Alabama, May 1976.

25. MITRE Report. Life Cycle Events: Software Acquisition Management Guidebook. Dayton, Ohio: Aeronautical Systems Division, Air Force Systems Command, February 1977. (AD-A037-115).

26. MITRE Report. DOD Weapon System Software Acquisition and Management Study, Briefing for DOD Software Steering Committee, 10 June 1975, Volume I. (AD-A034-802).

27. Nelson, E. A. "Management Handbook for the Estimation of Computer Programming Costs," Systems Development Corporation, Santa Monica, California, March 1976.

28. Nelson, Eldred, "Developing a Software Cost Methodology, Proceedings, IEEE Computer Society International Conference, Washington, DC, 7-10 September 1976. New York: IEEE, 1976.

29. Putnam, Lawrence H., A Macro-Estimating Methodology for Software Development, Proceeding, IEEE Computer Society International Conference, Washington, DC, 7-10 September 1976. New York: IEEE, 1976.

30. Regulations, Specifications, and Standards. Aeronautical Systems Division Software Acquisition Guidebook. Wright-Patterson AFB: November 1977. (ASD-TR-78-6).

31. Reiffer, D. J., "Software Specification Techniques: A Tutorial," Proceedings, IEEE Computer Society International Conference, Washington, DC, 7-10 September 1976. New York: IEEE, 1976.

32. Report of the Commission on Government Procurement. Washington, DC: United States Government Printing Office, 1972.

33. Reviews and Audits. Aeronautical Systems Division Software Acquisition Guidebook. Wright-Patterson AFB. November 1977 (ASD-TR-78-7).

34. "Software Management Conference - Phase III," Proceedings, AIAA, DPMA, Los Angeles, California: 26-27 January 1978.

35. Software Reliability Work Group (SRWG) of the Joint Logistics Commanders. Findings and Recommendations of the SRWG, Volume I, Executive Summary. Andrews AFB, DC: Headquarters Air Force Systems Command XRF, November 1975. (AD-A018-881).

36. Software Reliability Work Group (SRWG) of the Joint Logistics Commanders. Findings and Recommendations of the SRWG, Volume II, Supporting Technical Information. Andrews AFB, DC: Headquarters Air Force Systems Command XRF, November 1975. (AD-A018-882).

37. Watson, J. K., "Acquisition of Embedded Computer Software: A Descriptive Model," A Master of Science, thesis in Engineering Management at the University of Missouri-Rolla, 1977.

38. Wolverton, R. W., "The Cost of Developing Large Scale Software," TRW Software Series, March 1972. (TRW-SS-12-01).

39. Zabriskie, R. J., "The Development of Weapons System, Computer Programs," A Study Project presented to the Defense System Management School, Fort Belvoir, Virginia, November 1975. (LD-35049a).

Appendix A

Interview Format

Interview Format

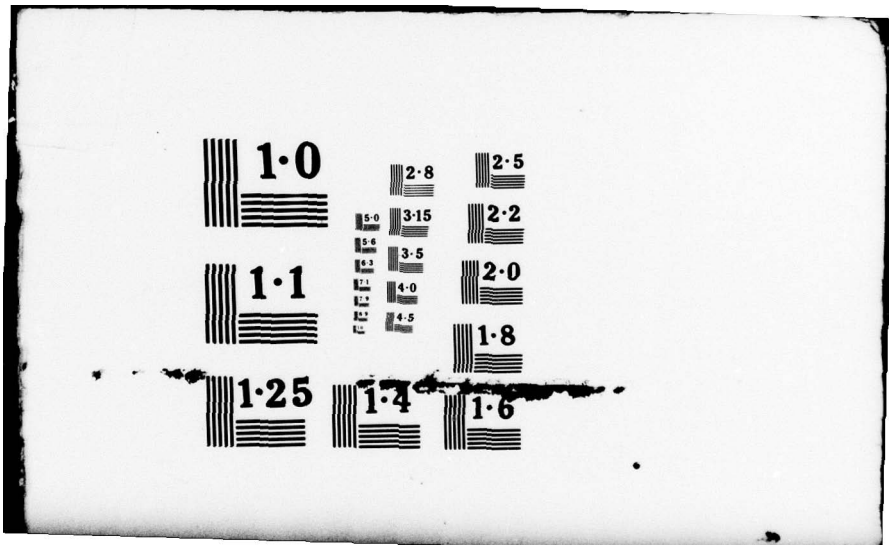
1. Have you been associated with the acquisition of ECS software?
2. How many years?
3. What is your current grade?
4. Where are you assigned?
5. What type of software experience do you have?
6. Describe your current job.
7. What order would you place the following ECS software problems in to reflect the greatest difficulty to your organization? Why?
   a. Dynamic state of the technical art
   b. Contracting policies
   c. Inadequate management techniques
8. What other problems have you encountered?
9. From your experience, do you agree that some of the ECS software acquisition problems are caused by management's inability to develop appropriate techniques as fast as the technical state of the art advances? Please comment.
10. From your experience, in what rank order of importance would you place the following problems? Why?
    a. Defining the specific software requirements.
    b. Defining and then implementing milestones for ECS software development.
    c. Tracking the software system's development progress.
    d. Defining and specify the software end product.
    e. Verification and Validation (V&V).

AD-A065 879 AIR FORCE INST OF TECH WRIGHT-PATTERSON AFB OHIO SCH--ETC F/G 9/2
A STUDY OF EMBEDDED COMPUTER SYSTEM SOFTWARE ACQUISITION MANAGE--ETC(U)
SEP 78 G M BARBEE

UNCLASSIFIED AFIT/GSM/SM/78S-1 NL

2 OF 2
ADA
065879

END
DATE
FILMED
4-79
DDC


1·0
1·1
1·25
1·4
1·6
1·8
2·0
2·2
2·5
2·8
3·15
3·5
4·0
4·5
5·0
5·6

11. Would you say that ECS acquisition managers are well prepared and trained or would you say that, for new personnel, a learn-by-experience education system is employed? Explain.
12. What experience and training do you feel are required for an adequate background?
13. Do you feel that good management practices and expertise are usually available but are not effectively used? Explain.
14. Do you believe that useful management information is often unavailable when needed because practices for evaluation, formatting, and feedback of software management information is inconsistent or loosely defined? Why?
15. From your experience, do software requirements definitions, risk analysis, development planning, preliminary design interface definitions occur during Full Scale Development (FSD) or earlier? Should software design and analysis begin earlier in the acquisition process than it does now? Explain.
16. Do you feel that hardware is usually initiated so early that software is forced to accept changes to relieve hardware difficulties even without the appropriate engineering and design? Explain.
17. Do you believe that software is so different from hardware, that hardware management approaches, techniques, and procedures will not work for software? What aspects of hardware and software development can be considered alike? Why?
18. Can most hardware problems be solved by changing software? What are the implications? Is this good or bad?
19. Does management of ECS software acquisition use a total systems approach for hardware and software combined? Should it now?

20. Do you feel that hardware design drives and limits software alternatives? Should more tradeoffs be made?

21. Should software be designed first and hardware designed or acquired off the shelf to match it?

22. Look at the separate list of "Recommendations for Improving Software Acquisition Visibility." In what order of importance would you place these suggestions?

23. Concerning these recommendations, do you:

    a. Completely agree

    b. Completely disagree

    c. Feel it needs improvements - what changes?

    d. Incomplete - what additions?

24. Do you have any suggestions on how to improve the management visibility of ECS software development?

25. Do you have any general comments on the subject of the interview?

SUMMARIZED - PRELIMINARY

RECOMMENDATIONS TO IMPROVE SOFTWARE ACQUISITION VISIBILITY

1. Require using command participation in and input to all requirements definitions and design reviews.
2. Require that software be included in all System Requirements Analysis during the Concept Formulation Phase of system acquisition.
3. Move software to a higher level in the Work Breakdown Structure and revise MIL-STD-881A to include software.
4. Establish measurable and achievable milestones for each software development.
5. Emphasis on software in the Program Management Plan and greater use of the CRISP.
6. Define support and operational software as separate deliverable contract line items with configuration item status.
7. Ensure that one person is accountable and responsible for software in the SPO.

Appendix B

Glossary of Terms

Glossary of Terms

ADP	Automatic Data Processing
AFB	Air Force Base
AFR	Air Force Regulation
AFLCM	Air Force Logistics Command Manual
AFSC	Air Force Systems Command
AFSCM	Air Force Systems Command Manual
APL	Applied Physics Laboratory of Johns Hopkins University
ASD	Aeronautical Systems Division
ASPR	Armed Services Procurement Regulation
BA/PA	Budget Authorization/Program Authorization
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CPCI	Computer Program Configuration Item
CPDP	Computer Program Development Plan
CRISP	Computer Resources Integrated Support Plan
C/SCSC	Cost/Schedule Control Systems Criteria
CSRDP	Computer System Resources Development Plan
DCP	Decision Coordinating Paper
DID	Data Item Description
DOD	Department of Defense
DSARC	Defense System Acquisition Review Council
ECS	Embedded Computer System
ECP	Engineering Change Proposal
ENA	Avionics Engineering Directorate of ASD
ESD	Electronics Systems Division
FCA	Functional Configuration Audit
FSD	Full Scale Development
FQR	Formal Qualification Review
FQT	Formal Qualification Test
IOC	Initial Operating Capability
IOT&E	Initial Operational Test and Evaluation
MTBF	Mean Time Between Failures
OFP	Operational Flight Program
OSD	Office of the Secretary of Defense
PCA	Physical Configuration Audit
PDR	Preliminary Design Review
PMD	Program Management Directive
PMP	Program Management Plan
PMRT	Program Management Responsibility Transfer
R&D	Research and Development
RFP	Request for Proposal
ROC	Required Operating Capability
SAM	Software Acquisition Management
SAE	Software Acquisition Engineering
SDR	System Design Review
SEAFAC	System Engineering Avionics Facility
SOR	Statement of Operational Requirements
SOW	Statement of Work
SPO	System Program Office
SRA	System Requirements Analysis

SRR	System Requirements Review
SRWG	Software Reliability Work Group
SSD	Source Selection Document
TO	Technical Order
TCTO	Time Controlled Technical Order
TRR	Test Readiness Review
USAF	United States Air Force
V&V	Verification and Validation
WBS	Work Breakdown Structure

Appendix C

Subjects Interviewed at Wright-Patterson AFB, Ohio

Subjects Interviewed at Wright-Patterson AFB, Ohio

Ajmel S. Dulai
PAVE TACK
ASD/ENAIA

Robert H. Gilmore
PLSS (SD-26E)
ASD/ENAIA

Kenneth L. Henry, Capt, USAF
B-1 Bomber
ASD/ENAIA

John M. Hoefirlin
Senior Software Engineer
ASD/ENAIA

John Y. Hung
Air Launch/Ground Launch Cruise Missile
ASD/ENAIA

C. Paul Johnson
Systems Software - Group Leader
ASD/ENAIA

Herbert R. McCarter
Technical Policy - Group Leader
ASD/ENAIA

Jack T. Sakai, Capt, USAF
Functional Software - Group Leader
ASD/ENAIA

Timothy A. Sparling
F-15 (TEWS)
ASD/YFEA

Beecher W. Vaughn
AE Software Focal Point
ASD/AECC

VITA

Gary Michael Barbee was born on 7 February 1950 in Columbus, Georgia. He graduated from Central High School in Phenix City, Alabama in 1968 and attended the University of Alabama for two years. In 1970, he transferred to Auburn University from which in March 1973, he received a Bachelor of Science degree in Electrical Engineering, as well as a commission in the USAF. He was then assigned to Aeronautical Systems Division of Air Force Systems Command at Wright-Patterson AFB, Ohio where he served as a research and development electrical engineer. During this time, he performed computer hardware and software design and development contract management as well as flight test functions for the AC-130 Gunship and HH-53 Pave Low III development programs. In June 1977, he entered the School of Engineering, Air Force Institute of Technology at Wright-Patterson AFB, Ohio.

Permanent address: 4903 16th Court
Phenix City, Alabama 36867

This thesis was typed by Pamela Glendening.

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER AFIT/GSM/SM/78S-1	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) A STUDY OF EMBEDDED COMPUTER SYSTEM SOFTWARE ACQUISITION MANAGEMENT AND RECOMMENDATIONS TO IMPROVE DEVELOPMENT VISIBILITY		5. TYPE OF REPORT & PERIOD COVERED MS Thesis 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Gary M. Barbee Captain, USAF		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS Air Force Institute of Technology (AFIT-EN) Wright-Patterson AFB, Ohio 45433		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS		12. REPORT DATE October 1978 13. NUMBER OF PAGES 98
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

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)

18. SUPPLEMENTARY NOTES

Approved for public release; AFR 190-17

JOSEPH P. HIPPS, Major, USAF
Director of Information

DEC 6 1978

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

Embedded Computer Systems
Computer Software
Software
Avionics Software
Software Development
Software Acquisition

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

The United States Air Force is the largest user of computers in the world and a major portion of that information processing capability is comprised of digital avionics computers. This thesis describes some of the major problems of acquiring Embedded Computer System (ECS) software for avionics systems. A description of the DOD avionics software acquisition process is included for background information as well as a discussion of the applicable guidance, policies, and regulations. (continued on reverse)

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Block 20. ABSTRACT (continued)

Recommendations to improve software acquisition were derived from literature research, refined by interviews with practicing software engineers and managers, and presented as a product of this thesis. The interviews were conducted with software acquisition personnel at the Aeronautical Systems Division of Air Force Systems Command at Wright-Patterson AFB, Ohio. A major conclusion of this thesis is that the development of a computer software management discipline is both necessary and feasible.