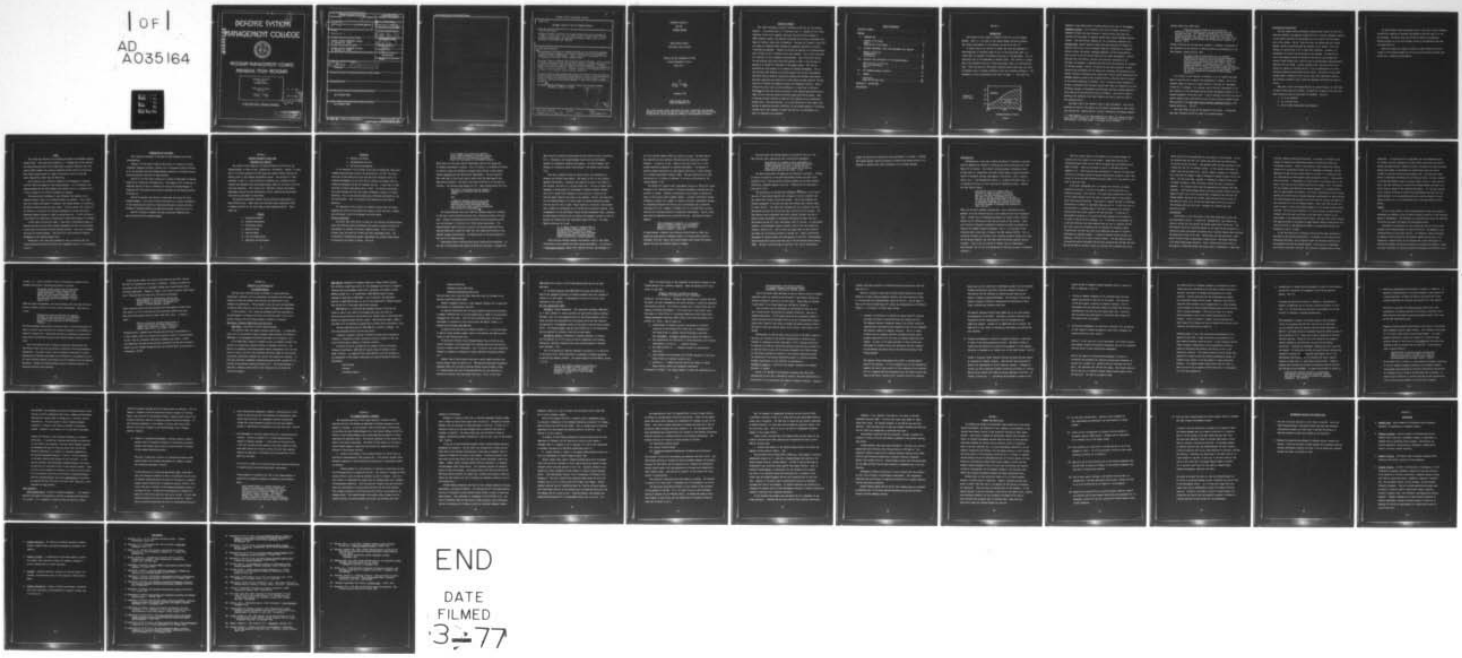
AD-A035 164 DEFENSE SYSTEMS MANAGEMENT SCHOOL FORT BELVOIR VA F/G 5/1
SOFTWARE VISIBILITY FOR THE PROGRAM MANAGER.(U)
NOV 76 A J DRISCOLL

UNCLASSIFIED NL

1 OF 1
AD A035164



END

DATE FILMED 3-77

AD A035164

DEFENSE SYSTEMS MANAGEMENT COLLEGE

PROGRAM MANAGEMENT COURSE
INDIVIDUAL STUDY PROGRAM

SOFTWARE VISIBILITY
FOR THE
PROGRAM MANAGER

Study Project Report
PMC 76-2

Alan J. Driscoll
LT COL USAF

D D C
RECEIVED
FEB 3 1977
A

FORT BELVOIR, VIRGINIA 22060

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS BEFORE COMPLETING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE (and Subtitle)

(6) SOFTWARE VISIBILITY FOR THE PROGRAM MANAGER.

5. TYPE OF REPORT & PERIOD COVERED

(9) Study Project Report, 76-2

6. PERFORMING ORG. REPORT NUMBER

7. AUTHOR(s)

Driscoll, A. J. / Driscoll

8. CONTRACT OR GRANT NUMBER(s)

(11) 10 Nov 76

9. PERFORMING ORGANIZATION NAME AND ADDRESS

DEFENSE SYSTEMS MANAGEMENT COLLEGE
FT. BELVOIR, VA 22060

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

(12) 57p.

11. CONTROLLING OFFICE NAME AND ADDRESS

DEFENSE SYSTEMS MANAGEMENT COLLEGE
FT. BELVOIR, VA 22060

12. REPORT DATE

76-2

13. NUMBER OF PAGES

54

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)

(10) Alan J. Driscoll

UNLIMITED

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

18. SUPPLEMENTARY NOTES

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

SEE ATTACHED SHEET

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

SEE ATTACHED SHEET

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

408462 dn

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

DEFENSE SYSTEMS MANAGEMENT COLLEGE

STUDY TITLE:

SOFTWARE VISIBILITY FOR THE PROGRAM MANAGER

STUDY PROJECT GOALS:

To determine what information the Program Manager must have to manage his software. To ascertain: (1) What are the current DOD/Air Force policies on software management? (2) What can the Program Manager learn from the practices of government and industry?

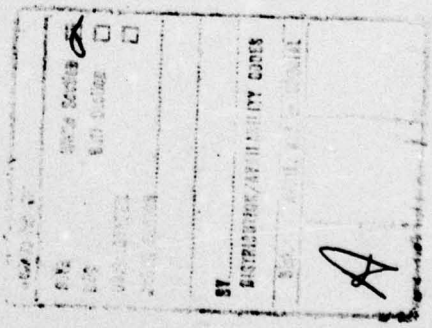
STUDY REPORT ABSTRACT:

The increasing importance of computer software in most acquisition programs is making increased demands on the Program Managers ability to manage and control the software development. This study project investigates the role of 'software visibility' in that management task.

The report looks at software visibility from three viewpoints. First, a review of current literature in the form of formal articles, studies and speeches to ascertain the current trends and ideas in software management. Secondly, a review of Department of Defense (DOD) and Air Force directives and regulations relating to software management. Thirdly, interviews with Program Managers to get their views on how to obtain software visibility.

The study concludes that although many ideas for improved visibility (such as making software a configuration item and putting it high in the Work Breakdown Structure (WBS)) have surfaced and found their way into DOD and Air Force directives, the key for the Program Manager is early and forceful planning to use them.

KEY WORDS: Software Management, Software Visibility, Embedded Computer Resources, Computer Programs



NAME, RANK, SERVICE	CLASS	DATE
DRISCOLL, A. J. LT COL, USAF	PMC-76-2	10 NOVEMBER 1976

SOFTWARE VISIBILITY

FOR THE

PROGRAM MANAGER

Study Project Report

Individual Study Program

Defense Systems Management College

Program Management Course

Class 76-2

by

Alan J. Driscoll
LT COL USAF

November 1976

Study Project Advisor
LCDR SUE ANDERSON, USN

This study project report represents the views, conclusions and recommendations of the author and does not necessarily reflect the official opinion of the Defense Systems Management College or the Department of Defense.

EXECUTIVE SUMMARY

This report documents a study of "Software Visibility for the Program Manager": to ascertain what it is and how to get it. Because of the virtual explosion in the use of computer resources and particularly software in modern weapons systems, the problems associated with software development have begun to receive a great deal of attention. The need for "software visibility" as a means of combating these problems has appeared repeatedly in current literature and speeches. Software visibility has meaning to a Program Manager only in terms of how it relates to his total program. This is best expressed as a function of cost, schedule and performance. One of the prime reasons for the problems which have afflicted software is that it has not been regarded as an element of importance in the weapon system. This lack of attention has been present in all phases of software development. The result has been that problems in the early phases such as poor requirements definition and not properly integrating hardware and software requirements have been aggravated and magnified in later phases by other problems like the inability to measure development progress and inadequate staffing. Taking affirmative action such as putting software at a high level in the Work Breakdown Structure and including software in the System Requirement Analysis (SRA) can go a long way toward alleviating these pervasive problems. None of the many actions required to avoid all the problems can be accomplished without early, long term planning. It is the conclusion of this report that the key to obtaining software visibility for any Program Manager is to elevate software out of the category of "data" and plan for its development on a level of importance with hardware.

TABLE OF CONTENTS

EXECUTIVE SUMMARY i

Section

I. INTRODUCTION 1

Purpose of the Study 3
Scope 6
Organization of the Report 7

II. SOFTWARE MANAGEMENT IDEAS FROM GOVERNMENT AND INDUSTRY . . . 8

Analysis and Design 9
Implementation 15
Verification 17

III. AUTHORITY AND CONSTRAINTS OF THE PROGRAM MANAGER 21

DOD Directives and Policies 21
Air Force Regulations 24
Other 33

IV. THE PROGRAM MANAGERS VIEWPOINT 36

V. SUMMARY 42

Conclusions 42
Areas for Further Study 45

APPENDIX A: DEFINITIONS

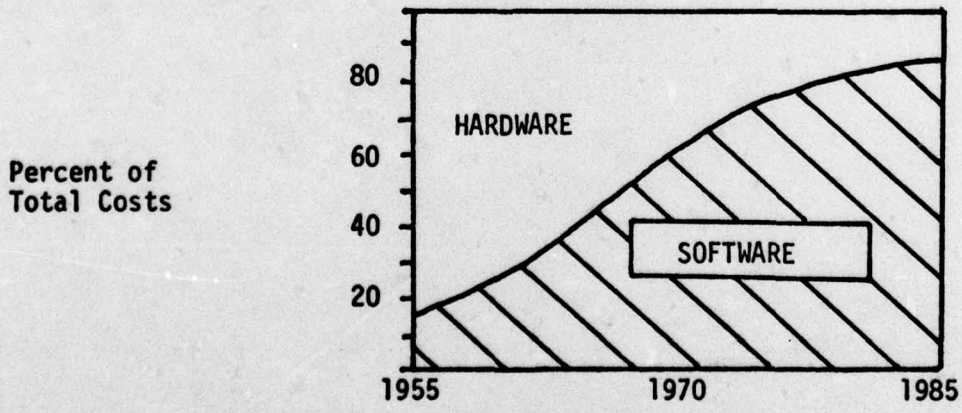
BIBLIOGRAPHY

SECTION I

INTRODUCTION

The subject of this study is "Software Visibility for the Program Manager: What is it in terms of the Program Manager and how he manages the software development in his program, and how can he get it?"

It should come as no surprise to anyone who reads the newspapers or trade magazines or otherwise keeps his ear to the ground that there is a rising level of interest in software on the part of the government. This is especially true of the Department of Defense (DOD). This interest is partly due to the extremely high cost of what many people think of as being merely data. Since the advent of the digital computer several years ago, the ratio of software costs to hardware (computer) costs has increased enormously. This phenomenon is well represented by the chart in Figure 1. This chart has



Hardware/Software Trends

Figure 1

appeared in many publications including being on the cover of the Defense Management Journal. This increase in the cost of software relative to computers becomes even more relevant in terms of total costs when you consider the high number of weapons systems which contain Embedded Computer Systems. (See Definitions, Appendix A) At a conference reported on by Aviation Week, Jacques Gansler, Deputy Assistant Secretary of Defense for Material Acquisition, said a Pentagon Study had "identified 115 different defense systems that employ "embedded computers" of which approximately half are now in service and the other half are under development." (36:43)[1] Relating this into dollars, Gansler also said that "according to our estimate, the Pentagon is spending more than $3 billion annually for software for defense systems." (36:41) He went on to say that 68 percent is spent during system development and 32 percent for operating and maintenance costs. Another estimate of software costs states that "current annual expenditures for Embedded Computer Systems (ECS) exceed $2 billion, with more than 70 percent of this amount dedicated to software." (20) It takes no great mathematical genious to determine from the above figures that there are something less than 60 defense systems in development (with perhaps that many program managers?) with close to $1 billion dollars worth of software. This kind of money is obviously not something to be taken lightly, yet cost is just one side of the software picture.

The other side of the software story is poor performance. This can be expressed in terms of not meeting its functional requirements or more simply, poor reliability. Quoting Barry C. De Roze, Directorate for Weapons Support

[1]This notation will be used throughout the report for sources of quotations and major references. The first number is the source listed in the bibliography. The second number is the page in the reference.

Systems Acquisition, OASD (I&L):

> Although hardware reliability has improved substantially, the corresponding gains in system reliability have not been realized. This apparent contradiction arises because software unreliability - the failure of software to satisfy the stated operational requirements - has become the "tall pole in the tent" in determining the reliability and operational readiness of systems. (20-3)

Further indication of the high level interest in software is expressed in the Statements by the Director of Defense Research and Engineering to the 94th Congress, Second Session, 1976:

> The urgent need for reducing the costs of computer software was described in last year's Posture Statement. A DoD Directive resulting from a comprehensive study is currently being coordinated that will require the use of improved procedures in software acquisition. A DoD software Management Steering Committee has been formally established to: (1) review DoD software technology programs, (2) recommend needed areas of research and emphasis, and (3) plan a balanced and coordinated software program. (12-VII-25)

As a result of this interest in software, a lot of studies have been made to ascertain how to improve the management of software. One of the foremost themes to come out of these studies is the need for "management visibility" of software. For instance, one of the four subelements of the main objective of the DOD Weapon System Software Management Program is "to promote management visibility" (21). "Lack of Software Visibility" was one of several problem areas contributing to the critical software problems facing DOD according to the DOD Weapon Systems Software Management Study by John Hopkins University. (23:1-2)

This then leads us back to the purpose of this study. To determine just what "Software Visibility" means to a Program Manager.

Discussion of the Problem

For most weapon system development programs which consist of both hardware and software, the computer software will be a critical component relative to the overall operation of the system, but it is usually not the primary part of the system. (30:13) There are then, two reasons why the Program Manager should be concerned with the software in his system. First, his software is going to get a lot of high level attention. Secondly, its performance is critical to the success of his program. The need for an error free computer program is obvious in the case of the quidance and control flight software for a missile such as the Minuteman Intercontinental Ballistic Missile (ICBM). A minor software error could cost millions of dollars by causing an inflight failure. Also, an undetected error could seriously degrade the operational missile force. But there are many other systems where, although an error may not cause such spectacular results, the resultant schedules slips, rework and degraded performance are just as costly.

What then, should the Program Manager be concerned about, or, what does he need to know about his software? Essentially, he needs to know the same basic things he needs to know about his hardware. They are:

1. Is it on schedule?
2. Is it within cost?
3. Does it meet requirements (performance)?

The Johns Hopkins report mentioned earlier, said that a lack of software visibility as compared to hardware contributed to the fact that it is not well managed. The report also said visibility could be increased by putting software on a par with hardware (23:2-4), and addressed some ways of accomplishing it.

What this study will attempt to address is how software visibility relates to the Program Manager and his concern with software problems that affect cost, schedule and performance.

SCOPE

This study was limited to the software associated with Embedded Computer Systems (ECS). (See definitions Appendix A). Although some of the thoughts and ideas discussed herein may be applicable to general Automatic Data Processing (ADP) systems, the research conducted and the thrust of writing has been toward systems which fall under the purview of the Air Force 800 series regulations rather than the 300 series.

This study was also limited to looking at the management of software strictly from the viewpoint of the program manager. It is directed at the program manager who has both hardware and software and it is assumed that he is not an expert in the field of computers or software.

The acquisition of software is very different from hardware in one important aspect; there is no production phase for software. (24:1, 20:3) Once the software development is complete, the program manager is essentially finished (assuming the job was done right). For this reason, this study will be limited to a discussion of software development up to the point when a completed computer program is ready for operational use. It will not discuss in any detail the operational, maintenance or modification aspects of software.

In the course of doing the study, I found that a considerable amount of material had been written about software management but that it was provided at the level of the person directly managing software - one level or probably two below the Program Manager. There was very little written on software management at the program manager level.

Nonetheless, this study has attempted to look at visibility for the Program Manager and delved into lower level management only as it was necessary to do so.

ORGANIZATION OF THE REPORT

This study was conducted in the form of three separate but related investigations.

Section II of the report contains the results of a search of current literature (magazine articles, studies, etc.) to ascertain current thinking as to the problems which the Program Manager encounters in software development and potential solutions to those problems.

Section III of the report consists of a review of Department of Defense (DoD) Policy, Directives and Regulations. The purpose of this review was to highlight sources of help or hindrance for use by the Program Manager in grappling with the problems and trying to implement the solutions described in Section II.

Section IV presents the results of interviews with three Air Force Program Managers. The intent of these interviews was to obtain information on the experience and opinions of those who have had to wrestle with the problems of managing software and getting the proper visibility to do it.

Section V contains a summary and some conclusions regarding software visibility for the Program Manager.

SECTION II

SOFTWARE MANAGEMENT IDEAS FROM GOVERNMENT AND INDUSTRY

The intent of this study was to discuss software visibility and the Program Manager in terms of Cost, Schedule and Performance. However, it seems that it is almost impossible to talk about the subject without covering the various steps in the software development process. The corrallory to that statement is that it is very difficult to separate cost, schedule and performance and determine what the program manager should do to obtain visibility into each separately. This section will, therefore, discuss the software development process and the relationship of Cost, Schedule and Performance visibility to each phase in the process.

The software development process has been defined by many people in many different ways. These range from the seven steps expressed by Eldon R. Mangold (30:2-8) to the three steps of Boyd Etheredge (25:21). These steps are:

Mangold

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

1. Analysis and Design
2. Implementation and Test
3. Delivery and Maintenance

R. W. Wolverton in his writings on the cost of developing large scale software discusses what he calls the 40-20-40 rule. (37:13) This rule, which was derived empirically says that the total resources (cost) for software development will be split 40% for analysis and design, 20% for coding and debugging and 40% for checkout and test. I have used a slight variation of these three phases which I label: (1) Analysis and Design - this step includes the first four steps of Mangold's process; (2) Implementation - this is the code and debug step of Mangold and Wolverton, and (3) Verification - this is essentially the checkout and test step of Wolverton.

The remainder of this section will address various tools and ideas, suggested by writers in government and industry, which link cost, schedule and performance visibility throughout the three phases.

Analysis and Design

The events that takes place in phase one, the analysis and design phase, start with defining system requirements and progress through allocation of requirements to software to having a complete design. This is a very critical step, one which will effect the total development cycle. The criticality of establishing systems requirements and software requirements is pointed out by Winston W. Royce. (30:1-13)

> In our judgment the single most important cause of poor management of software projects is the inability to successfully accomplish these first two phases of requirements analysis.

Royce goes on to say that without requirements analysis the first step in software development is design. This, of course, is a prelude to disaster. In actuality there are probably no projects where design is done without someone thinking they have defined the requirements. The real question then comes down to how do you know or ensure that the requirements have been defined properly? One step in the right direction is to get the user involved. The ultimate requirements are his. Again quoting Royce (30:1-21):

> The user of the software must be capable of injecting his expertise into the software product.
>
> and;
>
> A complete, detailed, accurate set of performance and implementation requirements which has been fully coordinated with the user is the first step in ensuring compliance with operational requirements.

The Program Manager must ensure that his software people are involved in the total system engineering process thus interfacing directly with the user. Software requirements must be considered from the beginning by the user and developer in relation to all other system requirements. The Program Manager should have this in mind as the program progresses from a Required Operational Capability (ROC) thru the system specification to the computer program specifications. He should insist on the participation of the user in all design reviews.

Requirements must be defined early and as specifically as possible. In the case of requirements that cannot be defined at the start, a schedule for

their definition should be established and the software planned accordingly. (1:6) Conversely, the Program Manager should plan the development schedule to accommodate changing requirements. He should remember that software is affected by nearly every change in the weapon system design. (1:6)

The latter statement brings up another point; the integration of hardware and software requirements. One aspect of this is the software/computer relationship. Obviously, the computer can have a big effect on the software, but there is a reverse effect also. The use of higher order languages is being pushed for development of defense software although the use of a higher order language is less efficient in terms of the required memory capacity and speed of the computer. (36:41) The issue here for the Program Manager is to ensure that his systems engineering people plan for the fact that a larger, faster machine is required or the software cost, schedule or performance (or all three) may be adversly affected. One recommendation of the DOD Weapon Systems Software Management Study, conducted by Johns Hopkins University was to require that computer systems be sized to provide for uncertainties and requirement growth (23:6-22). The Johns Hopkins study went on to say:

> It is a basic feature of software that it can accommodate change provided it is not limited by hardware capacity or speed. Accordingly, an important part of software systems engineering is the judicious and controlled provision of growth capability.

Aside from the software computer relationship, there is the larger relationship of the software and total system requirements. In an article in Government Executive, August 1975, General Phillips, the Commander of

Air Force Systems Command (AFSC) was quoted as saying: "We have come to the conclusion we must engineer software much the same way we engineer hardware." He went on to say: "What all this boils down to is a full systems engineering approach to software development." The process of applying systems engineering to requirements definition is often referred to as Systems Requirements Analysis (SRA). The more effective use of SRA, particularly as it relates to software, is an area to be pursued by the Program Manager (1:14).

The effects of a good or poor requirements analysis or definition ripple through all the remaining phases of software development including the next one which is design. Changes in requirements cause changes in design and these in turn usually cause schedule changes. (Software continually changes throughout a project for a number of reasons including: requirements changes, to accommodate hardware deficiencies and to accommodate new or modified interfaces (27:1-2)) Changes in schedule will at a minimum drive up costs: they may also preclude meeting all performance requirements. Ideally, every step would be completed prior to starting the next. According to Eldon R. Mangold:

> From a management standpoint, it is essential that the successive steps in the development process be restricted until the preliminary design is complete. (30:2-13)

In some programs, schedules have characteristically been so tight that coding was begun before an adequate analysis of program design could be performed. (27:5-24) Again, the Program Manager must protect the program against the cost and schedule impacts of changes.

Two areas where the Program Manager can accomplish this are, one, that familiar item, planning and two, configuration management.

> The planning of a computer program development is probably the source of fifty to seventy-five percent of all development problems. Planning does not have to be bad to lead to problems; but it must be exceptionally good to avoid them. (1:10)

The above quote serves to emphasize the important of planning. Planning of course is pervasive to the whole development process and good, early planning can go a long way toward avoiding problems by providing schedule flexibility, adequate computer size, etc. Planning will be discussed in more detail later.

The application of configuration management to software is not really new, but it does deserve mention here, because it can be a powerful tool for, among other things, controlling changes. One of the problems with software management in the past has been that software was treated simply as "data" (36:41). This meant that software was not a contract line item deliverable and thus did not get the same visibility as did hardware. The John Hopkins study recommended that major computer software involved in weapon systems development be designated Configuration Items (CIs) and deliverables during Full Scale Development (FSD) to include: (1) operational software; (2) development support software, and (3) test and integration software. (23:2-5,2-11) The Air Force has taken steps in that direction and these will be discussed further in Section III. Formal Configuration Management of software begins with the approval of the Part I (Development) specification which occurs about the time of the Preliminary Design Review (PDR). The PDR is also the time of the first real look at the software

design and reflects the matching of the requirements to a design. Although, the Program Manager cannot be expected to attend every design review in his program, he should stress their importance to his software manager.

Implementation

Implementation is the step in which the design is converted to program code and debugged (or tested) to eliminate any errors which may exist (and there will be errors). The analysis and design phase, although not easy to keep track of, nonetheless, had some visible means of measuring progress similar to hardware (written requirements, flow charts, a Part I specification, etc.). The implementation phase is a slightly different story; it is extremely difficult to find a good way of determining status. Quoting the John Hopkins report:

> The abstract nature of software makes it difficult to measure progress and, hence, makes it even more necessary to formalize the steps in design, implementation and test. The lack of such definition leads to difficulies in interface management and to the late discovery of inadequate requirements or design errors, with resulting slippages in schedules and increases in cost. (23:2-6)

There are two basic problems involved here. One is the nature of programmers to do the interesting work at the expense of dull work (documentation). Visible signs of programming progress are almost totally lacking. Another set of difficuties arises from the nature of the product. There are virtually no objective standards or measures by which to evaluate the progress of computer program development. (37:1) It is because of this situation that we have what is known as the 90% syndrome (36:42). This is expressed in Golubs Law #12 which savs; "projects progress quickly until they are 90 percent complete, and then they remain 90 percent complete forever" (26:252). (This is one of a series of "Golub's Laws of Computerdom," some humorous, but all too true expressions of what can go wrong in computers/ software.)

There are several means at the disposal of the Program Manager for tracking cost and schedule in any project. Among those are the Cost Performance Report (CPR) and for smaller programs the Cost/Schedule Status Report (C/SSR). These are discussed in detail in AFR 800-6 and AFSC Pamphlet 173-3. These have not been satisfactory in the past for obtaining visibility of software primarily because software was treated as "data" and/or was so low on the Work Breakdown Structure (WBS) (if it was there at all) that it was never seen.

It has been recommended that to increase the visibility and understanding of software, it be put in the WBS at a level equivalent to a hardware subsystem. (2:31; 23:2-4) This would make software more visible at the Program Manager level, but there is another aspect to the problem which also deserves attention. That aspect is whether the CPR or C/SSR type information on software is meaningful when you get it. It will tell you if the contractor is spending money at the rate he projected, but probably nothing about what actual progress has been achieved. If some set of measureable milestones is not established then progress will be measured in terms of time or dollars expended, i.e., if 100 hours are estimated for a task and 100 hours have been expended then the task is complete (29:55). The track record for estimating cost and schedule for software (coding in particular) has been very poor which emphasizes the need for discrete milestones to evaluate progress. A series of design reviews (system, preliminary, critical) is at least a partial answer. The John Hopkins report discussed the use of milestones (23:2-6) and referenced MIL-STD-490, MIL-STD-483 and AFR 800-14, Volume II which it said call out milestones but do not

define the work to be accomplished or the products to be delivered. It did not reference MIL-STD-1521 (11), which does define work and deliverables for design reviews and audits. One approach which has been proposed is what I would call the "all or nothing" system. In this system, the project is broken into discrete tasks (e.g., coding a module, checkout of a component, etc.) and then no attempt is made to estimate or measure progress within the tasks. For any given task, progress is reported as either 0% (from start to almost finished) or 100%, the point at which the task is physically complete. This approach takes away the guesswork and eliminates the "90% syndrome." The Program Manager should not have to concern himself with this degree of detail but it was discussed here to emphasize the fact that the Program Manager should be very careful about how much faith he places in the reports he gets (e.g. the CPR and C/SSR reports are only as good as the estimates which go into them: Estimates for software have been notoriously poor).

Verification

Verification is the third phase in the three which makes up the software development process. In phase two, Implementation, the software was coded to satisfy the requirements and design established in phase one. The Verification Phase will determine whether or not phase two was successful in translating those requirements and design into a computer program that satisfies the operational needs of the user. Verification can take many forms, from a manual review of code to operational flight testing. One author describes it as three interrelated functions. First is Code Verification, which is the process of determing whether the actual code is implemented in compliance

with the computer program specifications. The second is Validation, the process of testing the coded program against the specified design and performance criteria. Third is Certification, which extends the testing process to an operational (either real or simulated) environment. (34:22) Regardless of how the Verification Phase is defined, it is imperative that all requirements be tested against some measureable criteria. Requirements must be testable. A requirement for which there is no feasible test or for which a test has not been defined should not be allowed. (1:6) The problem for the customer (and therefore the Program Manager) is to somehow ensure that all requirements are testable. To do this, consideration of the testing methods and criteria must be accomplished when requirements are being defined. The Program Manager must maintain visibility into this process and emphasize the importance of it to his software people. Again, this must be done at a level of detail in which the Program Manager will not be able to get directly involved. One way he can influence it is thru insistance on good planning. Software test plans must be written to the same level of detail as the requirements in the development specification. The test plans should require that <u>each</u> performance requirement of <u>each</u> computer program configuration end-item be verified in some appropriate manner, and they should specify the acceptance criteria. (35:68)

So far the discussion of software verification has been restricted to that done by the developing contractor. An adjunct to verification by the developing contractor, and an excellent means of providing visibility for the program office and the Program Manager, is the use of an independent verification contractor. (27:5-9) This practice is becoming more and more

widespread. It originated with a requirement for the independent check of software from the standpoint of nuclear safety (hence the term Nuclear Safety Crosscheck Analysis (NSCCA)) wich was instituted to provide assurance against such things as inadvertent or unauthorized launch of systems (primarily missile) carrying nuclear weapons (12). The practice, widely known as Independent Validation and Verification (IV&V), has been expanded to include performance as well as nuclear safety criteria. When used properly, the IV&V contractor can provide the Program Manager excellent visibility into all phases of the software development. For instance, because of his interest in having to verify that requirements have been met, the IV&V Contractor is in an excellent position to provide feedback at early design reviews, etc., as to the testability of the requirements and the design.

General

So far in this Section, I have discussed several ideas from people in government and industry on how to obtain software visibility in the different phases of development. The following are some additional means for obtaining visibility that apply throughout the development process.

The method of contracting for the software can have an affect on the ability of the Program Manager to obtain visibility. The methods of contracting referred to are those such as: (1) choosing a single contractor to develop a total system with the software treated as only one of the contractor's several tasks, or (2) choosing one or more contractors to develop the software and another contractor to develop the hardware. There are other variations of course and there are advantages and pitfalls in all

of them. N. E. Bolen, writing in "An Air Force Guide to Contracting for Software Acquisition," addressed the subject as follows:

> The single system contract has the advantages of making one organization responsible for system performance. However, there is the danger that the software development effort will not receive proper management attention and resources within the contractor's organization. (3:7)

Under the above circumstances, the Program Manager must take some delibrate action to ensure visibility into the software development. Bolen went on to say:

> Dividing the system acquisition into separate contracts so that one contractor is responsible for software alone provides the potential for better Air Force visibility of the contractor's progress; .... .

The Program Manager should keep in mind that there is only the potential for better visibility and contracting for software separately is not without its potential problems also. Among these could be an integration problem which could place a considerable burden on the Program Managers organization and staff.

One of the problem areas in software acquisition identified by the John Hopkins report (23) was the technical staffing of the Program Manager's organization. The report cited a lack of personnel experienced in systems engineering and software development as contributing to the problem areas of; (1) lack of policy guidance and planning, and (2) inadequate cost and schedule monitoring. Because the Program Manager cannot do everything himself, he should get the best staff possible.

Aside from the number and quality of personnel on the staff, the way the staff is organized can also make a difference. The ways of organizing vary widely from vertical or aggregate through matrix and different forms of project approaches. Stephen P. Keider, in an article entitled "Why Projects Fail," discussed what he called the "Utilization Philosophy":

> A most fundamental problem which affects many large companies is one which demands maximizing the utilization of personnel, as opposed to a project - oriented approach. (29:55)

Keider explained that he was talking about the reassignment of people whenever there is a lull in the work and the continuity problems that occur when later these people return or others take their place. In Keider's words:

> This is a disastrous approach because while it assures that people are always assigned to a project and utilization is high, it places an emphasis upon effort, not results. (29:55)

Although Keider's comments were directed toward the use of programmers in a large company, there is a lesson to be learned in terms of using personnel in such a way as to maintain continuity throughout any project. Keider also emphasizes the need to have one man responsible for the entire software project and not fragment responsibilities to the point where no one person is accountable. (29:54)

SECTION III

AUTHORITY and CONSTRAINTS OF THE PROGRAM MANAGER

The high level attention given to software in recent years has, predictably, fostered a raft of regulations and manuals with the avowed purpose of producing software that satisfys the constraints of cost, schedule and performance. These documents cover the span from Department of Defense (DOD) Directives to Air Force System Command (AFSC) pamphlets.

In this section, I will review the documents which are pertinent to the development of software and discuss the portions which I feel are of salient interest from the Program Managers viewpoint of software.

Department of Defense (DOD) Directives and Policy

DODD 5000.1, Acquisition of Major Defense Systems.

This directive does not address software specifically. It establishes policy for the acquisition of major programs (major programs are defined in DODD 5000.1) and management principles applicable to all programs. (5:1)

All of DODD 5000.1 applys to software as well as hardware, however, one portion is of particular interest relative to software visibility. The portion on management information/program control requirements calls for the use of a single, realistic, Work Breakdown Structure (WBS) for each program to provide a consistent framework for control and reporting of progress. It also says contractor management information/program control systems will be utilized to the maximum extent practicable (5:7). The Program Manager is therefore, limited to some extent in the information he can obtain for visibility purposes.

DODD 5000.29, Management of Computer Resources in Major Defense Systems. This directive establishes policy for the management and control of computer resources during development acquisition, deployment and support of major Defense systems (6:1) As stated above, DODD 5000.29 applys to major programs as described in DODD 5000.1, and in addition, the principles outlined in DODD 5000.29 apply as well to the acquisition of Defense systems that do not fall in the major acquisition category.

DODD 5000.29 is a relatively new document (26 April 1976) and the intent is that it will not be in existence very long, but that its policies and principles be assimilated as an integral part of the established process of acquiring major Defense systems. DOD Directives 5000.1, 5000.2, and 5000.3 will be modified as appropriate to reflect this assimilation. (6:2)

The most significant part of DODD 5000.29 is Section V, Policy. The next few paragraphs briefly outline that policy.

In general, computer resources in Defense systems must be managed as elements or subsystems of major importance during all phases of the life cycle with particular emphasis on computer software.

To ensure the early consideration of computer resource (including software) requirements, DODD 5000.29 requires they be included in the DSARC II Review. To accomplish this, DODD 5000.29 lists the following to be implemented in the Concept Formulation and Program Validation Phases of development:

Risk Analyses

Planning

Preliminary Design

Security Definition

Interface Control Definition

Integration Methodology Definition

The risk areas, and a plan for their resolution shall be included in the Decision Coordinating Paper (DCP).

Another statement of policy is that computer software will be specified and treated as a Configuration Item (CI).

To identify acquisition and life cycle planning factors and guidelines, a computer resources plan will be developed prior to DSARC II and maintained throughout the life cycle. (The Air Force plan which meets this policy requirement is discussed in this section under AFR 800-14, Volume II, a document which preceded DODD 5000.29).

In parallel with the policy of making software a CI, is the requirement to specify unique software support items as deliverables with DOD acquiring rights to their design and/or use.

Of particular interest to the Program Manager from a visibility and control standpoint is the requirement for milestone definition and specific criteria to measure their attainment. This, of course, fits with considering software as elements or subsystems of major importance and making software a CI.

Another item of policy which could have a severe impact and which the Program Manager should be aware of is: "DOD approved High Order Programming Languages (HOLs) will be used to develop Defense system software, unless it is demonstrated that none of the approved HOLs are cost effective or technically practical over the system life cycle." (6:3) At the time

DODD 5000.29 was issued, a list of DOD approved HOLs had not yet been published.

It is readily apparent that DODD 5000.29 has made into DOD policy many of the suggested solutions to software problems that were reviewed in Section II of this report. A continuation of this will be seen in the remainder of this section.

Air Force Regulations (AFR)

AFR 800-2, Program Management. This regulation implements DODD 5000.1. It is short, general in nature, and does not specifically mention software. AFR 800-2, does list the responsibilities of the Program Manager, which include tailoring the organization of the program office and the selection and application of management systems to the needs of the particular program (14:2). The Program Manager thus has some flexibility in his organization and can optimize for software visibility to some extent.

AFR 800-14, Volume I, Management of Computer Resources in Systems. This AFR applies to "all Air Force activities responsible for planning, developing, acquiring, supporting and using systems managed or acquired under AFR 800-2." (15)

This is an especially important document in as much as it is the first in the series of Air Force acquisition or management oriented regulations to specifically address software. The stated objective of AFR 800-14, Volume I, is to;

> "insure that computer resources in systems are planned, developed, acquired, employed, and supported to effectively, efficiently and economically accomplish Air Force assigned missions." (15:1)

There are several parts of the regulation of particular interest to the Program Manager from a software viewpoint. Under the heading of Air Force Policy it says that;

> "Computer resources in systems are managed as elements or subsystems of major importance"

during all life cycle phases. Although some programs had in effect done this for years, for most, implementing this would represent a significant departure from previous practice. This policy of course, has a ripple effect into all aspects of software development. I will treat this in more detail later. Also under Air Force Policy is a paragraph enumerating those things which "Program Management Directives (PMDs) require and Program Management Plans provide for" (15,2). Some of these are:

a. Establishment of computer technical and managerial expertise responsive to the Program Office (PO) which is independent of the system prime or computer program development contractor and, preferably, an organic capability of the PO (my emphasis).

b. The specification and allocation of system performance and interface requirements to be met by .... computer programs.

c. The identification of .... computer programs as Configuration Items (CIs).

d. Work Breakdown Structures(WBS) (MIL-STD-881) designed to facilitate identification of computer resource costs.

e. Coverage of ... computer programs during the conduct of system design reviews, audits and management assessments.

In Section B of Volume I, the Program Manager is given the responsibility to:

Provide management and technical emphasis to computer equipment and computer program requirements identified in the PMD. (14:3)

Each of the above items mentioned an area of concern from past software experience which was identified and discussed in the various studies and writings reviewed in Section II of this report. These items are examined in more detail in Air Force Regulation, AFR 800-14, Volume II.

Item (a) above can be interpreted as the charter for one very real way in which the PM can provide for software visibility. That way is through organization. The PM should exercise the considerable amount of flexibility given him by AFR 800-2 (14) to set up a Program Office organization with a specific focal point to manage the program software efforts. The need for this was noted by Keider in his article, "Why Projects Fail." (29:54)

Looking back at item (b) above, this provision of AFR 800-14 should in any case, be a fallout of the earlier stated policy of software being ".... elements or subsystems of major importance." (15:1) Because the PM can't be expected to review all system and interface specifications personally it is important that he see that the Program Management Plan (PMP) provides for the proper treatment of software in the systems engineering process. The importance of the systems engineering process in correctly defining software requirements is very difficult to overemphasize.

AFR 800-14, Volume II, Acquisition and Support Procedures for Computer Resources in Systems.

Volume II of AFR 800-14 consolidates procedures that apply when implementing the policies of AFR 800-14, Volume I and other publications as they pertain to the acquisition and support of computer resources. Volume II

restates applicable portions of related publications and must be used with them (16:1-1).

Volume II contains a great deal of detail which should be of concern primarily to those directly managing software, but there are several items in it with which the Program Manager should be familiar. One of these is that an Air Force Working Group will review the implementation of AFR 800-14, Volume II. A discussion of these items follows:

a. Planning is discussed as it relates to several specific functions and there are three in particular which are software peculiar. The Computer Resources Integrated Support Plan (CRISP) identifies organizational relationships and responsibilities for the management and technical support of computer resources. This is a cradle to grave plan for computer resources including software, which assigns responsibility for all areas of software acquisition and support. As such, it has great potential to aid or hinder the development of software within the constraints of cost, schedule and performance and should receive the effective backing of the Program Manager.

The Computer Program Development Plan (CPDP) is the development plan for the software. It is the responsibility of the implementing command, but may be (and usually will be) prepared by the contractor. This is a complete detailed development plan and should contain some items of particular interest from a software visibility viewpoint.

Among these are the contractor's development schedule for each computer Program Configuration Item (CPCI) (and the proposed milestone review points) and the procedure for monitoring and reporting the status of computer program development. The writings of the various authors reviewed in Section II underscored the importance of these items in maintaining software visibility.

The Computer Resource Working Group (CRWG) has as its prime purpose the preparation of the CRISP. The CRWG is initially chaired by the Program Office with representatives from the implementing and supporting commands. Because of its membership and its purpose, the CRWG could be very useful in integrating requirements and getting the user involved.

b. Engineering Management as applied to computer resources is described in terms of the system engineering process. AFR 800-14, Volume II states that one objective of engineering management is "that computer resources are managed as an integral part of the total system." (16:4-1)

Volume II discusses Formal Technical Reviews and describes what should be reviewed for computer software. (MIL-STD-1521 more specifically details the requirements of formal technical reviews). Although all reviews can aid in obtaining software visibility one aspect of critical design reviews should allow them to provide additional visibility. As Volume II points out, "... the CDR may be performed in stages as the

logical design of Computer Program Components (CPCs) or groups of CPCs is completed." (16-7-4)

c. Testing of Computer Programs will be conducted under the same general groundrules as the rest of the system. The principles of AFR 80-14 apply to testing of computer resources. (16:5-1) Development Test and Evaluation (DT&E) is divided into two areas, Configuration Item (CI) test and system level test. Each CPCI must be tested and established as a qualified item suitable for the system level test program.

d. Configuration Management, as specified in AFR 65-3, will be applied to each Computer Program Configuration Item (CPCI) throughout the system acquisition cycle. (16:6-1)

Volume II is very explicit in this requirement and further requires that computer program configuration management must not be fragmented from the overall system configuration management,.

Most of the chapter on Configuration Management in Volume II reiterates the procedures for applying configuration management to any part of a system (i.e., software being no different from hardware). MIL-STDS-480, 483, 490 and 1521 apply. One software peculiar item is the use of a Computer Program Identification Number (CPIN) for each CPCI. The CPIN is assigned by AFLC.

The identification of computer programs as configuration items in recent years has alleviated one of the major software problems of the past, the control of changes and maintenance of a known baseline. Software baselines are now established and changes are controlled through the ECP system. Because of the relative ease with which software can be changed there is a great tendency to use software to correct hardware/design errors and deficiencies late in system development. This isn't all bad, it is merely taking advantage of one of the inherent characteristics of software. There is, however, a danger in not recognizing the great impact minor changes can have on schedule and cost due to the retesting and reverification required.

Although Volume II says, that configuration management will be applied to each CPCI, it does not provide any guidelines as to what computer programs should be CPCIs. In many instances certain items of support software built by the development contractor should be designated deliverables (and CPCIs) as well as the operational software. This support software could be either that required for support during the production/deployment phases, or software tools useable in the development of other operational software. The Program Manager should ensure that his staff is aware of his policy on support software and that it is designated as a CPCI when appropriate.

e. Documentation is needed during development in order to track progress and provide information for management visibility and decision making. (16:7-1)

The above was quoted from Volume II, Chapter 7, "Documentation" because it emphasizes one of the more frustrating aspects of software, i.e., the fact that the only way to see progress (or the end product) in software is through documentation.

Data management in general is handled through a data management office in accordance with AFR 310-1 and the use of techniques such as deferred ordering, deferred requisitioning (described in AFR 310-1 and Armed Services Procurement Regulation (ASPR) Section 7) and an accession list. These can be used to advantage to optimize the data received. Volume II lists five categories of documents usually prepared by the contractor and used for performance monitoring: Configuration management, engineering, test, operation, and support. Of particular interest are specifications (engineering documents), because they (1) document requirements (Part I, the development specification) and (2) document the actual computer program as coded (Part II, the product specification). Computer program specifications are written in accordance with MIL-STD-490, MIL-STD-483 and MIL-STD-83490. To quote from AFR 800-14, Volume II:

> Specifications provide the basis for documenting requirements, controlling the incremental development between major program milestones and <u>providing visibility</u>. (my emphasis)

f. Contractual requirements are discussed in Volume II, Chapter 8. It describes different means of including software in the contract for a prime contractor, but does not mention the possibility of contracting separately for software (see Section II of this report).

The point is made that the Program Manager should ensure that instructions to offerors provide for preliminary contractor plans which describe the computer program development concept (this would be the CPDP described earlier).

Computer programs should be identified at level three in the project Work Breakdown Structure (WBS) (16:8-2). This statement from Volume II is of major significance to the Program Manager. In a major program particularly, such as an airplane or a missile system, putting software at level three of the project WBS could pose some problems as well as provide some real benefits. On the benefits side, to quote Volume II:

> Identification of computer program configuration items at level three of the WBS will provide the visibility necessary to evaluate cost, schedule, and performance of contractor efforts. (16:8-2)

On the other side, if there is a large amount of software, consisting of several computer programs associated with different hardware (and perhaps being developed by different contractors) then putting all software at level three of the WBS might not be feasible. It might be quite difficult to correlate the WBS and the specification tree with the way the software is actually being procured.

MIL-STD-881A, Work Breakdown Structures for Defense Material Items, (9) has a series of appendices which show a summary Work Breakdown Structure for several types of systems (e.g., aircraft, missile, electronics). The only system for which software (computer programs) is listed at level three of the WBS is electronics systems. Software is not shown at all in the other system WBSs.

Chapter 8 of Volume II, also discusses software as a contract deliverable. It states that "contract deliverables are specified as line items in the contract," and that "while computer programs and documentation must be listed on the DD 1423, the DD 1423 should be identified as an exhibit or attachment depending on the required management emphasis." (16:8-2) An AFSC supplement to the ASPR, Section 9-603 expands on this direction. It also requires computer software/computer programs/computer data bases be specified as line/subline items in the contract schedule. There is still a dual treatment of software as a line item and as "data." This is handled by requiring that delivery of computer/software/computer programs/computer data bases documentation be specified on separate DD Forms 1423 from the actual cards, tapes etc., which represent the real product.

Other Documents

AFSC Pamphlet 800-3, A Guide for Program Management. This pamphlet describes the general considerations involved in managing the acquisition of a system. It is intended as a guide and does not specify a single

inflexible procedure through which all program goals are achieved. (19:1-1) Chapters 1 through 5 trace the acquisition process through its different phases, then the rest of the document presents a general description of the principal functions involved in managing systems acquisition programs. In the following paragraphs, I will attempt to extract and discuss those portions that are of interest to the Program Manager from a software peculiar viewpoint.

a. Chapter 8, "Engineering Management," mentions computer programs several times for purposes of emphasis in differeniating between hardware and software and to point out peculiarities of software in the systems engineering process.

The point is made that a portion of a System Design Review (SDR) should address the allocated requirements for computer programs and interfacing equipment. (19:8-5)

In the discussion of Critical Design Reviews (CDR), AFSCP-800-3 says, the purpose of a CDR for a CPCI is to establish the integrity of computer program design at the level of flow charts or computer program logical design prior to coding and testing. (19:8-5) This view of the CDR in relation to the software development process is an idealistic one. In practice, the exigencies of schedule and money will often force coding to start prior to CDR. In fact, some software managers go so far as to consider the CDR as a logical event to separate coding from the start of validation/verification.

b. Under Configuration Management, software is mentioned only to the extent of pointing out that the selection of Configuration Items below prime-item level is a management decision accomplished through the system engineering process and that each computer program is identified and documented by one top flow chart. (19-9-4)

c. Data Management is the only chapter which has a section devoted to software. Section D, Chapter 16, is titled "Acquisition and Support of Computer Programs." I must state at this point that I feel this section has been placed in the wrong chapter and that AFSCP 800-3 should be revised to reflect the fact that computer programs are <u>not</u> data, in accordance with the guidelines of AFR 800-14 and the ASPR.

This section of 800-3 lists several things which should be addressed in determining how to satisfy operational requirements.

A heavy emphasis is placed on the role of the Program Manager in acquisition of computer programs as evidenced by the following quote:

> Early identification of computer resources, and technical and management expertise within the Program Offices is needed to manage and engineer the acquisition of functional subsystems that incorporate computer programs. The Program Manager must provide the management expertise to focus attention on computer program development and integration across the total system. (19:16-8)

SECTION IV

THE PROGRAM MANAGER'S VIEWPOINT

The preceding portions of this study involved a review of current published articles and studies and Department of Defense documents on the subject of software. In this section I have tried by means of interviews, to ascertain how some Program Managers do in fact obtain software visibility. Three Air Force Program Managers were interviewed and the results of these interviews are presented below. The opinions expressed in this section are those of the person interviewed. The intent of this section is to provide actual Program Manager's experience and opinions for comparison with the contents of the previous sections.

a. Colonel Larry McKenna is the Program Director for the EF-111A, an Electronics Countermeasures (ECM) program (32). This project includes three different computers in three different ECM systems and a considerable amount of software.

Colonel McKenna is a firm believer in the matrix organization and the EF-111A Program Office is organized that way. The software is managed by three project officers, each of whom has hardware as well as software. Each project officer is responsible for seeing that his software meets cost, schedule and performance thresholds. From the technical viewpoint the three software areas are coordinated by a branch chief who works for the chief engineer. Four software engineers, two military and two civilian, are assigned to the program office. The Program Manager holds weekly status reviews with the project officers to monitor progress and discuss any problems with cost,

schedule or performance.

Software is located at level four in the Work Breakdown Structure (WBS) and is broken out three levels below that (to Level 7). According to Colonel McKenna, this was essential in obtaining proper visibility. In fact, he indicated he could go to the work package level to ascertain the cause of problems if required. The reader might note that AFR 800-14, Volume II suggests identifying software configuration items at level three of the project WBS (16:8-2).

In the use of Cost Performance Reports (CPR), Colonel McKenna exercises management by exception. The top levels are reviewed and if there are problems then a more detailed investigation is made down to whatever level is required to determine the cause of the problem. A detailed review of software could not be made unless the high level review indicated a need to go deeper. Another report received monthly from the contractor is a Research and Development (R&D) Status Review. This document provides the technical status of each work package. The program office staff beat the CPR and the R&D Status Review together to compare cost, schedule and technical status to ensure the data matches and that no problems are masked by looking at one or two elements only.

Colonel McKenna emphasized the need to track software progress by having measureable, achievable milestones. He indicated the EF-111A program employs the use of incremental Critical Design Reviews (CDRs) in order to track progress closely. (This technique is recommended in MIL-STD-1521 (11)). The use of incremental CDRs has been used in other programs, such as Minuteman, and has the advantage of allowing a review of individual Computer Program

Components (CPCs) to a level of detail not practicable with a single CDR for an entire computer program.

The EF-111A program office has a contract with an independent agency (a contractor independent of the software developing contractor) for Independent Validation and Verification (IV&V). This is another means of ensuring compliance with requirements and gaining technical visibility. (See Section II of this report).

In summary, Colonel McKenna emphasized treating software with the same importance as hardware, and not ignoring or trying to hide software although there is a tendency to do so, because it is not as well understood as hardware and people are therefore uncomfortable with it.

b. Colonel Delbert H. Jacobs is the Deputy System Program Director for the F-16 Lightweight Air Combat Fighter program. (28)

Colonel Jacobs cited several software problems which the F-16 program office was working to overcome or avoid, some of which are typical of many software efforts and some peculiar to the F-16. Typically software is designed last (after hardware) and the software is used to correct hardware errors. (See Bibliography item No. 4 for a discussion of "Software First Concepts"). This has a schedule and technical impact which can drive the software cost 3 to 4, or even as much as 10 times, over budget. Another problem cited was that of ineffective means for tracking software progress. This often means that all of the software must be completed before you find out whether any of it works or not. Treating software like hardware and establishing milestones for its development helps, but is no cure - all.

The organization of the F-16 Program Office is being changed relative to software to provide better visibility and control. Under the old organization the Deputy Chief Engineer was responsible for software as part of his duties. There were no people dedicated to software and there was a lack of continuity among the people who were working on it. The new organization being implemented puts responsibility for software in the Projects Directorate with the creation of a Software Division in the Projects Directorate. The Software Division will consist of three branches as follows:

(1) Operational Software Branch

(2) Support Software Branch

(3) Interface/Integration/Independent Validation and Verification Branch

The functions of the first two branches are apparent from their titles. The third branch performs an integration function between hardware and software, oversees the IV&V effort and also monitors cost, schedule and performance. Colonel Jacobs expects this new organization to be much more responsive to their software needs. (See Section II, General, of this report for a discussion of organization).

The contractor organization was also named as a problem. The software is buried 4 to 5 levels down in management and is not managed as an entity.

The contractor organization problem is also related to the placement of the software in the WBS. It is at level five or six and individual pieces of software are at different levels. An attempt was made to raise the software to level three, but was dropped due to contractor claims of huge cost increases to do it.

The F-16 approach to Independent Validation and Verification (IV&V) is different from most in that it is being done by other government agencies rather than a contractor. The IV&V task is shared by the Avionics Laboratory at Wright-Patterson Air Force Base and the Ogden Air Logistics Center, Hill Air Force Base, Utah. There is also a low level of independent contractor support to the Program Office.

Colonel Jacobs indicated the F-16 Program Office was well aware of the software problems which other programs have encountered and hope to avoid them or overcome them.

c. Colonel Dale Newbold is the System Program Director for Drones and Remotely Piloted Vehicles (RPVs). (33)

The Drone/RPV System Program Office (SPO) has a large number of separate development programs in it, each with a Program Manager who reports to the System Program Director, Colonel Newbold. The SPO is heavily matrixed, an organization type receiving strong support from higher echelons. Each individual Program Manager is responsible for the software on his program but there is no individual whose sole responsibility is software or one (other than the Program Manager) who has sole responsibility for the software. Because of the matrix type of organization there are no engineers assigned full time to the software. To provide assistance and continuity in managing the software effort of the development contractor, software engineering support is obtained from a separate contractor.

In all programs the software was contracted for as a subelement of the prime contractor. Software has not been acquired from associate contractors.

Software, in all instances, has been at a low level in the Work Breakdown Structure (WBS). Low level here means some number of layers below level three. The software breakout in the WBS has not been very detailed. There had been plans to make the software breakout more detailed but the effort was dropped due to projected high costs.

Cost/schedule reporting from the contractor includes a breakout for software in terms of dollars and manhours expended in the software development effort.

Colonel Newbold, when asked about AFR 800-14, said that only one new program had been initiated since the publication of AFR 800-14. This regulation requires establishing a Computer Resources Working Group (CRWG) (16:3-5) and writing and maintaining a Computer Resources Integrated Support Plan (CRISP) (16:3-4). Colonel Newbold stated that the requirements for the CRWG and CRISP had not been completely implemented due to the cost involved.

Independent software verification is used to ensure that the software is error free and meets its performance requirements. This independent validation and verification is acquired by contract with a company separate from the developing contractor.

Colonel Newbold emphasized his belief that software must be considered an important part of the total system development and must be monitored closely to ensure adequate results.

SECTION V

SUMMARY AND CONCLUSIONS

As software has become an increasingly large proportion of most weapon system developments, the problems of cost, schedule, and performance in the software have become critical to the successful fielding of those weapon systems. These costs, schedule and performance problems have pervaded all phases of software development and have been the result of some seemingly unsolvable problems and various sins of omission as well as commission. Some of the more important are: (1) poor requirements definition; (2) not properly engineering the software into the weapon system as a total system; (3) being unable to track progress, particularly as it relates to schedule and performance during the implementation and verification phases; (4) poor change and configuration control which allows changes to drive cost and schedule beyond acceptable limits; (5) test and verification not properly related to requirements; and (6) support software is not available when needed thus causing maintenance problems and higher maintenance costs.

Software does not have exclusive rights to all of these problems; hardware is often subject to them also. However, software has been more prone to suffer from them, due to its newness and the failure to recognize its importance. There are ways to alleviate most of the problems. If the Program Manager is going to maintain visibility of, and control cost, schedule and performance problems, he must recognize the potential for them to occur and take the necessary steps to prevent them. Among the more significant steps the Program Manager can take are:

(1) Get the user involved early. Insist on early statement of user requirements and meaningful user participation in design reviews.

(2) Insist on full incorporation of software into the system requirement analysis (SRA) process. Software must be engineered as an integral part of the weapon system.

(3) Place software at a high level in the WBS and get it out of the category of "data." It will be extremely difficult to get proper contractor attention on it otherwise.

(4) Make full use of such planning aids as the program management plan and the CRISP to ensure all members of the program management team know what is expected and required.

(5) Plan to make support software a deliverable item and make it a configuration item when applicable (particularly appropriate when it is to be transferred to the support or using command).

(6) Organize the program office to provide adequate technical support for software and to have someone responsible and accountable for it. Otherwise, control will be lost, because the Program Manager cannot be the integrator.

(7) Plan the total program budget to provide adequate funds to implement the total software development program.

If there is any one thing that is present in all aspects of what the Program Manager must do to obtain visibility it is planning. There is an old saying in the Real Estate Business that lists the three most important things to consider when buying a house: they are location, location and location. An analagous comment on software would be that the Program Manager who wants software visibility would do well to pay prime attention to planning, planning and planning. Experience has shown that; if you don't plan to include software in the SRA, it won't be; if you don't plan for the use of a HOL, there won't be enough computer memory to handle it; if you don't plan early for the funds to support proper software development, they won't be there.

One point to be made from this is, that the Program Manager can do little to alleviate problems of poor visibility and control late in the development effort. It is in the early stages that the proper steps can be implemented to assure the needed visibility later on. Therefore, the extent to which a Program Manager has visibility and control over the potential problems of software is a direct function of how he plans for its development.

RECOMMENDED SUBJECTS FOR FURTHER STUDY

1. The cost of placing software at level three in the WBS. Two of the three Program Managers interveiwed stated that they were prevented from putting software at a high level in the WBS and making it a detailed breakout by high costs claimed by the contractor.

2. Methods for measuring the progress of software coding, checkout and testing. There are currently no known methods for accurately measuring software development progress such that it can be effectively reported through such means as the CPR or C/SSR.

APPENDIX A

DEFINITIONS

A. Computer Data. Basic elements of information used by computer equipment in responding to a computer program.

B. Computer Equipment. Devices capable of accepting and storing computer data, executing a systematic sequence of operations on computer data or producing control outputs. Such devices can perform substantial interpretation, computation, communication, control, and other logical functions.

C. Computer Firmware. The logical code of computer equipment which interprets the control functions of that equipment.

D. Computer Program. A series of instructions or statements in a form acceptable to computer equipment, designed to cause the execution of an operation or series of operations. Computer programs include such items as operating systems, assemblers, compilers, interpreters, data management system, utility programs, and maintenance/diagnostic programs. They also include application programs such as payroll, inventory control, operational flight, strategic, tactical, automatic test, crew simulator, and engineering analysis programs. Computer programs may be either machine dependent or machine independent, and may be general purpose in nature or be designed to satisfy the requirements of a specialized process of a particular user.

E. Computer Resources. The totality of computer equipment, computer program, computer data, associated documentation, personnel, and supplies.

F. Computer Software. A combination of associated computer programs and computer data required to enable the computer equipment to perform computational or control functions.

G. Embedded. Adjective modifier; integral to, from the design, procurement, and operations point of view espoused in DOD Directive 5000.1.

H. Software Engineering. Science of design, development, implementation, test, evaluation, and maintenance of computer software over its life cycle.

BIBLIOGRAPHY

1. Bartlett, Jan C., et al, "Software Validation Study," Logicon Incorporated, March 1973.

2. Barklund, C. W., "Getting the User Into the System," Government Executive, August 1975.

3. Bolen, N. E., "An Air Force Guide to Contracting for Software Acquisition," Mitre Corporation, Bedford, Ma., January 1976, (AD-A020 444).

4. Bullen, Richard H., "Engineering of Quality Software Systems: Software First Concepts," Mitre Corporation., Bedford, Ma., January 1975, (AD A007 768).

5. Department of Defense, Directive 5000.1, Acquisition of Major Defense Systems, 22 December 1975.

6. Department of Defense, Directive 5000.29, Management of Computer Resources in Major Defense Systems, 26 April 1976.

7. Department of Defense, MIL-STD-480, Configuration Control of Engineering Changes, Deviations and Waivers, Washington, D.C., October 1968.

8. Department of Defense, MIL-STD-483, Configuration Management Practices for Systems, Equipment, Munitions and Computer Programs, Washington, D.C., 1 June 1971.

9. Department of Defense, MIL-STD-490, Specification Practice, Washington, D.C., 18 May 1972.

10. Department of Defense, MIL-STD-881, Work Breakdown Structures for Defense Material Items, 1 November 1968.

11. Department of Defense, MIL-STD-1521 (USAF), Military Standard: Technical Reviews and Audits for Systems, Equipment and Computer Programs, Washington, D.C., 1 September 1972.

12. Department of Defense, Program of Research, Development, Test and Evaluation, FY 77, Statement by the Director of Defense Research and Engineering to the 94th Congress, Second Session, 1976.

13. Department of the Air Force, "Air Force Regulation 122-9, The Nuclear Safety Crosscheck Analysis and Certification Program for Weapon System Software," 1 July 1974.

14. Department of the Air Force, Air Force Regulation 800-2, Program Management, Headquarters, US Air Force, Washington, D.C., 16 March 1972.

15. Department of the Air Force, Air Force Regulation 800-14, Volume I, Management of Computer Resources in Systems, Headquarters, US Air Force, Washington, D.C., 12 September 1975.

16. Department of the Air Force, Air Force Regulation 800-14, Volume II, Acquisition and Support Procedures for Computer Resources in Systems, Headquarters, US Air Force, Washington, D.C., 26 September 1975.

17. Department of the Air Force, Air Force Regulation 800-6, Program Control - Financial, Headquarters, US Air Force, Washington, D.C., 14 July 1972.

18. Department of the Air Force, Air Force Systems Command Pamphlet 173-3, Cost Management for Small Projects, 1 October 1975.

19. Department of the Air Force, Air Force Systems Command Pamphlet 800-3, A Guide for Program Management. 9 April 1976.

20. De Roze, Barry C., An Introspective Analysis of DOD Weapons System Software Management," Defense Management Journal, October 1975.

21. De Roze, Barry C., Weapon Systems Software Management, an address Presented to the Software Task Groups of the Defense Science Board, 25 July 1975.

22. "DOD Weapon System Software Acquisition and Management Study," MITRE Corporation, MTR 6908, Volume I and II, June 1975.

23. "DOD Weapons Systems Software Management Study," John Hopkins University Applied Physics Laboratory Technical Report, June 1975. (AD-A022 160)

24. "Effects of Management Philosophy on Software Production," MITRE Corporation, January 1975. (AD-A 007 76)

25. Etheredge, Boyd, MAJ, USAF, "Computer Software Management From the Point of View of the Systems Manager," Air Command and Staff College Research Study, Air University, Maxwell AFB, Alabama, May 1974. (AD 920559)

26. Grooby, John A., "Maximizing Return on EDP Investments," Data Management, September 1972.

27. "Investigation of Software Problems," Space Transportation System Software Concepts Development Study, Volume II, TRW Systems Group, Redondo Beach, California, 31 July 1972. (AD 902103L)

28. Jacobs, Delbert H., COL, USAF, Deputy, System Program Director for the F-16 Lightweight Fighter ASD(YP), Wright-Patterson AFB, OH, 45433, Telephonic Interview, 16 September 1976.

29. Kieder, Stephen P., "Why Projects Fail," Datamation, December 1974.

30. Mangold, Eldon R., "Software Visibility and Management," Proceedings of the TRW Symposium on Reliable, Cost - Effective, Secure Software, March 1974.

31. Manley, John A., LT COL USAF, "Embedded Computer Systems Software Reliability," Defense Management Journal, October 1975.

32. McKenna, Lawrence, COL, USAF, System Program Director of the EF-111A Electronics Countermeasures Program ASD(SD25), Wright-Patterson AFB, OH, 45433.
Interview at the Defense Systems Management College, 1 September 1976.

33. Newbold, Dale, COL, USAF, System Program Director for Drone/RPV Programs ASD(YMR), Wright-Patterson AFB, OH, 45433.
Telephonic Interview, 21 September 1976.

34. Reifer, D.J., "A New Assurance Technology for Computer Software," The Aerospace Corporation, El Segundo, California, 1 September 1975. (AD A020483)

35. Richards, Russell F., "Computer Software: Testing Reliability Models and Quality Assurance," Naval Postgraduate School, Monterey, California, July 1974. (AD-A001260)

36. "Software Improvement Plan Pushed," Aviation Week, 5 April 1976.

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