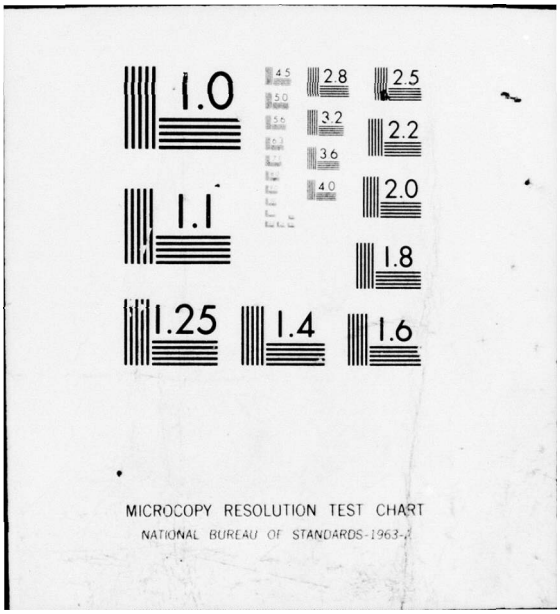
AD-A042 937 DEFENSE SYSTEMS MANAGEMENT COLL FORT BELVOIR VA F/G 9/2
SOFTWARE MANAGEMENT: CURRENT TRENDS AND A NAVY APPLICATION.(U)
MAY 77 H W TAYLOR

UNCLASSIFIED NL

OF
AD
A042937

END
DATE
FILMED
9-77
DDC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

DEFENSE SYSTEMS MANAGEMENT COLLEGE

PROGRAM MANAGEMENT COURSE
INDIVIDUAL STUDY PROGRAM

SOFTWARE MANAGEMENT: CURRENT TRENDS AND A NAVY APPLICATION

STUDY PROJECT REPORT
PMC 77-1

Harold W. Taylor Jr.
GS-12 DNC

FORT BELVOIR, VIRGINIA 22060

SOFTWARE MANAGEMENT:

CURRENT TRENDS AND A NAVY APPLICATION

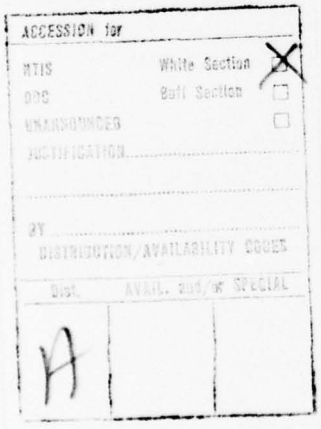
Study Project Report

Individual Study Program

Defense Systems Management College

Program Management Course

Class 77-1



by

Harold W. Taylor Jr.

GS-12 DNC

May 1977

Study Project Advisor
LCDR Sue Anderson, USN

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)* SOFTWARE MANAGEMENT: CURRENT TRENDS AND A NAVY APPLICATION		5. TYPE OF REPORT & PERIOD COVERED Study Project Report 77-1 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) HAROLD W. TAYLOR JR.		8. CONTRACT OR GRANT NUMBER(s)
9. PERFORMING ORGANIZATION NAME AND ADDRESS DEFENSE SYSTEMS MANAGEMENT COLLEGE FT. BELVOIR, VA 22060		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS
11. CONTROLLING OFFICE NAME AND ADDRESS DEFENSE SYSTEMS MANAGEMENT COLLEGE FT. BELVOIR, VA 22060		12. REPORT DATE 77-1 13. NUMBER OF PAGES 57
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)*

UNLIMITED

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution 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

SECURITY CLASSIFICATION OF THIS PAGE *(When Data Entered)*

DEFENSE SYSTEMS MANAGEMENT COLLEGE

STUDY TITLE: Software Management: Current Trends And a Navy Application

STUDY PROJECT GOALS: To examine the current trends in software management for weapon systems acquisitions and to understand the approach being taken in the Navy's AEGIS Combat Systems.

STUDY REPORT ABSTRACT: Management of embedded computer software in weapon systems is an important and current consideration. A large portion of the Defense budget is spent in software acquisition and support and the management techniques for making this a productive and cost effective process must be developed. This paper addresses the issue from two aspects. First, a study of the current trends in software management was conducted with particular emphasis on DoD sponsored activities. This phase was conducted by studying the current literature and DoD publications concerned with software management.

Secondly, the software management and development approach being taken in the Navy's AEGIS Combat System was studied. This aspect of the report was conducted through personal interviews with representatives of the AEGIS Program Office (PMS-403) and the Navy Lead Laboratory (Dahlgren, Virginia) and through an in-depth study of the available documentation.

Conclusions reached in this report include the identification of some major cornerstones that have the common goal of converting an art form into technology by application of sound planning, implementation, and monitoring techniques. One must consider the intangible nature of software to understand the problems that have plagued its management. It is now realized that the powerful black art cannot be practiced in a vacuum when it represents such a large contribution to the system in which it resides.

Subject Descriptors: Software Management, Computer Software, Computer Programs, Embedded Computer Programs, Embedded Computer Systems, AEGIS Computer Programs

NAME, RANK, SERVICE	CLASS	DATE
Harold W. Taylor Jr. GS-12 DNC	DMC 77-1	May 1977

EXECUTIVE SUMMARY

Management of embedded computer software in weapon systems is an important and current consideration. A large portion of the Defense budget is spent in software acquisition and support and the management techniques for making this a productive and cost effective process must be developed. This paper addresses the issue from two aspects. First, a study of the current trends in software management was conducted with particular emphasis on DoD sponsored activities. Some major cornerstones of successful software management are developed throughout the paper and are summerized in the conclusion section of this paper. Secondly, the software management and development approach being taken in the Navy's AEGIS Combat System was studied. This proved to be an informative and constructive endeavor that tends to give real-world meaning to the application of software management techniques.

ACKNOWLEDGEMENT

Appreciation is extended to LCDR Sue Anderson, of the DSMC staff, who served as my Study Project Advisor in the formulation and conduct of this project. Her expertise and guidance proved invaluable in making this project a truly rewarding experience.

A special note of appreciation goes to my wife, Fran, and my children , Tony and Tracy, for their patience and understanding during the twenty long weeks of the Program Management Course at DSMC.

TABLE OF CONTENTS

TABLE OF CONTENTS (con't)

SECTION I

INTRODUCTION

1. Purpose

The purpose of this paper is two-fold. First an attempt is made to examine the issue of software management from a "textbook" viewpoint. Recent publications, articles, reports, books and studies will be reviewed and anaylzed for determining current trends in the area of software management. Under the full realization that no cookbook procedure will be found, hope remains that some basic cornerstones will be identified that are essential to effective management of software. An attempt to arrive at a singular solution to the problem would be an act of simplism. However, if an interrelated group of events can be established that identify the hurdles that must be cleared, progress will have been made.

Secondly, a real-world application of software management will be discussed. One of the current major weapon system acquisitions in the Navy, the AEGIS Combat System, will provide the data base for this discussion. The intent of this portion of the paper is not to compare software management in AEGIS with the check-off list of good practices because by previous admission, no such list exist. It is intended to be an education process for the author and the reader. Perhaps some of the "textbook" trends will be

reinforced by the AEGIS approach; perhaps some new lights will glow.

As for the motivation to choose a topic such as software management, two statements of need are offered:

> J. S. Gansler OASD(I & L):
> The functional applications of software within Department of Denfense (DoD) weapon systems pervade almost every program.
>
> The most critical issue facing DoD is the increasing use of and dependency on software in weapon systems without the proven management and production methods necessary to control its direct and indirect cost.
>
> The second major issue concerns the need to convert software from an art to a technology. The notion that software advancements are the sole domain of "rugged individualists" or "artists" must be discarded. (6:1)[1]

DODD 5000.29

> Annual expenditures by DoD on the design, development, acquisition, management, and operational support of computer resources embedded within, and integral to weapons, communications, command and control, and intelligence sensor systems are measured in the billons of dollars. Unreliability, particularly of software, diminishes DoD mission effectiveness in many major Defense Systems.
>
> Computer resources in Defense Systems must be managed as elements or subsystems of major importance during conceptual, validation, full-scale development, production, deployment, and support phases of the life cycle, with particular emphasis on computer software and its integration with the surrounding hardware. (8:2)

1* This notation will be used throughout this paper for sources of quotations and major references. The first number is the source listed in the bibliography. The second number, if listed, is the page in the reference.

2. Goals

The author of this paper has five years experience in providing the life support for software contained in Navy missile systems. This has been primarily in the acquisition phases of full scale development, production and deployment. This paper is viewed as an educational opportunity to examining the entire acquisition process of major weapon systems, including conception and validation, and the considerations given to software management therein. It was felt that an examination of the current literature was a prerequisite for such a venture and forms the basis for the first of two goals, ie, examine the current trends in software management.

Once the textbook teachings are filed away it becomes a natural follow on to examine the real world. Hence goal number two; understand the software management approach in a real world application. The Navy AEGIS Combat System was chosen for several reasons which include its currentness, its complexity and its heavy use of computer programs.

3. Scope

This paper is intended to provide a condensation of the current trends in software management and to relate the findings of an educational examination of a real world application. It is intended for the use of anyone involved in software management of embedded computer programs. A point made in the Purpose section needs to be re-emphasized. Is is not in the

scope of this paper to critique the software management approach being taken in the AEGIS Combat System. The review of a real world application is being used to provide additional coverage of the topic.

4. Limitations

Due to the author's occupational interests and the limited time available for the generation of this paper the subject matter has been limited to embedded software applications. The interrelation between the weapon system acquisition cycle and the software acquisition cycle will necessitate that a certain amount of the big picture to be portrayed. We are concerned here with the software management of the embedded computer programs in a weapon system acquisition.

The choice of a Navy application for the real world part of the paper has been made for the reasons previously stated. This is not to imply that the concepts and realizations presented in this paper do not apply equally as well to the other Services.

5. Organization and Methodology

In addition to this introductory section, the paper has been organized into two major sections and a summary section. Section II examines the current trends in the area of software management from a textbook approach. Various types of reference literature, including articles, reports, study

papers, textbooks and DoD publication and directives, were studied and analyzed in an attempt to gain insight on the key considerations in software management. Many areas of software management and development will be discussed with varying degrees of intensity. Of particular interest will be the identification of basic cornerstones of successful software management and their relation to the Defense acquisition cycle. Any conclusions drawn will be based on the authors evaluation of the information studied combined with personal experience and involvement in the software acquisition and management arenas.

Section III will be an educational process in which a real world weapon system acquisition will be studied. In order that such an effort be accomplished, it was necessary to obtain and study the available documentation for the selected acquisition process. In addition, personal interviews were conducted with the Program Manager's office and the Navy Lead Laboratory. The interview questionnaire used for this purpose is enclosed as Appendix A to this report.

Section IV is the summary containing conclusions and recommendations.

SECTION II

CURRENT TRENDS IN SOFTWARE MANAGEMENT

1. Background

The application of computing technology to our evolving life style has been natural and rapid. Man has continually been in search of ways to advance his state of being and improve his environment. Computational advancements have been based primarly on the physical sciences which have had the effect of creating faster and larger computing systems with which to solve our problems. This has been a successful and efficient evolution, for the most part, because of the well developed skills that were directed to the developments. Man has been actively concerned and involved in advancing the state of the art, particularly in areas that are not art. Generation of smaller, faster and more reliable circuits are a direct result of our time tested techniques in hardware design, development and testing. We know how to specify what we want, how to build it, how to nurture our technology base for future breakthroughs, how to test it and how to maintain it. In summary we are very effective in advancing the hardware aspect of computing technology.

In the world of weapon system acquisition it is not too difficult for officals to visualize and understand the process of hardware development and hence, the application of management skills, which have been developed from our previous

experiences and experiments involving hardware technology, often prove to be successful. We are adept at specifing a widget to be twelve meters, plus or minus two centimeters. It is fairly simple to determine if the manufacturing technology (state of the art) can make it. Also it is easy to test; we just measure it against a standard unit. We certainly know how to maintain it by establishing a physical set of conditions to which it must always conform. If it doesn't, we tweek it, repair it or replace it. The management approach throughout this entire process is based on the physical presence of the acquisition. If we need to know how our widget is coming along, we take a look at it.

All the above is not intended to simplify the management of hardware acquisitions. One of the things we know to be fact is change and with it we must adapt. This fact makes management of anything a real challenge. The point of the above is to provide a jumping off point for the topic of this paper, ie, software management. If we accept the position that in fact we are experiencing undue difficulties in the area of software management, we can direct our attention to possible causes and corrections. Most any of the references in the bibliography go a long way toward the affirmation of this position. We will take advantage of some of the key points made in the literature as we further develope the subject.

2. Guidance Documents

The following statement of concern contained in an memorandum from the Office of the Secretary of Defense in 1974 characterizes the increased emphasis on the part of DoD on the solution to the problems of increasing software cost, schedule slippages and technical reliability that many weapon system acquisitions are encountering.

> 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 Department of Defense (DoD) weapon systems constitutes serious technical and management problems that must be solved if we are to have the weapon systems that are needed for national security (10:1)

Many of the management and technical problems associated with the acquisition of weapon system software arise from the attempt to use hardware acquisition techniques and guidance documents. The Department of Defense and Service Components have many policies and procedures and related documentation (Directives, Instructions, Standards, etc.) that are concerned with hardware acquisition. Very few of these deal with computer software acquisition and even fewer deal with embedded computer software. In recent years there has been a trend to treat Automatic Data Processing (ADP) separately from Embedded Computer System (ECS). This is a step in the right direction in that a basic difference exists in the two areas. In ADP the product is a computing system and the

associated programming. In ECS the software product is often difficult to define, or evaluate for that matter, except in terms of the weapon system in which it resides.

The OSD memorandum referenced above established a joint DoD/Service Software Steering Committee with an objective to determine methods to reduce and control software costs and improve software reliability, standardization and maintainability. The initial task of the committee was to oversee software management studies by the Applied Physics Laboratory of John Hopkins University (APL/JHU) and the Mitre Corporation. Each was to conduct separate, but coordinated, four month studies to identify and define:

1. the nature of the critical software problems facing the DoD,
2. the principal factors contributing to the problems,
3. the high payoff areas and alternatives available,
4. and the management instruments and policies that are needed to define and bound the functions, responsibilities and mission areas of system software management (10:1)

The second phase of the study was an indepth study into the critical areas identified as a result of the initial four month studies.

APL/JHU conducted their study in three parts. First, ten recent DoD sponsored studies in software management were analyzed. Secondly, the software design and management approach being employed in ten Navy and two Army Weapon Systems were reviewed. Thirdly, discussions with service and industry

organizations involved in software acquisition were conducted. The findings presented in the study report included seven categories of problem areas with recommended actions in each. In essence the categories and recommendations are as follows (3:2-1):

a. Management Policy - Comprehensive analysis and definition of software requirements down to the major function level should be conducted in the Validation Phase. Promote software visibility in terms of configuration control items, DSARC reviews, design reviews and other aspects of acquisition management. Specify that software be designated as Configuration Items (CI) and deliverables during Full Scale Development Phase including computer programs and computer data for operational software development, support software and test and integration software.

b. Acquisition Planning - Establish milestones and achievement criteria in the Full Scale Development Phase to ensure the proper sequence of analysis, design, implementation, test and integration. Require a detailed Computer System Resource Development Plan as part of the bid package of Full Scale Development contracts.

c. Systems Engineering - Establish functional segments in accordance with the operational requirements and conduct hardware/software trade off analysis

in the Validation Phase. Ensure the software design makes provision for growth to accommodate uncertainties and changes in system requirements. Specify the use of modular software architecture and top-down design.

d. Implementation Procedures - Include in Full Scale Development, provision for adequate modern support tools and facilities and the transferring of same to the Operational Support (Maintenance) Agent. Apply highly disciplined engineering practices to software development and require a progressive system integration and test capability.

e. Program Management Support - Require appropriate technical staffing in the Program Manager's Office with experience in systems engineering and software development. This should include a Systems Engineering Agent and a Software Operational Support Agent.

f. Acquisition Management Standards - Establish a set of requirements and criteria to be applied in the acquisition and support of weapon system computer resources by all services and prepare a series of handbooks and guides covering important aspects of software acquisition.

g. Development of Tools and Techniques - Support the development of software test and validation

tools to reduce the cost and time involved in software verification.

The Mitre study resulted in the identification of four high payoff areas in the software acquisition process in which corrective actions should be taken (13:1)

a. Software Performance Specification - Establishment and consistent application of engineering principals/practices to the process of specifying and validation software requirements.

b. Software Acquisition Planning - Early and complete software life cycle planning and establishment and application of management practices/strategies specifically for software.

c. Software Technology - High leverage technology programs needed to further improve software practices and techniques.

d. Personnel - Provision for knowledgeable and experienced DoD software management and software engineering personnel.

In addition to the two reports described above there have been many similiar studies undertaken at all levels in the Department of Defense, Joint Logistics Commanders, Service Components and Industry. Department of Defense Directive

5000.29 of April 1976 is the result of the DoD Software steering Committee actions and recommendations. Some of the major areas addressed are (8);

* Validation of computer resource requirements, including software, interface control and integration methodology definition will be conducted during the Concept and Validation Phases, prior to DSARC II
* Computer software will be specified and treated as configuration items.
* A computer resource plan will be developed prior to DSARC II and will be maintained throughout the life.
* Support items required to develop and maintain computer resources will be specified as deliverable.
* Milestone definition and attainment criteria that applies to system and support software will be utilized throughout the development.
* DoD approved High Order Programming Languages (HOL) will be used to develop Defense system software.

In addition, the DoD Software Steering Committee established the DoD Defense System Software Management Program whose objective is:

> to devise and carry out a comprehensive and integrated solution to the problem of embedded computer system resource acquisition, management and use. (5)

Specific elements of the objective are (1) to establish engineering discipline and rigor, (2) to promote management visibility, (3) to improve cost and schedule control, (4) to determine the best methods for improving software quality (5).

On 16, 17 June 1976, the Defense Systems Management College

(DSMC) conducted a workshop on the management of software acquisition for embedded computer systems. The workshop focused on management issues encountered at the program manager level. Representatives from all services and OSD were in attendance and their background and current involvements provided user, developer and logistician insights into the discussions and issues. The findings and recommendations of the workshop can be summarized as follows (7)

* Lack of discipline in specifying and validating requirements.

  Require comprehensive analyses of system requirements and defined software performance specification prior to Full Scale Development.

* Delay in the development of support software leads to delay in the operational readiness of embedded computer system.

  Include individuals knowledgeable of software subsystems in higher level budget reviews, specifically in areas involving training requirements, software maintenance, configuration control and system to system integration.

* Increased use of embedded computer systems in defense systems in creating new post deployment software support problem.

  Recognize the sizeable investment needed for the support of defense systems incorporating

embedded computer system, particularly the need for centralized support facilities for systems that are operationally integrated.

* There is a shortage of software engineers in the military and civil service.

  Establish appropriate specialty codes for software engineers and develop paths and incentives to attract and retain software engineers.

* There are many documentation standards for software within DoD.

  The Joint Logistics Commanders' Software Reliability Work Group should undertake the development of a DoD standard for software documentation.

3. Software Development Phases

Computer software development can be grouped into definite sequential phases which occur at least once, but in the usual case will be repeated many times in a system life cycle as major changes in system requirements occur. This dynamic environment seems to be a reality and if software development is going to be successful in terms of cost, schedule and performance, it must be dealt with in the management process of software acquisition. This does not mean that change should be accepted without challenge but rather that flexibil-

ity should be designed into the software acquisition process. We cannot stop the dynamic nature of weapon systems that are built on the leading edge of technology and we cannot deny that one of the underlying attributes of computer software is its flexibility. However, this realization must be made at the outset in order that a responsive software development can be envisioned and implemented.

3.1 Requirements Definition

This phase of software development is probably the number one culprit that adversly affects the cost, schedule and performance of a computer program development. The ability to specify the requirements of a software element in a weapon system has been lacking. It is the opinion of the author that two major factors are responsible for this deficiency. First, the prime attribute of computer software, flexibility, has given cause to such opinions as; let's allocate this function to a computer program and we can define it later. After all, a computer can be programmed to do most anything. That last statement is probably true but you may not be willing or able to pay the price in terms of cost and schedule. The subset of this argument is the approach often taken which is to get the program working and document later. This should be cause for administering a fate worse than death, ie, providing support for somebody else's un-documented computer program.

The second factor which relates to the inability to

accurately specify the requirements of a computer program is the reluctance to utilize the software experts in the definition process. This should be a ground rule for the staffing requirement of a PMO involved in a weapon system making use of computer programs. By software experts, we are talking in terms of both operational and support functions. This would include the facilities, tactical programs, simulation programs, test programs and procedures and the software support activity. Only through direct involvement in the requirements definition phase can these types of experts map realistic approaches to an efficient software development. The software functional requirements should be derived through analysis and tradeoffs of the operational requirements for the system or modification to the system. Requirements should be formulated from the analysis in terms of defining preferred configuration and design approaches as part of the systems engineering process. This software requirements baseline is normally identified and controlled as a Program Performance Specification (PPS) which is submitted for approval at the Preliminary Design Review (PDR).

3.2 Program Design

Following approval of the PPS, the next phase of the software development is the development of a design approach based on mathematical models, functional flow charts and supportive analysis and testing at several levels. In essence,

the program architecture and test requirements should be formulated. Allocation of functions to modules or subprograms and associated interface requirements are established in this phase. Testing methodology should be established that is sufficient to ensure successful attainment of performance at various levels of design completion. In his book "Techniques of Program Structure and Design", Mr. Edward Yourdon effectively relates the interdependance of the various functions of program development. Although primarily addressed to the ADP world, the basic program development considerations discussed in the book apply equally as well to embedded weapon system computer programs. In regards to program design and the effect on subsequent phases of program development,

> ---the most logical way, and in some cases, the only way, to make a program easy to test, easy to maintain, easy to upgrade, and easy for someone else to take over is to keep it simple and straight-forward (15:27)

At this point, decisions must be made concerning the software design approach; ie, top-down, bottom-up, top-down design/ bottom-up implementation, etc. This decision does not appear to be critical in terms of implementation success but is critical in terms of test philosophy. Testing that will occur at both the module level and system level will necessarily depend on the design approach and system requirements. The results of the design phase is normally the Program Design Specification (PDS) which is submitted for review and approval

at the Critical Design Review (CDR).

3.3 Implementation

Following approval of the PDS, the number crunching activity occurs which converts the performance and design requirements of the PPS and the PDS into working code. Many aspects of implementation could and should be discussed in connection with the total software acquisition process. In the interest of scope, only some of the more predominant consideration will be addressed in this paper. The first five words of this paragraph should be held near and dear to the heart. Never start coding until design has been defined and approved. There has been progress made in this respect in recent years but it is still the number one consideration to successful implementation. Never turn a room full of black-art practitioners loose to work their magic until everyone knows what is to be done and what is not to be done. Use higher order languages (HOL) when practical. The reasons for this simply stated are; (1) HOL are easier to implement than assembly languages, (2) the resultant program is simplier to test, (3) the resultant program is easier to change and (4) the transferability of the program is maximized. Often HOL will be less efficient than assembly language in terms of core requirements, but in light of the relatively high cost of generating software as compared to computer hardware, it represents a cost effective tradeoff.

The application of structured programming techniques in recent years has gone a long way in bringing the implementation phase out of the black-art arena. The literature now contains structured programming material and should be given serious attention. Mr. Yourdon devotes an entire chapter to the theory and techniques of structured programming.

> ---the notion of structured programming is a philosophy of writing programs according to a set of rigid rules in order to decrease testing problems, increase productivity, and increase the readability of the resulting program. The primary technique of structured programming is the elimination of the GO-TO statement and its replacement with a number of other well-structured branching and control statements. (15:144)

One must be careful not to let the pendulum swing too far in the opposite direction. There is probably some mid-stream approach between the un-disciplined black-art approach and the disciplined structured programming approach that will win out in the end. Given the choice of the two, the structured programming approach seems to more closely fit the total acquisition process from a technical and management point of view.

During the implementation phase, the Program Description Document (PDD) and Data Base Document (DBD) are generated which provide disclosure of the program design and implementation. In addition, the test procedures and results of the module testing (debugging) should be documented and retained for historical reasons. During this phase, module integration and system integration Test Plans and Procedures (TPP) should

be generated.

3.4 Module Integration

Up to this point testing of the implementation and design has been primarly an individual process involving the person directly responsible for generating the code. The modules will have been debugged to insure proper implementation of the design intent. This often involves the use of static test generators which allows the programmer to ascertain that for a predefined set of data, the module is capable of providing an expected response. In the module integration phase, the emphasis is on combining individual modules into larger and larger program elements. At this point, testing should be accomplished in accordance with an approved TPP and should be conducted by a group or person whose primary function is test. The individual completed program modules would have been placed under configuration control and any changes to the implementation or design of an individual module would be considered in terms of system impact. This is a crucial phase in the program development cycle. Any program errors (bugs) that survive past this point will be more difficult to find and correct in the future. Therefore, we must recognize the necessity of extensively verifying proper program operation before it is made part of an integrated hardware and software environment. Obviously we cannot be completely certain of program operation until it is tested with the total system but all possible risks should

be minimized. Mr. Yourdon makes the observation,

> from a philosophical point of view, you must remember that if you write a program so clever, so intricate, and so complicated that only you can understand it, then the program is worthless; if your program makes use of undocumented, esoteric features of the (computer) hardware, it is worthless; if your program is not commented and documented, it is worthless. All of these practices lead to a worthless program because they make it much more difficult to test. As a result, the program becomes unnecessarily expensive in terms of time and money. (15:8)

3.5 System Integration

In the system integration phase the computer program is tested as an integral part of the total system. This is normally a progressive process involving different levels of functional performance. In a weapon system development this should be a highly structured procedure in which the functional allocations resulting from the system engineering process establish the testing criteria. During this phase the computer program will tend to lose its identity as it becomes an embedded part of the weapon system. This is a natural occurrence and should be recognized as such. Tests should be designed to verify system performance that includes all the functions that have been allocated to computer programs and hardware.

In cases where actual system hardware is not available, simulations are often used to complete the system environment. In addition, systems utilizing computer programs have the inherent capability of extracting critical data parameters that can be used for evaluating system performance. Data

reduction programs are required to convert the extracted data into a meaningful form for analysis. These types of support programs should be managed and developed throughout the acquisition cycle as deliverable items of the system. They will also be required in the software support phase and therefore should be fully documented and certified. Consideration of support software should be made early in the acquisition process, which is often not done due to fiscal and schedule constraints. In analogy to hardware, it would be unthinkable to field a missile launcher without providing the necessary support equipment to maintain it.

3.6 Support

The final phase in the software development cycle is most often termed computer program maintenance. This appears to be a carry through from hardware technology. Unlike hardware, a digital computer program does not break and it does not degrade and if it ever worked it will continue to work for a given set of requirements. For this reason, the final phase would more accurately be described as the support phase. During this period the bugs that survived the module and system integration phases start to come out. Therefore one activity of this phase is the removal of program errors that evaded the earlier testing. Secondly, the probability of requirements remaining constant throughout the life of a weapon system is zero. Therefore the capability to modify

the computer programs to meet the new requirements must be retained. The two types of activities mentioned above are not a complete discription of the support phase but do point out the need for managing and developing support software as deliverable items.

The six phase software development cycle described above is based on the author's experience and study. It is not presented as the one and only way to develop software. It does provide a reasonable approach to the development of embedded computer programs for the weapon system. In general, the first two phases of requirements definition and program design should represent about 40% of the total development effort when the final phase of support is excluded based on its continuing nature. Implementation, which includes the coding and debugging of individual modules, would represent about 20% of the total effort and the module and system integration represents the remaining 40%. In an even more general sense, the software cycle can be correlated to the weapon system acquisition cycle. The requirements definition and program design phases would normally occur in the Conceptual and Validation phases of weapon system acquisition. In most cases however, the Validation phase will demonstrate the feasibility of a system design through actual implementation of intermediate levels of hardware and software configurations. In such cases, some portions of the first five software development phases will occur. This can be considered as a

subset or advanced development of the total system development. During the Full Scale Development phase the implementation, module integration and system integrations activities would occur. This leads to a complete validation of the software that is to be part of the system.

In addition to the software development cycle discussed above, various other approaches can be found in the literature. In Philip Metzger's book,"Managing a Programming Project", a six phase process is described (11)

Definition - a project plan is written and the technical problem is defined; solutions are deferred.

Design - a design document is written which describes an acceptable solution to the problem.

Programming - build and test a program system according to the solution.

System Test - separate group test the program in a "live" environment.

Acceptance - finished program system, including documentation, is demonstrated to the customer, and tested against acceptance criteria mutually agreed to earlier in the development process.

Installation and Operation - programs are introduced and tested on customer's equipment in the ultimate operating environment.

Mr. Metzger provides a thorough development of all aspects of computer programming and software management and is highly recommended for all software managers.

4. Software Costs

The concern that has been expressed in recent years at

all levels of DoD in regards to software development costs have probably been based more on estimate errors than on cost-effectiveness issues. Cost estimating for software development is a subject deserving of an in-depth treatment and will be addressed in this paper in a general manner. For additional information on the subject the reader is referred to Defense Systems Management School study reports prepared by A. W. Andres (2) and R. A. Findley (9).

Cost estimate errors in software development programs appear to be directly related to the management approach. It has only been in recent years that adequate attention has been directed to software acquisition. As a result we have been moving along the learning curve and paying our dues in cost, schedule and performance. Several major factors that have negatively influenced the accuracy of cost estimates are also the targets of the management improvement efforts. Of prime importance is the definition of requirements for computer programs. We must achieve the ability to accurately specify software requirements if we ever hope to accurately assess costs. These requirements must include the tactical or operational functions, the support software functions, documentation and facilities. Application of system engineering concepts and software expertise in the early concept formulation of software requirements is one method of improving this ability. All aspects of software development must be considered when arriving at cost estimates.

Once requirements have been defined, complete and detailed planning must follow which establishes the methodology of implementing a design which will meet the requirements. Proper generation of the Program Performance Specification (PPS), and Program Design Specification (PDS) and Test Plans and Procedures (TPP) will identify the scope of the tasks to be accomplished. From this point further planning should occur in the form of Work Breakdown Structures (WBS), Network Diagrams and Flow Charts which identify the tasks to be accomplished in individual definable units which can be accurately scheduled and priced. In general an experienced programming organization can make accurate estimates on definable segments of code. The real problem lies in arriving at that level of definition and holding it constant.

Two forms of cost estimating will normally be required. In the initial requirements formulation period, historical analogies will serve as the data base for making estimates. These are the estimates that have been most prone to error due to the lack of corporate memory resulting from the short history of weapon system software management. As we improve our management techniques, our cost data base should improve. The management approach must be disciplined to account for all aspects of software development, especially in the support programs and documentation areas. Secondly, the inherent flexibility of software must be prudentially applied. If functions are planned in the earlier periods that are allocated

to software flexibility, the cost estimates are sure to suffer.

The second form of cost estimating which should be applied in development process is based on detailed breakout of software tasks. Once this level of definition is achieved through WBS and the like, accurate estimates can be achieved.

SECTION III

SOFTWARE MANAGEMENT IN A MAJOR NAVY ACQUISITION (AEGIS)

1. Prelude

Up to this point in the report, the intent has been to examine the current trends in computer program management based on an analysis of related publications and the author's experience. In this section we will focus our attention on a real world application of computer program management in a major weapon system acquisition. No attempt will be made to provide an exhaustive analysis of the management approach or the technical design considerations incorporated in the weapon system acquisition. Instead our attention will be directed toward understanding the computer program management concepts that are being employed. The choice of the AEGIS Weapon System for this exercise was based on the currency and complexity of the computer program contribution to the total integrated weapon system.

The information contained in this section has been compiled from the AEGIS Combat System Computer Program Development Plan (1), the Applied Physics Laboratory, John Hopkins University DoD Weapon Systems Software Management Study, (3 & 4) and through interviews conducted with personnel in the AEGIS Program Manager Office (PMS-403) and in the AEGIS Lead Laboratory Office (Naval Surface Weapons Center, Dahlgren

Laboratory).

2. System Description

The AEGIS Weapon System is an integrated shipboard detection, command and weapon control system that is being designed for installation in a wide variety of ship classes. It is a fast-reaction, high performance Weapon System engineered to provide the Fleet with a wide area surface-to-air and surface-to-surface defense through the 1980's and beyond. When used with long range surface-to-surface and extended range surface-to-air missiles the system will provide the Navy with a major offensive surface strike capability.

An AEGIS ship Combat System, of which the AEGIS Weapon System in an integral part, consists of twenty-three separate elements, ten of which require computer programs. The major elements of the AEGIS Weapon System include a multifunction phase-phase array Radar System (AN/SPY-1A), Weapon Control System (WCS Mark 1), Fire Control System (FCS Mark 99), Command and Decision System (C&D Mark 1), Operational Readiness Test System (ORTS Mark 1), Guided Missile Launching System (Mark 26), and the Standard Missile-Medium Range (SM-2/MR). The first five of these elements contain embedded computer programs which are being developed by the AEGIS Weapon System prime contractor with participating sub-contractors. Computer programs for the elements outside of the AEGIS Weapon System

are being developed separately by Navy Acquisition Managers (NAM's). All embedded computer programs will be integrated into the Combat System by the AEGIS Weapon System prime contractor.

The AEGIS system is fully automated and includes the latest Naval Tactical Data System (NTDS) and makes wide use of the standard AN/UYK-7 and AN/UYK-20 digital computers. Modular system design was employed which is adaptable to a variety of ship hulls. System development includes the construction of two Engineering Development Models (EDM's) prior to implementation in the first operational ship. The first EDM has been completed and is currently aboard USS NORTON SOUND. It is a limited performance system intended to provide verification of critical AEGIS capabilities. The second EDM, which represents a proto-type of the operational system, will be installed and tested in a landbased Combat System Engineering Development Site (CSEDS). The purpose is to provide verification of system engineering, interfaces, ship design support, and operational computer programming.

Computational requirements in the AEGIS Weapon System are complex and critical to system performance. The Radar System which conducts area search, automatic target detection and tracking, and provides midcourse guidance communication with the SM2 missile, is controlled by a four-bay, memory-shared suite of AN/UYK-7 computers consisting of 256,000, 32 bit memory locations. An identical AN/UYK-7 computer configuration

is used in the C&D System to provide tactical decision support, multisensor data correlation and management, air intercept control, data link with other units and various display control functions. A third identical AN/UYK-7 computer suite is employed in the WCS for weapon assignment and scheduling. In addition to the above mentioned AN/UYK-7 computers there are also several of the smaller scaled standard AN/UYK-20 computers used throughout the System. Computer programs that are being developed fall into one of three broad categories. Operational Programs consist of the Radar System, WCS, FCS, C&D, ORTS and Training programs. The Executive Program category contain the AEGIS Tactical Executive System (ATES) and the Standard Executive for the AN/UYK-20 (SDEX/20). The third category contain a host of Support Programs such as the standard Compiler Monitor System (CMS-2), AEGIS Development Operating System (ADOS), Interface Simulator System (ISS) and the AEGIS Data Reduction (ADAR) program.

In a system such as AEGIS, the above description is at best the tip of the ice-berg in regards to the considerations that are being addressed by the computer program management function. The view held by the PM office that computer program development will be managed as an integral part of the system as opposed to a separate entity and the following realization found in the AEGIS Combat System Computer Program Development Plan in essence sums the approach being taken;

> It is fully recognized that a development program of this magnitude and complexity cannot be undertaken without assuming some inherent risk. The objective of this plan is to structure a disciplined and orderly program that will reveal problems sufficiently early to minimize long-term risk. Part of this plan includes the Build-a-Little, Test-a-Little, Integrate-a-Little concept to assure an orderly, efficient computer program development and integration process (1:5)

In discussions with the PMO, it was pointed out that specified Reaction Time in the Anti-Aircraft Warfare (AAW) mode is the overwhelming and dominant requirement that drives all parts of the computer program development.

3. System Acquisition History

The Advanced Surface Missile System was born in concept in 1963. Design, development and demonstration of essential elements of the multifunction array radar were accomplished by APL/JHU and provided a firm basis for full Scale Development of the radar. In April of 1968, the Secretary of Defense signed Development Concept Paper 16 which resulted in DSARC I approval for initiation of Contract Definition. In 1969, RCA was awarded a contract for the Engineering Development of the AEGIS Weapon System. DSARC II approval was obtained in June 1974 which supported acquisition of the AEGIS System. As noted in the APL study, early consideration was given to computer program development:

> During the initial engineering phase, program decisions were made to develop a functionally modular computer program and provide a Tactical Executive Program structured for AEGIS. The specifications and planning

reflected strong emphasis on the software acquisition procurements (4:6-2)

4. Computer Program Definition, Design and Implementation

4.1 Definition

Mission requirements contained in the Development Concept Paper (DCP), Tactical Operational Requirements (TOR) and the Top Level Requirements (TLR) are translated into specific performance requirements for the AEGIS System and are documented in the AEGIS System Specification. The first stage of Functional Allocation, which relates to computer program development, is the generation of the prime system elements specifications. Further Functional Allocation occurs as the prime system elements requirements are allocated to the specific equipment and program subsystems constituting each system element. MIL-STD-490 (12), augmented by SECNAVINST 3560.1(14) for the computer program specifications, governs the hierarchy and format of specifications which document the total system design.

A tool developed in the AEGIS Project for the allocation of system-level function to subsystems is the Functional Flow Diagrams and Descriptions (F^2D^2). Documents are produced under the F^2D^2 process at the system level by functional analysts with the support of system designers. In the APL/JHU study the F^2D^2 process was described as follows:

> It translates the requirements of the AEGIS Weapon System into hierarchically ordered functional block diagrams and associated textual descriptions at every level of system operation. Inputs to each block and their sources are identified as are outputs and their destinations. Required functions are indicated and, at lower levels, allocated appropriately to computer programs, equipment items, or operator stations. At every level, traceability is provided to higher and lower levels. At the lower, more detailed levels, direct reference is made to specific paragraphs within specification and/or design documents. This enables a complete system audit by making all functions tracable to approved documentation and all documentation tracable to allocated functions. (4:6-50)

The definition of the computer program requirements begins at the Tier 2 level of the F^2D^2 which defines the individual element requirements and the element-to-element functional interfaces. Inherent to this comprehensive system engineering approach is the early consideration of computer program development as an embedded part of the system development. As functions are allocated to either equipment or computer programs at the system element level, equipment/program tradeoffs are of major concern.

Once the system elements functions have been allocated to a computer program, the definition of the perfromance requirements for the program drive the generation of the Computer Program Performance Specification (PPS) for the element. The PPS is generated by the systems engineering activity with assistance from the programming activity. Further development of the F^2D^2 to the Tier 3 level provides further definition

of the function allocation to a level sufficient for detailed program design. Various tradeoff studies and supporting analysis are conducted at this point to demonstrate the feasibility of achieving the required performance with a given design approach. This material along with the preliminary PPS is presented to the Navy for a Preliminary Design Review (PDR) of the element computer program. Approval of the PPS establishes the allocated baseline for the computer program performance and is the basis for further development of the computer program.

4.2 Design

At this point, program functions are allocated to sublevels in the program. Functional description of the modules, module interfaces, core and timing estimates, program control logic and other considerations of the computer program architecture form the basis for generation of the Computer Program Design Specification (PDS). Following an iterative refinement process, a draft version of the PDS is reviewed at a Preliminary Critical Design Review (PCDR). Completing this milestone in the computer program development process starts the generation of the final PDS and the Build process. This activity is characterized by the generation of the preliminary Program Description Document (PDD) and the Data Base Design (DBD) and the Build In-Progress Review (IPR's). The term

Build refers to a partial or complete computer program that can execute several or all computer program functions. Processing within Builds which are not available is simulated with core and time stubs in order to maintain realism in the operational environment. As Builds are coded and tested, they become part of a more encompassing Build and eventually part of the total program. Following the Build IPR, the program Build is released for code and test. Throughout the definition and design process, interfaces between the computer programs and equipment are control via computer program interface documentation.

Since operational functions and modules do not always relate directly, a top-down design technique has been used that involves the use of System Verification Diagrams (SVD) and threads.

> SVD's illustrate the inputs, conditions, tasks, and outputs constituting an element computer program functional allocation. SVD's may be developed first at the most general level. Subsequent progressive analysis furnish the more detailed (subfunction) levels until each SVD represents a group of related tasks in terms of computer program logic (algorithm, procedures, decisions, etc). Each such task is then represented as a thread, which represents a flow of data through various process steps from stimulus through response. (1:3-9)

A major characteristic of the above design approach is its supportive nature to the verification and validation of the computer program as an integral component of a system element.

In addition it provides a ready means of evaluating the impact of change in system or element requirements. The Critical Design Review (CDR) marks the formal review and approval of the Computer Program Design Specification (PDS) which represents the allocated program design baseline.

4.3 Implementation

Once the Build IPR has been completed the partial subprograms which constitute the Build are released for coding which utilizes the CMS-2 Higher Ordered Language (HOL) and structured programming techniques. Builds have been designed to segment the programs into manageable sized tasks which represent a defined low-level subfunction. As coding and testing progresses, the Builds are linked together with additional subprograms to perform major functions associated with an element. The Build evolution is functionally defined to establish an orderly growth in capability which eventually interfaces with other elements of the system. The build and test process has been an effective approach in identifying and correcting coding errors in the early stages of program development.

5. Computer Program Management

5.1 Organization

The management function of the computer program develop-

ment in the AEGIS System is performed by the Program Manager Office (PMS-403) and the prime contractor (RCA) for the system development as depicited in Figure 1. Support is provided to the PMO by various other Navy and industrial organizations. In addition to the Defense Contract Administration Services (DCAS) Office, the PMO maintains a Technical Representative Office at the prime contractor's facility. Computer system specialists are included in the PMO staff and at the contractor's facility as part of the Technical Representative Office in an effort to focus expert attention on the computer program development throughout the system acquisition process. As noted in PMO interviews, the emphasis is placed on developing computer programs as an integral part of the total system development. In AEGIS this concept is of particular importance in light of the extensive use made of computer programs in meeting the mission requirements.

Within the prime contractor organization there has been established a multidisciplined computer program development team consisting primarily of Navy and contractor management, system engineering, design activities, test and evaluation and subcontractor management. The AEGIS Combat System requirements are established by the Combat System Development Group. Computer program management, which is under control of a centralized office, controls budget, funding, definition, design, coding, test, integration, configuration management, a centralized Program Generation Center (PGC), a Computer

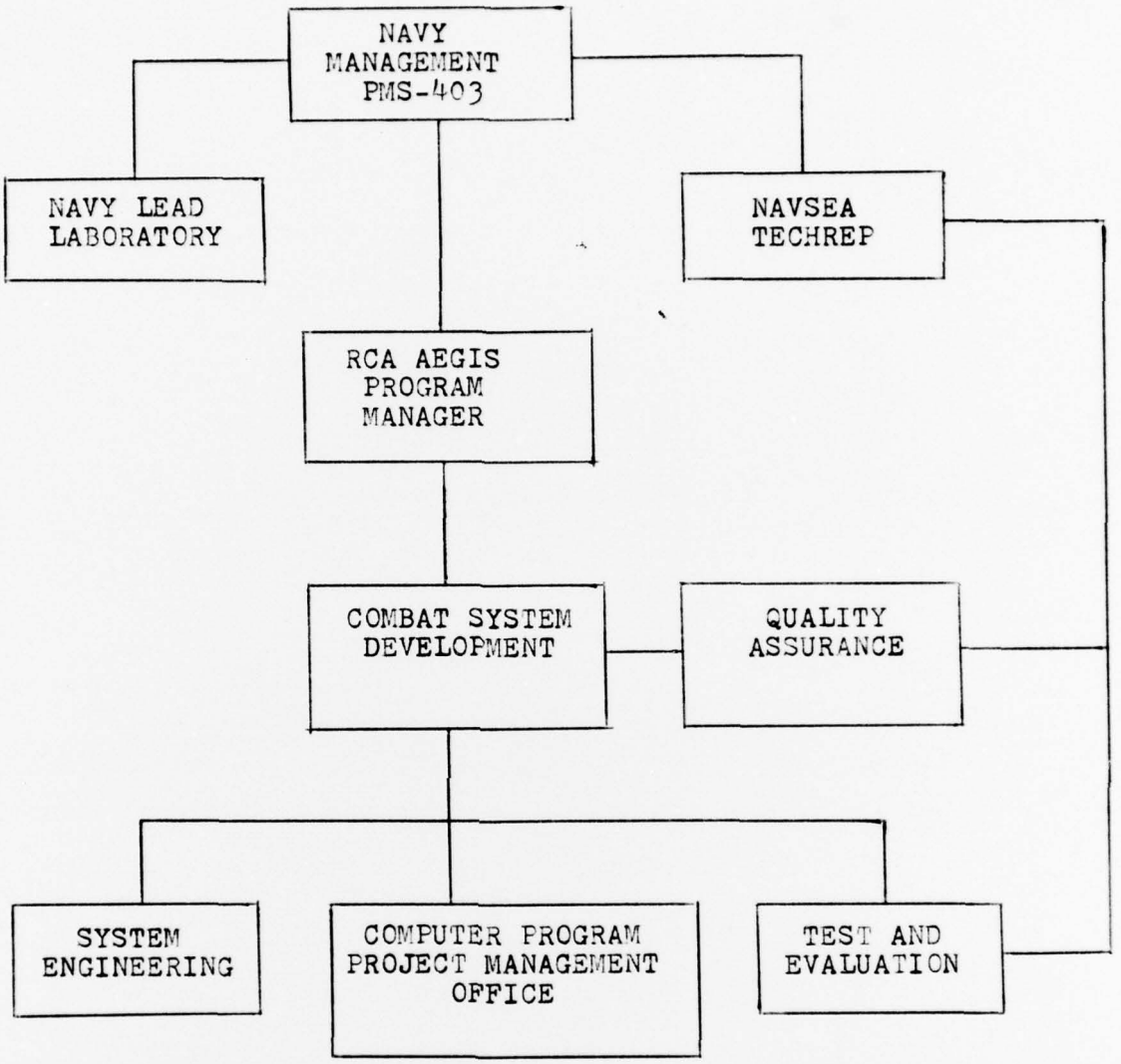


FIGURE 1. AEGIS COMPUTER PROGRAM DEVELOPMENT TEAM ORGANIZATION (1:2-2)

Program Test Site (CPTS) and the Central Computer Program Library (CCPL). Additional comments concerning the PGC, CPTS and CCPL will be made later in this report.

5.2 Design Review Process

As in the management of any activity it is necessary to establish methods and procedures, standardization and feedback systems that are capable of accurately monitoring the status of the development process. In AEGIS a variety of management tools has been employed to provide control of the computer program development process. Emphasis has been placed on the definition of computer programs as deliverable configuration items and the documentation of this definition by a set of uniform specifications. Requirements contained therein form the basis for formal technical reviews and subsequent test planning documentation and testing. AEGIS uses technical reviews, Preliminary Design Reviews (PDR), Critical Design Reviews (CDR) and Configuration Audit Reviews (CAR) for baseline management and contractual milestones.

The PDR is used as a vehicle wherein all participating activities, especially the Navy, review and approve the Computer Program Performance Specifications (PPS) that are written in accordance with SECNAVINST 3560.1 (6). Reviewers include the PMO, users, system engineering activities and design activities. At PDR, the preliminary design concepts

are reviewed to assure that performance requirements can be met within the constraints of the equipment and environment and that the computer program performance matches the performance levied by the item specification for the elements. The output of the PDR process is the final draft of the PPS. It should be noted that the computer program functional allocations are derived from the prime item specification of the element and that the first draft of the PPS is generated by system engineering. During the PDR process, numerous In-Progress Reviews (IPR) are held and post PDR conferences serve to resolve all questions concerning the performance requirements as reflected in the PPS.

CDR serves as the vehicle for review and approval of the Computer Program Design Specification (PDS). The review process is basically the same as for the PDR and results in formal identification and approval of the specific computer programming documentation that will be released for coding and testing. All PDR and CDR actions are reviewed and approved by the Executive Panel. It should be noted that coding and testing is not started prior to achieving the CDR milestone. During the CDR process, work leading to the first Build will start.

5.3 Cost and Schedule Control

In addition to normal contractual cost reporting, provisions have been made in the Computer Program Development

Office to provide detailed control and accounting for costs associated with the computer program development. Work to be performed is broken down by statements of work to the implementing activity and identified into Work Breakdown Structure (WBS) cost levels. For each WBS, expenditures are budgeted, funded and monitored based on scheduled and planned accomplishment. Expense elements include operation and maintenance of the PPG and CPTS as well as contractor furnished computer equipment.

Schedule control is accomplished with the use of computer generated network diagrams at the project, computer program, multi-element program and element program levels. Compatibility of connecting events and critical paths are determined by the network generation program. Changes and updates to any of the networks are operator actions at an interactive terminal. The network program automatically generates alerts if schedule problems are to be encountered as a result of the new information.

5.4 Verification and Validation

As defined by the AEGIS Combat System Computer Program Development Plan,

> Verification and validation of the AEGIS computer programs is systematic evaluation of the specification implementation, and operation of the program throughout the development cycle. (1:2-17)

In essence the verification process determines that require-

ments and specifications of earlier stages in the development process have been translated properly into more detailed requirements and objectives. This process involves an iterative review and revise cycle that translates system performance requirements into the program design requirements. In encompasses the PDR, CDR and IPR processes and all associated documents.

While verification ascertains that the requirements have been properly allocated to the computer programs, validation demonstrates that the program performs as required in its intended operational environment. The validation process is directly related to the Build-A-Little, Test-A-Little, Integrate-A-Little philosophy. As defined earlier, the Build is a logical grouping of functions in computer program modules or computer program elements. As part of the validation process, Builds are tested at increasing levels of program requirements and thereafter become integral parts of higher level Builds. Module or module-part testing is conducted as an Engineering Test & Evaluation (ET&E) using informal programmer generated test data and procedure. This type of testing uses program drivers and the AEGIS Tactical Executive System (ATES) as test tools and are conducted in Computer Program Test Site (CPTS). Results and scenarios of these tests are recorded in the programmer's notebook. Build testing is similiar to module testing but represents a validation

of a more complete program requirement. The Interface Simulator System (ISS) programs which are certified by the developing managers, are utilized in the Build testing. As Build testing and ISS development progresses to the element level, a validation process called Phase II testing measures the element computer program test results against the PPS requirements. Phase II testing takes place in the CPTS using the ISS as the primary test tool and is conducted as defined in the Phase II Test Procedure document which is submitted to the Navy as a Contractual Data Requirements List (CDRL) item. The Test Procedure document defines a pass-fail criteria for each test item of the procedure. As element computer programs and equipment integration progresses, multielement computer program and equipment are integrated at the Combat System Engineering Development (CSED) Site.

5.5 Configuration Management

Configuration management of the element computer programs is maintained throughout the development cycle via approved configuration management plans. The level of control exercised varies from minimal in the initial code and debug period to formal Configuration Control Board (CCB) reviews following the completion of Phase II testing. The PPS and PDS establish the specification baseline for the computer programs against which configuration is managed. Intermediate configurations, that relate to Build plans, are designated as working baselines

which converge to the specification baseline configuration as element computer program delivery is approached. Configuration control of the computer programs addresses control of the changes to the programs and control of the Computer Program Configuration Item (CPCI) specifications in relation to each other and to other system requirements. Two tools for performing this function have been established in the Configuration Control Board (CCB) and the Central Computer Program Library (CCPL). The CCB reviews and approves change packages and ensures consistent and formally approved specifications in accordance with contractual requirements. The CCPL objective is to control the acquisition, maintenance and dissemination of approved program configurations. Program change control and history is maintained through Phase II testing. Configuration control starts when the module has been tested and approved at the design activity level. Thereafter coding changes can only occur through official channels.

> Experience on AEGIS EDM-1 shows the majority of program errors to be design related and not coding related, with the major portion found during program testing. This multiple build approach attempts to find and correct the problems early in program development.(4:6-41)

SECTION IV

CONCLUSIONS AND RECOMMENDATIONS

1. Conclusions

The primary conclusion that one draws from a study in software management is that a vast amount of attention is being directed toward improvement. In general the improvements have the common goal of converting an art form into technology by application of sound planning, implementing and monitoring techniques. There is an overriding objective of seeking out a roadmap to successful software development. One must consider the intangible nature of software to understand the problems that have plagued its management. It is now realized that the powerful black art cannot be practiced in a vacuum when it represents such a large contribution to the system in which it resides.

This study has identified some cornerstones that should be considered in software management.

* Responsibility for computer programs and equipment of a particular system or element should be assigned to a single individual.
* Include systems engineering and software support considerations in the requirements definition phase.
* Assume a realistic attitude toward software flexibility.

* Consider support programs and facilities as essential and necessary components of the software development process.
* Specify all software required for operation and support as contract deliverable items.
* Require complete documentation for all deliverable software.
* Ensure management visibility through the use of a computer Program Development Plan, structured Program Reviews and Milestones. Specify attainment criteria.
* Staff the PMO for performing technical evaluation of the software development.
* Use Top-Down design and modular construction for program development.
* Develop the test plan in parallel with the program design. Develop test procedures in parallel with program implementation.
* Use High Order Language (HOL) and Structured Programming when possible for implementation.
* Perform module integration and system integration within a Test Group separate from the Programming Group.
* Impose configuration control prior to module itegration.

2. Recommendations

In terms of specific recommendations, the cornerstones of software development listed above should be applied to any software acquisition. They, by all means, do not represent an exhaustive treatment of the management issue but do identify the type of considerations that should be made in the development process. The underlying theme of their use is to provide the definition, design, planning, implementation and testing requirements of computer programs that are embedded in a weapon system. Each one of them constitutes an area in which complete studies could and should be made.

Several essential areas of software management have not received detailed consideration in this paper. These would include cost control, configuration management, documentation standards, testing methodology and operational support. It is recommended that future DSMC study projects address these areas in detail.

A general recommendation that should be made regarding software development is akin to flag waving. Keep it simple! Computer programs should be developed in an orderly and logical manner in which each task is expressed in terms of simpler tasks until unique and definable units are established which relate to higher level tasks through functional interface definitions. This establishes the environment in which sound management techniques can be employed. In essence, we should design software to be manageable and then manage the design.

APPENDIX A

INTERVIEW QUESTIONNAIRE

I. Planning

1. Guidance documents?
2. Where in the acquisition cycle were s/w requirements considered? Documented?
3. What office manages the s/w acquisition?
4. WBS level?
5. Cost & core estimates?
6. S/w management visibility? Monitoring?
7. Is one person in charge?
8. Milestones?

II. Implementation

1. When were specifications finalized?
2. When did coding start?
3. How was language decided on?
4. Were structured programming and/or PDL used?
5. What support s/w is required? How will it be documented? Do you have data rights?
6. Is s/w a configuration item?

III. Evaluation

1. What is the s/w test philosophy? Top-down/Bottom-up?
2. What simulations are used? Are they deliverables?
3. What agencies are involved in the s/w testing?
4. Who buys the s/w?

5. Will LBTS be used for s/w testing? How?

6. What s/w Q.A. approach is used?

7. What is the acceptance criteria?

IV. Integration

1. Who has integration responsibility for s/w and hardware?

2. What integration documents exist?

3. Will s/w and hardware integration include operator/user integration?

V. Life Cycle Support

1. When was LCS planned?

2. What was the plan?

3. When did the LCS agent get involved?

4. What considerations were given to maintainability?

5. Will the s/w be Service maintainable?

VI. General

1. What is the configuration control plan?

2. What s/w inputs were included in DSARC/DCP?

3. What is the difference in s/w and hardware management approaches?

4. What is your position on s/w vs hardware flexibility?

5. What percent of s/w development cost went to design, implementation, testing, integration and LCS?

6. What do you see as the major risk areas in s/w development?

BIBLIOGRAPHY

1. AEGIS Combat System Computer Program Development Plan, CDRL Sequence Number 0178AB, May 1976

2. Andres, A. W.; Estimating Computer Software Development Costs; DSMC PMC 75-1, May 1975

3. APL/JHU SR75-3; DoD Weapon System Software Management Study, June 1975

4. APL/JHU SR75-3; DoD Weapon System Software Management Study, Appendix B, Shipborne Systems, June 1975

5. DeRoze, B. C.; Weapon System Software Management, presentation to the Software Task Group of the Defense Science Board, 25 July 1975

6. Defense Management Journal; Volume 11, Number 4, October 1975

7. Defense System Management College workshop report; Management of Software Acquisition for Embedded Computer Systems, 16-17 June 1976

8. DODD 5000.29; Management of Computer Resources in Major Defense Systems, 26 April 1976

9. Findley, R. A.; Computer Software Development Costs, Predictable or Not?, DSMC PMC 74-1, May 1974

10. Management of Weapon System Software; Memorandum from the Office of the Secretary of Defense, 3 December 1974

11. Metzger, P. W.; Managing a Programming Project, Prentice-Hall Inc., 1973

12. MIL-STD-490; Specification Practices; Change 2, 18 May 1972

13. Mitre Corporation; DoD Weapon Systems Software Acquisition and Management Study, June 1975

14. SECNAVINST 3560.1; Department of the Navy Tactical Digital Systems Documentation Standards, 8 August 1974

15. Yourdon, E.; Techniques of Program Structure and Design; Prentice-Hall, Inc., 1975