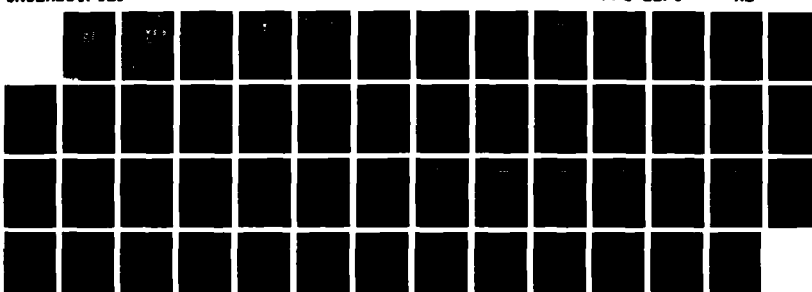
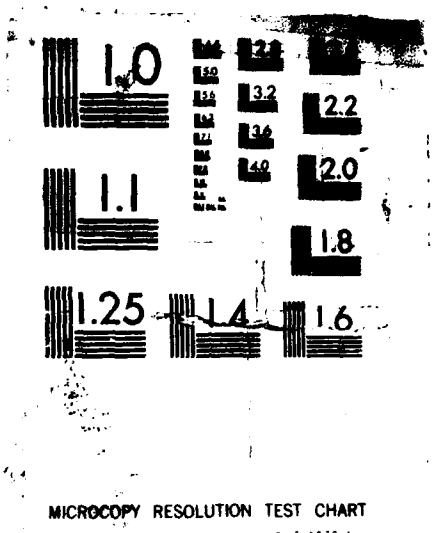
AD-A182 309 THE SOFTWARE CRISIS AND A SENIOR LEADERS AWARENESS COURSE(U) AIR COMMAND AND STAFF COLL MAXWELL AFB AL D A TAYLOR APR 87 ACSC-87-2445 1/1

UNCLASSIFIED F/G 12/5 NL





MICROCOPY RESOLUTION TEST CHART

AIR COMMAND AND STAFF COLLEGE

STUDENT REPORT

THE SOFTWARE CRISIS AND
A SENIOR LEADERS AWARENESS COURSE

Major David A. Taylor 87-2445

"insights into tomorrow"

DISCLAIMER

The views and conclusions expressed in this document are those of the author. They are not intended and should not be thought to represent official ideas, attitudes, or policies of any agency of the United States Government. The author has not had special access to official information or ideas and has employed only open-source material available to any writer on this subject.

This document is the property of the United States Government. It is available for distribution to the general public. A loan copy of the document may be obtained from the Air University Interlibrary Loan Service (AUL/LDEX, Maxwell AFB, Alabama, 36112) or the Defense Technical Information Center. Request must include the author's name and complete title of the study.

This document may be reproduced for use in other research reports or educational pursuits contingent upon the following stipulations:

-- Reproduction rights do not extend to any copyrighted material that may be contained in the research report.

-- All reproduced copies must contain the following credit line: "Reprinted by permission of the Air Command and Staff College."

-- All reproduced copies must contain the name(s) of the report's author(s).

-- If format modification is necessary to better serve the user's needs, adjustments may be made to this report--this authorization does not extend to copyrighted information or material. The following statement must accompany the modified document: "Adapted from Air Command and Staff Research Report (number) entitled (title) by (author) ."

-- This notice must be included with any reproduced or adapted portions of this document.

REPORT NUMBER 87-2445

TITLE THE SOFTWARE CRISIS AND A SENIOR LEADERS AWARENESS COURSE

AUTHOR(S) MAJOR DAVID A. TAYLOR, USAF

FACULTY ADVISOR MAJOR CHARLES E. ZIMMER, JR., ASC/YF

SPONSOR MAJOR DEL TACKETT, AFSC/ESC

Submitted to the faculty in partial fulfillment of requirements for graduation.

AIR COMMAND AND STAFF COLLEGE
AIR UNIVERSITY
MAXWELL AFB, AL 36112

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

A182309

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION: UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT: STATEMENT "A" Approved for public release; Distribution is unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S): 87-2445	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION: ACSC/EDCC; 6b. OFFICE SYMBOL (If applicable)	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State and ZIP Code): Maxwell AFB, AL 36112-5542	7b. ADDRESS (City, State and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION; 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State and ZIP Code)	10. SOURCE OF FUNDING NOS. PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT NO
11. TITLE (Include Security Classification): THE SOFTWARE CRISIS AND A SENIOR	
12. PERSONAL AUTHOR(S): Taylor, David A., Major, USAF	
13a. TYPE OF REPORT; 13b. TIME COVERED FROM _____ TO _____	14. DATE OF REPORT (Yr., Mo., Day): 1987 April; 15. PAGE COUNT: 50
16. SUPPLEMENTARY NOTATION: ITEM 11: LEADERS AWARENESS COURSE	
17. COSATI CODES: FIELD / GROUP / SUB GR	18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)

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

Project Bold Stroke is a HQ USAF program to combat problems in Air Force software management. A specific Bold Stroke requirement given to Air University is to develop a short course for general officers and senior ranking civilians. This paper attempts to address that requirement by reviewing the documented problems in software development and conclude with a list of recommended course topics and readings. A by-product of the research is a survey of the software crisis facing the DOD.

Keywords: theses

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT: UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT XXX DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION: UNCLASSIFIED
22a. NAME OF RESPONSIBLE INDIVIDUAL: ACSC/EDCC Maxwell AFB AL 36112-5542	22b. TELEPHONE NUMBER (Include Area Code): (205) 293-2483; 22c. OFFICE SYMBOL

DD FORM 1473, 83 APR — EDITION OF 1 JAN 73 IS OBSOLETE.

UNCLASSIFIED
SECURITY CLASSIFICATION OF THIS PAGE

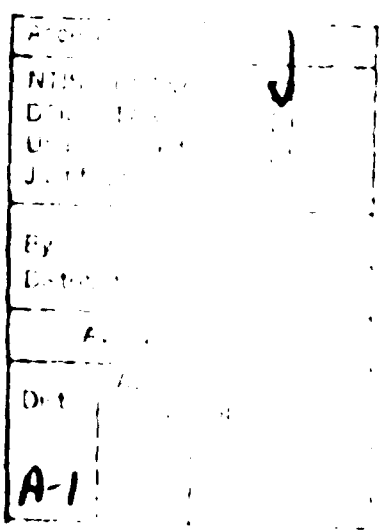
PREFACE

Project Bold Stroke is a HQ USAF program to combat problems in Air Force software management. A specific Bold Stroke requirement given to Air University is to develop a short course for senior officers and civilians. This paper attempts to address that requirement by reviewing the documented problems in the development arena and concludes with a list of recommended course topics..

This material is being submitted to the faculty of Troy State University in partial fulfillment of the requirements for the Master's of Science degree.

The author wishes to thank Major Chip Zimmer and Mr. Dennis Gibson for their enthusiasm in assisting and proofreading this project. A special thanks also to Major Del Tackett for the idea of this topic and his countless hours of guidance during the research. It is gratifying to work with such a professional.

Most of all, my deepest appreciation for my family: to my wife, Sandy, for giving me the two greatest joys of my life; and, to my parents, for having made all things possible.



ABOUT THE AUTHOR

The author, a native of New Albany, Indiana, was awarded a Bachelor of Science degree in Computer Science and a Bachelor of Science degree in Psychology from Purdue University, West Lafayette, Indiana, in 1974. He was a distinguished graduate of ROTC and commissioned that same year. His initial tour of duty was at HQ Military Airlift Command. There, he was a computer programmer/analyst before selection as an Education With Industry student at Draper Labs, Cambridge, Massachusetts. At the HQ European Defense Analysis Center he supported the DOD Intelligence Information System as a computer systems analyst. He also gained experience in contracting and fiscal management as Contracting Officer's Technical Representative and Chief of Data Services Center's Resource Management Office, HQ European Command. At HQ Pacific Air Force his ADP software programming and maintenance branches supported tactical intelligence missions. He is a top third graduate of the Squadron Officers School resident program, an honor graduate of the Computer Systems Staff Officer Course, and a graduate of the Air Command and Staff College correspondence program. He is currently assigned to Air Command and Staff College, Maxwell AFB, Alabama and attends the graduate program at Troy State University, Montgomery. Major Taylor is married to the former Sandra L. Redlingshafer of Eugene, Oregon. They have one daughter, Angela.

TABLE OF CONTENTS

CONTINUED

Chapter One

INTRODUCTION

PROJECT BOLD STROKE

In July 1985, the Assistant Secretary of the Air Force for Research, Development and Logistics wrote to the Air Force Vice Chief of Staff expressing his concern about software problems adversely affecting the Air Force. The Assistant Secretary concluded with a remark the Air Force must make a "bold stroke" towards a remedy. In response, the Vice Chief commissioned a study team to survey the problem and report its' findings. Hence, the Air Staff Software Management Program began, commonly called Project Bold Stroke (12:100). The study team concluded its' findings with four objectives and accompanying action items issued as an action plan to Air Force major commands and separate operating agencies. One of the objectives, "Training and Education," tasked Air University to develop a "short (3-5 day) software literacy course" for "all new brigadier general selectees" and "generals/senior civilians with responsibility for software" (26:3).

PURPOSE OF THIS STUDY

Given the above tasking, this paper reviews the literature and highlights the problems in software development. It posits a list of topics and recommended readings for the course. A by-product of the paper is a summary of recent thinking on the severe technical problems facing the DOD and the private sector in the software field.

While much of the data available to the Bold Stroke team is no doubt replicated here, the objective of this paper is different. By way of analogy: assume Smith's auto headlights do not work and Jones wants to borrow Smith's car. Smith can either take the car to a mechanic or advise Jones of the problem. The mechanic will look for a solution to the problem. Advising Jones of the faulty headlights does not solve the problem but alerts Jones to the hazards of driving at night. As applied to this paper, the Bold Stroke Team was looking for solutions to the software crisis whereas this paper's objective is to help the students avoid common pitfalls while they oversee the software development process. Solutions would have to come from a lengthy

and intensive course, beyond the scope of the Bold Stroke software literacy course.

ASSUMPTIONS AND LIMITATIONS

The reader must be familiar with software concepts and terminology since brevity precludes a tutorial approach. Even for the knowledgeable reader, length constrains the amount of detail that can be presented on the problems in software development. Only the highlights appear. An in-depth treatment is available through the citations and bibliography.

Although there are several ways to classify software, e.g., by host hardware or by function, this paper assumes only two categories of software with only one significant difference. The scientific, business, administrative, or command and control software, running on commercially available hardware, will be called "ADP systems." "Embedded software" is the label given to software operating as an integral part of a system, e.g., an avionics package. The only significant difference between the two is in the procurement process because the principles involved in the development, maintenance and management are similar (38:11).

This study will not argue the findings, objectives and taskings of the Project Bold Stroke Team. Though some may take issue with the team's conclusions (32:--), it is not the purpose of this paper. Today, the course is a "go" and the need exists for planning the instruction.

It would seem to be a safe assumption that general officers, as students attending this course, will have a wide breadth of experience. It is also assumed most will lack experience in software development. This last assumption follows from the Project Bold Stroke Team's findings. They concluded senior leaders needed an increased awareness of the software development problems.

OVERVIEW

Chapter Two introduces the idea of a software crisis by showing the immensity of the problem within the DOD. It summarizes the Project Bold Stroke initiatives to counter the problem, and presents the specific objective and tasking for the senior management software course.

Chapter Three highlights the problems encountered during the software system life cycle as documented in the literature. The issues fall into three categories: concept development, system development, and system deployment.

The literature survey shows other software development problems do not fit into the above three categories. One commonality among these items is management is ultimately responsible for them. Chapter Four, Management Issues, describes these items.

With the nature of the software crisis introduced and the life cycle and management issues explored, Chapter Five assesses the topics for the course. It begins with a few preface remarks and ends with suggested course offerings.

Lastly, Chapter Six concludes with a summary and a series of recommendations. The recommendations include the course topics (also Appendix A), course readings (Appendix B) and items for further research.

Chapter Two

THE SOFTWARE CRISIS

THE PROBLEMS

> In any attempt to define the problems facing the Air Force in the computer area, the pervasiveness of the technology becomes immediately and readily apparent. Practically every area of Air Force operation today is involved with some facet of computer technology. The explosive growth in the use of the technology over the past twenty years has been segmented and largely unplanned in an overall sense (30:9).

This quote is from a 1970 report on the Air Force's ability to exploit computer technology. The point to underscore is how long problems have been studied and documented. While organizational, hardware and software problems continue to impact the Air Force today, the emphasis has shifted.

There have been many organizational changes since that report. For example, the original ADP'ers were assigned to staff elements and each staff element purchased its' own hardware. Later, the Air Force created the "single manager" concept in which ADP personnel and hardware belonged to one centrally controlled organization. This organization became a service facility for the staff elements. The most recent organizational change has been the merger of the ADP and communications fields under the control of HQ Air Force Communications Command.

On the topic of hardware, there has been a dramatic change. Once, the hardware to software cost ratio had been 80 to 20. This was the era of transistors and before the advent of high scaled integrated circuitry. Now, hardware is very inexpensive compared to software with a cost ratio of 20 to 80.

The lessening costs of hardware is only one side of the cost equation. The expense and proliferation of software has tremendously influenced the cost ratio. For example:

1. Software costs the DOD $4 - 8 billion per year according to a 1985 report (37:2). Some predictions say embedded software alone, will hit $32 billion by 1990 (39:9; 6:2541).

2. The "Report of the SECDEF to the Congress on the FY85 Budget" listed 160 key programs. Of these, 75 percent had a significant software component (24:2).

3. F-16As, in 1981, "had seven computer systems with fifty digital processors and 135,000 lines of code." 1986 F-16Ds have "fifteen computer systems with 300 digital processors and 236,000 lines of code" (5:49).

4. Weapon systems software expenditures account for 5 percent of the total Air Force budget. It will increase to 10 percent by 1990 (5:46).

5. Software is the limiting aspect in both mission critical and support systems applications. The demand for software far exceeds the supply. With current technology the supply to demand ratio will be 1 to 4 by 1990 (28:17).

Clearly, software is a growing portion of systems and budgets. Furthermore, DOD faces severe problems with developing, maintaining, and managing software:

1. The DOD wastes $800 million per year because of poor software management (12:100).

2. Of 10 systems in trouble today, 7 were in trouble because of software (29:2-2).

3. The Pentagon's IG found 75 percent of the software contracts reviewed had management problems (15:2).

4. "It has often taken the military services as long as nineteen years to get software into systems from the time of its conception" (5:46).

5. The Defense Systems Management College ranks control of software cost as one of the Armed Forces largest management problems (42:2).

One can easily discern the impact. If a large portion of the budget and a deliverable system is in trouble, operational readiness can suffer. If the USAF in FY86 could have saved one-tenth of what it expected to spend on software, it could have bought 26 more F-16Ds (5:51).

Interestingly, the DOD is not the only one suffering the software crisis. In many corporations, there is a two- to four-year backlog of programming requests. One bank reported a seven-year wait. When the waiting time is this long, users do

not submit requests for programming support. Researchers call this the "invisible backlog." The Sloan School of Business attempted to measure this "invisible backlog" and estimated it to be 168 percent of the formal backlog (4:5-6). Though the DOD and the private sector have many of the same problems there are unique aspects to DOD's software computing.

Complexity, size, specialized application, and managerial restraints are some of the aspects that distinguish DOD computing from private industry. Weather forecasting and ballistic trajectory calculations are examples of complex computing. The size of these software programs can be millions of instructions. Size can also refer to the costs as evidenced by the dollar figures cited above. Command and control, avionics fire control and inertial navigational guidance systems are examples of specialized applications. Managerial restraints can be procurement regulations, mandated personnel end-strengths, and budget cuts. Adding to this uniqueness is that the DOD leadership can change abruptly with each new administration (2:16; 31:12-13). Perhaps it is these unique aspects and the difficulty in achieving progress that has influenced the DOD to do several studies on the software crisis (28:--; 23:--; 24:--; 27:--; 33:--; 29:--; 30:--).

In 1982, the USAF Scientific Advisory Board was studying the application of advanced electronics. In many instances, software was a notable component of those systems. They said weapon system cost, availability, lead time, utility and survivability would increasingly become a function of the software. "The software impact on schedule, cost and performance may become the dominant factor in fielding advanced technology systems" (29:2-1). Their findings led to a 1983 follow-up devoted specifically to software in weapon systems. By 1985, USAF top management became involved and Project Bold Stroke was born.

PROJECT BOLD STROKE

Representatives from each of the functional areas on the Air Staff and from several major air commands made up the survey team. The Air Staff's Chief of Education, Directorate of Personnel Programs, chaired the team. The direction given to the team was to investigate problems of the "availability of expertise, sufficiency of software orientation and reliance on the enlisted force" (26:15). They surveyed the literature and conducted interviews. Once word spread of their investigation, they received many unsolicited opinions from the field.

In August 1985, the team briefed its' findings. Primarily, they found a need for greater awareness and in-depth expertise (26:16; 5:47). Having determined a scope to the problem, they were tasked to come up with a "bold stroke" towards a solution. They made their recommendations in the form of an action plan composed of four objectives:

1. Awareness - to create an overall Air Force awareness of the criticality of software and computer-based technology to information and mission critical systems.

2. Training and education - to provide the training and education necessary to ensure the Air Force fully uses the potential available through software and computer-based technology.

3. Personnel - to survey Air Force software personnel requirements and develop more comprehensive approach to recruiting and managing military and civilians in this career area.

4. Future planning - to project future software personnel requirements with emphasis on the appropriate mix of officers, civilians, NCOs, and contractor support.

A purpose statement and a series of required actions or tasks supports each objective in the action plan. The basis for this paper is objective two, paragraph "A", hereafter called Task II-A. Given the second objective above and the "need to ensure that personnel working directly with software development, employment, and follow-on support are properly trained/educated in their responsibilities" (26:3), Task II-A required Air University to develop a software literacy course as quoted in the problem statement, Chapter One. Since the intent links "literacy" with those having responsibilities for software development, it becomes important for the curriculum planner to determine what is meant by "literacy" and what problems and issues are causing this software crisis. Armed with this information the planner can develop a course for senior leaders who in turn will be able to avoid some of the documented pitfalls. The next two chapters then, highlight common problems and issues of the software crisis.

Chapter Three

THE LIFE CYCLE ISSUES

FRAMEWORK FOR PRESENTATION

Most of the reports surveyed for this paper presented different problem areas in software development. There does not appear to be one systemic problem (unless one concludes industry suffers from a lack of a "true" software engineering discipline). The only common theme among authors is there are several contributing parts to this software crisis. In order to show them, this chapter uses a life cycle model to provide a logical framework. There are many descriptions of each step of the life cycle. One model labels the steps: requirements, specifications, design, programming, testing, integration testing, deployment, and maintenance. The DOD's model labels the phases as: concept, feasibility study, requirements definition, design, coding and checkout, testing, integration, operational test and evaluation, and deployment maintenance (4:178-179). Newer models have appeared to accompany the methods of structured analysis, design and programming. Since no standard set of labels nor model exists, it is necessary to discuss problems about the software life cycle in terms of a higher level of abstraction. Using an adaptation of another model, it should be safe to classify all steps of the life cycle in terms of three categories: concept development, system development, and finally, system deployment (34:3).

CONCEPT DEVELOPMENT

This phase of the software life cycle begins with the user's conceptualization of an automated need, be it an accounting and finance system or a fire control processor on an advanced fighter. The user, often with the help of a computer systems analyst, states the requirement and writes the specifications. The "specs" later become the criteria for judging an acceptable product. Finally, the phase ends with some type of broad or general design.

A common problem in this phase is incomplete requirements definitions (1:385). Several reasons are: the user does not know what is desired; the user changes the requirements; the user does not know how to express the requirement; the analyst incorrectly

interprets the requirement; and, all parties are under pressure to proceed into the system development phase (31:10). The latter is a common problem even though many stress the importance of having adequate time to define and design. The pressure starts because the concept development phase shows the least visible measure of personnel productivity. How can one measure thinking as opposed to lines-of-code or programs constructed? Hence, there is strong pressure to move on to an area that begins to show visible results rather than devoting the needed time to requirement definition. For example, in a review of 84 contracts, 23 percent had unclear statements of work or incomplete definitions of requirements. Of the products delivered, the majority were unsatisfactory, delayed, incurred cost overruns or had a combination of the three (28:7-3).

Even with requirements defined completely, there often is another problem: errors in the requirements documentation and design step (1:241). In fact, the larger the product, the more often the number of design errors exceeds the number of coding errors (1:242). One estimate was 83 percent of the bugs were in the requirements documentation and design steps. These bugs are expensive to correct (4:57-58).

Since requirements are often incomplete or contain errors, it is no wonder it is nearly impossible to predict the complexity, cost and schedule of any project. A method to reduce problems in this phase is to conduct an open review of a preliminary design. From the review, designers obtain agreement this rough sketch approaches the stated requirement. Having an approved design permits the developers to better estimate project complexity, length, and cost. Hence, until preliminary design, there is no "manageable level of software predictability and control" (29:3-10).

One study reported most Air Force software acquisitions are fixed before this design review. This means the statement of work, used in the acquisition, may not match the desired requirements. If they do not, then the budget process will use erroneous cost estimates based on invalid requirements and falsely limit the fixed price contracts. The ripple effect ends with "best and final offers," based on improper requirements and poor cost estimates, guaranteeing built-in cost and schedule overruns (29:3-11).

Having a design and conducting a review reduces the risk but do not eliminate all problems with the requirements and specifications steps. Embedded software specifications use MIL-STD 483/490 the same as hardware procurements. The difference is "form, fit, function, materials, components and previous designs may be described in considerable detail" for the

hardware, whereas the software specifications "concentrate almost exclusively on describing performance and function. . . . The control that an Air Force procurring or developing agency has over cost, schedule, and maintainability is ultimately limited because of such a general specification" (42:14). Furthermore, given that the design, by definition, is preliminary, or at the least conceptual in nature, then the development will be a function of the designer's "logic and ideas". This places the "cost, maintainability, and schedule at the mercy of the experience, efficiency, and effectiveness of the developer's design team" (42:14).

SYSTEM DEVELOPMENT

This life cycle phase includes the detailed design, programming, integration, testing (component testing, integrated systems testing and operational testing), and evaluation. From the viewpoint of the system developer, in-house or contract, the software development issues are the same. From the Air Force perspective, there are additional risks in software development contracting.

Development Issues In General

Robertson (40:74-94) categorized the issues affecting software development into four categories. They are: physical environment, structural environment (staffing, awards and incentives, suggestion programs, flextime), development and maintenance tools (hardware considerations, software tools), and intellectual skills. These four categories can positively or negatively affect productivity. A discussion of the first three follows while intellectual skills will be deferred until the next chapter's discussion on training.

The physical environment describes the building, office setting, and the office equipment. Terminals designed to reduce glare and eyestrain, chairs designed to reduce backache, desks equipped to handle several unfolded computer printouts and bookcases to store documentation are examples of office equipment necessary for motivating workers and increasing productivity. The numbers, types and locations of the terminals can be a consideration. Areas to help exchange ideas, like conference rooms, and quiet office space to further concentration are important too. While there are few documented studies to confirm these ideas, IBM estimated their well-designed facility at Santa Teresa generated an 11 percent improvement in programmer productivity (40:76). The Bold Stroke Briefing Team claims if the environment is poor it could decrease productivity 20 percent; if good, increase it 20 percent (33:Tab 3). The

physical environment is important because programming is such a mentally intensive activity.

Staffing, one of the structural environment aspects, equates to management learning how to motivate programmers, especially if management wants an increase in productivity. Some programmers are ten times more productive than others (40:77). With an increasing backlog of programming requests and increasing shortages of computer personnel, there is a real need for increased productivity. The USAF Scientific Advisory Board estimates the annual increase in staffing and productivity will be 4 percent each but the demand will still outpace the growth by 12 percent. This means an ever widening gap between supply and demand (29:3-27; 24:136-138). One opinion is productivity must increase by a factor of 100 in the next 10 years if development is to match need (4:21).

Development and maintenance tools refer to the computer hardware and software support in doing development. Productivity suffers from poor hardware tools when terminal response times are long, program execution speeds are slow, main storage is small or constraining, and job turnaround time is long (40:83-85). If the organization lacks software tools to do prototyping, project managing, budgeting, auto documenting, interactive debugging, auto coding or database maintaining, then productivity suffers again.

Contracting Issues

"In a recent software contract the Air Force estimated that the cost would be $1.5 million. The contractor's low bid was $400 thousand. But the final cost. . . turned out to be $3.7 million" (15:2). While this gets attention, it is only symptomatic. The General Accounting Office surveyed 163 software firms and 113 Federal project officers experienced in software development contracts (21:--). They concluded their report with five important causes to the contract problems.

The first was a lack of central guidance on contracting. This can lead to using inappropriate contracts, e.g., fixed price and unspecified deliverables. A second cause is top management not committing enough qualified people to the contract. Often, the government's and the contractor's staffing remains below sufficient levels for the entire life cycle (1:305). Spending limitations can cause staffing problems too. The author recalls one project staffed incrementally because of fiscal constraints. A third cause, cited before, is committing to a project before specifications and requirements are completed. A fourth cause is poor contract management. Lastly, because of incomplete, inadequate requirements and specifications, the government may

not adequately test the final product (21:27-28).

Another problem occurs with embedded systems when the size of the development is large. It is not atypical for a development to span three years. In this instance, the contract is often let before the hardware design occurs. Hence, the software must be developed without specifying the interface requirements (31:18) or the hardware delivered does not do what the software designers expected. In either case integration is difficult and engineering change proposals inevitable (1:305). Another likely possibility is the user's needs will change during a long span time, again, making changes necessary.

Lastly, though attributed to problems of management, discussed in the next chapter, the following quote appears here to complete this section on contract issues:

> Seventy-five percent of the 84 software contracts examined had at least one type of management problem. Software contracts were written that did not contain clear and complete statements of work; rights of technical data clauses; adequate documentation and software standards; and comprehensive test and acceptance criteria. Guidance was not available to clarify and provide an overview of existing DOD and non-DOD publications on software quality assurance programs and standards. There were no reviews of purchase requests for software contracts by personnel familiar with data processing, procurement and quality assurance. Contracting officer technical representatives (COTRs) for software contracted did not have adequate training on how and when to recommend the use of contract clauses to protect the interests of the DOD. As a result of these problems, some software contracts cost more than planned, were not completed on time and/or were not satisfactory when delivered or were never delivered (28:7-2).

SYSTEM DEPLOYMENT

This phase assumes all testing is complete and the system becomes fully operational. Maintenance is the key function in this phase. However, software "maintenance" may be a misnomer, i.e., there are no parts to wear out or to replace. True, some logic errors exist, but these account for only 18 percent of the total maintenance effort (37:3). The bulk of maintenance is correcting design flukes and responding to changes in requirements. According to the Scientific Advisory Board Study, maintenance is fixing the immediate errors and achieving full

capability because of inadequate design or changes to the hardware. These efforts are really "mini-developments" rather than software "maintenance" (29:3-50 - 3-51).

Besides inadequate design driving maintenance activities several other issues are common. Long development times can lead to new systems fielded on obsolete hardware (40:52). Missions change. For example, in the Falkland hostility, navigational systems had to be changed to fly in the southern hemisphere (24:6). Lastly, problems arise when the software development organization completes its' task and transfers the product to the support organization. Often the support organization has had no prior involvement with the system.

Previous paragraphs have cited development costs and schedule overruns. It is interesting to note the significance of the maintenance issue. One estimate is development costs $30 per line of code whereas maintenance costs $4000 per line of code. In 1990, maintenance will be 70 to 80 percent of the software budget (38:52-53). It is also 40 to 95 percent of the personnel effort (37:3). Supporting this last statistic, GAO noted in a review of 15 federal computer sites, software maintenance accounted for an annual expenditure of $38 million. Of this amount 71 percent was for salaries. They concluded the government spends $1.3 billion annually (22:i). Martin provides a more liberal estimate: the DOD spends $2 billion per year and expects to pay $16 billion by the end of the 1980s (4:7).

In short, the inherent flexibility of software to meet new mission needs dictates maintenance. However, maintenance is very costly, far exceeding the cost of development. To reduce costs one must: design for maintenance while in the concept development phase; and, employ automated maintenance tools wherever possible. Unfortunately, neither is simple as discussed in the previous two phases.

LIFE CYCLE SUMMARY

This chapter has highlighted many of the problems occurring in each phase of the life cycle. However, the two most important points are: the critical phase is concept development because incomplete and erroneous requirements are the biggest causes of later problems; and, the deployment phase is the costliest.

Chapter Four

THE MANAGEMENT ISSUES

The topics in this chapter are distinct and do not lend themselves to a logical framework such as in the previous chapter. Their only common link is they relate to management's ability to influence their impact. The topics are: management's technical competency, manpower shortages, personnel management, productivity, training, and project management oversight.

Management's Technical Competency

The USAF Scientific Advisory Board's recommendations, were in large part, the impetus for Project Bold Stroke, concluded with the basic "belief that the true essence of the solution is effective, enlightened management" (29:4-9). Several reports echo this sentiment:

> Being knowledgeable about technology is essential if management is to avoid giving up control of major areas of their job. . . . Having management that is not technically oriented can be a critical factor in the failure of any technological system. . . . They do not have to be experts, but they need to understand it because they must be the ones to ask the penetrating 'what if' questions. . . . In fact, before making any technical decision, the first step should be training top management to understand it (8:103).
>
> . . . new software technologies create some serious challenges. Air Force managers must be able to evaluate new systems and insure requirements are satisfied (41:10).
>
> Computer technology is complex and changing so rapidly that skilled technicians at senior levels (GS-13 and above) are not a luxury but a necessity (40:14).
>
> A major campaign should be conducted to improve the computer literacy of Air Force personnel (28:21).
>
> Technological and scientific advances do not require all Air Force officers to be engineers or scientists, but all officers will need to be technically literate (36:25).

Manpower Shortages

From 1972 to 1984, Air Force end-strength fell 20 percent. An increasing use of automation helped to offset this manpower decline. Yet, ADP spaces increased only 12 percent in the same period (28:5). This meant an increasing demand for a slowly increasing pool of resources. More important is the concern for competition between the DOD and industry for this resource. "The national shortfall of some 80,000 civilian and military software professionals is expected to swell to 1,000,000 by 1990" (5:46). Also, there is evidence of a growing attrition rate among lower ranking enlisted within the ADP field (29:3-26). "Good military programmers are lured away in droves by industry and mesmerizing salaries" (13:12). The implications are clear: the military and industry demand for ADP professionals will increase; competition will be strong; and, the DOD will have an added constraint of facing stagnant or decreasing manpower authorizations in the face of austere budgets (41:6; 5:48; 29:3-53). If the competition is keen and resources are dwindling, effective, efficient management of personnel is obligatory.

Personnel Management

"Education, training, assignment, progression, and retention of both military and civilian personnel in the computer area are recognized as significant problems at all levels in the Air Force" (30:16). It is interesting to note this quotation is from a report published in 1970. Ten years later, a student officer published a review of career irritants adversely affecting science and engineering officers (including the ADP field). He raised several issues. Among them:

1. The utility of training is particularly important in an officer's career. After the sixth year, officers do not feel the Air Force is providing this utilization (35:10).

2. Most feel rated as technicians by their supervisors but must compete in the "whole-person concept" for promotion boards. A civilian scientist or engineer is always evaluated within the organization based on job performance (35:22).

3. The 1974 Air Force Career Motivation Conference recommended a dual track career ladder for science and engineering officers. The idea was to let the officer remain a technician until retirement. The idea was rejected for two reasons: civilians could be used and there were not enough field grade slots to permit the concept (35:25).

4. The science and engineering officer "at the five year point in his career must carefully weigh the prudence of investing six more years of his life for a 50 percent chance of making O-4 and allowing him to continue a full career in the Air Force" (35:28).

Though striking the institutional notion of generalist versus specialist, the last two points surface again in the USAF Scientific Advisory Board's report:

> The Air Force must provide adequate career paths for both military and civilian technical staff personnel to ensure that the most highly qualified have the headroom necessary to advance in the career field. Currently, qualified engineers and computer science personnel must either leave the service or move laterally into management career fields in order to advance (29:3-54).

Productivity

With growing requirements and a shortage of people, "doing more with less" will be the norm. Since an increasing requirements backlog is unacceptable, it becomes important to focus on productivity. A 1984 "Report on Data Systems Management and Manpower Impacts" estimates the Air Force needs a productivity "increase of 19%, compounded annually," to eliminate the ADP requirements backlog by 1990 (28:11-7). However, there are several management impediments to increasing productivity. The following are seven barriers:

> Improving productivity is often a slow process. Unfortunately there is strong pressure to produce quick results. Assignment rotation may even be a contributing factor. Not 'getting around' is another barrier to increasing productivity and motivating subordinates. Managers unwillingness to experiment with new ideas hurts initiative. Regulations and policies created 'on high' may not address the real problems. Lack of incentives and poor management training are two more hurdles. Finally, the 'never enough time to do it right but always enough time to do it over' is a common problem in all disciplines (40:96-98).

Training

Recall from the previous chapter four categories of issues affecting software development. One of those issues, intellectual skills, dealt soley with training. Robertson made several points. First, develop a good training program. Second, ensure workers and managers attend. Third, realize training meets the intrinsic growth needs of professionals, keeps them current with technology, increases productivity, and serves as a reward. Finally, understand a short term decrease in productivity will be offset by a long term increase in productivity (40:92-93).

While the DOD has several training facilities and programs, there has been some criticism. One surveyed report, by Hedges, dealt solely with this issue. His view was the DOD trains only for the programming step (23:22-23) but there are five other areas of computer science: design, acquisition, implementation, operations and maintenance (23:38-41). Furthermore, he claims problems in software cannot "always be placed at the doorstep of changing requirements, lengthy approval times for acquisition, and so on." "Austere training budgets" fail to keep software people "smart" (23:26). With the continual flux in methods, keeping abreast of new techniques is paramount. "Disciplines like systems engineering, data base engineering, and computer networking offer tomorrow's challenges. Aggressive training in all of these areas should be a high priority" (41:7).

Project Management Oversight

Oversight is important because cost overruns can impact budgets and cause funds to be unavailable for other projects. Also, delays in delivery can hurt operational readiness. Though important, there are difficulties in managing software development projects. One problem is finding a reliable method for estimating costs. A popular method, by Boehm, is successful only 68 percent of the time in estimating costs within 20 percent of the actual expenditure (40:109). Unfortunately, it appears Boehm's method is the best predictor at the moment. Scheduling is another problem, i.e many managers fail to keep up with cumbersome PERT charts or do not have access to automated project management tools. It is not only important to estimate cost but to estimate schedules and then manage to cost and schedule (29:2-3).

A problem with embedded software in large weapon systems is it is not reported as a line item in the status reports and becomes overlooked by management (28:23). However, the software component may be the essential element deserving senior level attention. For example:

> The X-29 supersonic aircraft with the forward swept wing design illustrates the critical nature of software to system performance. Totally dependent on software controlled electronic commands, the aircraft requires instant response and feedback at subsonic speeds to compensate for its instability at those speeds. (e.g., Any pitch divergence can rapidly double in tenths of seconds). Rapid and accurate data feedback is required to insure balance and reliable performance. In brief, while the software component is not necessarily the largest element of the system, it is vital to the functioning of the aircraft (24:3).

Sometimes the opposite occurs and status reports are too detailed, becoming useless and costly.

Another study suggested high level management has little time for the subject or is software illiterate (29:3-18), harking back to the issue of technical competency. "Every software oversight manager should be trained in both computer software/hardware engineering design concepts and practices and oversight management techniques" (29:3-23). Whether it is little time or software illiteracy, lack of oversight can result in a reduction of mission capability, cost overruns and schedule delays. "For example, the Air Force and the Navy had to postpone making some much-coveted changes for the better in their respective F-16 and F/A-18 fighters when the software that they had counted on for such changes - as in radars - was not ready on schedule" (5:47).

Security

In conducting this research, it was disconcerting to note few articles exist on the software crisis and security. The author believes the following merits reflection:

1. As the Air Force increases its' emphasis on contracting for software development (28:7-4), more and more systems will be developed "out-of-house." While "in-house" development does not guarantee uncompromised security, it can limit access to software and documentation. At the least, administration, phone calls, and mailings between the requirement activity and the contractor complicates security procedures.

2. The software development process may involve computer networking. One should note networks and security have conflicting goals (41:10).

3. Usual security thinking dwells on preventing compromises. However, there are at least two other concerns: sabotage by disgruntled workers and damage to data by poorly written software. One article (16:--) referred to the latter as "unsafe" or "hazardous" software.

4. Who will check commercial, "off-the-shelf" software for reliability and security (44:--)?

MANAGEMENT ISSUES SUMMARY

This chapter has highlighted seven broad areas of management issues. All of these issues concern human resource and project management. While the problems vary in nature, their complexity is attested to by the fact some of the issues are nearly two decades old. The need for a "bold stroke" is evident.

Chapter Five

PROPOSED COURSE

This chapter begins with the groundwork of whom the students will be, the focus of the course, and what is meant by "computer literacy." The chapter concludes with the suggested course length and topics supporting Task II-A of Project Bold Stroke. The intent is to provide a course for Air Force senior leadership to address the issues and avoid the pitfalls presented in the previous two chapters.

PREFACE TO THE PROPOSAL

Analysis of the Audience

Task II-A has not been modified as of this writing. This means students in the short course will be general officers and equivalent grade civilians who have responsibility for software development. (If they have responsibility for software development, they may be familiar with software issues.) Task II-A also specifies newly selected brigadier generals attend. Likely, most personnel at this level have varied background with a strong dose of command and operations experience. In other words, specialization in software development, a support field, is unlikely for most students. The course should aim at students with little knowledge of the course material.

The "ADP" Versus "Embedded" Focus

Initially, Project Bold Stroke was about embedded software systems. During the team's investigation it became clear the problems incurred in ADP and embedded systems were similar (43:--). However, there appears to be an attempt to stress one facet over the other. A senior official has emphasized Project Bold Stroke should focus attention on software engineering in the operational and acquisition worlds rather than being a generalized "information systems" education (19:--). This is an accepted euphemism for ADP vice embedded software. Several months later a senior official expressed the same view and criticized the Project as drifting towards "AFCC type information systems"; Bold Stroke should focus on embedded weapon system software managed by AFSC and should be about the acquisition and management of mission critical software (20:--). Another senior official agreed Project Bold Stroke was for weapon systems (10:8)

but wanted to create a project similar to Bold Stroke for increasing awareness of ADP type systems (18:--). It is hoped that personal viewpoints will not override the Bold Stroke team's findings: the problems are serious, affecting both ADPE and mission critical systems (25:--; 43:--). Recall from Chapter One, this paper assumes there are few differences in the problems affecting both types of systems i.e., there are many similarities in managing ADP and embedded software.

The Goal of Computer Literacy

"Because each profession or discipline has different specific needs that can be met by computers, it is impossible to develop a generic computer literacy course" (3:74). Bold Stroke Task II-A targets senior leaders having software management responsibilities as the "profession or discipline" needing the short course in software literacy. For them, a Rand study provides a good definition of computer literacy:

> Computer literate is a loosely defined term implying an awareness on some level of understanding about aspects of computer affairs. It is usually applied to individuals who are not formally trained in a computer discipline but who need an overview, often in depth, of the field for some purpose. From the viewpoint of the professional practitioner, 'computer literate' would describe the lay person who can speak knowledgeably about computers in general. . . . The term emphasizes the insights and perspectives relevant to overseeing computer-related project development (27:2).

PROPOSED COURSE OF INSTRUCTION

Course Length

A "minimum course in computer literacy," teaching only microcomputer fundamentals, "would require about sixteen hours of carefully planned instruction" (3:74). This falls far short of teaching the issues presented in the previous two chapters. Task II-A suggested a course of three to five days; the author recommends five days. While allowing adequate time for instruction on the complex issues, this length has an added benefit: if the Air Force shows its' concern by encouraging senior leaders to devote valuable time to these subjects, then it lends more credibility to the criticality of software in Air Force systems.

Course Topics (Appendix A)

The author recommends the first day of instruction serve as a motivator and a "break-in" period for the students because computer technology can cause apprehension and hostility among the uninitiated and senior staff (8:183; 11:34; 14:43).

> A new jargon has emerged to describe people s negative reactions that make it difficult for them to use the technology effectively. Such terms as 'computerphobia, cyberphobia, technophobia,' and 'technostress' characterize the resistance to change in the work place and emphasize how critical it is to understand and plan for the human perspective when installing new technology (9:12).

This "break-in" and introductory phase on day one, should use personal computers (PC). They are small, individual computers: easily used and demonstrated. Contrast this vehicle of instruction to a tour of a large (overwhelming) ADP facility or displaying a ruggedized, embedded (mysterious) system. Furthermore, the instructor can open the PC to show the internal hardware, destroying the notion of computers as "black boxes". PCs are popular conversation items and widely available commercially. This can be a motivation to learn (3:3). Instruction should cover PC hardware and software. In doing so, the instructor must avoid one common error.

> . . . in computer literacy classes is the tendency to teach about a particular computer as if knowledge of its special features were important to computer literacy. . . . Computer literacy instructors must make every effort to stress the general principles, the transferable characteristics, and the universality of the concepts they teach. This can be done even while illustrating the particular implementation of a specific machine. . . . Moreover, a general discussion of the principal features of small computers offers an accessible and relatively simple approach to the general functions of all computers, however actually implemented in hardware (3:74-75).

After the introductory phase, the author recommends day two and three cover chapters two through four of this paper. At the least, these chapters cover the design, acquisition, implementation, programming, operation and maintenance subjects recommended by Hedges in his study of computer science training within the DOD (23:37).

Day three should end with at least one case study on the software impact to readiness. For example, improvements to the F-16 were postponed because of delays in software delivery (5:47). The case study will reinforce days two and three while reminding the students of the criticality of software.

It now becomes appropriate, on day four, to present solutions. The author reiterates only highlights can be taught. Since in-depth presentations of possible solutions is not feasible, recommended readings or a desk reference set with checklists should supplement the lectures. The goal is to arm the students with the "right questions" to pose to their staffs when they have software oversight responsibilities. Examples of topics are: incremental or evolutionary acquisition; measuring programmer productivity; negotiating contracts; motivating personnel; predicting costs and controlling schedules; etc.

Finally, the course should conclude with a description of the organizations involved in embedded and ADP information systems within the Air Force, the DOD and the Federal government. The key is to give more than just wiring diagrams. The lecturer should identify who is the "expert help" and where it is available. If time permits, current DOD research in long term solutions could also be mentioned (37:14-20). Day five ends with a course review and student critique.

Chapter Six

CLOSING

SUMMARY

Project Bold Stroke is a HQ USAF program to combat problems in Air Force software management. A specific Bold Stroke requirement given to Air University is to develop a short course for senior officers and civilians. This paper attempts to address that requirement by reviewing the documented problems in the development arena and conclude with a list of recommended course topics.

In researching the topic this paper noted various reports have dealt with different aspects of the software development crisis. Some studies treat the issues within the context of command and control, embedded, or general purpose ADP systems. Some publications concentrate on contracting, cost estimating, or programmer productivity. Some simply address management. This paper tries to highlight most software issues so students in a software development literacy course, under the Project Bold Stroke program, may become aware of the pitfalls. Having assumed students of varied backgrounds but with little software expertise, the author argues for a course length of five days and concludes with a suggested list of topics.

RECOMMENDATIONS

Course Topics

The author suggests 30 to 40 percent of the course highlight software development problems in parallel with the presentation in chapters two through four. About half of that time should be given to highlighting solutions and guidance. A portion of the course should provide hands-on experience and discussion of other Federal agencies the student could contact for help. Appendix A contains a list of topics. Appendix B is a recommended student reading list and a start towards a deskset reference.

Items For Further Study

In developing this paper, several references spawned questions and ideas beyond the scope of this research. Appendix C contains a list of items for further study.

FINAL REMARKS

> Software is the most troublesome current problem in. . . system design and development. It is the primary reason why some systems in existence or under development have not achieved their performance goals. It is now also the most costly part of a. . . system and software costs continue to grow while hardware costs continue to decline (1:33).
>
> The Select Committee heard presentations from over 300 representatives of 97 government, military and industrial agencies. . . . The briefings. . . covered the original mode of operation of the agencies, their successes, failures, and problems with computers. . . . The overall impression conveyed by these presentations was one of uncertainty created by a mass of detailed problems. . . . It became apparent to the committee, at an early date, that most of these problems. . . were in reality symptoms of the more basic problems. The evidence points to the fundamental problem of system definition and acquisition, cost and complexity, and expertise (30:10-11).

These quotes, written in 1974 and 1970, respectively, sum up much of today s thinking. Unfortunately, the age of these comments point out the software crisis was recognized long ago. "That crisis has been building for years. Jointly and separately, the military services and the Office of the Secretary of Defense have studied software problems off and on ever since programmable digital electronics began entering combat systems in the 1970s" (5:49). A break-through is needed and Project Bold Stroke may be a unique idea by focusing on awareness at the top through formal, Air Force-wide programs. Its success will depend upon support from the same community it will attempt to educate.

BIBLIOGRAPHY

A. REFERENCES CITED

Books

1. Boyes, Jon L. (ed). Issues in C3I Program Management: Requirements, Systems, and Operations. Washington: AFCEA International Press, 1984.

2. Head, Robert V. Federal Information Systems Management: Issues and New Directions: A Staff Paper. Washington, D.C.: Brookings Institution, 1982.

3. Lombardi, John V. Computer Literacy: The Basic Concepts and Language. Bloomington: Indiana University Press, 1983.

4. Martin, James. An Information Systems Manifesto. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

Articles and Periodicals

5. Canan, James W. "The Software Crisis." Air Force Magazine, Vol 69 (May 1986), pp. 46-52.

6. Corrigan, Richard. "The Software Bottleneck." National Journal, Vol 17 (9 November 1985), pp. 2540-2541.

7. Crafts, Ralph. "The Pitfalls of ADA Translation." Defense Science and Engineering, (June 1986), pp. 73-74.

8. Denton, D. Keith. "Success with High Tech: It's Up to You." Personnel Administrator, (March 1986), pp. 96-105.

9. Faerstein, Paul H. "Fighting Computer Anxiety." Personnel, (January 1986), pp. 12-17.

10. Famiglietti, Leonard. "AF Managers to Receive Computer Training." AF Times, Vol 40 (27 January 1986), p. 8.

CONTINUED

11. Mainiero, Lisa A. and DeMichiell, Robert L. "Minimizing Employee Resistance to Technological Change." Personnel, (July 1986), pp. 32-37.

12. Marcus, Daniel J. "Project Bold Stroke." Signal Magazine, (April 1986), pp. 100-101.

13. Robertson, Jack. "Harsh Words on Software." Electronic News, Vol 30, no 1483 (6 February 1984), p. 12.

14. Poth, Terence. "Finished at Forty." The Wall Street Journal, (16 September 1985), pp. 43c, 52c.

15. Royce, Knut. "AF Illiterate in Computer Software." Current News, Part 2 (9 December 1985), pp. 1-2.

16. Scherer, Maurice E. Jr. "Unsafe Software - The Missing Security Perspective." Computer Security Journal, Vol 3, No 1 (Summer 1984), pp. 41-52.

17. Smith, Jeffrey. "Some Problems with Ada in Real-Time Embedded Systems." Defense Science and Electronics, Vol 4 (October 1985), pp. 45-46.

Official Documents

18. Carver, Richard C. Memorandum for AF/CV, 11 July 1986.

19. Cooper, Thomas E. Memorandum for AF/CV, 6 May 1986.

20. Cooper, Thomas E. Memorandum for AF/CV, 17 Nov 1986.

21. General Accounting Office. Contracting for Computer Software Development - Serious Problems Require Management Attention to Avoid Wasting Additional Millions. FGMSD-80-4. Washington, D.C.: 9 November 1979.

CONTINUED

22. General Accounting Office. Federal Agencies Maintenance of Computer Programs: Expensive and Undermanaged. AFMD-81-25. Washington, D.C.: 26 February 1981.

23. Hedges, Robert L. Computer Science Training in the Department of Defense: The Silent Problem. Washington, D.C.: National Defense University Press, 1983.

24. Institute for Defense Analyses. DOD Related Software Technology Requirements, Practices, and Prospects for the Future. June 1984.

25. Piotrowski, John P., Gen, USAF. Memorandum: Project Bold Stroke - A Plan to Improve Air Force Software Management Expertise. 29 November 1985.

26. Ramsey, Joseph C. Jr., Col, USAF. Project Bold Stroke: USAF Software Management Action Plan. 29 November 1985.

27. Rand Corporation. Perspectives on Oversight Management of Software Development Projects. July 1983.

28. USAF Management Analysis Group. Report on Data Systems Management and Manpower Impacts. Washington, D.C.: 1 September 1984.

29. USAF Scientific Advisory Board. Report of the USAF Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission-Critical Software. December 1983.

30. USAF Select Committee on Computer Technology Potential. An Air Force Study of Air Force Organizational Ability to Exploit and Manage Computer Technology. Washington, D.C.: 1970.

CONTINUED

Unpublished Materials

31. Albert, Cecilia C., Maj, USAF. "A Primer for Program Managers: Embedded Software Acquisition." Research study 86-0045, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

32. Amodeo, James M., Lt Col, USAF. "Computer Resource Education and Management." Research study AU-AWC-86-010, prepared at the Air War College, Air University, Maxwell Air Force Base, Alabama, May 1986.

33. Bold Stroke Briefing Team. "Bold Stroke Reference Book." 21 April 1986.

34. Brass, Edward M., Lt Col, USAF. "Computer Software Development: Managerial Insights." Research study AU-AWC-86-027, prepared at the Air War College, Air University, Maxwell Air Force Base, Alabama, May 1986.

35. Cochoy, Robert C., Maj, USAF. "Career Irritants Affecting Retention of AF Science and Engineering Officers." Research study 0435-80, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, May 1980.

36. Estep, Roger W., Maj, USAF. "An Analysis of the Impact of Technology on Professionalism in the USAF." Research study prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, June 1986.

37. Hanlon, Robert C., Lt Col, USAF. "Software Modernization." Research study AU-AWC-86-088, prepared at the Air War College, Air University, Maxwell Air Force Base, Alabama, May 1986.

38. Hughlett, Eric C., Lt Cmdr, USN. "A Framework for Software Development." Master's thesis, Naval Postgraduate School, Monterey, California, September 1984.

CONTINUED

39. Marvin, Kenneth L., Maj, USAF. "Desirability of Software-First Development in Systems Acquisition." Research study 85-1740, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1985.

40. Robertson, Timothy F., Capt, USA. "A Productivity Enhancement Study for the US Army Information Systems Engineering Command." Master's thesis, Naval Postgraduate School, Monterey, California, September 1985.

41. Sweeder, James, Maj, USAF. "Trends in Data Automation and Implications for the Air Force." Research study 86-2465, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

42. Wiedemer, Michael P., Maj, USAF. "Controlling Cost, Schedule, and Maintainability in Major, Air Force Unique, Real-time Control System Applications Software (MAFURCSAS)." Research study 85-2855, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1985.

Other Sources

43. Ramsey, Joseph C. Jr., Col, USAF. Telephone interview with author, 13 January 1987.

44. Tackett, Del, Maj, USAF. Interview with author, September 1986.

CONTINUED

B. RELATED SOURCES

Books

Auer, Joseph. Computer Contract Negotiations. New York: Van Nostrand Reinhold, 1981.

Articles and Periodicals

Famiglietti, Leonard. "AF Seeks to Boost Software Literacy." Air Force Times, Vol 46 (6 January 1986), p. 12.

Gallant, John. "Puzzling Over Productivity." Computerworld, Vol 20, No 10 (10 March 1986), pp. 25, 32.

Ghazarian, Sarc B. "Taking a Look at Development, Maintenance Techniques--Then, Now, the Future." Defense Science and Electronics, Vol 5 (June 1986), pp. 67-68+.

Hannon, John. "Military Computing." International Defense Review, Vol 18, No 9 (1985), pp. 1429-1440.

Harris, Charles E. "Negotiating Software Contracts." Datamation, (15 July 1985), pp. 52-58.

Hirsch, Edward, BGen, USA (ret). "Evolutionary Acquisition of C2 Systems." Signal Magazine, (September 1985), pp. 39-45.

Holden, Constance. "Soviets Launch Computer Literacy Drive." Science, Vol 231 (10 January 1986), pp. 109-110.

Jones, Capers. "How Not to Measure Programming Productivity." Computerworld, Vol 20, No 2 (13 January 1986), pp. 65-66, 70+.

CONTINUED

-----. "How Not to Measure Programming Quality." Computerworld, Vol 20, No 3 (20 January 1986), pp. 73, 76-79, 82.

Leford. "New Approach to Answering Software Acquisition." TIG Brief, Vol 38 (January 1986), pp. 18-19.

Michaelis, Marc, Maj, USA, and Scott, Dennis M. CW3, USA. "Computer Literacy and Beyond." Military Police Journal, Vol 12 (Summer 1985), pp. 26-29+.

Naatjes, C.S., Col, USA. "A Layman's Look at Fourth Generation Languages." Armed Forces Comptroller, Vol 30 (Summer 1985), pp. 24-25.

Schmidt, F. L., et al. "Impact of Valid Selection Procedures on Work-Force Productivity." Perspectives on Employee Staffing and Selection, pp. 82-95.

Welsh, Rikki L. "Qualities of Good Software." MIS Week, (13 October 1982), p. 38.

Official Documents

Drezner, Stephen M. and Ware, Willis H. Information Systems: The Challenge of the Future for the AFCC. Rand Note N-2162-AF. Rand Corporation. Santa Monica, CA: May 1984.

General Accounting Office. Federal Agencies Could Save Time and Money with Better Computer Software Alternatives. 20 May 1983.

General Accounting Office. Wider Use of Better Computer Software Technology Can Improve Management Control and Reduce Costs. FGMSD-80-38, 29 April 1980.

Hosley, David L., Col, USAF. Memorandum: Requests for Software Management Short Courses. 27 November 1985.

CONTINUED

Orr, Verne and Gabriel, Charles A., Gen, USAF. Memorandum: Air Force Software Management - Action Memorandum. 15 October 1985.

Unpublished Materials

Brown, Gary L., Lt Col., USA. "Effects of Generalization and Specialization on the Development of Commanders." Research study prepared at Carlisle Barracks, PA., Army War College, 16 April 1982.

Burg, Joan R., Maj, USAF. "Job Attitudes of AFCC Personnel." Research study 86-0410, Air Command and Staff College, Air University, Maxwell AFB, April 1986.

Clark, Gregory A., Maj, USAF. "Software Cost Estimation Models - Which One to Use?" Research study 86-0545. prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

Cristiani, Steven J., Maj, USAF. "Guidelines to Establish Qualifications for Computer Related Jobs." Research study 86-0620, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

Helbling, Victor M., Capt, USAF. "Microcomputer Software System Development: Suggested Revisions to MIL-STD-1521A for Cost-Effective Acquisition of Custom Software Through Software Engineering." Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, September 1983.

Ransbotham, James I. Jr., Lt Cmdr, USN and Moorehead, Donald F. Jr., Lt Cmdr, USN. "A Program Manager's Methodology for Developing Structured Design in Embedded Weapons Systems." Master's thesis, Naval Postgraduate School, Monterey, California, December 1983.

CONTINUED

Totsch, James P., Maj, USAF. "Implementing Automated Information Systems in the Air Force." Research study 84-2605, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1984.

APPENDICES

Appendix A

COURSE TOPIC OUTLINE

Day 1

- Course Introduction and Objectives
  - Bold Stroke Reference Book
- Introduction to Microcomputers
  - Hardware (with tear-down model)
    - CPU
    - Memory
      - ROM, RAM
    - Input Devices
      - Keyboard, Mouse, Lightpen, Tablet
    - Output Devices
      - Parallel/Serial Ports, Printers
    - Storage Devices
      - Floppies, Hard Disks, Tape Backup, Bubble
    - Add-on Cards
  - Software
    - DOS
    - Word Processing, Spreadsheets, DB Managers
- Hands-on Experience

Day 2

- Hands-on Experience (cont'd)
- Introduction to Software Development
  - Life Cycle Model
  - In-house Development
  - Contract Development
- Embedded Computer Systems
  - Embedded Software
    - Avionics Packages, Sensors, Heads-up Displays
- ADA

Day 3 AFR 700 vs. 800-series

Software Development Crisis
- Life Cycle Issues (Chapter 3 this report)
- Management Issues (Chapter 4 this report)

Case Studies on the Impact to Readiness

Day 4 Avoiding the Pitfalls

James Martin's Manifesto

Concept Development
- Incremental Design
- Prototypes
  - Screen Generators
  - Fourth Generation Languages
  - ADA

Contracting
- Evolutionary Method (BGen Hirsch)
- Deliverables

Project Oversight
- Costing
- Scheduling
- Validation and Verification (Includes testing)
- Deployment

Day 5 Organizations for Help
- AFSC
- AFCC (SISC, EID, AFCAC)
- DOD (NAVCOSACT, TRADOC)
- GSA
- Software Engineering Institute (SEI)
- Joint Logistics Commanders (JLC) Initiatives and Computer Management Committee

Review

Course Critique

Appendix B

RECOMMENDED COURSE READING LIST

Books

Auer, Joseph. Computer Contract Negotiations. New York: Van Nostrand Reinhold, 1981.

Boyes, Jon L. (ed). Issues in C3I Program Management: Requirements, Systems, and Operations. Washington: AFCEA International Press, 1984.

Brooks, Frederick P. Jr. The Mythical Man-Month. Reading, Massachusetts: Addison-Wesley Publishing Co., 1978.

Glaseman, S. Comparative Studies in Software Acquisition. Lexington Books, 1982.

Martin, James. An Information Systems Manifesto. Englewood Cliffs, N.J.: Prentice-Hall, 1984.

----- and McClure, Carma. Software Maintenance: The Problem and Its' Solutions.

Putnam, Lawrence H. Software Cost Estimating and Life Cycle Control: Getting the Software Numbers. New York: Computer Society Press, 1980.

Articles and Periodicals

Boehm, Barry W. "Software and its Impact: A qualitative Assessment." Datamation, (May 1973), pp. 48-59.

Gallant, John. "Puzzling Over Productivity." Computerworld, Vol 20, No 10 (10 March 1986), pp. 25, 32.

Harris, Charles E. "Negotiating Software Contracts." Datamation, (15 July 1985), pp. 52-58.

Hirsch, Edward, BGen, USA (ret). "Evolutionary Acquisition of C2 Systems." Signal Magazine, (September 1985), pp. 39-45.

Horton, Len. "Could Project Management Be the Next Super Product." Software News, (May 1985), pp. 41-44.

-----. "Users Find a Tool to Manage Themselves." Software News, (May 1985), pp. 45-49.

Houghton, Raymond C. "Software Development Tools: Profile." Computer Magazine, National Bureau of Standards, (May 1983).

Jones, Capers. "How Not to Measure Programming Productivity." Computerworld, Vol 20, No 2 (13 January 1986), pp. 65-66, 70+.

-----. "How Not to Measure Programming Quality." Computerworld, Vol 20, No 3 (20 January 1986), pp. 73, 76-79, 82.

Lynn, Bernie, Capt. "Operational Test and Evaluation Requirements for Contractor-Supported Software." Air Force Journal of Logistics, Vol 9, (Fall 1985), pp. 29-30.

Naatjes, C.S., Col, USA. "A Layman's Look at Fourth Generation Languages." Armed Forces Comptroller, Vol 30 (Summer 1985), pp. 24-25.

Scherer, Maurice E. Jr. "Unsafe Software - The Missing Security Perspective." Computer Security Journal, Vol 3, No 1 (Summer 1984), pp. 41-52.

Schmidt, F. L., et al. "Impact of Valid Selection Procedures on Work-Force Productivity." Perspectives on Employee Staffing and Selection, pp. 82-95.

Technical Management Committee of the Aerospace Industry Association. "Suggestions for DOD Management of Computer Software." Concepts, Vol 5, No 5 (Autumn 1982).

Welsh, Rikki L. "Qualities of Good Software." MIS Week, (13 October 1982), p. 38.

Official Documents

General Accounting Office. Contracting for Computer Software Development - Serious Problems Require Management Attention to Avoid Wasting Additional Millions. FGMSD-80-4. Washington, D.C.: 9 November 1979.

General Accounting Office. Federal Agencies Could Save Time and Money with Better Computer Software Alternatives. 20 May 1983.

General Accounting Office. Federal Agencies Maintenance of Computer Programs: Expensive and Undermanaged. AFMD-81-25. Washington, D.C.: 26 February 1981.

General Accounting Office. Greater Emphasis on Testing Needed to Make Computer Systems More Reliable and Less Costly. GAO-IMTEC-84-2. Washington, D.C.: 27 October 1983.

General Accounting Office. Ways to Improve Federal Management and Use of Productivity Based Reward Systems. FPCD-81-24. Washington, D.C.: 31 December 1980.

General Accounting Office. Wider Use of Better Computer Software Technology Can Improve Management Control and Reduce Costs. FGMSD-80-38, 29 April 1980.

Rand Corporation. "Perspectives on Oversight Management of Software Development Projects." July 1983.

United States Air Force Management Analysis Group. Report on Data Systems Management and Manpower Impacts. Washington, D.C.: 1 September 1984.

United States Air Force Scientific Advisory Board. Report of the USAF Scientific Advisory Board Ad Hoc Committee on the High Cost and Risk of Mission-Critical Software. December 1983.

Weiss, David M. The MUDD Report - A Case Study of Navy Software Development Practices. Naval Research Laboratory Report 7989, 21 May 1975.

Unpublished Materials

Albert, Cecilia C., Maj, USAF. "A Primer for Program Managers: Embedded Software Acquisition." Research study 86-0045, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

Bold Stroke Briefing Team. "Bold Stroke Reference Book." 21 April 1986.

Brass, Edward M., Lt Col, USAF. "Computer Software Development: Managerial Insights." Research study AU-AWC-86-027, prepared at the Air War College, Air University, Maxwell Air Force Base, Alabama, May 1986.

Clark, Gregory A., Maj, USAF. "Software Cost Estimation Models - Which One to Use?" Research study 86-0545, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1986.

Cochoy, Robert C., Maj, USAF. "Career Irritants Affecting Retention of AF Science and Engineering Officers." Research study 0435-80, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, May 1980.

Hanlon, Robert C., Lt Col, USAF. "Software Modernization." Research study AU-AWC-86-088, prepared at the Air War College, Air University, Maxwell Air Force Base, Alabama, May 1986.

Helbling, Victor M., Capt, USAF. "Microcomputer Software System Development: Suggested Revisions to MIL-STD-1521A for Cost-Effective Acquisition of Custom Software Through Software Engineering." Master's thesis, Air Force Institute of Technology, Wright-Patterson AFB, Ohio, September 1983.

Hughlett, Eric C., Lt Cmdr, USN. "A Framework for Software Development." Master's thesis, Naval Postgraduate School, Monterey, California, September 1984.

Marvin, Kenneth L., Maj, USAF. "Desirability of Software-First Development in Systems Acquisition." Research study 85-1740, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1985.

Ransbotham, James I. Jr., Lt Cmdr, USN and Moorehead, Donald F. Jr., Lt Cmdr, USN. "A Program Manager's Methodology for Developing Structured Design in Embedded Weapons Systems." Master's thesis, Naval Postgraduate School, Monterey, California, December 1983.

Robertson, Timothy F., Capt, USA. "A Productivity Enhancement Study for the US Army Information Systems Engineering Command." Master's thesis, Naval Postgraduate School, Monterey, California, September 1985.

Totsch, James P., Maj, USAF. "Implementing Automated Information Systems in the Air Force." Research study 84-2605, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1984.

Wiedemer, Michael P., Maj, USAF. "Controlling Cost, Schedule, and Maintainability in Major, Air Force Unique, Real-time Control System Applications Software (MAFURCSAS)." Research study 85-2855, prepared at the Air Command and Staff College, Air University, Maxwell Air Force Base, Alabama, April 1985.

Appendix C

ITEMS FOR FURTHER STUDY

1. What are realistic possibilities of automatic code generators? Will they exceed glorified report generators?

2. Are there differences between mainframe and embedded systems beyond hardware and procurement approaches (1:313-314)?

3. Programming productivity is sometimes measured as useable lines-of-code (LOC) per man-month. What factors contribute to Japan's rate of 3,500 LOC compared to the US's 183 LOC (5:52)?

4. It is known this crisis has been building for years. What has prevented the DOD from attacking the problem?

5. If today's problems are a function of computer illiteracy, is this a temporary crisis until the younger, computer literate generation matures? (For example: every academy student has access to a computer.) Can a get-well date be determined? What are the impacts on readiness till then?

6. No recent study of generalist versus specialist or whole-person versus technician was discovered? Could a dual track ladder work? What career management concept did Electronic Data Systems (EDS) of General Motors undertake?"

7. Has there been any survey of Air Force satisfaction with computer support? If one were done, could the adverse impact of the software crisis on user satisfaction be subtracted and valid feedback obtained?

8. How can today's computer specialist keep abreast of technology? Where is the Air Force headed and how is that information passed on to planners so as to glean from today's computer advancements the ideas for tomorrow's weapon systems?

9. Is there any mechanism to create an Air Force lobby to de-conflict congressional oversight from budgeting and acquisition (41:14)?

10. What will the impact of artificial intelligence and fourth generation languages have on the overall Air Force software development and maintenance problems?

11. Will ADA, when fully employed, provide the envisioned portability of software packages and reduce the need for some software development? What problems are associated with ADA (7:--; 17:--)?

12. What is the impact on security with increased contracting and off-the-shelf buys?

END

8-81

DTIC