A182.340 (3)

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE

1a. REPORT SECURITY CLASSIFICATION Unclassified	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for Public Release: Distribution Unlimited.
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S)	5. MONITORING ORGANIZATION REPORT NUMBER(S) BRMC-85-5096-III
6a. NAME OF PERFORMING ORGANIZATION D. Blosser & Associates 6b. OFFICE SYMBOL *(If applicable)*	7a. NAME OF MONITORING ORGANIZATION Air Force Business Research Mgt. Center
6c. ADDRESS *(City, State and ZIP Code)* 301 Princeton Dr., SE Albuquerque, NM 87106	7b. ADDRESS *(City, State and ZIP Code)* AFBRMC/RDCB Wright-Patterson AFB, OH 45433-6583
8a. NAME OF FUNDING/SPONSORING ORGANIZATION AFBRMC 8b. OFFICE SYMBOL *(If applicable)* RDCB	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-5096
8c. ADDRESS *(City, State and ZIP Code)*	10. SOURCE OF FUNDING NOS.

PROGRAM ELEMENT NO.	PROJECT NO.	TASK NO.	WORK UNIT NO.
71113	0	013	0

11. TITLE *(Include Security Classification)* (Unclassified) In-Plant Technical Assistance for Software*

12. PERSONAL AUTHOR(S) D. Blosser

13a. TYPE OF REPORT	13b. TIME COVERED	14. DATE OF REPORT *(Yr., Mo., Day)*	15. PAGE COUNT
Final	FROM 85-09-30 TO 86-09-29	86 09 29	116

16. SUPPLEMENTARY NOTATION

17. COSATI CODES		
FIELD	GROUP	SUB. GR.
14	4	7

18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)*
Contract Administration, software, quality assurance,

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

* Contract Administration

ABSTRACT: Differences in hardware and software contract administration were documented. Technical functions required are identified and recommendations for logical partition between the AFPRO and SPO focal points are made. Training of personnel in software contract administration is identified as well as policy changes needed to perform these recommended functions are identified.

20. DISTRIBUTION/AVAILABILITY OF ABSTRACT UNCLASSIFIED/UNLIMITED [✓ XXX] SAME AS RPT. ☐ DTIC USERS ☐	21. ABSTRACT SECURITY CLASSIFICATION Unclassified
22a. NAME OF RESPONSIBLE INDIVIDUAL Capt D. Smith	22b. TELEPHONE NUMBER *(Include Area Code)* (513) 255- 6221 22c. OFFICE SYMBOL RDCB

DD FORM 1473, 83 APR EDITION OF 1 JAN 73 IS OBSOLETE.

SECURITY CLASSIFICATION OF THIS PAGE

FINAL REPORT

IN-PLANT TECHNICAL ASSISTANCE

FOR

SOFTWARE CONTRACT ADMINISTRATION

F33615-85-C-5096

86 OCT 29

AFBRMC/RDCB
Area B, Bldg. 125, Room 2063
ATTN: Captain Dennis W. Smith
Wright-Patterson AFB, OH 45433-6503

Contractor:	Dennis Blosser & Associates
	301 Princeton Drive SE
	Albuquerque, NM 87106
	(505) 268-3538
	Attn: Dr. Dennis F. Blosser

TABLE OF CONTENTS

TABLE OF CONTENTS

EXECUTIVE SUMMARY

One purpose of the study was to document the differences in the development of hardware and software for USAF weapons systems. A further purpose was to document the functions presently being performed by the buying activity (SPO) and by contract administration services (CAS) personnel in software acquisition. Another purpose of the study was to document the technical functions which should be performed by SPO and CAS for more effective management of software acquisition. A final purpose of the study was to document personnel skills (for CAS) and policy changes needed to support the recommended technical functions.

(For convenient reference, the headings of this summary are identical to the chapter headings of the full text.)

General Differences in Hardware and Software

An increasing amount of the dollar and manpower expenditure for weapons system development is going into software. The following figure shows the dramatic shift for development personnel over two decades.

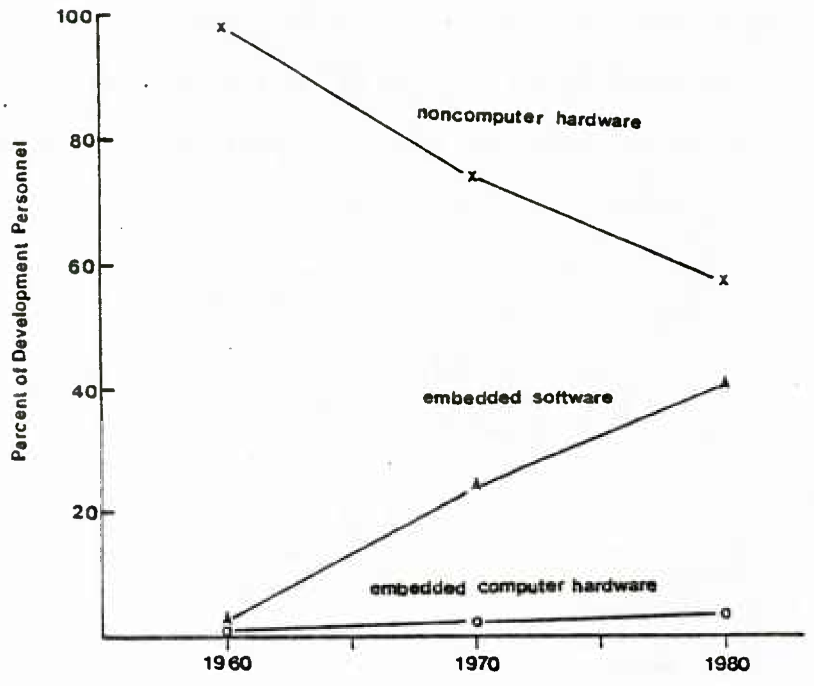

FIG. ONE

GROWTH IN SOFTWARE PERSONNEL
(Glaseman,1982)

Currently, control aspects of weapons systems are coming to be dominated by computer software and the appropriate methods and approaches for the acquisition of weapons system software bear subtle and critical differences from parallel methods for hardware.

Although the engineering <u>principles</u> applicable to software and hardware are similar, the tremendous differences in the essential nature of the two products requires that the <u>methods</u> for applying these principles be different. Since these differences and organizational tendencies to deal with them less effectively than desired form the basis for this study, the differences are discussed in the text in some detail. Table One shows the nine areas of software versus hardware similarities and differences which are discussed.

NINE AREAS OF DIFFERENCE
IN SOFTWARE AND HARDWARE
(Overview)

- Physical Existence
- Physical Dimensions
- Measurability
- Standard Elements
- Life-Cycle Relationships
- Design Precision
- Cost of Design Correction
- Quality Assurance
- Comprehensive Testing

TABLE ONE

The text includes suggested applications in each of these nine areas for the management of software acquisition. However, two recommendations based on these differences are especially clear and potent:

1. Early Involvement and Early Error Correction

In spite of any theoretical positions or current management philosophies, the orientation of DOD to quality assurance is understandably hardware-dominated, and as such, places its greatest manpower effort in the quality evaluation of end products. In the case of hardware, this is probably a good strategy. In pre-production and production development, the government definitely participates, but puts the large burden of technological manpower on the contractor. But, in the final phase of quality assurance, the government greatly steps up its participation to assure the final end-product, before it is accepted.

This orientation is inappropriate for software. The most critical activities in software take place in the design and testing phases. Not only is this in contrast to hardware orientation, but the <u>production phase for software</u> may even be considered virtually <u>nonexistent</u>. This is not to say that the production of <u>copies</u> of software is trivial. Reproduction of tapes, disks, etc., can be expensive and time consuming and must be quality assured like other production processes.

In place of a significant production phase for software, there is instead the software maintenance phase. This is far from being a phase where corrections to software might be made on a cost-effective basis, thus taking a marginally functional system and "tweaking it up" at reasonable cost. Instead, the maintenance phase for software has been repeatedly shown to be (Fairley, 1985):

Comm. Pages

(a) the most costly phase of the software life cycle, and,

(b) the life-cycle phase in which any given error costs the most to correct.

For an illustration of the cost of correcting errors at different points in the Life-Cycle, note the following figure.

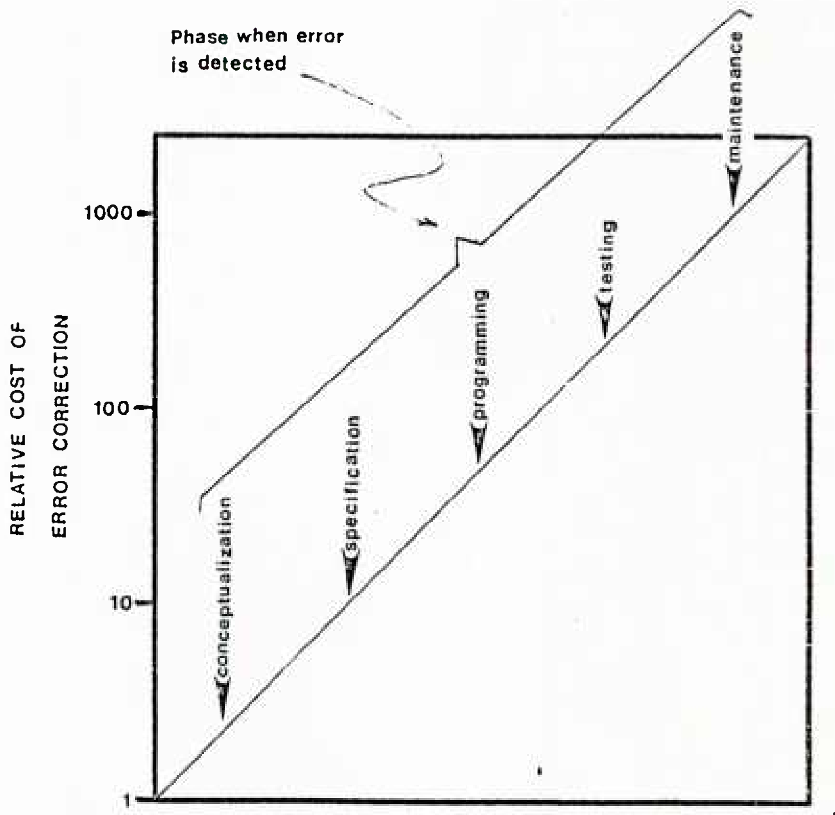

FIG. TWO
ECONOMY OF EARLY ERROR CORRECTION
(Martin,1985)

2. Early and Intense Attention to Testing.

In hardware systems, testing can usually be nearly exhaustive, if desired. Certainly, there are trade-offs and not all testing may be desireable relative to some other cost factors, but testing of components and complete systems is generally within option. The case of software is quite different. In the case of computer programs that are large enough to be significant (and the programs involved with weapons systems are very large), their complexity is so great that even automated testing by intelligent computer programs cannot permit exhaustive testing. The following diagram illustrates dramatically how the complexity of a conceptually fairly simple module quickly leads to impossible requirements for exhaustive testing.

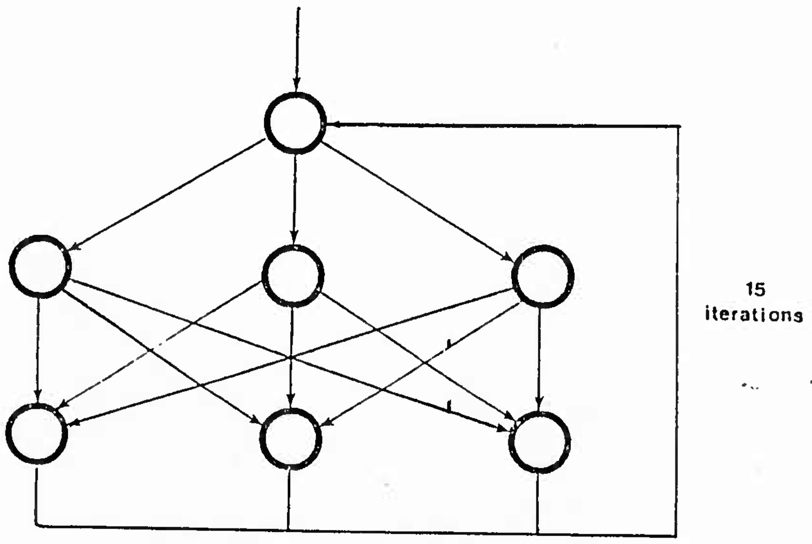

FIG. THREE
OVER 206 TRILLION UNIQUE PATHS

The capability for the program to be effective in the test phases must be laid down and assured from the earliest design phases. Thus, not only is it cost effective to correct problems in the design phase, when feasible, but it is also necessary to assure the quality of testability from the earliest design phases. Otherwise, the test phase can be explosive in terms of cost, schedule, and morale (Evans, 1984).

Life-Cycle Differences

The main differences between software and hardware in contract administration can best be viewed in terms of the weapons system life cycle (WSLC) and the software development cycle (SDC). The overall WSLC consists of four sequential and essentially discrete phases:

* Concept Exploration
* Demonstration and Validation
* Full-Scale Development
* Production and Deployment

Within the phases of the WSLC, hardware and software have development life cycles which are somewhat comparable. (NOTE: THE PHASES ARE SIMILAR IN FUNCTION AND HAVE SIMILAR

NAMES, BUT THEY OPERATE VERY DIFFERENTLY WITHIN THE WSLC. See table, following.)

SOFTWARE DEVELOPMENT CYCLE	HARDWARE DEVELOPMENT CYCLE
1. System/Software Requirements Analysis	1. System/Hardware Requirements Analysis
2. Software Requirements Analysis	2. Hardware Requirements Analysis
3. Preliminary Design	3. Preliminary Design
4. Detailed Design	4. Detailed Design
5. Coding, Unit Test and Computer Software Component Integration Testing	5. Fabrication
6. Computer Software Configuration Item Testing	6. Hardware Configuration Item Testing

TABLE TWO

The terms used here are self-explanatory and the sequences are intuitively meaningful, as well as the parallels in software and hardware (some kind of parallelism should be present, since they develop as <u>linked</u> subsystems of a parent system). But here is one of the most critical differences for contract administration: phases of the hardware development cycle are expected to proceed in a linear fashion with virtually no overlap -- by contrast, the phases of the software development cycle are expected _a priori_ to:

* Overlap one another significantly
* Proceed _non_sequentially with an indeterminate number of loops between successive phases (this implies possible paths from phase 6.0 clear back to phase 1.0).
* Overspan the phases of the WSLC (the hardware cycle is expected to be finished complete within a given phase of the WSLC).

(It should be noted that both the hardware development cycle and the software development cycle are likely to appear in somewhat modified forms, as is appropriate, in the first two phases.)

The following figure attempts to illustrate the complexity of these relationships.

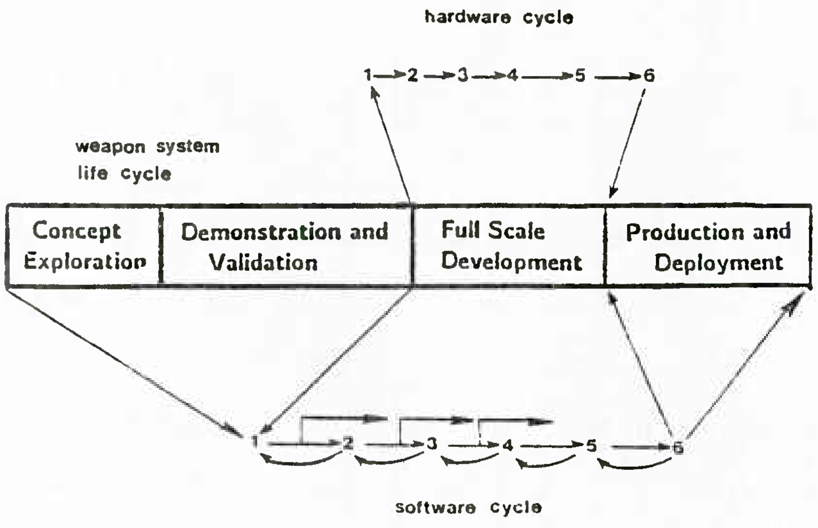

FIG. FOUR
COMPLEXITY OF SOFTWARE
DEVELOPMENT CYCLE

Training and Selection of Software Specialists

Personnel designated as software focal point usually have some, but typically minimal or somewhat irrelevant, training and background in software. Although the software focal point is not necessarily supposed to be highly competent in software, this person is typically looked to in their organization for leadership and technical competence in software problems. NOTE: There appears to be considerable difference of opinion on this from AFSC, HQ/AFCMD, and AFPRO viewpoints. If they are indeed to fulfill these expectations, they should be provided with:

* Development of their leadership abilities through training
* Development of their leadership roles
* Technical training (for them)
* Technical training (their support for their colleagues)

It is clear that training and personnel development needs for software duties are strong throughout the contract administration organization. Hardly anyone is expressing competence to do this job. The few who have strong competence know there is much more than they can do.

Nondeliverable Software

When nondeliverable software is used in the development and production of weapon systems, we have to ask the question:

* What are the consequences if nondeliverable software does not function properly?

Obviously, the consequences could range from trivial to very serious both in terms of cost and safety.

This leads to the conclusion that, even though some support software is nondeliverable, it still requires careful management and surveillance from both contractors and the government. Currently, there is little attention being paid to the various aspects of management of nondeliverable software. Notably, one approach to the management of nondeliverable software did surface with some consistency. This is a report by Major George Trevor which surveys the concerns of nondeliverable software and offers some guidelines. A draft version of this report is included in the Appendix.

It is hypothesized that the lack of attention to nondeliverable software comes from four sources:

(1) The very heavy and specific focus on deliverable software,

(2) Existence of clear-cut standards and procedures for deliverable software that are lacking for nondeliverables,

(3) Assumption that if it is nondeliverable, its development cannot be important enough to warrant management effort,

(4) Assumption that if the nondeliverable software is obtained from the government or a commercial source, it has to be O.K.

<u>Technical Tasks in Software Acquisition</u>

What technical tasks must the System Program Office (SPO) and in-plant Contract Administration Service (CAS) organizations perform in the acquisition of mission critical computer resources (MCCR) software?

Tasks in software acquisition were selected on the basis of these criteria:

(1) Task should represent an appreciable volume of work to be done, usually by more than one individual.

(2) Task should represent a process rather than an event.

(3) Task should be software dominated.

(4) Task should, as much as possible, represent a description of "what is to be done" rather than "how to," or criteria, or policy.

A total of 89 technical tasks were identified and are displayed in detail in Table Three of the text. While Table Three is intended to be comprehensive, there are some restrictions which must be made in this sense. First, there is the problem of diversity. Although one criterion in selecting tasks for this analysis was concreteness, this criterion itself (though useful) mitigates against comprehensiveness. Software applications in embedded systems are so diverse that any given project may have a great deal of uniqueness in its software. For this reason, the only way to cover all possibilities is to present tasks that are extremely generic or abstract. Hopefully, a useful middle ground has been achieved here.

However, any given application may certainly require tasks not represented here.

The second problem for comprehensiveness involves the problems of the extent of the effects of software. With increasing applications of computers and complexity of software, it is difficult to delimit the extent of the effects of software. The selection of tasks for this study implies a delimitation of the systemic influences of software. It is noted that this is sensible but arbitrary.

<u>Relation of Tasks to SPO and CAS</u>

The separation of responsibilities for technical tasks (shown in Table Five of the text) between SPO and CAS contains a great deal of information. However, the general implications of this separation can be characterized as reflecting:

(1) An increased amount of <u>required</u> consultation of the CAS by the SPO than is currently reflected in written policy and/or practice. This is especially true in the areas of pre-award survey and RFP development. Interviews conducted in the study indicate that under current practice, valuable knowledge developed by CAS agencies through their long-term relationships with day-to-day contractor activities is under utilized and wasted in these areas. If followed, the

recommendations of Table Five should alleviate this situation, which should not be allowed to continue in the critical, costly, and problem-ridden area of software acquisition.

(2) More compulsory participation of CAS personnel in formal reviews during the development process. As specified here, required CAS participation would begin with the System Design Review (Table Three, number 66). Such early involvement is necessary in order for CAS personnel to do their jobs effectively through the later stages of the software development process. Also, CAS familiarity with day-to-day problems of software development offers the potential to predict and prevent potential problems earlier in the development process. This offers the possibility of great dollar and time/schedule savings to the government.

CAS Personnel Skills

The tasks identified in this study suggest the designation of three main areas of training/knowledge for CAS personnel:

(1) A core area, referred to as Software Development Management Evaluation. This area is introductory,

foundational, and basic, and recommended for <u>all</u> affected CAS personnel.

(2) An area specific to Engineering, referred to as <u>Software Development Technical Engineering Evaluation</u>.

(3) An area specific to Quality Assurance, referred to as <u>Software Quality Assurance Evaluation</u>.

More specific areas for each main heading above are outlined in the text.

A specific recommendation is made that further development of training proceed in the direction of establishing systems of training objectives similar to the system of KSAs (Knowledges, Skills, and Abilities) currently used by Quality Assurance. Such systems of specific training objectives should lend to more effective and more cost-effective training, in addition to other benefits.

Further needed increases in manpower acquisition for CAS software support, estimated by the Electronics Institute of America, average 14% per year for the next three years. Currently, the effects of Gramm-Rudman have not seriously eroded numbers of software CAS personnel. However, given that there was already a shortage of well-trained software personnel and a needed increase of 14% per year, it is clear that holding the status quo amounts to losing ground.

<u>Policy Recommendations</u>

The task-relations specified in Table Five are designated as recommendations for policy, <u>and</u> it is recommended that policy should be altered to <u>enforce</u> the relationships of Table Five. This contractor is aware that many of the relationships shown in Table Five already exist in written policy as "suggested," "where appropriate" (or similar phrases) or are even <u>required</u>. However, interviews conducted in this study show that in spite of guidance or requirements of written policy, in practice, there is insufficient involvement of CAS capabilities in the development of RFPs and contracts and too little early involvement of CAS personnel in formal reviews and audits to fully utilize and support their potential. Therefore, policy should be rewritten to <u>require and enforce</u> CAS involvement as shown in Table Five and discussed in Chapter 6.0 and 8.0 of this report.

<u>Ancillary Recommendations</u>

Based on informal observations and considerations, two additional recommendations are offered.

The first involves a "services-marketing" approach for CAS. This is based on the idea of not relying solely on the rearrangement and enforcement of policy to attain increased involvement for CAS in preaward activities and greater early contract involvement. The idea is to package these CAS services in terms of benefits to the SPOs and aggressively and persuasively market them.

The second involves streamlining of training. This could be done by establishing an entry or core course for all CAS personnel (horizontal and vertical) involved with software. More advanced and specific training could then be placed in a dynamic data base which could be continuously expanded and updated. This data base could be accessed on an ad lib basis by individuals and groups. This would provide highly tailored and individualized training at low dollar and time cost.

1.0 GENERAL DIFFERENCES IN HARDWARE AND SOFTWARE

1.1 Introduction.

Historically, weapons systems have been dominated by hardware orientations. Until very recent times, weapons were essentially hardware systems operated entirely by humans. Although the weapons themselves could be adjusted, these adjustments were determined and carried out (mostly) by fairly direct human judgment and action. With the advent of computers in weapons systems, many adaptive procedures and criteria could be programmed into systems via software which extended and removed (in time and space) the effects of human operators in many cases. This essentially amounts to turning over a certain amount of control of a weapons system to computer software.

For historical reasons, hardware-oriented methods and approaches to weapons system acquisition predominate. Currently, control aspects of these systems is coming to be dominated by software, and the appropriate methods of approaches for the acquisition of weapons system software bear subtle and critical differences from parallel methods for hardware. A reflection of the increasing amount of software involvement in one organization is shown in Figure One. (The directness of the relationship between

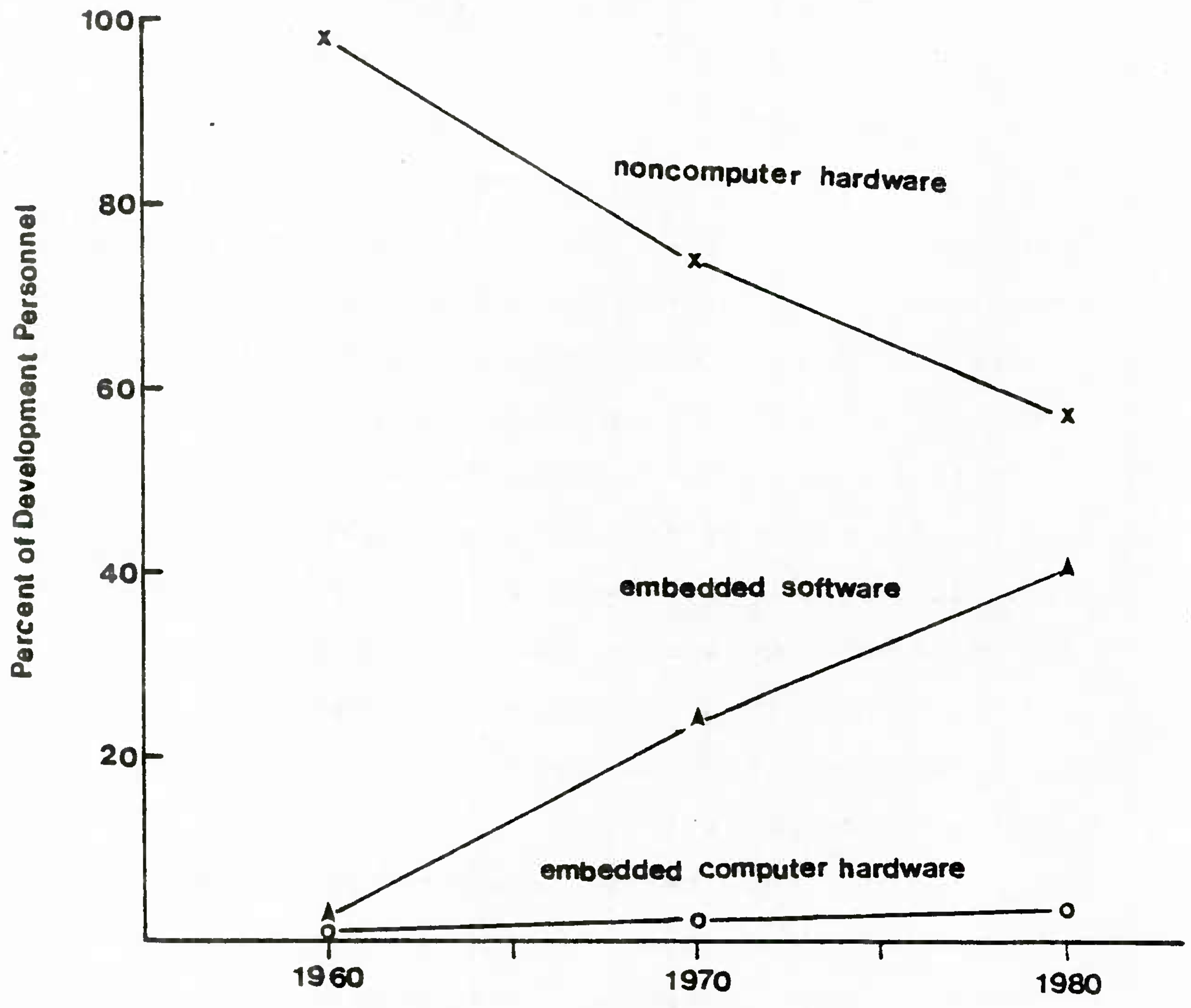

FIG. ONE

GROWTH IN SOFTWARE PERSONNEL
(Glaseman,1982)

percentage of personnel and amount of software produced is clearly an open question.)

Chapters 1.0 through 4.0 cover the preliminary phase (Phase II) of an investigation into the differences in hardware and software for contract administration, how these differences impact contract administration technical functions, training, needs and staffing needs, and implications for administration of nondeliverable computer software. The observations of these chapters serve the purpose of laying groundwork for more definitive observations in Chapter 5.0 through 9.0.

1.2 Method

Investigation for the preliminary part of the study consisted of document study and interviews. Its general goal was to give the contractor first-hand contact with some of the agencies involved in order to provide a realistic foundation for issues to be examined more formally in the final portion of the study. In addition to the examination of documents such as:

* DOD-STD-2167
* MIL-S-52779A
* MIL-STD-1679
* DOD-STD-2168 (May 9, 1985, draft version)
* AFCMD Regulation 800-3
* AFSCR 800-42

* AFCMD PAMPHLET 800-2
* AFCMDR 74-1
* DOD-STD-480A

and other government guides, etc., the contractor examined Statements of Work and Requests for Proposals. The contractor also gained valuable information and guidance from on-site meetings with personnel at the following agencies:

* AFCMD/HQ Kirtland AFB, NM
* SD/SPV Space Division, Los Angeles, CA
* AFPRO/Martin-Marrietta, Denver, CO

The observations begin with a general discussion of differences in software and hardware. The subtle similarities and differences in the engineering of software and hardware must be clearly differentiated. Insistence on detailing these differences does not imply that either government technical assistance personnel or DOD contractors are ignorant of them. More sophisticated members of these groups are well aware of these differences, and less sophisticated members no doubt easily recognize these differences when they are pointed out. Nevertheless, these differences are worth noting, because in one form or another, they continue repeatedly to be the underlying causes of the unique and serious problems that arise in the development of large software projects. Also, although differences in hardware and software are often elsewhere referred to by implication or in a passing manner, they are seldom dealt with directly or in detail. Failure to make these differences distinct has already contributed to a great deal of confusion in the development of software. The confusion will be compounded unless these distinctions are clearly highlighted.

Following the discussion of the differences in software and hardware, the critical differences for contract administration in life cycle are discussed. This is followed by possible implications for staffing and organization and a brief discussion of implications for nondeliverable software.

1.3 <u>Hardware vs. Software -- Nine Important Differences</u>.

Many of the differences between hardware and software that are outlined and discussed here may seem obvious to those with a good deal of experience in these matters. Still, these differences are worth discussing for several reasons:

(1) Although many who work for the government in MCCR software are quite sophisticated, many others are definitely <u>not</u> so sophisticated,

(2) Although the differences discussed here may have been noted by others, they are usually given partial, limited, or elliptical treatment,

(3) Currently, neither government nor the commercial sector engineers and manages the development of MCCR software with satisfactory effectiveness. Virtually all of the critical problems of software development arise from or are profoundly affected by the basic differences between hardware and software (Fox, 1982). Thus, until we can say we have significantly controlled the problems of software development, it is worthwhile to continue to consciously re-examine these matters both for the sophisticated and the not-so-sophisticated.

MCCR software is part of an engineered weapons system. As such, this software itself must be engineered and correspond to the life-cycle development process of weapons systems. The application of life-cycle concepts and engineering principles to software marked a great advance in the history of computer applications. However, there appears to be a strong organizational tendency to exert a hardware-engineering mind-set when applying engineering principles to software. This mind-set overlooks the profound differences between hardware and software.

Although the engineering <u>principles</u> applicable to software and hardware are similar, the tremendous differences in the essential nature of the two products requires that the <u>methods</u> for applying these principles be different. Since these differences and organizational tendencies to deal with them less effectively than desired form the basis for this study, the differences are discussed here in some detail. Table One shows the nine areas of software versus hardware similarities and differences which will be discussed.

1.3.1 <u>Physical Existence</u>. Hardware has physical existence. It consists of matter and takes up space. Software, by contrast, does not enjoy physical existence in that sense. The nonphysical existence of software is sometimes termed "virtual" as opposed to physical. Similarly, sometimes software is referred to as information or data consisting of instructions for a computer. Often, we casually speak of a computer program (software) as an object. However, the closest software comes to being an object is actually a <u>representation</u> of that software in some medium (e.g. print, magnetic disk). These may seem like fine distinctions, but they are legitimate, and they turn out to be crucial. Most, if not all, of the problems unique to developing MCCR software arise from its nonphysical property. Consider any traditional field of engineering. Consider the various fundamentals of measurement, observation, modeling, testing, etc., in that field of engineering. Consider how those fundamentals could (or could not) be applied to a component that is invisible, intangible, and has no physical existence.

```
* * * * * * * * * * * * * * * * * * * * * * * * * *
* Application Note for In-plant Technical          *
* Assistance:  Observation of the software as      *
* it develops (at the module level, say) is        *
* always indirect.  This means AFPRO personnel     *
* may be looking at code, or being given an        *
* electronic representation, or looking at         *
* plans, or specifications, or test results.       *
* Making a useful choice among these and inter-    *
* preting them requires different knowledge and    *
* criteria than would be applicable for compar-    *
* able observations of hardware development.       *
* Without adequate orientation and training,       *
* AFPRO personnel may feel that software doesn't*
* give them anything to "get a handle on" and      *
* thus avoid software duties when possible.        *
* * * * * * * * * * * * * * * * * * * * * * * * * *
```

1.3.2 Physical Dimensions. Hardware has physical dimensions, the simplest being those such as length, width, mass. Since software lacks physical existence, its dimensions, similarly, are less concrete. The tangible physical dimensions of hardware form the basis for some very realistic guidelines and limits in the design and development of systems. For example, if a system component is proposed to a mechanical engineer which requires a large number (say 500,000) of moving parts, he will probably quickly reject its feasibility. It is too large in number of moving parts -- it just wouldn't work. However, the size of software systems seems to be limited only by computers with memories large enough to execute them and the imagination of designers. Because of concrete physical dimensionality, a too-large hardware system becomes obviously untrustworthy at a relatively small level.

Lacking physical dimensionality, there is no clear, simple guide in software as to how big is too big, as long as we think we understand it and have a big enough computer to run it. [Not too long ago, we would have thought that physical and memory size of computers would soon provide clear-cut limits for size of software systems, but current trends are just the opposite. Vastly more and more computer (hardware) power available at less and less cost in dollars, time, space, and weight is pushing software in the direction of larger and larger systems.] Some attempts are underway at developing a software "physics," but these are at a very early and tentative level, and their results are not available or appropriate for immediate engineering applications. In all likelihood, developments in these areas will proceed slowly due to their academic and theoretical nature.

* *

<u>Application Note for In-plant Technical Assistance</u>: About the only practical control AFPRO personnel can exert on size (and thus complexity) in software in MCCR applications is at the module level. The trend is clearly toward larger and larger software systems. So, the best control for size is to keep modules to a size limit. A common rule of thumb is that a module should be less than 100 lines of code on the average and should not exceed 200 lines of code at the extreme. A better approach would be to design modules that exhibit loose coupling (i.e., relative independence) to other modules and that exhibit internal cohesiveness or high internal relationships and then post-apply the numeric rules of thumb as contraints. In any case, the appropriate control point for AFPRO surveillance is the Unit Development Folder/Module level.

* *

1.3.3 Measurability. Due to its physical dimensionality, hardware is directly measurable. This is not to say that direct measurements are the only useful measurements for hardware. However, if it weren't for the feasibility of precise, reproducible measurements, we could never have produced hardware whose functions would be interesting to measure. Indeed, it has often been argued that measurement is the fundamental basis of science and the fundamental basis of engineering.

Software, having no physical dimensions, offers difficulties for the proposition of measurement. Consider, for instance, the simple question "How large is Program A?" A very simple answer would be to state how long Program A is in lines of code. But this leaves open questions such as: Is the code assembly language, or a high-order language? Which high-order language? Are all programs with the same number of lines of code in the same language truly of equal size [No.]? And so on.

These measurement difficulties, in turn, pose difficulties for AFPRO personnel in just about all areas of contract management regarding software. Evaluation of the propriety of software analysis, software design, and software implementation methods is made more difficult by the measurement problem. Progress in relation to cost and schedule expectations is difficult to evaluate. The various factors constituting final product quality in software are difficult to evaluate.

The measurement of software is being studied intensively, and developments are occurring at a rapid pace. However, there are as yet no thoroughly quantifiable <u>and</u> satisfactory answers to many of the basic needs of AFPROs in the area of measurement. At the conceptual level, good clarifications have been made of some of the basic factors which are <u>desireable</u> to measure in software. However, these factors are functional (e.g. understandability, testability) and as yet very few can be quantified to any degree, if at all. As more and more satisfactorily quantifiable measures of software become available, software engineering will benefit greatly and mature. This will lead to better contractor capability and an increase in the ability of AFPROs to perform critical surveillance. It should be clear, however, that even with such maturation, the basic differences between software and hardware will make the methods of measurement different, although the principle of measurement spans both areas.

```
* * * * * * * * * * * * * * * * * * * * * * * * *
* Application Note for In-plant Technical        *
* Assistance:  Currently, AFPRO personnel could *
* conceivably choose among metrics which are     *
* numerically quantifiable and metrics which are*
* conceptual, but not numerically quantifiable  *
* (or in some cases numeric, but questionable to*
* interpret).  There are metrics for software   *
* development process, metrics for software     *
* development output, and some which are not    *
* clear cut.  There are also measures available *
* which are useful management indicators which  *
* are not technically software metrics.  The    *
* best pay-off is to identify the decisions     *
* critical for MCCR contract management (and at *
* what state in the development process) and    *
* then search for a combination of measures (in-*
* dicators and/or metrics) which are affordable *
* to collect and compute and which can be use-  *
* fully interpreted, possibly, in combination.  *
* AFPRO personnel should be aware that metrics  *
* and indicators with low collection/computation*
* cost and high interpretation/decision value   *
* may not exist for all desireable situations.  *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

1.3.4 Standard Elements. Many of the elements of hardware engineering are highly standardized. Their make-up and functions are transportable with high reliability and accuracy, and much is known about their behaviors and limits. Furthermore, this standardization spans a very wide scope. For example, it can include single, elementary units of electronic components, such as transistors, that are extremely small and separate, up to complex electro-mechanical devices whose operation is considered a fully standardized procedure under proper conditions of calibration and usage.

By comparison, the current state of standardization for software is almost the opposite (this is in no way meant to demean efforts such as STARS and Ada, but to emphasize a situation that remains seriously problematic in spite of those admirable efforts). There are many different major languages used in MCCR by DOD, and even many of these languages have different dialects and versions which differ depending on the hardware system they utilize (and there are very many different hardware systems). Some studies have counted the number of languages and variants utilized within DOD to run literally into the hundreds. It seems hard to believe, but counts notwithstanding, there is a definite and pernicious proliferation of languages in MCCR software. Grady Booch (1983) has enumerated several of the costs linked to proliferation of languages in the DOD such as: dilution of effects of training, almost no technology transfer among projects, and a general diffusion of resources. A number of studies of the proliferation of computer languages in DOD-embedded systems pointed to the desirability of a single standard language and the issuance of Directives 5000.29 and 5000.31 which served to narrow the range of acceptable languages.

Nevertheless, the effects of these studies, the advent of Ada, and directives do not as yet appear to have deeply affected contractor software development practices.

The result of this is that relatively few standard elements or systems of software exist. Even the fundamental elemental statements of a computer language (e.g., FORTRAN) may not execute identically in different, but functionally similar environments. While this lack of standardization may not represent an intrinsic property of software as contrasted to hardware, it is currently and will continue to be, influenced by the peculiar nature of software.

1.3.5 Life-Cycle Relationships. Hardware engineering and the engineering of modern weapon systems gave rise to the Life-Cycle concept of weapons development. The Life-Cycle concept has provided great advantages in the successful development of systems, the management of resources throughout the Life-Cycle, assurance of conformation to standards, and other desireable achievements. The Life-Cycle concept initially evolved around systems that were not at all computerized. So, the mind-set associated with the Life-Cycle originated in, and has a long history of, purely hardware orientation. When computers began to be introduced into weapons systems, there was certainly not much initial motivation to view the situation as being different in any important sense. Although computers were novel, and their functions were novel, computers are hardware. There was already a well-established history of experience with complex electronic hardware, and the computer, as hardware, fit quite well into that mind-set.

Also, and this is the <u>most critical point</u>, the early computers were small (operating capacity) and weak. Correspondingly, the software which controlled those computers was small and easily managed. It seemed quite appropriate under those circumstances to continue with the hardware Life-Cycle concept for the system, including the computer hardware, and simply require the software as a data item on contracts, much like documentation for operation of the computer hardware. For that time, that thinking was quite appropriate. It would still be appropriate except that some sweeping and profound changes have occurred.

In the brief thirty or so years that have elapsed since digital computers first began to be integrated into weapons systems, their functions have increased tremendously in two important ways. First, computers have come to operate in a greater and greater <u>diversity</u> of roles. Early on, computers were used in only a few, isolated applications in weapons systems. Now, computer applications in weapons systems have multiplied to the point of seeming to be everywhere. As an example, in early applications, a very large weapon system such as a bomber aircraft might have one or two on-board computers to assist in such things as sighting for bomb-drops or navigation. Currently, a smaller system, such as a fighter aircraft, has multiple on-board computers that control many functions of the aircraft much like a biological nervous system. There are computers to manage the aiming and firing of various weapons on board, and some weapons (e.g., missiles) are under articulated computer guidance after firing. (Even this description greatly simplifies and understates the involvement of computers in this example.)

The second important change that is linked to increased diversity is increasingly complex function. The almost unbelievable advances in microminiaturization of computer circuitry have supported more different applications by providing more computer power (complexity of function) per dollar and per unit of weight and space than was dreamed of thirty years ago.

Given the peculiar nature of software and the problems of engineering software as outlined here, the results for scientific and/or engineering development of software have been virtually disastrous. Obviously, the function of computers in weapons systems is to control or assist in control of a system or its components. But, and this is <u>critical</u>, the computers themselves are in turn <u>controlled by software</u>. So here is the crisis: while the potential for control in weapons systems through advances in computer hardware has grown like a super-exponential, the realization of that potential is severely hindered, jeopardized, and made expensive by a definite lack of similar advancements in the software required to control the computers.

When it first became clear (circa 1968) that software development was getting out of control to the point that the idea of "engineering" software began to emerge, the Life-Cycle concept began to be applied to software. Like the other concepts discussed here, in <u>principle</u>, this is appropriate and valuable. Problems arise, however, if a hardware mind-set is maintained in applying the Life-Cycle principle to software. This is like the case of the other issues discussed here. The principles are the same, but due to the basic differences in hardware and software, the applications must be different in order to effective.

The Life-Cycle concept as a <u>principle</u> applies fruitfully to software. Problems arise when, in the context of a hardware weapons system, the subtle differences in the meaning of the concept of Life-Cycle for software, and specifically how the software Life-Cycle is integrated into the overall weapons system Life-Cycle, are neglected.

A detailing of how some of these problems can arise in DOD contract administration appears in Section 2.2. However, it is likely that the implications of Life-Cycle differences in hardware and software are likely to be neglected where:

(1) The history of the organization is heavily grounded in hardware systems.

(2) The organization's concepts of Life-Cycle originated in hardware and has a long history where hardware orientation predominates.

(3) Few of the organization's front-line workers have extensive, formal training in software, but do have extensive training/experience with hardware.

(4) Practices of hiring, promotion, retention, and compensation mitigate against rapid adjustment to new technology.

Obviously, an organization that meets the above requirements is DOD.

This is not to say that there have not been definite and appropriate attempts made to provide DOD workers with conceptual and legal tools to work with software. Simply, given the organizational nature of DOD, the historical DOD hardware mind-set, the problems facing the software industry in general, and the infant nature of software as compared to the demands of hardware potential, efforts have fallen short.

1.3.6 Design Precision. Creativity and perhaps even some degree of artistry are no doubt valuable abilities in any design undertaking. But, the orientation of the hardware system design is to always provide as rational and quantifiable a means as possible for evaluating different alternatives of design. Of course, not even in hardware can all design decisions be suitably or profitably quantified. But, the limitations for quantitative evaluation of designs in software is much more limited than in hardware. These limitations go back to the non-physical nature of software and current problems of software measurement. There are definitely methodologies available in software design. And, they offer definite rationales and advantages. However, software designs are ultimately like very precise verbal statements. Hardware designs, on the other hand, are like very precise descriptions of objects, measurements, and actions. One way of comparison is to consider the use of diagrams in hardware vs. software. Design diagrams in hardware ultimately are a way of creating images representing physical objects. Design diagrams in software are a way of creating images which represent data, or information, or commands--non-physical entities. In final analysis, design of software is more abstract than the design of hardware. As an abstraction, it lends itself to more possible instantiations than does a design for hardware. Thus, it is essentially less precise. Some writers (e.g., Martin, 1985; Hoare, 1975) have argued that the practice of software design and development lacks so much precision in areas such as cost control, reliability, and control of complexity that it scarcely dares to be associated with the term "engineering" as it is commonly understood.

1.3.7 Cost of Design Correction. In many hardware components, minor, sometimes even relatively major, design alterations can be reasonably allowed during production. Thus, a component may meet minimal acceptance standards and yet receive profitable alteration at a later phase. In other words, some changes in hardware design exhibit beneficial cost-effectiveness well after the design phase.

By contrast, alterations in software systems, even apparently minor ones, after the design phase become tremendously expensive. So expensive, in fact, that a rule of dollar and time economy is to make as many required changes as can be anticipated before leaving the design phase. Thus, relative to hardware, the design phase should be intensive and highly test-oriented. The primary reason for these differences has to do with the quality of modularity. Most components of hardware systems are highly functionally encapsulated. They affect each other only through clearly specified and controlled interfaces. Currently, the components of a software system seldom exhibit strong modularity. That is, they link to other components not only via their designed interfaces, but often through unintentional and/or unpredictable links to other components. Unwanted effects going from one software component to another in this fashion are called side effects.

Thus, when highly modular hardware components are redesigned during post-design phases, their system-wide effects are highly predictable. In the case of software, this predictability does not hold. Due to weak modularity, redesign of a software system component in post-design phase may have wide ranging side effects which will then require affected components to need redesign, which may in turn have wide ranging side effects, which will require redesign of other components, etc. The problem of modularity is well noted but by no means conquered. At present, the best solutions appear to be to anticipate and solve as many problems as possible and to try to build in modularity in the design stage -- easy to state, difficult to accomplish.

* *

<u>Application Note for In-plant Technical Assistance</u>: In practice, in spite of philosophies of "early involvement," AFPRO surveillance is most heavily oriented to testing, with little or virtually no strong involvement in design phases. However, the capability for the program to be effective in the test phases must be laid down and assured from the earliest design phases. Thus, not only is it cost effective to correct problems in the design phase, when feasible, but it is also necessary to assure the quality of testability from the earliest design phases. Otherwise, the test phase can be explosive in terms of cost, schedule, and morale (Evans, 1984).

* *

1.3.8 <u>Quality Assurance</u>. In simplified terms quality assurance is concerned (1) that quality be designed into the product, (2) that quality be designed into the production process and (3) that quality be executed in the production process as evidenced by evaluation of end-products. In spite of any theoretical positions or current management philosophies, the orientation of DOD to quality assurance is understandably hardware-dominated, and as such, places its greatest manpower effort in the third area of quality assurance described above. In the case of hardware, this is probably a good strategy. In pre-production and production development, the government definitely participates, but puts the large burden of technological manpower on the contractor. But, in the final phase of quality assurance, the government greatly steps up its participation to assure the final end-product, before it is accepted.

AFCMD Quality Assurance policy for Software (AFCMDP 74-3, Apr 83) clearly dictates this orientation as inappropriate for software.

```
* * * * * * * * * * * * * * * * * * * * * * * * *
*   Planning should start as soon as possible   *
* after contract award...                       *
* There are four computer software disciplines  *
* which are needed to assure a quality computer *
* software product.  They are:  1) Design,      *
* 2) Documentation, 3) Configuration Management *
* and 4) Quality Assurance.  If one of these is *
* missing, there is increased likelihood the    *
* computer software development effort will     *
* experience problems.    (AFCMDP 74-3).        *
* * * * * * * * * * * * * * * * * * * * * * * * *
```

As was alluded to in the immediately prior section and AFCMD policy, the most critical activities in software take place in the design and testing phases. Not only is this in contrast to hardware orientation, but also the <u>production phase for software</u> is so trivial, it may even be considered <u>nonexistent</u>.

In place of a significant production phase for software, there is instead the software maintenance phase. This is far from being a phase where corrections to software might be made on a cost-effective basis, thus taking a marginally functional system and "tweaking it up" at reasonable cost. Instead, the maintenance phase for software has been repeatedly shown to be (Fairley, 1985):

1) the most costly phase of the software life cycle, and,
2) the life-cycle phase in which any given error costs the most to correct.

For an illustration of the cost of correcting errors at different points in the Life-Cycle, note the following figure:

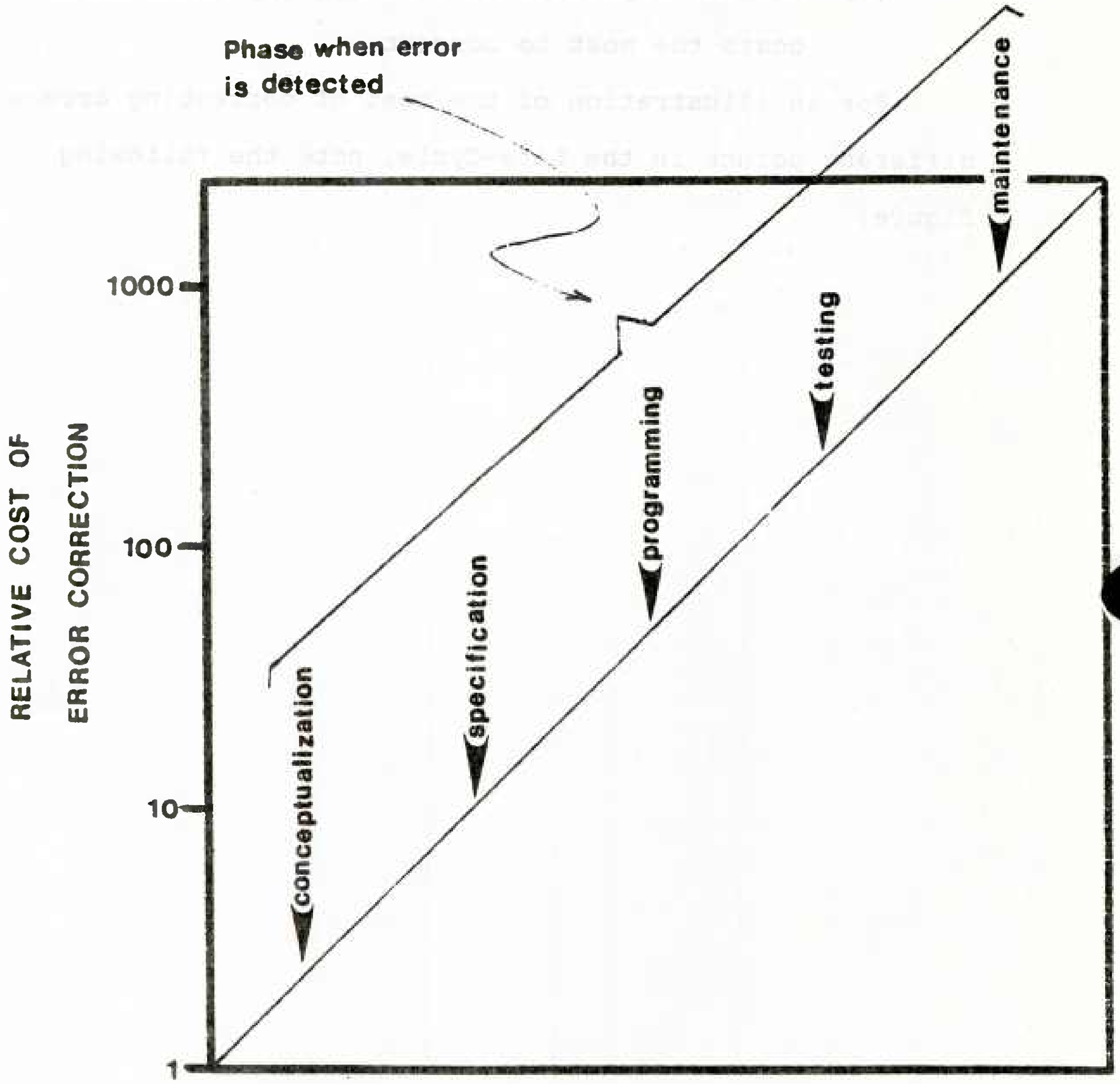

FIG. TWO
ECONOMY OF EARLY ERROR CORRECTION
(Martin,1985)

1.3.9 Comprehensive Testing. In hardware systems, testing can usually be nearly exhaustive, if desired. Certainly, there are trade-offs and not all testing may be desireable relative to some other cost factors, but testing of components and complete systems is generally within option. The case of software is quite different. In the case of computer programs that are large enough to be significant (and the programs involved with weapons systems are very large), their complexity is so great that even automated testing by intelligent computer programs cannot permit exhaustive testing. The following diagram illustrates dramatically how the complexity of a conceptually fairly simple module quickly leads to impossible requirements for exhaustive testing. This module contains in excess of 206 trillion unique paths. Assuming a mythical test generator that could devise and execute tests for the paths in such a module at a rate of one per millisecond, it would take over 6,000 years to exhaustively test all the unique paths.

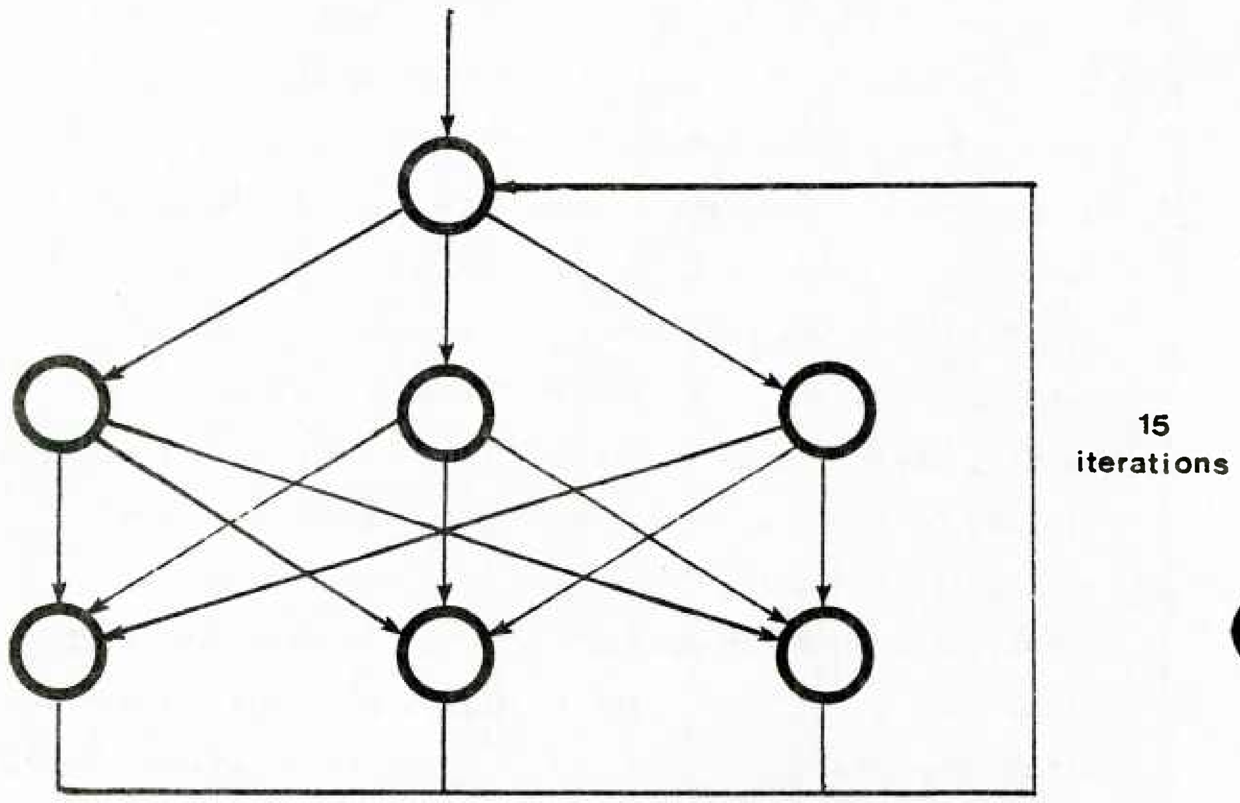

FIG. THREE
OVER 206 TRILLION UNIQUE PATHS

As a result, software developers are left with the task of deciding by <u>judgment</u> which paths of a program are critically important enough to test. This tends toward the paradoxical, since this judgment is necessary because the complexity is too great to permit analytical understanding. Due to this impossibility of exhaustive testing and the need to rely on judgment-based partial testing, software systems have a quality of being "haunted" by errors which are almost certainly there (since programming is <u>highly</u> error prone), yet which have not been detected and whose criticality and time of appearance cannot be assessed. Therefore, the development and application of better test methods must be relentlessly pursued.

2.0 LIFE-CYCLE DIFFERENCES

The main differences between software and hardware in contract administration can best be viewed in terms of the weapons system life cycle (WSLC) and the software development cycle (SDC). The overall WSLC consists of four sequential and essentially discrete phases:

* Concept Exploration
* Demonstration and Validation
* Full-Scale Development
* Production and Deployment

The first two phases of the WSLC are more study-like than the latter two. Concept exploration is oriented mostly toward paper study and computer simulations, examinations of trade-off effects, feasibility, etc. Demonstration and Validation is also study-oriented, but involves commitment of resources to development of some actual subsystems for study and testing. The outcome of the Demonstration and Validation phase is the capability to produce a fully functioning and reproducible prototype of the weapon system.

Full-scale development represents the first complete development of enough versions of the weapons system (including training and support) to test the fully operating system and move into development of large volumes of production in the final phase.

Within the phases of the WSLC, hardware and software have development life cycles which are somewhat comparable.

(<u>NOTE</u>: THE PHASES ARE SIMILAR IN FUNCTION AND HAVE SIMILAR NAMES, BUT THEY OPERATE VERY DIFFERENTLY WITHIN THE WSLC. See table, following.)

SOFTWARE DEVELOPMENT CYCLE	HARDWARE DEVELOPMENT CYCLE
1. System/Software Requirements Analysis	1. System/Hardware Requirements Analysis
2. Software Requirements Analysis	2. Hardware Requirements Analysis
3. Preliminary Design	3. Preliminary Design
4. Detailed Design	4. Detailed Design
5. Coding, Unit Test and Computer Software Component Integration Testing	5. Fabrication
6. Computer Software Configuration Item Testing	6. Hardware Configuration Item Testing

TABLE TWO

The terms used here are self-explanatory and the sequences are intuitively meaningful, as well as the parallels in software and hardware (some kind of parallelism should be present, since they develop as <u>linked</u> subsystems of a parent system). But here is one of the most critical differences for contract administration: phases of the hardware development cycle are expected to proceed in a linear fashion with virtually no overlap -- by contrast, the phases of the software development cycle are expected <u>a priori</u> to:

* Overlap one another significantly
* Proceed <u>non</u>sequentially with an indeterminate number of loops between successive phases (this implies possible paths from phase 6.0 clear back to phase 1.0).
* Overspan the phases of the WSLC (the hardware cycle is expected to be finished complete within a given phase of the WSLC).

(It should be noted that both the hardware development cycle and the software development cycle are likely to appear in somewhat modified forms, as is appropriate, in the first two phases.)

The following figure attempts to illustrate the complexity of these relationships.

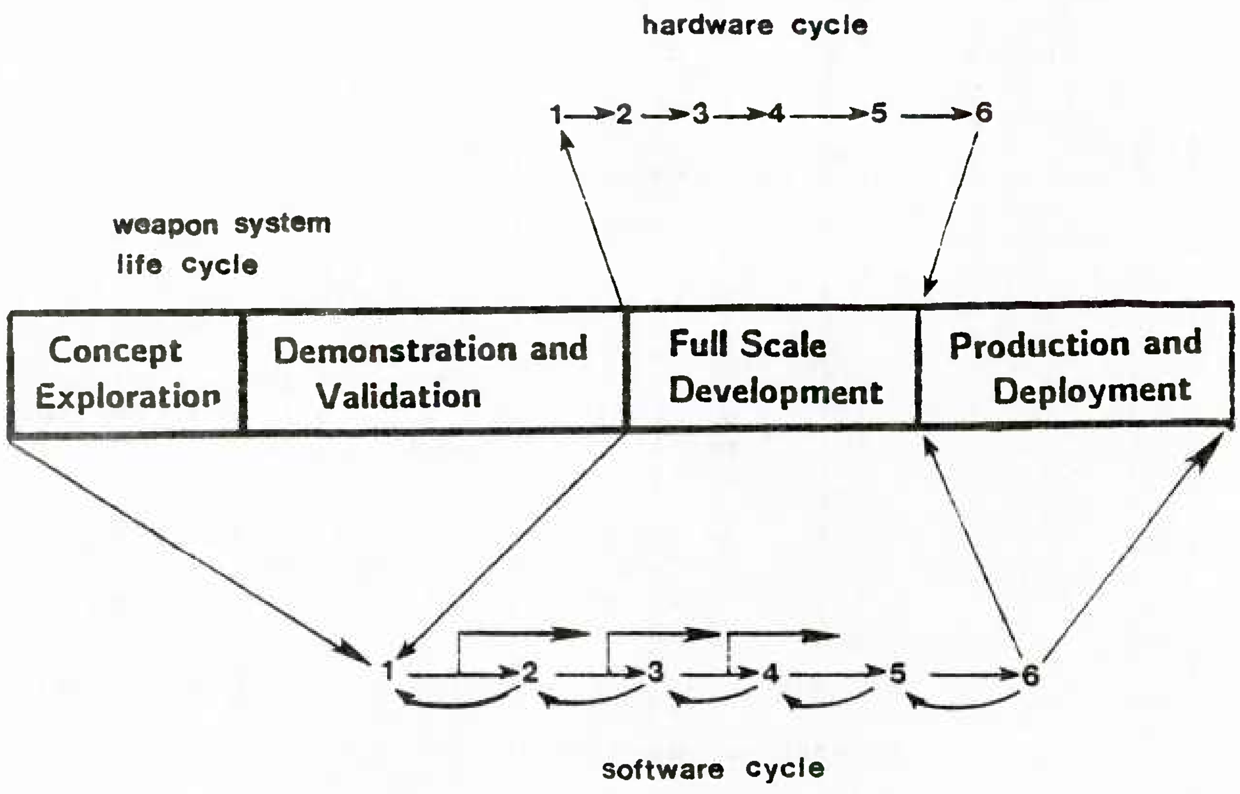

FIG. FOUR

COMPLEXITY OF SOFTWARE DEVELOPMENT CYCLE

This increased complexity of the software development cycle in its relationship to the WSLC has probable implications for manpower training and allocation in contract administration.

3.0 TRAINING AND SELECTION OF SOFTWARE SPECIALISTS

A popular myth has risen in the defense contracting industry over the past five years. The need for software engineers is tremendous. At the same time, avenues of formal education and experiential resources are producing qualified people at a very slow rate. Hence, the myth, born of need, pressure and misconception: "Just take a bright engineer who has had a few programming courses (or send him to a few), and then he will be your software engineer." (Pressman, 1982.)*

Generally, it doesn't work (exceptions noted). First, because software engineering and computer programming are not the same thing. Second, because a "few courses" in anything will not change the hardware engineering mind-set. Third, because people designated as "the software person" are looked to for leadership and authority. It is a very rare individual who can carry off that role with inadequate training and experience.

---

* The quotation is used here stylistically and does not represent a direct quote, but rather a paraphrase.

Regarding government contract administration, there seems to be a similar myth. Take whoever has any kind of computer training on his transcript (COBOL?, DP Systems? AI?) and designate him as software focal point. The point here is not to criticize software focal points, or to dampen creativity in selection practices of management, or to deter anyone from seeking and accepting career challenge. We need to examine what can be done to support software focal points through:

* Development of their leadership abilities through training
* Development of their leadership roles
* Technical training (for them)
* Technical training (their support for their colleagues)
* Other areas

Although statements have been made here about casual and general assumptions regarding training of people with special roles regarding software, it is clear that training and personnel development needs for software duties are strong throughout the contract administration organization. Hardly anyone is expressing competence to do this job. The few who have strong competence know there is much more than they can do.

(It has been pointed out that software focal points are not intended to be software specialists, but only administrative functionaries. However, the observation in

AFPROs is that software focal points are seen by themselves and others as specialists and that the duties of this role tend to expand over their other responsibilities.)

One of the assertions made frequently when AFPRO personnel note that a contractor is not abiding by some commonly accepted (or even common sense) software standard is that the Program Office did not require it on the contract; hence, it was not enforceable.

Assumedly, with the advent of DOD-STD-2167, much of this kind of situation will subside. Prior to DOD-STD-2167, the main written standards for software were MIL-S-52779A and MIL-STD-1679. These were inadequate by themselves and so many of the specifics necessary to bring a Statement of Work up to acceptable criteria had to be added as specific items to each contract. Clearly, this left much room for variance and ambiguity. Perhaps DOD-STD-2167, which is very comprehensive, articulated and up to date, will improve this situation to a degree. Nevertheless, it must be acknowledged that no set of standards, no matter how up to date or comprehensive, can substitute for a well-written SOW or the oversight of development by a competent government staff.

However, there is some reason to believe that part of Program Offices' not requiring some specifics grew out of a belief that the AFPROs lacked sufficient expertise to support and enforce the requirements. This (along with other direct assertions) suggests less than effective trust and communication between Program Office and AFPROs.

One of the needs for this study is to examine ways to insure that the positive potential of CAS personnel to improve software acquisition is not dampened by historical lack of trust and communication. New patterns of organizational linkages are one route that will be explored.

4.0 NONDELIVERABLE SOFTWARE

Nondeliverable software refers to that which is intended to serve as a support in developing systems (Computer Assisted Design software or Automatic Test software, for example) or in production of systems (Computer Assisted Manufacturing software, for example), but is not a primary end-product (deliverable) of the system development process. This software may be developed by a contractor as needed, or acquired from a commercial source or from the government.

Tremendous advantages are available through the use of such support software. In the case of test software, for example, a computer program can often generate and control the execution of numbers of test cases which would represent a number of man-years effort that would be literally impossible if done by human operators. In the area of production, Computer Assisted Manufacturing software can control processes with tolerances, speeds and continuity that could not be approached with human operators.

However, when nondeliverable software is used in the development and production of weapon systems, we have to ask the question:

* What are the consequences if nondeliverable software does not function properly?

Obviously, the consequences could range from trivial to very serious both in terms of cost and safety.

This leads to the conclusion that, even though some support software is nondeliverable, it still requires careful management and surveillance from both contractors and the government. There is currently very little attention being paid to the various aspects of management of nondeliverable software. Notably, one approach to the management of nondeliverable software did surface with some consistency. This is a report by Major George Trevor which surveys the concerns of nondeliverable software and offers some guidelines. A draft version of this report is included in the Appendix. At present, there is a level-of-awareness problem. In general, the lack of attention to nondeliverable software comes from four sources:

(1) The very heavy and specific focus on deliverable software,

(2) Existence of clear-cut standards and procedures for deliverable software that are lacking for nondeliverables,

(3) Assumption that if it is nondeliverable, its development cannot be important enough to warrant management effort,

(4) Assumption that if the nondeliverable software is obtained from the government or a commercial source, it has to be O.K.

A slightly different position than that characterized above was also observed. It may be characterized as "surveillance of nondeliverable software is important and MIL-STD-2167 says we should do it, but we don't even have enough manpower to fulfill the needs on deliverable software."

5.0 TECHNICAL TASKS IN SOFTWARE ACQUISITION

5.1 Introduction and Background

This portion of the research was undertaken to investigate, from the point of view of the Air Force Contract Management Division (AFCMD), the following general question: How can AFCMD and the System Program Offices (SPOs) carry out their joint responsibilities in the acquisition of mission critical computer resources (MCCR) software more effectively?

More specifically, under this general question, the following questions were to be addressed:

Should policies, practices, etc., be changed for more combined effectiveness?

What skills/training for contract administration personnel would support recommended changes?

This part of the research was conducted by three main methods in order to accumulate basic results relevant to the research questions:

(a) Review of Air Force documents concerning the formal policies for operations of SPOs and AFCMD in the acquisition of MCCR software,

(b) Interviews with SPO and AFCMD personnel at five various locations each throughout the country regarding the actualities of their operations (as compared to formal policy), and their viewpoints and suggestions regarding the basic questions of this research.

(c) Consideration of: (1) discrepancies between formal policy and reported actualities, (2) policy itself, as written, and (3) expressed attitudes and suggestions of those interviewed.

The existence of this study implies that there is a felt need to improve the effectiveness of MCCR software acquisition. Studies have shown (cf. Booch, 1983) that throughout the Department of Defense, software costs are increasing at a tremendous rate (six billion dollars in 1980, predicted to exceed 32 billion by 1990) while at the same time software is frequently behind schedule and of poor quality, which in turn impacts overall system delivery schedule and increases overall system costs.

One of the underlying factors in this state of affairs is a severe shortage of personnel trained in software engineering. As of 1984, only three institutions of higher education were training software engineers (Schaar, 1984). Although this situation is improving, it is predicted that a lag and shortage of well-trained and experienced personnel will be a problem for contractors and government for the foreseeable future.

Thus, one of the fundamental sources of problems in MCCR software acquisition is a <u>shortage of well-trained and</u>

<u>experienced personnel that strongly affects both the contractors and the government acquisition organizations</u>.

A second underlying factor in the problems of MCCR software acquisition is time pressure. A repeated observation by both SPO and AFCMD personnel was that many critical facets of the MCCR software acquisition process were swept over by the time pressure of schedules. This ranged through the whole acquisition process, from not having time to prepare Statements of Work and contractual requirements with sufficient thoroughness, to not being thorough enough with pre-award surveys, to glossing over failures in quality assurance, to glossing over software review and test problems, etc., all because of time pressure.

It is likely that the shortage of well-trained software personnel interacts with the time-pressure problem source.

A final source of problems should be considered from the AFCMD point of view. In light of the problematic nature of MCCR software acquisition, the shortage of well-trained personnel in this area, and the interactive critical effects of time pressure, <u>it is seriously ineffective and a waste of government funds to under utilize the capacity of any component of the government's MCCR software acquisition organizations</u>. Nevertheless, an Inspector General study in 1981 showed that in general, SPOs regarded AFCMD personnel as significantly lacking capability in the software area.

AFCMD headquarters personnel contend that this is currently not the case, but that the SPOs still regard AFCMD in this fashion and thus tend to under utilize their capabilities.

5.2 Identification of the Tasks

The first major requirement for this Phase of the study was to identify technical tasks to be performed by the government when acquiring software products. This corresponds to task 4.3.1 of the Statement of Work.

The strategy in identifying tasks for software acquisition was to limit the scope of tasks to those aspects of the software acquisition process which primarily involve SPO (System Program Offices) and CAS (Contract Administration Service) activities. This strategy was used, since the context of the State of Work (SOW) makes it clear that these aspects of software acquisition and these two agencies are the focus of the study.

Tasks in software acquisition were additionally selected on the basis of these further criteria:

(1) Task should represent an appreciable volume of work to be done, usually by more than one individual.

(2) Task should represent a process rather than an event.

(3) Task should be software dominated. (This

distinction is difficult, not always clear, and probably arguable).

(4) Task should, as much as possible, represent a description of "what is to be done" rather than "how to" or criteria or policy. (Once again, a sometimes less-than-clear distinction.)

Many sources were reviewed and influence the selection of tasks specified for this study (see References). In the final analysis, it was decided to move as close as possible to concrete tasks and away from generic or more abstractly stated tasks. For this reason, and due to ultimately arbitrary choices, the documents which were relied on to the greatest extent in the final analysis were:

(1) _Airborne Systems Software Acquisition Engineering Guidebook for Contracting for Software Acquisition_, ASD/ENAI.

(2) _Contracting for Software Acquisition_, Software Acquisition Guidebook Series, ASD/ENE

(3) _Contract Administration of Mission Critical Computer Resources in Systems Acquisition_, Joint Contract Administration Coordinating Council Guidebook, draft.

(4) Captain Dennis W. Smith, personal communication.

(5) _Guide to the Management of Embedded Computer Resources_, Space Division.

Table Three, below, contains the 89 tasks identified for software acquisition. The context here is software, so that is specifically implied for each task, even if not stated. The order of the tasks shown is in roughly chronological order. However, strict chronological order and sequentiality must not be expected.

While Table Three is intended to be comprehensive, there are some restrictions which must be made in this sense. First, there is the problem of diversity. Although one criterion in selecting tasks for this analysis was concreteness, this criterion itself (though useful) mitigates against comprehensiveness. Software applications in embedded systems are so diverse that any given project may have a great deal of uniqueness in its software. For this reason, the only way to cover all possibilities is to present tasks that are extremely generic or abstract. Hopefully, a useful middle-ground has been achieved here. However, any given application may certainly require tasks not represented here.

The second problem for comprehensiveness involves the problems of the extent of the effects of software. With increasing applications of computers and complexity of software, it is difficult to delimit the extent of the effects of software. The selection of tasks for this study implies a delimitation of the systemic influences of software. It is noted that this is sensible but arbitrary.

TABLE THREE

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

1. Use the following checklist as appropriate to prepare RFP package.

<u>Checklist</u>

A - Introduction

(1) Define software to be developed

(2) State function of software

(3) Identify users

B - Scope

(1) Define the development phase

(2) Define technical objectives in each phase

(3) Brief previous history

(4) Reference documents

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

Checklist Continued

C - Tasks

(1) Specific requirements for contractor

(2) No "to be determined" or informal tasking

(3) "Identify" and "design" tasks giving specific module functions

(4) Requirements for design management

(5) Requirements for test methodology planning, execution and evaluation

(6) Requirements (if any) for independent verification and validation

(7) Requirements for definition of acceptance certification

(8) How user training will be supported

(9) Requirements for operational simulations and/or rehearsals

(10) Requirements for methodology and capacity to modify and upgrade software

---

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

<u>Checklist Continued</u>

D - Review compliance documents for applicability

- Work Breakdown Structure (MIL-STD-881A)
- Configuration Management (MIL-STD-483) (tailored as required)
- Specification Practices (MIL-STD-490) (tailored as required)
- Technical Reviews and Audits for Systems, Equipment and Computer Programs (MIL-STD-1521A) (tailored as required)
- Software Quality Assurance Program Requirements (MIL-S-52779A)
- Specifications, Types and Forms (refers to MIL-STD-483/490) (MIL-S-83490)
- Software Development Standard (MIL-STD-2167A) (tailored as required)

---------- End of Checklist ----------

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

2. Request inclusion of <u>all</u> detailed assumptions and rationale for submitted software cost estimate.

3. Request that the proposal includes appropriate reuse of appropriate software previously developed by the contractor.

4. Refine the statement of work by internal review and revision.

5. <u>ALTERNATELY</u> (for large projects > $1M)
   - Compile a list of interested bidders.
   - Circulate draft RFP and get questions from bidders.
   - Hold a briefing for interested bidders to answer (anonymous) questions.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

6. Assess whether to separate conceptual validation and development phases into separate contracts.

7. Understand the contractual data items relevant to the specific procurement.

8. Customize the application of relevant data items to the specific environment and acquisition strategy.

9. Conduct a pre-award survey. Review a guidebook such as "Pre-Award Survey for Contractor Software Developments" (SD/ALR Guide, June 1982).

10. Submit the pre-award survey document to bidders upon receipt of their proposals and allow a minimum of four weeks for them to organize their responses prior to conducting the survey.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

11. Prepare a comprehensive set of questions for each major software area relevant to the contract and tailored to the contractor's environment.

12. Conduct the interviews on site with allowance for the answers to questions to spontaneously generate new questions or areas of investigation/audit as appropriate (i.e., highly skilled and software-experienced interviewers).

13. Assemble a team of highly skilled software management experts to conduct the pre-award survey.

14. Assess the survey results and submit the assessment to the appropriate AFPRO or DCAS.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

15. AFPRO or DCAS has responsibility to finalize pre-award report and provide the report to the SPO.

16. AFPRO/DCAS can conduct software pre-award, but SPO has final responsibility for assembly of total team.

17. If the contractor's (or contractor's division) average net profit is small relative to the size of the software contract and the contract is "firm, fixed-price," the financial capability aspect of the pre-award survey must be critically considered along with the software component. This is because estimates of software costs are extremely volatile, and actual costs may exceed estimates as a factor of ten. This situation poses a possibly severe risk for receiving the desired product.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

18. Obtain general assistance and information on contractor's general capabilities in software areas from AFCMD. Augment AFCMD capability with SPO personnel during conduct of the pre-award survey.

19. Establish channel of communication between SPO and AFCMD specific to new program/project.

20. Determine requirements necessary for letters of delegation to subcontractors.

21. Conference contract administration, quality assurance, engineering, and other AFCMD support as necessary to ensure flow of information to SPO.

22. Identify contract requirements for performance measurements.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

23. Define the relationships and responsibilities of the SPO and AFPRO for the contract, including a memorandum of agreement, if appropriate.

24. General review of draft RFP for consideration of all components to ensure enforceability of contract requirements, proper subcontract or management, complete pricing requirements, and consideration of delegation of some monitoring efforts to AFPRO by SPO not already included in the RFP.

25. Review of the draft SOW to determine whether the management requirements, software development of criteria, procedures, referenced to CDRLs and specifications, and other consideration provide adequate means to properly monitor the contract.

26. Review of the draft CDRLs to determine the effectiveness of the DIDs and detailing of CDRLs provide adequate means to properly monitor the contract.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

27. Review of the draft RFP to determine the reasonableness of the overall time duration; whether the delivery schedule for various plans supports their valid useability; identification of critical needs for long lead times; appropriateness of milestones and sequences; areas of high potential risk to schedule maintenance.

28. Review of the draft RFP to determine the presence of appropriate and current versions of standards and specifications and to check tailoring against requirements for valid administration of contract.

29. Prepare and review sample contract to ensure inclusion of necessary clauses and forms, deletion of unnecessary forms and clauses, proper cross references to MIL-specs included in SOW/CDRL, and other technical and clerical details.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

30. Prepare source selection plan.

31. Obtain approval of source selection plan from Source Selection Authority.

32. Establish criteria for evaluation.

33. Establish factors and standards.

34. Prepare an independent cost estimate.

35. Evaluate proposals.

36. Determine the competitive range.

37. Identify deficiencies in proposals.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

38. Carry out written/oral discussions with bidders.

39. Obtain final and definite contracts.

40. Negotiate prices with bidders.

41. Complete evaluation.

42. Prepare Source Selection Evaluation Board Report.

43. Prepare Source Selection Advisory Council Report.

44. Render final Source Selection Authority decision.

45. Document the source selection decision.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

46. Evaluation of proposal by CAS representatives based on past experiences with administration of contracts with similar requirements and characteristics.

47. Participation in Source Selection panels by CAS representatives specifically in regard to historical knowledge of abilities of bidders under consideration and experience with technical and management aspects of bidders' proposals.

48. Review contract as awarded to determine presence of software functional areas such as Work Breakdown Structure, Military Standards and Specifications, Contract Data Requirements Lists, Software Development Plan, Software Configuration Management Plan, Software Quality Assurance Plan.

49. Identify and document potential contract administration problems due to terms and conditions of the contract.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

50. Insure that formal agreements (such as Memorandum of Agreement) exist between SPO and CAS identifying their specific duties and responsibilities concerning software.

51. Develop an internal CAS plan for assignment of manpower, test and review participation, inspection points, and communication responsibilities.

52. Identify and develop plan for desired contract administration activities for subcontracted software.

53. Optional SPO and CAS Post-award Orientation Conference of software concerns and responsibilities.

54. Ongoing evaluation and review of contractor's management system for software development.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

55. Monitor critical subcontracts for flow-down of software contract requirements.

56. Monitor cost, schedule, and performance of software development tasks.

57. Resolve technical and management problems in software development process.

58. Review and evaluate software Engineering Change Proposals.

59. Review contract modifications for any possible effects on monitoring software development.

60. Monitor development and deliveries of software Contract Data Requirements List documents.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

61. Monitor and review compliance with automated test system standards and documentation.

62. Audit contractor compliance with software requirements for configuration management and quality assurance.

63. Review and report to SPO on contractor's technical planning and management of software development prior to each formal review.

64. Follow up all action items designated by SPO as a result of each formal review.

65. Evaluate the adequacy of the contractor's system requirements definition through the formal System Requirements Review.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

66. Evaluate the completeness of the allocated technical requirements and the technical and management risks of the software development through the formal Systems Design Review.

67. Assess the contractor's system development tools and management and technical abilities to carry out the necessary development functions.

68. Review the Computer Software Configuration Item's functional, performance, data base, qualification, interface, and delivery requirements through the formal Software Specification Review.

69. Assess proceeding with contractor's designs for each Computer Software Configuration Item through the formal Preliminary Design Review.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

70. Determine preparedness to proceed to the coding phase through evaluation of the contractor's Software Detailed Design through the formal Critical Design Review.

71. Monitor software coding phase.

72. Monitor software integration and test.

73. Evaluate test procedures, tools, facilities, personnel, configuration control procedures, and other necessary factors to determine preparedness for qualification testing through the formal Test Readiness Review.

74. Evaluate the performance of the software, the adherence to approved test procedures, and the accuracy of test results through Formal Testing.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION OF SOFTWARE

75. Evaluate the validity, compliance, and accuracy of test results of the software integrated into the system through formal System Integration and Test.

76. Document the completeness of meeting requirements and specifications, and testing of each Computer Software Configuration Item and Software Product through formal Functional and Physical Configuration Audits.

77. Monitor planning and execution of Independent Verification and Validation.

78. Monitor development of automated tools used to prepare technical manuals, operating procedures, engineering drawings, etc., to assure compatibility with end-user systems.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

79. Monitor assurance that configuration management and historical data systems are maintained.

80. Determine compatibility of maintenance procedures to human factors and equipment.

81. Evaluate training requirements for operation and support of the software system.

82. Evaluate software system documentation to assure support of post-acceptance maintenance.

83. Evaluate logistical support to system users for post-acceptance maintenance.

84. Complete software acceptance procedures.

TABLE THREE CONTINUED

TECHNICAL TASKS IN THE ACQUISITION

OF SOFTWARE

---

85. Review the Configuration Management Plan for post-development software support.

86. Review the Software Quality Assurance Plan for post-development software support.

87. Review Software Test Plans for post-development software support.

88. Determine that proposed engineering changes, deficiencies, and latest defects in software are corrected by the contractor and the Configuration Management Plan adjusted accordingly.

89. Establish a Configuration Management Plan for post-development software support.

The tasks in this table are drawn from sources which predate MIL-STD-2167. However, they are generic enough that it is unlikely that 2167 would have much effect on them. Also, the policy which these tasks reflect will probably not be reconsidered in the light of 2167 for some time, since 2167 is a contractor standard.

6.0 RELATION OF TASKS TO SPO AND CAS

The second major requirement of this phase of the study was to identify those technical tasks more appropriately performed by SPO personnel and those more appropriately performed by the in-plant CAS personnel. This corresponds to task 4.3.2 in the SOW for the study.

It creates problems to devise exclusive divisions of tasks, such as those in Table Three, between agencies such as SPO and CAS, i.e., the acquisition and development of software is complex and nonlinear. It is not reasonable to expect that a totally simple, clear-cut division of tasks would be feasible. The notion of concise separation is certainly appealing administratively. But in terms of effectively acquiring the development of complex software systems, this would amount to a useless oversimplification.

In order to move toward a useful separation and coordination of tasks which would result in effective software acquisition, the contractor has derived a simplified schema for the relationships among tasks and the two agencies involved in the acquisition of software. The aim here is to:

(1) produce clarification,

(2) support analysis and evaluation,

(3) establish effective roles,

(4) acknowledge interrelated roles, and

(5) avoid oversimplification.

For this purpose, a very simple schema was established for the assignment of task relationships to SPO and CAS.

The schema is shown in Table Four.

The schema was kept as simple as possible, which seems appropriate for the present study. However, if these recommendations are considered for implementation, it may be that a more articulated schema would be useful.

TABLE FOUR

SCHEMA OF TASK-RELATIONSHIPS

FOR SPO AND CAS

Code Use In Table III	Meaning
I	Primary Responsibility
II	Must be Consulted
III	May be Consulted
---	No significant responsibility (Does not imply exclusion or that responsibility may not be enlarged by agreement.)

Table Five shows the tasks previously identified for software acquisition displayed against task-relationship codes for SPO and CAS. In Table Five, only a key word or phrase is given for each task. However, the number of each task in Table Five corresponds to the number of the full task description in Table Five, for reference.

There is a great deal of information contained in Table Five, and of course, the distinctions it makes should be considered in detail by anyone undertaking to implement its recommendations as policy. However, without addressing each item individually, the general implications of Table Five can be usefully summarized.

Table Five reflects an increased amount of required consultation of the CAS by the SPO than is currently reflected in written policy and/or practice. This is especially true in the areas of pre-award survey and RFP development. Interviews conducted in the study indicate that under current practice, valuable knowledge developed by CAS agencies through their long-term relationships with day-to-day contractor activities is under utilized and wasted in these areas. If followed, the recommendations of Table Five should alleviate this situation, which should not be allowed to continued in the critical, costly, and problem-ridden area of software acquisition.

TABLE FIVE

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
1. RFP Package Checklist	I	---
2. Request cost assumptions	I	---
3. Previously developed software reuse	I	---
4. Internal SOW refinement	I	---
5. Alternate bidder input	I	---

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
6. Assess contract separation	I	III
7. Understand data items	I	---
8. Customize data items	I	III
9. Conduct pre-award survey	I	II
10. Pre-award survey to bidders	I	---
11. Prepare pre-award questions	I	III

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
12. On-site interviews	I	II
13. Assemble software expert team	I	II
14. Submit assessment to CAS	I	---
15. Finalize pre-award report	---	I
16. CAS pre-award survey	I	II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
17. Evaluate contractor profit-volume/risk	I	---
18. Input from AFCMD on pre-award	II	I
19. Establish SPO/AFCMD communication	I	I
20. Subcontractor letter requirements	I	II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
21. AFCMD internal conference	---	I
22. Contract requirements for measurement	III	I
23. SPO/CAS relationships	I	I
24. Review draft RFP	III	I,II
25. Review draft SOW	III	I,II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
26. Review CDRLs	III	I,II
27. Review draft RFP scheduling	III	I,II
28. Review standards, specifications, tailoring	III	I,II
29. Review sample contract validity	III	I,II
30. Source selection plan	I	III

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
31. Source selection plan approval	I	---
32. Establish evaluation criteria	I	III
33. Establish factors and standards	I	III
34. Independent cost estimate	I	II
35. Evaluate proposals	I	II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
36. Determine competitive range	I	III
37. Identify deficiencies	I	II
38. Confer with bidders	I	III
39. Final contracts	I	---
40. Negotiate prices	I	III

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
41. Complete evaluation	I	III
42. Source Selection Evaluation Board report	I	---
43. Source Selection Advisory Council report	I	---
44. Final Decision	I	---
45. Document Decision	I	---

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
46. CAS proposal evaluation	---	I,II
47. CAS in Source Selection Panels	I	III
48. Review awarded contract	---	I
49. Identify potential contract problems	---	I
50. Insure specificity of SPO/CAS agreements	I	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
51. Internal CAS plan	III	I
52. CAS subcontract plan	III	I
53. Post-award orientation conference	II	I
54. Ongoing management evaluation	II	I
55. Subcontract flowdown	II	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
56. Monitor software development measures	II	I
57. Technical and management problem resolution	I,II	I,II
58. Evaluate engineering change proposals	I,II	I,II
59. Review contract modifications	III	I
60. Monitor CDRL delivery	III	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
61. ATS standards compliance	III	I
62. CM and QA compliance audits	III	I
63. Pre-review reports	II	I
64. Follow-up review action items	II	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
65. Systems Requirements Review	I	III
66. System Design Review	I	II
67. Assess development capabilities	II	I
68. Software Specification Review	I	II
69. Preliminary Design Review	I	II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
70. Critical Design Review	I	II
71. Coding monitor	II	I
72. Integration and test monitor	II	I
73. Test Readiness Review	I	II
74. Formal Testing	I	II

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
75. System Integration Test	I	II
76. Monitor Configuration Audits	II	I
77. Monitor IV & V	II	I
78. Monitor tools compatibility	II	I
79. Assure CM maintenance	II	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
80. Evaluate Human and equipment compatibility	II	I
81. Evaluate Training Requirements	II	I
82. Evaluate documentation	II	I
83. Evaluate logistical support	II	I
84. Software acceptance	II	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
85. Review CM for post-development	II	I
86. Review SQA plan for post-development	II	I
87. Review Test Plans post-development	II	I
88. Determine corrections and updates	II	I

TABLE FIVE CONTINUED

TASK-RELATIONSHIPS CODED FOR

SPO AND CAS

Task	SPO	CAS
89. Review CM plan for post-development	II	I

Another general effect of Table Five is to recommend more compulsory participation of CAS personnel in formal reviews during the development process. As specified here, required CAS participation would begin with the System Design Review (Table Five, number 66). Such early involvement is necessary in order for CAS personnel to do their jobs effectively through the later stages of the software development process. Also, CAS familiarity with day-to-day problems of software development offers the potential to predict and prevent potential problems earlier in the development process. This offers the possibility of great dollar and time/schedule savings to the government.

A final phrase in the SOW for this component of the study is "for each task assignment, give rationale and identify benefits accrued to the government." To do this for each task assignment would be extremely redundant, as the rationale for each assignment (in that it differs from current policy/practice) is pretty much the same and is outlined in the immediately preceding paragraphs along with potential benefits. In short, current policy and practice fails to utilize available CAS potential to produce better software contracts and fails to involve CAS personnel early enough to support their potential contributions to early problem prevention and correction. Since software development is so costly in dollars and schedule and so burdened with errors and setbacks, fuller use should be made of these obvious potentials.

Probably the most specific and clear-cut policy statement regarding the task-relationships in Table Five appears in AFSCR 800-42:

"3.C. System Program Office will -- ...

(8) Involve AFCMD (before initial contract award) or the AFPRO in the development and review of the following program documents, and as a participant in the following precontract activities. Except for sole source acquisitions not administered by AFCMD, AFCMD should be involved to help make sure the contract can be properly administered, even if the resulting contracts may be administered by another CAO agency.

(a) Program management plans (PMP).

(b) Acquisition plans (AP).

(c) Statements of work (SOW).

(d) Requests for proposals (RFP), including draft RFPs.

(e) Drafts of proposed contracts.

(f) Subcontract management plans (SMP).

(g) Proposal evaluations.

(h) Source selections.

(i) Contract negotiations.

(j) Business strategy panels (BSP).

(k) Solicitation review panels (SRP).

(l) Manufacturing management and production capability reviews (MM and PCR).

(m) Production readiness reviews (PRR).

(n) Quality or product assurance assessments and reviews.

(o) Any other areas in which AFCMD support would be available."

The above, along with other areas in AFSCR 800-42, are essentially spelled out in detail specific for software in Table Five. Although AFSCR 800-42 would appear to imply the increased CAS involvements recommended above, interviews with SPO, AFPRO, and AFCMD personnel indicate that in the software area, these involvements are happening infrequently or not at all. Assumedly, it is the prerogative of AFSC to enforce these involvements.

7.0 CAS PERSONNEL SKILLS AND STAFFING

The third major requirement of this phase of the study was to utilize the technical tasks to be performed by the CAS to identify personnel skills required within the CAS organizational functions to carry out those tasks. This corresponds to Task 4.3.3 of the SOW for the study.

As a preliminary to identifying personnel skills to carry out tasks which the prior analyses show to be prominent for CAS, an analysis of CAS responsibilities with regard to functional areas (Engineering, Quality Assurance, Manufacturing, Contracting, Program Management Support, Property Management, and Subcontracting) was carried out. Criterion for the inclusion of a task in this analysis was that it have either a I or II rating in the analysis shown in Table Five. The idea for this criterion was to provide a first approximation of those tasks where CAS personnel have more critical responsibility. A table showing the plot of selected tasks against responsibilities assigned each functional area is included in Appendix C. Visual inspection of this table led to the grouping of tasks into two areas:

(1) Those tasks where high level of responsibility was spread over <u>several</u> functional areas.

(2) Those tasks where high levels of responsibility were assigned to <u>few</u> functional areas. In the

case of all but one such task, the only two functional areas with high levels of responsibility were Engineering and Quality Assurance.

The following tasks were identified as having responsibility spread over several functional areas:

Task Number (refer to Appendix C)

15

18

19

21

23

48

49

50

52

53

All of the remaining tasks are evaluated as having higher levels of responsibility limited to few functional areas.

Since the tasks representing higher responsibilities limited to few functional areas included (virtually) only Engineering and Quality Assurance, the skills required to perform the tasks must reflect (1) a general area, (2) an Engineering area, and (3) a Quality Assurance area. Tasks which had high responsibility rankings only for Engineering and Quality Assurance are shown separately for these two functional areas in tables in Appendix C. These tables match tasks for Engineering and Quality Assurance against suggested training areas. These training areas are discussed below.

For the purposes of the present analysis, the tasks shown in Table Five as requiring major CAS input only suggest a partitioning of CAS functional units into the three following categories:

(1) Engineering

(2) Quality Assurance

(3) All other functional units.

The implication of this breakdown is that Engineering and Quality Assurance appear to have the most general, specific, long duration, and continuous contact with the development of software. Other units tend to have more general, briefer, or more technically limited contact with software development. It is admitted at the outset that any statements of this nature are clearly generalizations, and there are likely to be important exceptions. The exceptions should be responded to in accordance with their specific training needs, whatever those should be.

The information in Table Five and the three categories just outlined above lead to describing three areas of training for CAS personnel (the term "training" is used for convenience here to refer to any combination of formal or informal training and/or experience which results in the knowledge/skill areas outlined):

(1) Training of a basic and general nature which should be obtained by <u>all</u> affected CAS personnel (includes Engineering and Quality Assurance). This area is hereafter referred to as -- <u>Software Development Management Evaluation</u>.

(2) Specific training areas relevant to affected Engineering personnel. This area is hereafter referred to as -- <u>Software Development Technical Engineering Evaluation</u>.

(3) Specific training areas relevant to affected Quality Assurance personnel. This area is hereafter referred to as -- <u>Software Quality Assurance Evaluation</u>.

The <u>Software Development Management Evaluation Area</u> is intended to give a general and basic background in software acquisition for all affected CAS personnel. This broad area is seen as probably sufficient coverage of the area for all personnel except for Engineering and Quality Assurance. In the case of Engineering and Quality Assurance personnel, <u>Software Development Management Evaluation</u> is seen as introductory and basic to the other areas more specific to

Engineering and Quality Assurance. One advantage of identifying an area such as this is that it establishes a "core" and a common basis of knowledge, understanding, and communication for all CAS personnel involved with software acquisition.

The more specific training areas under the label of Software Development Management Evaluation are:

- Acquisition and Development Cycles for Software
- Cost Estimation Methods and Problems in Software Development
- Management Principles in Software Development
- Quality Factors throughout the Software Life Cycle
- Cost Factors throughout the Software Life Cycle
- General Requirements and Intentions of MIL-STD-2167.

Engineering Training. Using the background described above as a prerequisite foundation, specific areas of training for Engineering are grouped under the heading, Software Development Technical Engineering Evaluation:

- Technical Cost Aspects of Software Planning
- Technical and Quality Factors in Requirements Analysis
- Software Specification Techniques
- Engineering Management in the Software Life Cycle
- Software System Development Techniques
- Software Test Development Techniques

- Software Program Design and Coding
- All of the above are assumed to involve MIL-STD-2167 and current or future regulations and additional standards as appropriate.

Quality Assurance Training. At the outset, it must be commented upon that Quality Assurance has a system (called "KSAs" for Knowledges, Skills, and Abilities) which arises from and/or is linked to job-task analysis. The KSA system is very advantageous for analyzing and planning training needs. The analysis presented here is based on broad tasks, whereas the KSAs are based on a much more detailed job-level task orientation. The data presented here may possibly shed light on KSAs, but should in no way be seen as an approach to critique or change the KSA system.

The following areas of software training are assumed to rest on a foundation of the knowledge in the Software Development Management Evaluation, or core area, and are here labelled Software Quality Assurance Evaluation:

- Software QA plans and Software QA plan development
- Software QA in test plan auditing and test execution auditing throughout the Software Life Cycle
- Software configuration management auditing throughout the Software Life Cycle
- Software Quality Assurance standards for software development throughout the Life Cycle

- All of the above are assumed to involve MIL-STD-2167 and current or future relevant regulations and standards.

<u>Further Recommendations for Training Development</u>. The areas outlined here arise clearly from the task analyses which comprise this study. A next step in the production of high-utility training would be to utilize the areas identified here as guidelines to conduct more detailed job-task analyses and arrive at highly specific training objectives such as those currently found in the Quality Assurance KSAs.

It is recommended that all CAS training implied here be refined to the level of specificity exemplified by the KSAs. Such systems of specific training objectives offer a number of benefits, such as:

(1) A basis to evaluate the comprehensiveness of training considered

(2) A system for continuous evaluation of specific training needs of individuals

(3) A system for determining needs of groups

(4) A system for establishing priorities for training when distributing limited resources

(5) A system for tracking specifically where training funds, resources, and time have been applied

(6) A system to clarify training needs requested from training contractors.

The net result of these advantages should be more effective training and more cost-effective training. As an example, the Quality Assurance KSAs are included in an Appendix to this report in their entirety.

The foregoing discussion is oriented exclusively to recommendations for the type and content of training required by the various CAS organizations. Another important aspect of the topic of CAS software skills is the availability of manpower resources.

Interviews with CAS personnel at AFPRO sites and at HQ/AFCMD consistently reflected an overall shortage of manpower in addition to a shortage of personnel adequately skilled in software. The prevailing point of view might be characterized by the rationale that since you have too few personnel to adequately do the parts (nonsoftware) of the job you're best prepared for, it doesn't make much sense to push hard into software, where you know you have an extreme shortage of skills.

The Electronics Institute of America (E.I.A.) estimates the total USAF personnel need in 1986 at 8,800 people for software support (which includes CAS personnel). The E.I.A. estimates that this requirement will increase at an average rate of 14% per year for the next three years. Given the widely acknowledge observation that CAS software is currently undermanned, the emphasis required to overcome the current deficit <u>and also</u> keep up with an average 14% per year growth rate (assuming proportionality of CAS to other software support personnel) would be very weighty. In fact, given the current pressures to cut back government expenditures due to Gramm-Rudman, it seems very unlikely that the necessary growth rate could be expected even if there were not the initial deficit to surmount.

Although cut-backs due to Gramm-Rudman have not yet seriously eroded the number of current CAS software personnel, the preceding discussion makes it clear that for this area, simply holding the status quo actually means losing ground at a rapid rate. Thus, if the future effects of Gramm-Rudman on other policies result in zero or slow growth of CAS software personnel, the current manpower shortage will quickly worsen.

Another critical factor for CAS personnel skills is verticality of training in software. Availability of more and better trained personnel at the journeyman level will not effectively impact USAF's needs for CAS support in software acquisition unless there is a pervasive elevation of the organizational visibility and importance of software. This can come about only through training of <u>all</u> levels of management, from the very top through supervision. The concern and attention that management gives to the various functional areas of an organization affects the degree to which people involve themselves in that work, and how effectively they do that work, independent of their training. Management at all levels can give the appropriate type and degree of emphasis to software only with sufficient training.

In closing, some comments should be made regarding the current status of software training in Engineering and Quality Assurance. As stated above, Quality Assurance has a powerful base for their journeyman training program in their system of KSAs. This is especially important since hiring in this area is based on the assumption that incoming personnel with general or limited expertise will be hired and then continuously assessed and trained to expand and deepen their capabilities. Currently, software is a top priority for journeyman training. The big catch is funding. In prior years, this training program was well-funded and delivered many hours of specifically targeted training for its journeymen. However, for the last three years, funds allocated for this training have been cut so low that they represent a virtual zero. So here is a program geared by need, premise, and capability that is severely impaired by recent lack of funding. (Quality Assurance also operates three-year training intern programs which are not considered in this discussion which is targeted to journeyman capabilities.)

Lack of funds is also a currently critical factor for training in Engineering. In Engineering, the hiring premise differs from Quality Assurance in that the incoming employee is a professional engineer by virtue of training and experience. However, CAS engineers, unlike civilian engineers, are expected to become generalists and grow to operate effectively in more areas than the specialty of their training or prior experience. MCCR personnel at HQ/AFCMD are well aware that training engineers from other areas in software is a necessary source for developing manpower capability in this area. Although their training program is not developed to the level of specificity and delivery capability of the current Quality Assurance program, Engineering aggressively studies their needs for training. Currently, lack of funds have prevented them from mounting a systematic program. Currently, most of their training is provided by HQ staff or administered by them via circulation of training videotapes to the AFPROs.

8.0 <u>RECOMMENDATIONS REGARDING POLICY</u>

The fourth requirement for this phase of the study was to recommend policy changes that would provide clear and concise functional tasking for MCCR acquisition. This corresponds to Task 4.3.4 of the SOW for the study.

The problems and limitations involved in attempting to produce completely clear and concise separation of functional tasks for SPO and CAS have been discussed in Chapter 6.0 of this report. To reiterate, as appealing as complete separation of function may appear from a purely administrative viewpoint, oversimplification threatens the functional effectiveness of the complex process of software acquisition.

Thus, the task relations specified in Table Five are designated as recommendations for policy, <u>and</u> it is recommended that policy should be altered to <u>enforce</u> the relationships of Table Five. This contractor is aware that some of the recommendations following Table Five and many of the relationships shown in Table Five already exist in written policy as "suggested," "where appropriate" (or similar phrases) or are even <u>required</u> (e.g., AFSCR-800-42). However, interviews conducted in this study show that in spite of guidance or requirements of written policy, in practice, there is insufficient involvement of CAS capabilities in the development of RFPs and contracts and too little early involvement of CAS personnel in formal reviews and audits to fully utilize and support their potential.

Therefore, policy should be rewritten to <u>require and enforce</u> CAS involvement as shown in Table Five and discussed in Chapter 6.0 of this report.

The SOW for this study requests changes be recommended "to DOD policy, including the Federal Acquisition Regulation (FAR), the DOD FAR Supplement (Sup), the Air Force FAR Sup, and the AFSC FAR Sup...." Recommendations are made as indicated above. However, they are tentative. The contractor would deem it precipitous to institute such sweeping changes in policy without further study.

Thus, it is further recommended that policy requirements and enforcements such as those recommended above be tested on several projects, and the results evaluated prior to instituting more widespread or higher level changes in policy.

9.0 <u>ANCILLARY RECOMMENDATIONS</u>

As in any study, there was, of course, more information observed than is strictly relevant to the focal issues. Some additional recommendations (or perhaps substitute "suggestions" for "recommendations," if preferable) are included here.

There are many interesting findings, but the most salient for additional recommendations are informally:

(a) There are some significant overlaps of function in SPO and AFCMD activities for the acquisition of MCCR software. However, these appear to be well-noted by the individuals involved in actual operations.

(b) The primary formal instrument for coordination of activities and responsibilities between the SPO and AFCMD for a given contractor operation is usually referred to as the MOA (memorandum of agreement). This instrument is stipulated by policy, but it is admirably flexible. <u>In most cases, however, it is reported as minimally functional.</u>

(c) Generally, SPO personnel involved in software activities are spread very thin, feel very overloaded and believe that AFCMD personnel contribute little or nothing to MCCR software acquisition.

(d) Generally, AFCMD personnel believe their capabilities and contributions to MCCR software acquisition are ignored/under-utilized by SPOs.

(e) There are a few, but notable exceptions to (c) and (d) above, which point the way to positive recommendations.

(f) There is a lack of sufficiently trained and experienced personnel in both SPOs and AFCMD to handle the extremely rapidly growing requirements of the MCCR software acquisition process.

Ancillary recommendations emerging from the findings of this study offer the possibility (but not the guarantee) of more effective operations by SPOs and AFCMD in MCCR software acquisition. Such increased effectiveness could lead to better cost and schedule control and the increased assurance of receiving quality-controlled software products.

Unfortunately, the ancillary recommendations of this study would require considerable commitment and yet address only a segment of the larger and complex overall problem of MCCR software acquisition.

The problem of time pressure discussed in Section 5.1 pervades every aspect of MCCR software acquisition and also affects the other two root problems of insufficient trained/experienced personnel and under-utilization of available capability in AFCMD. Specifically, when a work force is under heavy time pressure, it is unlikely to find adequate time to identify training needs, develop training plans, and find available time to release personnel for training. Further, when an organization such as an SPO is under heavy time pressure, it is not likely to find time to develop or encourage uncontrolled critical input from an outside organization such as AFCMD. Some of the likely fundamental causes of time pressure, such as the difficulties of accurately estimating the time/cost parameters of software production scheduling and the difficulties of planning and executing strategic weapons development in a rapidly changing technology, are well beyond the scope of this study.

However, there is a reciprocal relationship between the time-pressure problem source and the other two fundamental problems. That is, if means could be identified to enhance the utilization of AFCMD software capability and also to increase the skill levels of personnel involved in MCCR software acquisition with minimal time requirements, these two problem areas would be improved and would have the potential to reduce at least some time pressure. Thus, the following recommendations:

1. Service-Marketing Approach for AFCMD. In the majority of cases where SPOs seem to make little use of AFCMD capabilities, there is also a benign ignorance typified by statements such as, "I don't know what they really do," or,

"I suppose they could help us, but they report to ...." In the case of AFCMD personnel, the typical attitude is, "We could offer the SPOs a lot of information that would help them and maybe lead to better software, but they're not interested."

This is particularly noteworthy in that the policy for both organizations specifically acknowledges benefits of AFCMD expertise for SPO operations, and MOAs are prepared to specify and assure the ways the organizations will recognize and support each other in individual projects.

In the apparent minority of cases where SPOs and AFCMD organizations have productive, effective relationships, the situation is characterized by the following key elements:

(a) A relationship was established by initiative from AFCMD.

(b) The major thrust of this initiative came from a high-ranking (e.g., EP division chief) in an AFCMD in-plant organization.

(c) The effort was persuasive in nature, based on how AFCMD can benefit the SPO and how as a team AFCMD and SPO can influence the contractor to produce better software.

(d) The effort was continued over a period of time (i.e., <u>not</u> a one or two presentation information offering, for example).

Observation of these factors led to the following recommendation of what is being termed here a "service-marketing approach" for AFCMD in MCCR software acquisition. The main objective of this recommended approach is to increase SPO utilization of AFCMD capabilities. The <u>basic features</u> of this <u>recommendation</u> are as follows:

(a) Specify exactly what services AFCMD wishes to market (in general) to SPOs.

(b) Determine specifically what actions by an SPO do/do not constitute acceptance of each possible component of the service product.

(c) Determine specifically what benefits each service product offers SPO and under what conditions.

(d) Determine specifically <u>**who**</u> in SPO or related organizations makes the decision, and when, to accept AFCMD service products or not.

(e) Set specific goals for level of acceptance of service products.

(f) Set specific goals for activities to obtain acceptance of service products.

(g) Evaluate and adjust (e) and (f) on experience.

(h) The activities of (f) constitute persuasive, repeated, timely demonstrations to decision makers which:

(1) clearly and specifically show benefits to SPOs and USAF.

(2) include a mechanism for commitment.

(i) These activities need to be continuous and long term in duration.

(j) A component of public relations should also be developed, along with seeking support from a wide and vertical constituency.

2. Streamlined training. In an age of rapid technological turnover (especially true in the software area) and with the personnel turnover and reassignment that is prevalent in government organizations, streamlining of training is very important. The government has courses available in software acquisition management. Although these could no doubt be improved, they appear to offer an adequate initial basis of training in the area. However, an observation of this study is that relatively few have been able to take these courses. Currently these courses are taught primarily at training centers. Perhaps less lengthy, videotaped, or otherwise mediated courses could be developed which could be disseminated more widely and used more time effectively.

As a follow-up to the basic or foundational training, it is recommended that all more advanced training be comprised in a computerized dynamic data base. This data base could utilize an artificial intelligence language such as PROLOG. Persons needing training beyond the basics of software acquisition management could simply query this data base. The data base could be continuously updated by experts from within AFCMD and external to AFCMD. This recommendation reduces the training requirements to two elements:

(1) Basic software acquisition management.

(2) <u>How</u> to utilize the training data base for advanced training and updating.

The contractor is aware that other proposals and efforts have been made in AFCMD that have been associated with the idea of "marketing." It should be noted that the approach suggested here is very specific and goal oriented and includes, but is not centered around, an image-promotion approach. Image promotion is useful and can be effective up to a point. However, it is difficult or impossible to gauge the effectiveness of such a program and thus to effectively modify it or make other decisions. The program outlined here supports direct evaluation and decision making.

Assumedly, leadership and direction for a service-marketing effort would come from Headquarters, AFCMD. The main thrust for effectiveness, though, has to come from the in-plant AFCMD organizations. In the final analysis it is the perceived capability of these organizations that SPOs "buy" or not.

A major question for the streamlined training recommendation is, "Who would be in charge of the data base/experts?" This is a political question beyond the scope of this study. However, at a minimum, the following need to be included. First, in spite of a shortage of adequately trained/experienced personnel in the field, there **are** some very competent, dedicated people in the AFCMD in-plant organizations. Their capabilities should be very valuable here. Second, some specific mechanism should be established to assure that the content and function of the data base continuously serves the intended end users well.

REFERENCES

Airborne Systems Software Acquisition Engineering Guidebook for Contracting for Software Acquisition, ASD/ENAI, Wright-Patterson AFB, OH, August 1980.

Booch, Grady, Software Engineering with Ada, Benjamin/Cummings Publishing Co., Inc., Menlo Park, 1983.

Contract Administration of Mission Critical Computer Resources in Systems Acquisition, Joint Contract Administration Coordinating Council Guidebook, October 1984 DRAFT.

Contract Management Engineering, AFCMD Regulation 800-1, May 1983.

Contract Management Engineering Guide, AFCMD Pamphlet 800-2, May 1983.

Contracting for Software Acquisition, Software Acquisition Guidebook Series, ASD/ENE, Wright-Patterson AFB, OH, January 1979.

Contractor Documentation Guide for Software Development Survey, Space Division, August 1982.

Developing Space Flight Software, Management Guide, Space Division, SD/ALR, April 1985.

Documentation Requirements, ASD Software Acquisition Engineering Guidebook, ASD-TR-79-5025, November 1978.

Evans, M., Piazza, P., and Dolkas, J., Principles of Productive Software Management, John Wiley & Sons, New York, 1982.

Evans, Michael W., Productive Software Test Management, John Wiley & Sons, New York, 1984.

Fairley, Richard E., Software Engineering Concepts, McGraw-Hill, New York, 1985.

Fox, Joseph M., Software and Its Development, Prentice-Hall, Englewood Cliffs, NJ, 1982.

Glaseman, Steve, Comparative Studies in Software Acquisition: Management Organization Versus the Development Process, Lexington Books, 1982.

Guide for Computer Resources Contract Deliverables, Space Division, SD/ALR, April 1985.

Guide to the Management of Embedded Computer Resources, Space Division, January 1984.

Hoare, C.A.R., "The Engineering of Software: A Startling Contradiction," Computer Bulletin, December, 1975, pp. 34-42.

Martin, James, System Design from Provably Correct Constructs, Prentice-Hall, Englewood Cliffs, NJ, 1985.

Martin, James and McClure, Carma, Structured Techniques for Computing, Prentice-Hall, Englewood Cliffs, NJ, 1985.

McClure, C., Managing Software Development and Maintenance, Van Nostrand Reinhold Co., New York, 1981.

Mission Critical Computer Resources Contract Management, AFCMD Regulation 800-3, May 1984.

Program Office/AFCMD Interface, AFSCR 800-42, November 1982.

Pressman, Roger S., Software Engineering, McGraw-Hill, New York, 1982.

Schaar, Brian, Paper representing the Ada Joint Program Office (AJPO) of the Department of Defense, Presented at "Ada on the Move" conference, University of New Mexico, April 1984.

Software Development Planning and Control, ASD Software Acquisition Engineering Guidebook, ASD-TR-80-5022, February 1980.

Statement of Work Preparation Guide, Acquisition Management, Space Division, Pamphlet 800-6, April 1980.

Statements of Work and Requests for Proposal, ASD Software Acquisition Engineering Guidebook, ASD-TR-79-5026, September 1978.

Wiener, Richard and Sincovec, Richard, Software Engineering with Modula-2 and Ada, John Wiley & Sons, New York, 1984.

APPENDIX A

A GUIDE FOR MANAGING

NONDELIVERABLE

COMPUTER RESOURCES

January 1984

Prepared by

Major George W. Trever

Air Force Contract Management Division

Air Force Systems Command

Kirtland AFB NM 87117

Foreward

While much attention has been placed on the life cycle of embedded computer systems, somewhat less attention tends to be directed to those computer resources supporting the delivery and maintenance of end products. Many lessons have been learned by painful experiences in acquiring and using computer systems for military weapon systems. Now the computer is finding its role in a wide range of applications, many directly affecting the design, configuration, manufacturing, testing, and inspection of contracted products. Some of these computer applications are acquired from vendors, some adopted from existing or like systems and still others are custom built. Regardless of its source of being, computer resources used in the work place will need some degree of management. Usually, the computer programs become the focus for management, as they are invariably the scape goat of user complaints.

So how should one approach the managment question of in-plant supporting computer resources? How can we avoid management overkill or costly decisions? Simplistically one could say, "Well, use the basic principles applied for deliverable products," or "What is good for the goose is good for the gander." Perhaps you have your own trite statement to insert here. In any event, this guide makes an attempt to bridge the gap between management practices for deliverable computer resources and doing nothing. We have attempted to address practical reasons for adopting certain management approaches. And, yes, even a touch of lessons learned has been thrown in. Management of any kind of computer resource is important, regardless of its application. this perspective of broad, non categorized CR management is presented. We are hopeful that this guide will provide the framework you need and can use.

TABLE OF CONTENTS

1.0 GUIDE OBJECTIVES: This guide was prepared to assist managers structure and understand policies and procedures in the management of nondeliverable computer resources (NDCR). It is the intent of this guide to capture the basic management methods and lessons learned attributed to deliverable embedded computer systems and translate them into management terms appropriate for NDCR.

1.1 Applicability: This guide should applied to any NDCR, i.e., hardware, software, and associated documentation, used by a manager's organization to plan, evaluate or produce a deliverable product. It is intended that the reader compare his/her NDCR environment to this guide and tailor one's own policies and procedures according to the management risks and disciplines involved. The guide user may identify NDCR used in their environment that do not exactly fit any categorization addressed by this guide. This should not be an inhibitor to using the guide. Not every idea or consideration will apply, but concepts addressed an initial step in establishing or refining NDCR management policies and procedures.

1.2 Need Recognition: Do you need this guide? Take a moment to evaluate your environment, i.e., look at how computers and computer programs are being assembled and are used to aid your job. Do you know how any one computer system or computer program affects your operations and to what degree? If you cannot confidently respond to this question, then we suggest you proceed into this guide. NDCRs possess risks to your working environment even if the computer system salesman did not point out such facts. It is very easy to be absorbed into the computer technology boom which makes it just as easy to make unfortunate management errors. This guide attempts to raise many environmental conditions for your consideration and proposes some guidelines to deal with those conditions.

2.0 DEFINITIONS: Technical terms used in this guide are consistent with IEEE Standard 729, "A Glossary of Software Engineering Terminology." Defined terms not contained in that standard are listed below.

2.1 Nondeliverable Computer Resources (NDCR): Resources, i.e., hardware, software and associated documentation, which are purchased, developed or used in the design, development, manufacturing, inspection, or test of a product deliverable to a customer.

3.0 MANAGEMENT OBJECTIVES: Nondeliverable Computer Resources (NDCR) represent a growing investment by management. Management will rely more and more on NDCR to assist every element of the work force to better support the systems acquisition process. However, with the gains in productivity have come new risks that threaten the cost and schedule savings. To safeguard the integrity of NDCRs, a series of management objectives is identified and discussed in the following sections.

One word of caution; tailoring management objectives is extremely important for NDCR. One should avoid creating elaborate methodologies or bureaucracies to manage NDCR. The key word is elaborate. Methodologies or bureaucracies will generally be needed to assure that workable discipline is formulated and used. In some instances, management may desire to distinuish between NDCR systems, e.g., ATE vs. CAD, to define different management methodologies. This maybe a form of typing or categorizing NDCR in a way to avoid the use of stringent controls or very formal procedures where simple, informal methods would suffice.

3.1 Planning: Project planning should include detailed consideration for the development, purchasing, control, evaluation, validation and use of nondeliverable computer resource products.

3.1.1 General: Where NDCR will be developed/acquired by a system developer, specific identification of schedules, labor and computer time, and activity interfaces for each NDCR system should be documented. Project and customer management need to be aware of potential risks and benefits to the project by engaging in parallel computer resource developments/acquisitions.

3.1.2 Interfaces: It is not uncommon for NDCR to depend on events outside of NDCR management control. NDCR usually will support a major project and, thereby, will suffer from changes dictated by the project. The NDCR manager can expect some suffering, but only to the degree that internal interfaces have not been established and effectively cultivated. The internal project change control and appropriate design review forums should be examined closely for potential direct interfaces. The decisions arising from these forums, in some cases, are still too late for the NDCR manager to adequately support project management, but it is a lot closer than waiting for formally published minutes, design change orders or other communiques.

3.1.3 Skills: Applying computer technology to a manual-labor task may be traumatic. It is not uncommon for management to take a task expert, e.g., an RF-engineer, and train that person how to program a computer, e.g., using BASIC. This may seem quite innocent and even logical, but the NDCR manager should beware of a major pitfall. If software or complex data-bus interfaces will be developed, the NDCR manager should seek computer system technical skills to supplement a staff of traditional electronics, manufacturing, structure, etc., skills. It is rare to have multi-qualified individuals who know the total computer system technically and have the discipline to get the job done. Discipline is the key. The methodology followed by a software designer must be highly disciplined to assure that the end product can be maintained and documented. It is not uncommon for a task expert (i.e., an electronic engineer, manufacturing specialist, etc.) not to receive the training on how to do efficient and effective software design and documentation. Hence, a "team" approach involving several disciplines can be most beneficial to the NDCR manager.

3.1.4 Schedules: Developing a computer system (or NDCR) to support some phase of a deliverable product development or acceptance can be very sporting. Management attempts to define a realistic schedule for the NDCR can be easy. Using and maintaining the schedule can be a real challenge. As discussed before, the NDCR manager will find that various interfaces with the deliverable product management will be essential. Relating as many key deliverble-product related milestones to the NDCR being developed is critical. Not only are interfaces better defined, but the NDCR manager can better assess the success of the NDCR development when tied directly to the deliverable product and its problems.

As with large computer systems, software development, if any, will typically pace the NDCR development. The "team" of skills doing the NDCR software development and the "firmness" of NDCR system requirements, will probably drive the NDCR software development. However, take note of other factors as security, safety, and state-of-the-art controlled equipment. In some situations, these factors may be very significant. The NDCR manager should not underestimate the risks and schedule realities associated with these factors. Upper management commonly fails to understand the complexity of automating any process. Automation may be seen only from a hardware or electronics point of view, i.e., simple electronic or electrical connections, straight forward electronics maintenance, etc. This perception must be altered to provide a visible picture of the software impact. In many cases, the schedule becomes an important vehicle to demonstrate the critical hardware and software factors. Failure to realistically account for these factors in the earliest schedule will haunt the project manager later on.

3.1.5 Return on Investment: To automate for the sake of automation is not wise. Although personal computers have made automation more feasible and less costly, management still needs to consider the "real" benefits to acquiring computer resources. Why? Because many factors come into play in supporting and using the NDCR. Computer system salespersons are very quick to show how inexpensive a new system may be. Unfortunatly, they may easily gloss over many hidden costs associated with expanding, interfacing, updating and maintaining the computer system software, hardware and documentation. In other words, look at any computer system from a life-cycle point of view.

3.1.6 System Support: Care must be taken by the NDCR manager to acquire the needed documentation or vendor support agreements for the computer system used. Again, long range planning should address the simple aspect of the computer operating system maintenance or equipment obsolescence. Knowing how long a vendor will truely support your computer system may determine your strategy to acquire specialized in-house support capability.

3.2 <u>Corrective Action</u>: Corrective action procedures should be established to apply to nondeliverable software to assure: proper identification/documentation of software problems; analysis of problem causes; validation of problem solution; authorization to implement changes; validation and/or verification of change implementation correctness and completeness.

3.2.1 Approach: Establishment of corrective action procedures should reflect a structured discipline (i.e., controls, reviews audits) commensurate with the project and product risks of using any NDCR system. The discipline established by the developer/user of NDCR should be a common sense sequence of actions. These actions, as described in the following section, should form a closed-loop process.

3.2.2 Recording Problems: Establishing a written record of software problems. The record should be a simple logging process or a more formal software problem record form. The log or form serves as a reference point for management, designers and programmers alike to understand the attention that a problem is receiving.

3.2.3 Solutions: With any problem, some record of the proposed solution and an analysis of its implementation is needed. This is the heart of the corrective action process. In complex applications, this might be non-trivial; many interfaces might exist and management needs to know the cost/schedule/technical implications BEFORE committing to a change of significance. Part of this step involves preliminary steps to validate a proposed change using working files and computer programs. These results are important to know in sufficient detail before the change is approved for final implementation.

3.2.4 Authorization: Authorizing the implementation of a change is a very important step in any corrective action process. Whether it is a change generated in-house or a new version of vendor software, appropriate management and technical people need to authorize the implementation. Sometimes the analysis of the proposed change will reveal some constraints that management might not wish to enjoin. Delaying a change to a latter time might prove more cost effective or less disruptive to an ongoing process. The decision needs to be made by the right people. It should not be arbitrary.

3.2.5 Verification: Verifying the integrity of the total system after implementing a computer program change is the last logical step. This includes change to the computer programs and documentation. It also provides the point at which the original log or form is revisited for the last time to close out the corrective action process. Depending upon the complexity of the change, additional test-result documentation may be generated and stored with the ongoing software history. Other documentation should also be updated to change history and function descriptions and source listings.

3.3 <u>Documentation</u>: Methods of documenting NDCR to support efficient computer resource modification and problem diagnostic efforts.

3.3.1 General: NDCR documentation is very important but not easy to categorize to everyone's liking. In general, two document types should be considered: (1) a combined systems requirements/design document and (2) a user/operator guide. Regardless of the source of the NDCR documentation (i.e., in-house or vendor supplied) sufficient documentation should be developed/acquired to efficiently operate and maintain the system.

3.3.2 System Descriptions: Adequately detailed documentation is needed to perform efficient maintenance on the system, unless the supplying vendor retains such information. Sufficient algorithm/process descriptions and source code listings are essential for successful and timely problem resolution. It is not uncommon for short cuts to be taken in documenting a system (e.g., hardware, software and interfaces). Typically such documentation is very "top-level." However, the systems analyst, component engineer or system engineer may not appreciate such brevity when a major problem has to be solved. Formal documentation is generally not needed; a set of file folders/notebooks and source code listings may be sufficient. But they do need to be well organized, and contain minimum information [illegible] ~~be~~ accessible. Using a "unit development folder" concept might be the best methodology for the NDCR management to consider.

3.3.3 Unit Development Folders: Many computer software developers use a documentation and development management tool, commonly referred to as a unit development folder (UDF) or a programmer's notebook. This tool is nothing more than an documentation organization scheme designed to be used in parallel to the development of software. The UDF is intended to be used at a low level in the software design, e.g., module level. Hence, a set of UDFs will define a total software design and configuration status. The UDF is typically organized to contain the following:

a. Introduction. The developer can place overall system description information, review and audit sign off sheets, overall software sizing data, (i.e., how much memory will be used in program execution), and development schedules.

b. Requirements. The developer should extract each requirement from system or subsystem specifications that pertain to the particular module. In addition to performance requirements, other important interface (i.e., hardware or software) and design convention information should be located in this section. This additional information serves as a good future reminder to the individual modifying a particular module.

c. Detailed Design. Each module component, or unit, should contain flow diagrams or program design language listings to show the structure details. If a large number of units are encountered, it is suggested to have one preceeding section that shows all units in a single flow diagram or pictorial overview. Along with each design, the code source listings (including all comments) should be included.

d. Test. This section should attempt to document the test criteria and results associated with each software unit or grouping of units. A thorough record of testing at the lowest possible level becomes an asset in future trouble shooting or module redesign.

e. Change Record. A history on the changes made to the module is very important. Change records can be used to recover from an unauthorized change to the module configuration. It can also assist in understanding, "Why did it work before and not now?" type of problem.

3.3.4 Users Guide: If the development of a support computer system is not tasking enough, the NDCR manager generally is asked to also support the systems' use. The use of a computer system will be driven by the man- machine considerations not addressed duri'ıg the design phase. The user will generally need to have a broad pictur- ,f how the system operates and adequate instructions and information to know what is going on and how to recognize incorrect operations. Once recognized as an incorrect operation, the user must then know what to do next. The users guide should lead the user to some logical conclusion if nothing more than calling some phone number to report or discuss the problem.

3.4 <u>System Support</u>: A methodology established to assure minimum interruption of an automated systems usage.

3.4.1 Organic Support: The NDCR managers needs to be concerned about how a particular computer system will be supported and assure that defined support is properly rendered. A link should exist between the supporting and the developing activities; however, don't depend on it. This is the reason for good documentation and good design practices. Having a good staff of experts that can agressively attack hardware, software and/or operator problems is essential. Likewise, having spare equipment and spare or backup experts can be a significant factor depending on the computer system complexity. However, in order to work problems, the NDCR manager must be assured to receive the problems, documented well enough to have a fighting chance to solve the reported problem. Hence, a workable communication system should be established using telephone "hot lines," electronic mail boxes, deficiency or problem report forms, etc. to capture user problems. The users guide and a problem communication system will determine the user acceptance of the automated system and, ultimately, the productivity realized by the organization.

3.4.2 Vendor Support: The NDCR manager should consider assigning one or several employees to act as liaisons with the vendor or vendors that have support agreements with the organization. In this manner, specific attention can be applied to the hardware and software problem resolution, and the vendor performance can be accurately followed. Computer systems are supposed to save time; if a vendor can't support you so you can't realize those time savings, then a change might be warranted.

3.4.3 System Change: A computer system's hardware or software configuration will always change. Change begins before formal usage begins and continues as long as management permits it. Change is generally good. It indicates that an interest in the system exists and new ideas are not being quashed. Change could also be an indicator of how poor the system was designed or that it has outgrown its usefulness. The NDCR manager needs to be very sensitive to the nature and trends to suggested change. A proper perspective on proposed changes can only be achieved if these changes are evaluated with the NDCR and deliverable systems properly taken into account. The totality of changes and support problems may suggest radical change vice a patch-quilt approach. Emotions will easily cloud the real problems and workable solutions. This suggests the need for a separate forum to handle the significant problems separate from the daily support activity.

3.4.4 Responsibility Assignment: The NDCR manager should not overlook explicit designation of responsibilities to respective individuals who directly affect the NDCR. Assignment should specifically address the integrated management of NDCR computer programs, equipment, documentation, using personnel training, supply support ard support services. Never under estimate the potential mismanagement of computer resources; it is too easy to do. Of particular note, the NDCR manager should give special consideration to the individual(s) responsible for the computer programs.

Maintaining a good, firm management handle on the computer programs, and associated documentation, is essential. Computer programs represent a growing investment; a valuable resource that, by the simple absence of visible properties, needs protection throughout the life-cycle. Assignment of responsibility is essential. Assignment of resonsibility to protect the resource is only part of the objective; establishing and maintaining communications with the other responsible resource managers is also important. Particular areas of interest should include: matters of security, configuration control, requirements analysis, fault analysis, and corrective action, customer or user support, special access control, documentation, maintenance and data base design and maintenance.

3.5 Library Control: Procedures to administer computer programs, data bases and associated documentation to assure disciplined use and maintenance of computer software resources.

3.5.1 Procedure Development: Library control procedures are essential to the integrity of computer programs and data bases. Whether a library consists of a few dozen cassette tapes or a building full of magnetic disks, procedures are needed. Making the physical magnetic storage devices, labelling the programs or data bases with control/identification numbers and dates, maintaining physically remote storage of duplicates, running comparisons of differing programs, etc. represent some of the areas where library control procedures play a very important part. Procedures should receive rigorous attention commensurate with the "high value" nature of a library. For instance, computer aided design (CAD), manufacturing (CAM), inspection (CAI) and related data bases should receive optimum treatment. Protection from inadvertent mistakes or intentional alterations is another significant factor in library control procedures. The library discipline is an insurance policy and one that needs adherence.

3.5.2 Library Operations: In preparing library procedures, the NDCR manager should consider addressing several specific tasks or provisions. First, consider the mechanics for the library function. As a service activity, the library must be responsive to the user's needs which includes a measure of discipline. Second, assure that the library data is backed up and that the backed up information is physically stored remote to the library. Fire, sprinkler systems, electromagnetic distrubances, tornados, and floods are all potential catastrophies that can rendor a good library useless. Third, keep documentation hand-in-hand with computer programs. By segregating the two or creating two separate systems will only enhance the potential for miscommunication. Fourth, think security. Unauthorized change to computer systems and its software is not too difficult. Even easier is the ability to copy computer programs which might violate copyright laws or present a threat to trade secrets.

3.6 Subcontracting: Establishment of a methodology to assure that subcontracting practices aequately address NDCR management discipline.

3.6.1 The Subcontracts: Should a manager be concerned with NDCR when subcontracting for a product or service? In today's automation environment, the answer is probably yes! Do you have a warrantee? If so, how long? Or, you may have a support agreement, but should you be paying for it in the first place? NDCR can probaly be found behind most products procured today. The bottom line question might be, "How concerned is the subcontractor about managing NDCR?" In contracting for a product or service, it might pay to study the subcontractor's behavior regarding NDCR. This might be the time to address specific NDCR issues either in the contract or as a business-to-business agreement. When using the contract vehicle, get relevent NDCR information as soon as possible, most preferably in a planning document.

3.6.2 Managing the Subcontract: Understand the role automated systems will play in the development and delivery of a product. A subcontractor's automated systems are most likely the reason you contracted with them in the first place. The subcontractors management attention to those automated systems represent an indicator of NDCR management health. The contract itself will inform the customer, to a limited degree, of the NDCRto be used by the subcontractor, providing NDCR was addressed in the first place. The rest of the information will have to come from customer - subcontractor discussions and face-to-face visits. Each stage of a product procurement necessitates an understanding of the NDCR involved and the potential risks at hand, not just the productivity enhancements advertised by company public relations. Dig into how changes are controlled for a computer aided manufacturing system. Understand the particular specifications or requirements set into a computer aided process controller and what protections exist to prevent unauthorized changes to the process. Find out how acceptance criteria will be programmed into an automated test system and determine how and what is the nature of raw data outputs. Examine the role of methodology in setting standards and evaluating calibration or correction factor data files used in testing systems. Question the credibility of simulation and analysis programs used with a computer aided design (CAD) system and find out how the CAD data base is handled to prevent unauthorized changes.

4.0 SYSTEM CONSIDERATIONS: There are various applications of computer resources (CR) that typically fall into the category of NDCR. With any specific CR application will come some special management considerations. For the most part, the NDCR manager will be exposed to the level of discipline that needs to be exercised for an application. With management discipline comes the hazzard of over constraint, a concern that has a good basis for management attention. Likewise will come the different support considerations both for the equipment and software, and the users. Finally, a manager will need the visibility of the resources at hand and will need various tools to assess the health of the management commitment.

4.1 Automated Test Systems: Automated test systems (ATSs) should receive rigorous management treatment to assure the integrity of ATS hardware, software, and associated documentation and ATS interfaces.

4.1.1 ATS Impact: ATS plays a major role in the acceptance of deliverable products at all levels of build-up and test. In some cases, an ATS may be as complex and essential as the deliverable product being produced and tested. Management should ensure that a rigorous life-cycle treatment is applied to ATS again commensurate with the project risks.

4.1.2 ATS Plans: ATS development is an integral element to overall program planning. Such development should be defined by a development/product specification, hardware and software configuration documents that are adequately defined and controlled. Also, the ATS should be rigorously validated/verified for all test requirements, demonstrate its accuracy and precision, and be highly maintainable. Management should pay close attention to an overall ATS usage strategy, that is look beyond the initial ATSappliction. With new equipment, sensors and advanced computer availability, the ATS investment will not be static. Management needs to look at long range transition strategies to capture basic hardware, software and documentation designs. Reusable software is a potentially significant cost saver. Employing a modularized ATS configuration employing standard data busses and hardware connection could be significant. Committing to a high order language as PASCAL, ATLAS, Ada, etc. might also be significant in reducing computer program development costs.

4.1.3 ATS Controls: ATS usage should be properly controlled through automated/manual procedure, adequately maintained and calibrated, adequately protected to avoid unknown hardware/software configurations, and properly documented testing conditions to assure test repeatability.

4.1.4 ATS Integrity: ATS computer programs should undergo rigorous library controls and corrective action discipline. Built-in test capabilities, calibration routines, problem diagnosis programs and ATS self-test results should receive particular control emphasis to assure ATS integrity.

4.1.5 ATS Documentation: ATS user and design documentation, including ATS training materials, should also be properly developed and controlled. Efficient use and effective maintenance of the ATS relies heavily on this factor. Reluctance to invest indocumentation is the most serious hazzard facing managment. Many times, an ATS may have a specific application for only one to two years. After this period, the ATS may scrapped or completely reconfigured for another project. But be observant. One might sell the hardware, erase the software media, but the documented algorithms and basic ATS configuration designs (both hardware and software) will most probably be used again. In fact, long term NDCR management should have addressed "software reusability" at the start to reduce reinventing ATS designs, implementation and usage instructions.

4.2 <u>Computer Aided Design Systems</u>: A computer aided design (CAD) system should receive adequate management attention to maintain the integrity of CAD analysis computer programs, user instructions, operating systems and generated data bases.

4.2.1 Data Bases: The generated CAD data base that describes the design of a product warrants very stringent controls. NDCR managers should establish very explicit procedures in establishing, changing and certifying a CAD data base. The media does offer additional risks in maintaining the purity of a data base and confidence in the controls established by management. The first risk is in the acceptance of design data into a controlled data base. Is the design ready for rigorous control, or is it long overdue? Management should consider requiring that various checks or steps have been completed before accepting a design for control. For example, have certain analyses been done and documented? Have certain interfaces (e.g., jigs, tools, software) been examined and results documented? Second, once the data base has been accepted, who is authorized to make changes and what procedures are to be followed? Are back-up copies of the data base kept at a remote site? Are old versions retained? When are updates performed and how often permitted or required? Finally, who can have access to the data base, even if only on a read only basis? Proprietary or security constraints should be of vital concern.

4.3 Computer Aided Manufacturing Systems: A computer aided manufacturing (CAM) system should receive adequate management attention to maintain the integrity of CAM computer program support tools, operating systems, user instructions, safety features and used data bases.

4.3.1 CAM Data: A set of commands designed to drive a particular numerically controlled (NC) device poses a configuration management challenge. Once a file has been established, whether by manual or automatic means, an operator is faced with the problem of retrieval, identification and verification. Finding a particular data file might be very difficult unless stored on a separate media, i.e., paper tape, diskette, etc. Even if the file is accessible, the next question is whether the content of the file reflects the correct version. Finally, feedback is necessary in verifying that the final product was properly made for a given NC output file used, i.e., a process to avoid the "impossible" errors. Herein rests the challenge; each one of these steps will involve man-machine interaction. The opportunity for "garbage-in, garbage-out" will exist. The NC machines are not intelligent (yet) to catch obvious input, set up or command errors. One must pay particular attention to each man-machine step and build-in preprocessing checks to detect common errors. Independent, data entry processes in addition to simulations might be considered as a trade-off to numerous "dummy" product manufacturing dry runs. Checking of "to-be-used" files with history files for command-by-command accuracy might be implemented when high-value materials or extreme precisions are required. If one is using a distributive system, it might be useful to use fault detection routines at NC site to assure accurate reception of data. (Remember the CAM environment is typically very noisy and can have drastic effects on simple communication methods.)

4.3.2 CAM Safety: Safety is a must. Computer controlled processes provide a greater opportunity to link many previously separate processes into one continuous set of operations. If the human must interact with the NC process, care must be taken. Typical integrations of robot devices wtih "traditional" NC devices also should enhance the managers safety concerns. But what to do about it? Perhaps, the best process should not involve human-machine interaction during any sequence; such interaction should occur only after the computer has released control of the machine. As an alternative, one should consider software routines to evaluate any given NC process to, if nothing else, identify potentially hazardous operations.

4.4 Computer Aided Inspection Systems: A computer aided inspection (CAI) system should receive adequate management attention to maintain the integrity of CAI computer programs, operating systems, user instructions, and used/developed data bases.

4.4.1 Criterion: In any inspection activity, an actual measurement will be compared to some expected measurement with an associated tolerance. Established or expected measures form a criterion data base whose accuracy is a must. With any computer comes the configuration management problem. Do I have the right version? Has anyone unauthorized altered my data base? Is the criterion being applied to the proper coordinates or locations in or on the product? These are typical configuration concerns that are extremely important to address where a CAI system is involved. Rigorous procedures in establishing, entering and maintaining criterion must be on hand. If possible, build such safeguards into the computer programs used to interface with the human.

5.0 MEASUREMENT PRACTICES: Just because the NDCR manager is concerned about such resources, are the resulting practices adequate to satisfy management's objectives? Perhaps the established management practices are too constraining or are ill-defined. What if changes to the practices are implemented, will the results on the personnel and computer resources, both good an bad, be visible? Management will need some form of measurement to address these questions. Common measurement methods involve evaluation, inspection and auditing; terms that can strike terror into the heart of a manager. So, how might one address needed measures on NDCR management and not cause cardiac arrest? Perhaps, the following guidelines can help.

5.1 Measurement Organization: The process of measurment can generally be satisfied through normal supervision activities or by establishing a separate organization/activity. Consider methods where "independency" exist, whether done by a computer (i.e., automatic measurement) or by an evaluator. Independency helps to reduce significant skewing due to personal or "directed" biases. Even if independent measures are achieved, be very careful of the handling of the acquired information. Constructive uses of the measures is absolute; primitive uses only destroy. Management should consider a feedback loop for measurement information to the lowest possible level of an activity and not just upward flow or "black-hole" collection of information.

Acquiring the information should be as integrated into ones way of doing business as possible. Collecting specific information at the computer level and as automatic as possible is the ideal situation. Using the computer to assist in colection, processing and feedback is generally desirable. Just be careful of sensitive, classified or proprietary information. Measures should reflect various system characteristics: configuration status of hardware, software and documentation; personnel productivity/performance and compliance with established rules; and product conformance (objective and subjective). Try to keep a reasonable perspective; be serious with your management objectives. To measure NDCR management system characteristics effectively requires "tailoring." What bounds tailoring will be rather subjective. There is no magic formula. The best guidelines are keep it simple, make it work for you, and get a good cross section of all system characteristics.

5.2 Evaluating Planning: As NDCR use increases within a development environment, management should evaluate how well the NDCR was integrated intc the overall deliverable product planning. Certainly, reviewing past p--gram problems with NDCR or its program integration are necessary to preclude similar future problems. However, as with any new applied technology, developers should evaluate how the potential risks to the program have been addressed. Automation advances, for the most part, create a "potential" for significant cost/schedule impacts resulting from inadvertent or intentional alterations to computer programs and data bases.

5.3 Auditing Corrective Actions: Writing procedures to correct problems to NDCRs is only the first step for management. Making sure that the discipline has been adhered to is essential. Many NDCRs comprise systems that service other company employees. User problems necessitate a responsive and thorough corrective action system. A disciplined approach to corrective action should be maintained to avoid causing new problems while fixing old ones. Likewise, the decision to implement a change or not should be an integral part of the corrective action system either through the control of a responsible manager or a group of key players. Verify these activities are properly supporting the users is the bottom line objective.

5.4 Evaluating Documentation: Feedback from the user is generally the best measure of how effective computer resource documentation really is. Getting feedback is the first challenge. Understanding and using the feedback is the second challenge. Characteristics of NDCR documentation that should be evaluated might include accessibility, readability, algorithm descriptions (degree of depth), general logic flow, test/validation information, requirements information and interface information.

5.5 Auditing Management: Being self-critical is never easy, especially for management when everything seems to be going well. Automation can lure management into a false sense of security. What might be a small problem frequently multiplies into one or a number of big problems when not treated in a disciplined fashion. Certainly, watching how management copes with such small problems can be amusing. When the problems are no longer small and costs mount, the humor can diminish rapidly. Checking management's resolve to follow its own prescription, directly reflects the health of the business. Perhaps it is only common sense, but it is only as common as management allows it to be.

5.6 Auditing Library Controls: Auditing library controls is a safety check or an insurance check. NDCR computer programs represent a sizeable monetary investment and, frequently are critical to the effectiveness of an activity. Again, checking one's own procedures, whether simple or complex, reveals the health of the control discipline.

5.7 Measuring ATS: The automated test system (ATS) is a very important component to the product delivery process. All the checks and balances placed on getting an ATS working and keeping it going should be complemented with an oversight activity. Again, compliance with established procedures and identification of "bad" practices are the primary objectives for measuring related ATS procedures. Common sense dictates rigor, providing the product developer is interested in producing a quality, deliverable product. Characteristics that might be addressed during procedure audits and reviews are:

a. Adequacy of ATS documentation that is needed to interface with the product to be tested (e.g., interface equipment and calibration factors.)

b. Adequacy of the test input/output documentation (e.g., supportive of test repeatability and diagnosis of problems).

c. Adequacy of procedures to determine the precision and accuracy of the ATS.

d. Adequacy of ATS library controls and ATS corrective action activities. Also, how do ATS problems interface with tested-product problems?

5.8 Evaluating Subcontract Management: Any customer wants to be assured of a good product/service from a subcontractor. The supporting cost (i.e., in-house, nondeliverable resources) might not get the proper attention unless addressed by the item developer. The quality and subcontractor interest in the NDCR will be a significant indicator of the development process health. In addition, the dependence upon NDCR by the deliverable product for maintenance may be an off stage factor and waiting in the wings not to be revealed until after product delivery. The unpleasantness associated with product deficiencies tied to NDCRs or the revelation of "unknown" backup resources are always a source for heated conversations and wasted revenue.

6.0 MANAGEMENT BENEFITS: Commitment to managing NDCR extends the basic principles of planning, responsibility, and control deemed applicable for deliverable resources. It instills the fact that NDCR are not mechanical devices, but complex electromechanical systems. This raises important management concerns about risk in the use of NDCR to the deliverable cost/schedule factors. It also associates the same "life-cycle" concerns identified for deliverable weapons systems and urges maintainability to be a part of the NDCR development consideration. As is fundamental to many computer systems, protection of the computer program and data information (both written and computer stored) becomes a focus for efficient use of NDCR. All these factors point to the protection of the productivity enhancement gained by automating a process. Hence, a substantial intangible cost saving is realized through preventative measures. Also, direct cost savings are achieved through the implementation of a standard management approach within an activity. Employee transfer with the activity will require less reorientation/training. To gain efficiency within an organization, management can centralize/decentralize such functions as software libraries, configuration control, procedure auditing, corrective action control/monitoring, documentation support, and subcontractor evaluation. Although a cost is incurred with implementing and maintaining such functions, such costs are substantially less than doing activities without automation. And incurring such costs are necessary if the automation benefits are to be sustained for any period of time.

7.0 MANAGEMENT CONCERNS: With the adoption of any discipline comes a cost of its implementation and maintenance. Implementation costs will be a direct reflection of the formality management elects to adopt. Implementation costs will normally be one-time. Maintenance costs will, again, be incurred and reflect the formality adopted by management. Hence, with some added costs expected, management could inflate these costs, whether deliberate or not, through the misapplication of the management objectives. For example, management could over control activities, preparing excessive procedures or developing restrictive policies in a business environment where simple policies/procedures could suffice.

APPENDIX B

KSAs

KNOWLEDGES, SKILLS, AND ABILITIES

AIR FORCE SYSTEMS COMMAND

QUALITY ASSURANCE

AIR FORCE SYSTEMS COMMAND

QUALITY ASSURANCE
KNOWLEDGES, SKILLS & ABILITIES

(KSAs)

1 CORE & SYSTEM DISCIPLINES

APPLY TO ALL QUALITY ASSURANCE POSITIONS

AA ACQUISITION QUALITY ASSURANCE (Orientation)

Familiarity with:

001. Defense Acquisition Regulation (DAR) purpose, structure and use.

002. Types of procurement, e.g., formal advertising, negotiation and special types and methods, and selection of contractors.

003. Types of contracts (e.g., fixed price, cost reimbursement); and contract structure, including format, numbering, CDRL, order of precedence, changes, clauses, etc.

004. Air Force Systems Command Acquisition process: phase of the ystem life cycle, establishment of Program Project Offices, and general rogram management responsibilities.

005. DAR, DOD, DCAS, other military services, Air Force and AFSC policies related to assignment of contract administration and supporting contract administration.

006. Five levels of contractual Quality Assurance (QA) requirements specified by DAR and their related DAR clauses.

007. Government specifications and standards related to Quality including MIL-I-45208, MIL-Q-9858, MIL-C-45662, MIL-STD-1520, MIL-STD-1535, MIL-S-52779, AFSC unique requirements and NASA documents.

008. Tailoring of QA requirements contained in specifications and standards, and policies related to contractor QA plans.

009. Other DAR provisions related to Quality, e.g., subcontracting policies, DD-250, pre-award practices, and Government property.

010. Program Office control of contracts: post-award meetings; contract administration delegations/MOAs/letters of instruction; program management reviews and engineering reviews.

011. In-plant Contract Administration QA program, i.e., DAR Section XIV, AF/AFSC/AFCMD policies and procedures (including AFCMDRs 74-1 d 178-1). Development of QA plan for contract implementation.

012. Range and Test Center QA program, to include basic understanding of mission, QA functions and tasks, test wing interface, remote site activity and user coordination/interface. (AF/AFSC and AFSC Component Policies and Procedures, including AFSCR 74-2).

013. Quality surveys by contractors, Government CAS and buying activities, and related scheduling, interface coordination and conduct.

014. Configuration Management policies and procedures, specification review, engineering documentation and change control, and configuration reviews/audits.

015. User Quality program and feedback systems.

016. Ability to search and find appropriate technical and contractual information within the government including the use of the "DOD Index of Specifications and Standards", libraries and other sources.

4B GENERAL COMMUNICATION SKILLS

001. Familiarity with the psychological aspects of the communication process.

002. Ability to communicate through written letters, memoranda or reports, including the ability to organize material, present it in an effective manner, and clearly convey the information necessary to assure the desired reader reaction and action.

003. Ability to orally: convey information to achieve the desired result; respond clearly and concisely to questions, and cope with both face-to-face and telephone communication problems.

004. Understanding of common listening and memory faults, and ability to listen effectively.

005. Understanding of verbal and behavioral skills needed for interpersonal negotiation/bargaining situations.

4C HUMAN RELATIONS

001. Familiarity with various factors influencing interpersonal relationships, e.g., ethnic differences, impact of voice tones, behavioral patterns, use of language, etc.

002. Ability to interact with other people effectively and resolve problems and conflicts without creating an adversary relationship.

003. Ability to analyze the impact of your personal behavior on others, and seek means for improvement if needed.

004. Ability to recognize needs of others, and employ appropriate communication strategies in order to obtain willing cooperating, consideration and respect of persons interacted with.

KNOWLEDGES, SKILLS & ABILITIES

(KSAs)

2 NON-TECHNICAL SPECIALIZED DISCIPLINES

AD IN-PLANT QUALITY ASSURANCE

001. Understanding of the in-plant Acquisition Quality Assurance Program as prescribed in DAR, DOD, AF and AFSC directives.

002. Thorough knowledge of, and ability to comply with requirements of AFR 74-1, AFSCR 74-1, and AFSC component peculiar directives, i.e. AFSCR 74-2, AFCMDRs 74-1 and 178-1.

AE CONTRACT ADMINISTRATION

Knowledge of:

001. DOD organization for Contract Administration.

002. Authorities and responsibilities of Principal Contracting Officers (PCOs) and Administrative Contacting Officers (ACOs).

003. Types of contracts, e.g., FFP, FPI, CPIF, etc., and general familiarity with cost/risk sharing practices.

004. Contract structure, including format; numbering, CDRL, order of precedence, changes, clauses, etc.

005. How contract modifications are authorized, i.e., changes, clauses, and methods of modifications, i.e., change orders and supplemental agreements.

006. Highlights of contract cost principles and procedures, and pricing theory and practices.

007. Policies and procedures for evaluation, prior to contract award of contractors' capabilities to meet the terms of the contract.

008. DAR, Air Force and AFSC policies related to assignment of contract administration and supporting contract administration.

009. Overview of basic Quality Control and Quality Assurance philosophies and their evolution.

010. Five levels of contractual QA requirements specified by DAR and their related clauses.

011. Inspection, acceptance and warranty provisions of contracts including Government and contractors' rights and responsibilities.

012. Differences among quality and inspection provisions in cost vs fixed price contracts.

013. Other CAS functions related to Quality, e.g., subcontracts, make or buy and procurement systems approval; Government property; production/manufacturing surveillance and Priorities and Allocations Systems.

014. Methods for resolution of disputes, claims and controversies.

015. Procedures for termination of contracts for default by the contractor or convenience of the Government.

016. Ethical considerations for CAS personnel and standards of conduct.

AF INSPECTION SYSTEM/QUALITY PROGRAM REQUIREMENTS

001. Comprehensive knowledge of all provisions of MIL-I-45208 and MIL-Q-9858.

002. Understanding of Government Interpretations of the MIL-I and MIL-Q requirements as reflected in Military Handbooks H-50 and H-51.

003. Understanding of the language of these specifications in terms of contractually enforceable vs non-enforceable.

004. Knowledge of USAF policy of placing QA plans on contract and contract impact.

005. Ability to evaluate adequacy of contractor procedures written to implement the specification requirements.

006. Familiarity with NASA publications NHB 5300.4(1B) and (1C).

AG NONCONFORMING MATERIAL/CORRECTIVE ACTION

001. Comprehensive knowledge of nonconforming material/corrective action provisions of MIL-I-45208, MIL-Q-9858, and NHB 5300.4(1B) and (1C).

002. Understanding of Government interpretations of the MIL-I and MIL-Q requirements as reflected in Military Handbooks H-50 and H-51.

003. Comprehensive knowledge of "Corrective Action and Disposition System for Nonconforming Material" requirements of MIL-STD-1520.

004. Ability to evaluate adequacy of contractor procedures written to implement the specifications and standards requirement.

AH CALIBRATION SYSTEM REQUIREMENTS

001. Understanding of calibration system requirements imposed by

contracts (as specified in MIL-I-45208, MIL-Q-9858, MIL-STD-45662, interpretations of HDBK-52, and NASA publications) including:

a. Relationship between certified measurement standards and national standards.

b. Accuracy of standards.

c. Use of calibration labels on measuring and test equipment.

d. Calibration certifications and reports.

e. Calibration procedures for production tooling used as a medium of acceptance inspection.

002. Ability to evaluate contractor's procedures written to implement the contractual system requirements, and the written calibration procedures, and the ability to evaluate for compliance.

-I <u>SUBCONTRACT QUALITY</u>

001. Understanding of basic provisions of DAR relating to subcontracting, including make-or-buy decisions, consent reviews, procurement system reviews, contractor responsibility for subcontractors, etc.

002. Knowledge of various contractual provisions which may be applied specifically related to quality of subcontracted supplies, e.g., MIL-I-45208, MIL-Q-9858, MIL-STD-1535, and NASA Quality publications.

003. Comprehensive understanding of requirements for contractor's pre subcontract activities, e.g., purchase documentation, selection of suppliers, preaward surveys, etc.

004. Comprehensive understanding of requirements for contractor's post award activities, e.g., periodic review of supplier performance, inspection of products, and corrective action.

005. Ability to evaluate the adequacy of contractor's documented system for control of subcontract-quality, its effectiveness and their compliance thereto.

006. Understanding of Government rights and responsibilities related to subcontractors, including standard inspection clauses, rights and limitations, Government review of contractor purchase documentation, conditions for requesting supporting contract administration or Government Procurement QA at source.

007. Ability to make logical, cost effective decisions relative to imposing Government actions at subcontract level.

4J <u>QUALITY SURVEY CONCEPTS AND TECHNIQUES</u>

001. Understand purpose of surveys conducted by various organizations (Program Project Office, cognizant AFPRO, DCAS, or other organizations)

002. Understand the essential elements of a survey such as preparation, conduct, reporting and follow-up.

003. Knowledge of planning and research required prior to conducting survey.

004. Understanding of need for checklists, and the ability to develop them.

005. Ability to apply effective techniques in conduct of surveys, such as in-briefings to supervisors, conducting interviews, reliance on objective evidence, observing courtesy and protocol, validation of findings with personnel surveyed and outgoing critiques.

006. Ability to prepare written reports of survey.

007. Understanding of means for obtaining corrective action and alternative methods for follow-up action.

4K SOFTWARE QUALITY ORIENTATION

001. Familiarity with basic elements of computer software design, coding, configuration control, and verification and validation.

002. Familiarity with digital computer hardware equipment, e.g., embedded vs automatic data processing systems, computer logic, languages, arithmetic operations, storage and peripherals.

003. Familiarity with microprocessors and firmware, including semiconductor theory, input-output, memory capability, discs, tapes, drives, displays, etc.

004. Understanding of requirements MIL-S-52779 (AD), Software Quality Assurance Program Requirements, for contractor software program.

005. Understanding of responsibilities and authorities of various AFSC organizations related to software quality.

4L CONFIGURATION MANAGEMENT

001. Understanding of concepts of acquisition life cycle and the relationship thereto of configuration management activities.

002. Knowledge of the family of documents and engineering drawings which identify or describe the system or item (Configuration Identification).

003. Familiarity with the methods of controlling and managing changes to the Configuration Identification baseline. (Configuration Control).

004. Familiarity with Configuration Status Accounting Management information system which provides for records of actions accomplished under the identification, control and audit functions of Configuration Management.

005. Familiarity with reviews and audits performed, including System Requirements Reviews, Design Reviews, Functional and Physical Configuration Audits.

006. Understanding authorities and responsibilities of the AFSC components and sub-components in the various aspects of Configuration Management.

[illegible] PROGRAM/PROJECT OFFICE (Purchasing Activity)

001. Understanding of the five major phases of activity in the life cycle of a system acquisition.

002. Knowledge of the key program control elements of a Program Office, i.e., estimating, budgeting, scheduling, planning, analyzing and forecasting.

003. Ability to develop and manage the total system acquisition QA requirements over the system life (concept, validation, full scale development, production and deployment).

004. Understanding of the principal functional processes accomplished during the various phases of the acquisition cycle, e.g.

Engineering Management
Configuration Management
Test and Evaluation
Manufacturing and Production Management
Integrated Logistics Support
Data Management
Etc.

005. Familiarity with the missions, authorities, and responsibilities of all AFSC components involved in Quality Assurance aspects of the Systems Acquisition process, including Air Force Flight Test Center, Arnold Engineering Development Center and all AFSC Program Management Divisions (AD, ASD, ESD, BMO, SD) and the Air Force Contract Administration Division.

006. Knowledge of Program Management Directives (PDM), AF Systems Acquisition Review council (ASARC), and Defense Systems Acquisition Review Council (DSARC).

007. Knowledge of Statement of Work (SOW) documents.

008. Ability to tailor QA program requirements in SOWs.

009. Knowledge of contract clauses, Source Selection Board (SSB), and Contract Data Requirements List (CDRL).

010. Knowledge of Program Assessment Reviews (PAR), Command Assessment Reviews (CAR), and Program Management Requirement Transfer (PMRT).

011. Considerations and requirements for preparing a Memorandum of Agreement (MOA) with cognizant AFPRO, and Letter of Instruction (LOI) for a cognizant DCAS office.

012. Knowledge of QA requirements needed for: (1) A Program Management Plan (PMP), and (2) Development of the total QA program planning for an acquisition program during validation, full scale development (FSED), and production phases. It should cover a program from time of inception of QA Program requirements, through to implementation of Contractual QA Program by contractor until inspection, test and final acceptance by the Government.

013. Understanding of QA involvement in and utilization of GIDEP and other government or industry failure data exchange programs that can involve QA activities.

014. Understanding of the QA role in System Requirements Reviews. System Design Reviews, Preliminary Design Reviews, Critical Design Reviews, FCAs and PCAs as defined in MIL-STD-1521.

015. Ability to plan and participate in PDR, CDR, Physical Configuration Audits, Functional Configuration Audits and Production Readiness Reviews.

016. Knowledge of component breakout process for implementation of DAR 1-326 and AFSC/AFLC 800-31.

017. Ability to prepare and monitor the QA sections of award fee and incentive contracts.

018. Ability to conduct QA assessments and independent QA assessments as required by AFSCR 74-1.

019. Ability to manage implementation of those portions of the AF Defective Parts and Components Control Program (DPCCP) which are allocated to AFSC Product Division.

020. Ability to identify critical components and their mandatory characteristics for CAO verification based on an assessment of the product improvement process or test parameters.

021. Ability to implement and negotiate mutual contract understanding.

022. Ability to establish, convey and maintain QA program policies and plans that are program unique.

023. Ability to prepare and issue QA program Operating Instructions (OIs).

024. Ability to provide QA coordination with other related program/project offices.

KNOWLEDGES, SKILLS & ABILITIES

(KSAs)

3 TECHNICAL - GENERAL DISCIPLINES

AN MEASUREMENT TECHNIQUES

001. General familiarity with a broad range of measurement techniques, including mechanical, electronic, optical, etc., e.g., micrometers, scales, vernier calipers, optical comparators, voltage meters, and gauges.

AO STANDARDS OF WORKMANSHIP

001. Familiarity with the necessity for standards of workmanship, how they are developed for manufacturing purposes, and how the criteria are used to determine the acceptability of products.

002. Ability to verify the adequacy of contractor's application of workmanship standards.

AP INTRODUCTION TO NDT

001. Knowledge and fundamental concepts of acceptance inspection/test of materials and supplies.

002. Understanding of applications and limitations of nondestructive inspection/test, and reasons for use.

003. Familiarity with various nondestructive methods, including radiography, magnetic particle, ultrasonics and liquid penetrant.

AQ DRAWINGS AND SPECIFICATIONS

001. Understanding of use of drawings and specifications in production.

002. Ability to read engineering drawings, including lines, symbols, projections, dimensioning, tolerancing, parts listed and administrative information.

003. Knowledge of military and Federal specification and standards, including format, coordination procedures, and methods of changing specifications.

004. Familiarity with DOD Index of Specifications and Standards.

AR METRIC CONVERSION

001. Working knowledge of metric system and familiarity with issues surrounding the transition to metric measurements.

002. Ability to identify the primary units of measure in metric system and how they are used.

003. Ability to convert back and forth between metric and customary U.S. System of Measurement in order to understand contracts, specifications, drawings, etc.

AS BASIC STATISTICS

001. Understand terms, symbols and formulas used in statistics.

002. Calculate descriptive statistics such as percentages, measures of central tendency and measures of variations.

003. Understand fundamentals of process control statistics and charting.

004. Knowledge of concepts of acceptance sampling procedures, including military sampling plans for sampling by variables and attributes.

AT PRESERVATION AND PACKING

001. Knowledge of various Government specifications and standards related to preservation,packing and marking.

002. Understanding of essential elements of preservation and packing such as cleaning, preservatives, containers, environmental controls, marking and labeling.

003. Ability to verify contractor adherence to preservation and packing requirements through personal inspection and test activity.

AU RELIABILITY AND MAINTAINABILITY

001. Knowledge of terms, definitions and concepts used in reliability studies, and the methods used in predicting and testing for reliability.

002. Skill in predicting reliability from sampling schemes and interpreting sampling and test results.

003. Ability to estimate, by category (minor, major) the effects of reliability on items, assemblies, systems, etc., of nonconformances in design, material and workmanship.

004. Knowledge of the terms, definitions, and concepts used in maintainability studies. Knowledge of the specifications and requirements used in maintainability studies and human factors design.

005. Ability to estimate the effects on maintainability of nonconformances in design, material, and/or workmanship.

KNOWLEDGES, SKILLS & ABILITIES

(KSAs)

4 TECHNICAL-SPECIALIZED DISCIPLINES

COMPUTER SOFTWARE

AU SOFTWARE DESIGN

001. Understanding "top-down" design, structured programming, software modules, design of software modules; tree structures and flow charts.

002. Skills in recognizing correct design decomposition steps and in asking relevant questions concerning individual tree structure elements (modules) and flow chart operations (meaning of symbols, data flow, timing and control).

003. Ability to compare the contractual requirements to description of modules and flow chart operations and ask questions concerning the existence, adequacy, and compliance with each requirement of the contract. (Note that ability in computer programming (coding) is not required in this skill area).

004. Understanding of MIL-S-52779, DOD-STD-480A, MIL-STDS-483, 490, 1521A and MIL-STD-1679 (Navy) requirements as they pertain to software design.

005. Familiarization with program design languages (PDLs) commonly used in Air Force applications.

006. Participate in computer design reviews to assess contractor progress and performance during computer software development cycle.

007. Ability to maintain technical surveillance over the contractor's application of computer software tools, techniques, and methodologies. Specific emphasis will be given to that effort which supports quality assurance objectives such as systems analysis techniques, modeling, simulation, and other software optimization techniques.

008. Ability to review any tailoring to the applicable specifications and standards and identifies the lack or over-specification of QA requirements for each computer product being procured.

AV SOFTWARE CODING (PROGRAMMING)

001. Understanding of the methodologies used for translating software design modules into a programming language. Commonly used languages are Fortran, Jovial, Pascal and Ada (for future programs). These HOL (higher order languages) must be supplemented by knowledge of machine languages and assembly language.

002. Skills in recognizing proper compliance with programming standards and conventions constraints in the contact (such as: use of a specific HOL, limitation of 50 to 100 lines of code per module, limitations on the use of GOTO statements, etc.).

003. Ability to discuss coding methodology with computer programmers and to determine if their methods and procedures of coding are in compliance with the contract. (This level of skill will require basic training in computer programming in one or more of the HOL's used in military contracts).

004. Understanding of MIL-S-52779 requirements and MIL-STD-1679 (Navy) as they pertain to software coding.

005. Ability to determine the adequacy of the documentation standards and programming conventions and practices utilized by the contractor.

AW SOFTWARE CONFIGURATION CONTROL

001. Working knowledge of the definitions of baselining, change control procedures, releasing various versions of software, control of master tapes, printouts, card decks and flow charts.

002. Skills in auditing the contractors compliance with configuration control procedures for computer software and in recognizing nonconformance or bad practices.

003. Ability to cite authority for each configuration control standard, the reason for each rule or procedure, and specify proper corrective action.

004. Understanding of MIL-S-52779 requirements as they pertain to software configuration control.

005. Ability to evaluate the contractor's computer program library controls for both deliverable and nondeliverable software to assure that the computer program versions are accurately identified and documented.

AX SOFTWARE VERIFICATION & VALIDATION

001. Working knowledge of methods used for software testing, debugging, simulation, emulation and V & V (verification and validation).

002. Skills in evaluating testing procedures and results for compliance with the contractual requirements for each test.

003. Ability to locate invalid tests and improper results and determine the logical correctness of the final computer program to satisfy functional requirements.

004. Understanding the MIL-S-52779 requirements as they pertain to software verification and validation.

005. Ability to understand proper methods for both verifying and validating both deliverable and nondeliverable computer programs and associated documentation.

COMPUTER HARDWARE

AY DIGITAL COMPUTER EQUIPMENT

001. Working knowledge of types of military computers, embedded vs automatic data processing systems, computer logic, computer languages, elements of Boolean algebra, arithmetic operations, storage and peripherals.

002. Skills in interconnecting computer elements (hardware), data bases, testing systems using pre-programmed checkout tapes and card decks, and in interpreting results of system checkout prior to operation use of the computer hardware.

003. Ability to understand different types of computer hardware and typical use and/or limitations in actual operation.

AZ MICROPROCESSORS AND FIRMWARE

001. Working knowledge of semiconductor theory, types of "chips" by function, size, input-output (I/O), memory capability, RAM, ROM, PROM, EPROM similiarities and differences, floppy discs, tapes, drives, displays.

002. Skills in interconnecting microprocessor elements, testing for proper operation, and analyzing results prior to operational use.

003. Ability to recognize different types of microprocessors and firmware "chips" and understand their uses and limitations in actual operation.

BA TECHNICAL DATA AND PUBLICATIONS

001. Knowledge of requirements and responsibilities of Quality Assurance in inspection and acceptance of specific technical data items in accordance with terms of contracts.

002. Ability to apply Quality Assurance to data; DD Form 1423, Technical Data Specifications and Standards, Artwork, Photolithographic Negative, Engineering Drawings, Microfilms and Tabulating and Aperture Cards.

ELECTRONICS/ELECTRICAL

BB BASIC ELECTRICITY

001. Knowledge of all necessary mathematical concepts, fundamental concepts of electricity including sources of electricity, magnetism, direct current theory, theories and uses of alternating current generation, AC circuitry, impedance and resonance.

002. Knowledge of associated Specs and Standards (i.e. MIL-HDBK 217, MIL-STD-750, MIL-STD-883, MIL-S-19500, MIL-M-38510, MIL-STD-1313, etc.).

8C BASIC ELECTRONICS

001. Knowlege of necessary mathematical techniques, fundamental electronic theory and the use of various electronic equipment. Knowledge of theory and use of vacuum-tube principles and circuitry, which must be mastered before attempting advanced training on electronics and related fields such as solid state circuitry.

8D MICROELECTRONICS

001. Knowledge of hybrid technology and terminology, thick/thin film hybrid design rules, packaging of the circuit, mask making and photoresist operations, materials, substrates and assembly methods including die attach, wire bonding, beam leads, flip chips, automatic assembly methods, tape bonding and carriers.

002. Knowledge of evaluation of integrated electronic devices, basic principles of technology and applications and descriptions of the various forms of ICs, such as large scales and medium scale integration, hybrid circuits and reliability.

8E DIGITAL DEVICES

001. Knowledge of principles of digital electronics, diodes, transistors, numbering systems used in digital equipment, logic circuits and their symbols, arithmetic circuits (adders, counters, registers, memories and converters). Applications for missile guidance, aircraft navigation, digital communications, radar, sonar, counter measures, data processing, digital test equipment, computers, calculators and microprocessors.

002. Ability to interpret digital circuit diagrams and understanding and testing of modern military digital electronic equipment.

8F INTEGRATED CIRCUIT DEVICES

001. Knowledge of the principles of operation of both digital and linear integrated circuits.

002. Familiarity with various types of digital circuits including transistor, diode and resistance logic, storage elements (flip-flop), counters, encoders, decoders, shift registers, timing and control elements. Linear RF/IF amplifiers and voltage regulators.

003. Skill in the technique of inspecting monolithic, multichip and hybrid microcircuits for specification nonconformance prior to capping or encapsulation, and the pertinent inspection techniques and related procedures required under MIL-STD-883.

8G RADAR AND MICROWAVE

001. Knowledge of wave and transmission line theory, antennas and microwave components; the standard radar block diagram; individual block

elements; pulse relationships; types of radar displays and various types of radar systems; and radar and microwave tests and measurements, and test instruments involved, including typical specification provisions.

3H SUBSTRATE TEST MEASUREMENT

001. Knowledge and understanding of the procedures in MIL-M-38510 and the test methods in MIL-STD-883 necessary for the acceptance of substrates and complete microcircuit devices.

002. Familiarity with procedures for certification, qualification and quality conformance inspection.

003. Ability to interpret inspection and test procedures and acceptance, rejection criteria.

AIRCRAFT AND SPACE

3I WEIGHT/BALANCE

001. Knowledge of weight/balance safety regulations, set-up of test equipment, and procedures such as TO 1-1B-40, TO 1-1B-50, AFR 81-6 and MIL-W-25140.

002. Ability to interpret and evaluate weight/balance test results.

3J FUEL/FUELING SYSTEMS

001. Knowledge of fuels/fueling system theory and procedures; operation controls; fuel line installations including striking, kinking and chafing during service; checks for leaking, routine, lashing, supporting, spacing; use of protective sleeving; capping of open ends; grommet securing; torqueing; fuel line color codes, and identification, damage and contamination.

002. Ability to interpret and evaluate fuels/fueling systems test results.

3K JET ENGINES OPERATIONS & MAINTENANCE

001. Knowledge of theory and principals of gas turbine and jet engine operation.

002. Knowledge of the major work efforts associated with the QA of new engine procurements including turbojet, turbofan, turboshaft, turboprop and gas turbine engines; related components and accessories, and specialized mechanical and electrical requirements associated with jet engine testing and acceptance. The specific function associated with jet engine accessories and components, and their interrelationship and engine operation/performance. Engine buildup, green run testing, teardown procedures, inspection methods and technique, final buildup, final accep-

tance testing, final inspection/acceptance and preparation for shipment. Contractor's QA program, suitability of engine logs/workbooks, test documentation and engine performance data review/approval.

3L POWER PLANTS-FUELS AND LUBRICANTS

001. Knowledge in testing for chemical and physical properties of light distillates, equipment used in testing disposition of off-specification petroleum producers, interpreting laboratory reports.

002. Knowledge of laboratory equipment used in testing light and heavy distillates, procedures used in performing physical and chemical laboratory tests and interpreting and evaluating test results.

003. Knowledge of preaward surveys, contractor QA programs, accomplishment of DD Form 250, Material Inspection and Receiving Report and various other QA responsibilities.

3M HYDRAULICS/PNEUMATICS

001. Knowledge of hydraulic system theory as related to open and closed systems; functional testing caution and warning procedures; hydraulic pump and motor functions; flight control system interrelationships for hydraulic actuators and flight control system rigging.

002. Familiarity with flow rates for hydraulics and water column use on pneumatic testing; pneumatic strut operation; and pneumatic system control for canopy seal and environmental control systems.

3N PRE & POST-FLIGHT ACCEPTANCE

001. Knowledge of preflight, postflight, acceptance, and delivery of aircraft including familiarization with the technical documents of the services which prescribe the specific details for the performance of preflight and postflight inspections; general procedures applicable to a flight line operation which involve safety, movement of aircraft, servicing of aircraft, use of support equipment, operation of aircraft systems, engine operation and briefing of flight crews prior to flight.

002. Certification in all of the following areas is desirable: Flight and engine instruments, installation and checkout; aircraft systems and subsystems (except electronic); and aircraft components and subassemblies.

003. Ability to interpret and evaluate pre and post-flight acceptance test results.

3O EGRESS SYSTEMS

001. Knowledge of qualification and safety criteria need for the aircraft.

002. Familiarity with applicable military standards and technical orders.

8P MAINTENANCE AND OVERHAUL

001. Knowledge of various types/models of aircrafts and the methods/procedures by which shop complete and final inspections of contract items are conducted; the interrelationship of the contractor versus PQA work effort; the many obligations of the Government such as, government furnished property, aerospace ground equipment, ground support equipment, special tooling, input of reparable assets, technical documentation, technical assistance, work statements, procurement specifications, flight tests, ferry crews, product audits and work request proposals.

002. Knowledge of a typical M & O operation including, but not limited to, receipt/shakedown inspection, shop complete inspection, parts replacement, rob back or cannibalization preflight and postflight inspections, and acceptance and delivery of the aircraft, contactor QA program, aircraft workbooks, over and above activity, and flight safety items.

8Q OXYGEN SERVICING/OPERATIONS

001. Knowledge of theory and procedures of oxygen servicing/operations including, but not limited to, MIL-STD-1551, QC of Gases and Liquid Aviators Breathing Oxygen; AFR 127-101, General Safety Precautions and; TO-0025-172, Static Ground.

002. Ability to interpret and evaluate oxygen servicing/operations test results.

8R PROPELLANTS AND OXIDIZERS (SPACE)

001. Knowledge in the use of laboratory equipment and the various methods of analyzing alcohols, esters, aldehydes, acids, alkalies, salts, propellants and oxidizers such as RP-1, RJ-1, hydrazine, UDMH, nigrogen tetroxide, inhibited red fuming nitric acid, liquid oxygen and nitrogen. Application of approved test methods, interpretation of test results and contractual testing requirements.

8S ACCEPTANCE TESTING (SPACE)

001. Knowledge of acceptance testing of spacecraft and missiles and the procedures that must be followed preliminary to and during the performance of this function; the specific technical requirements as prescribed in specifications and contract requirements, and the procedures applicable to pretest, during test and post-test requirements.

002. Certification in all of the following areas is desireable: Spacecraft and missiles systems and subsystems (except electronic); and spacecraft and missile airframe components and subassemblies.

MANUFACTURING PROCESSES

8T ADHESIVE BONDED STRUCTURE

001. Knowledge of characteristics, procedures, techniques, applications and limitation of adhesives, solvents and potting compounds for mechanically joining parts of structures.

002. Familiarity with advantages and disadvantages of using adhesives as a joint material; types, characteristics and uses of different adhesives; joint designs, bonding techniques and preparations necessary for a properly bonded joint; controls of adhesive materials prior to and during use; inspection and testing technqiues for suitability of joint or assembled bonded structure and techniques for repair of defective bonded areas.

003. Knowledge of materials and processes used in construction and inspection of composite structures such as graphite, boron, and glass fibers, honeycomb structures, layup and curing techniques, vapor deposition, and test and repair techniques.

8U PLATING

001. Knowledge of the theory and procedures of plating.

002. Knowledge of the chemistry of plating, ion vapor deposition, tin electroplating, nickle electroplating, chromium electroplating and other types of plating.

003. Skill in verifying the adequacy of contractor plating operations and the quality of the item being inspected.

8V CLEANING

001. Knowledge of the purposes, equipment, materials used, procedures and controls in the cleaning of new and returned from service items of material or equipment.

002. Familiarity with chemical, solvent, mechanical, sonic and electrical methods of cleaning.

8W WELDING

001. Knowledge of concepts associated with weld inspection: surveillance of on-going QA procedures in welding facility; and the visual, non-destructive inspection and mechanical testing of welded structures (fusion, spot/seam, and other welding processes).

002. Ability to determine acceptability with regard to the specified quality level.

8X HEAT TREAT

001. Knowledge of equipment and operations of the heat treat processes for ferrous and aluminum alloys.

002. Familiarity with the common heat treatment processes, effects of improper procedures, controls and test methods needed to determine compliance with requirements, methods of heating materials, principles of hardening and softening metals by heat treatment, and calibration of equipment.

BY SOLDERING

001. Knowledge of present techniques and requirements for soldering.

002. Familiarity with current inspection procedures, proper techniques, tools and materials according to the latest standards for soft soldering electrical connectors.

003. Ability to accept or reject soft soldered electrical connections according to pertinent specifications.

BZ ELECTROPLATING AND ANODIZING

001. Knowledge of the equipment and procedures related to electroplating and anodizing.

002. Familiarity with the purposes, preparation of base metal, processes and controls.

003. Ability to test the plated and anodized surfaces to specification.

CA FINISHES, PAINTS AND SURFACE TREATMENT

001. Knowledge of the various surface finishes applied to metals and metallic products for protection.

002. Familarity with the materials used as finishes, the processes of applying the finishing materials including painting, black oxide coating, phosphate coatings, chromium conversion coatings, and passivation of stainless steels.

003. Ability to perform tests required to accept or reject per specifications.

CB CORROSION CONTROL

001. Knowledge of the equipment and procedures used in corrosion abatement.

002. Familiarity with the types of corrosion such as water intrusion, galvanic, vapor corrosion, etc.

CC BRAZING

001. Knowledge of the various processes used in brazing.

002. Familiarity with the different alloy materials used in brazing, the flux materials, effect of joint thickness on strength and characteristics of satisfactory brazed joints.

003. Ability to accept or reject brazed joints per applicable specifications.

CD OPTICAL TOOLING AND ALIGNMENT

001. Knowledge of application of optical tooling and alignment.

002. Familiarity with optical tooling terminology and solving problems encountered when using optical tooling equipment.

003. Ability to operate and checkout optical tooling equipment.

CE DIMENSIONS AND TOLERANCES

001. Knowledge of precise and uniform drawings technology along with uniform interpretation of this technology.

002. Ability to read and interpret drawings.

CF LASERS

001. Knowledge of laser characteristics and various applications.

002. Knowledge of various laser applications in industry including applications such as heat treating internal and external surfaces, heat treating irregular geometries and corners, two sided bimetal welding, precision spot welding, long stand-off welding, bimetal contoured welding, high-yeild hermetic welding, hermetic high speed welding, omnidirectional distortion-free cutting, hard plated metal cutting, deep section cutting, partial cutting, engraving, refractory non-metal cutting and non-metal high-volume cutting.

CG FERROUS AND NON-FERROUS METALS

001. Knowledge of the principles and processes in the making, shaping, and treating of metals including extraction, refining, alloying, forming and heat treating of metals, and process controls and test methods used in metal production and manufacturing; the origin of metal defects in the cycle from basic metal to final product.

002. Knowledge of the purposes and methods of conducting mechanical tests commonly required by contract, including tensile, compression, shear, cold bend, hardness and impact testing.

NDT/NDI AND ENVIRONMENT TESTING

CH RADIOGRAPHY

001. Knowledge of the theory of radiography, including x-rays and radioactive isotopes.

002. Knowledge of the applications and limitations of test methods, application of specifications and standards pertaining to radiographic inspection.

003. Ability to make radiographic set-ups, positioning and operating radiographic equipment, and processing and interpreting films.

004. Skill in verifying adequacy of contractor radiographic operations and the quality of the items, including microcircuits being inspected through radiographic techniques.

C I MAGNETIC PARTICLE

001. Knowledge of magnetic particle inspection and magnetizing procedures and techniques.

002. Ability to interpret test results and pertinent specification requirements.

C J LIQUID PENETRANT

001. Knowledge of liquid penetrant inspection procedures, techniques, physical principles, pre-cleaning, dwell time and fluorescent applications.

002. Ability to interpret test results and pertinent specification requirements.

C K ULTRASONICS

001. Knowledge of ultrasonic inspection principles, procedures and techniques of applying ultrasonic energy to the inspection of materials for determining the presence of discontinuities in material, lack of bond between materials and for thickness measurement.

002. Knowledge of theory of ultrasonic energy, the generation and transmission of ultrasonic energy, couplant and transducer theory and function, flaw detection and thickness measurement applications, surface contact and immersion techniques.

003. Ability to interpret test results and pertinent specification requirements.

CL LASER BOND INSPECTION

001. Knowledge of laser bonding procedures and theory.

002. Knowledge of solid state and gas filled lasers, laser optics, aperture installation,, bond inspection principles including locating unbonded areas and determining unbonded bounderies, detecting void in bellows using laser beam sprekle-shift techniques, and thorough knowledge of personnel hazards such as retina burns and high voltage shocks in using laser equipment.

CM ADHESIVE BOND INSPECTION

001. Knowledge of characteristics, procedures, techniques, applications and limitations of adhesives, solvents and potting compounds for mechanically joining parts of structures.

002. Familiarity with advantages and disadvantages of using adhesives as a joint material; types, characteristics and uses of different adhesives; joint designs, bonding techniques and preparations necessary for a properly bonded joint; controls of adhesive materials prior to and during use; inspection and testing techniques for suitability of joint or assembled bonded structure, and techniques for repair of defective bonded areas.

CN EDDY CURRENT

001. Knowledge of eddy current inspection procedures and techniques.

002. Knowledge of detecting and evaluating surface and subsurface discontinuities in metals, including such items as raw tubing, extrusions, internal surfaces of holes in metals, sheet stock and thin parts, and machined parts.

003. Ability to operate eddy current equipment, read and interpret test results, and accept or reject parts.

CO ELECTROMAGNETIC INTERFERENCE COMPATIBILITY

001. Knowledge of theory and procedures of electromagnetic compatibility requirements and testing.

002. Familiarity with power measurements and spectral analysis to monitor test and measurement programs relative to radiation frequency interference, pollution and hazards; mathematical considerations including power ratios (decibels); theory and use of typical measurement equipment such as RF sniffers, spectrum analyzers, power density and field density meters; various techniques relative to noise and field intensity measurements and the compilation of data and the recording of data.

CP ENVIRONMENTAL TEST

001. Knowledge of theory and procedures of environmental testing.

002. Familiarity with various environmental tests including low and high temperature, humidity, pressure, rain, solar radiation, sand and dust, salt spray, fungus resistance, explosion, vibration, shock, acceleration and acoustic.

003. Ability to accept or reject test results per specifications.

MEASURING AND TEST EQUIPMENT

CQ ELECTRONIC MEASURING EQUIPMENT

001. Knowledge of the theory of electronic measuring equipment.

002. Ability to use all types of current, voltage, and resistance indicating devices such as meters; power supplies and regulators; resistance and impedence measurement equipment; waveform display and analysis equipment; signal generators; and frequency measurement instruments and devices including meggers, Hi-Pot testers, capacitance testers, transistor testers, and tube testers.

CR MECHANICAL MEASURING EQUIPMENT

001. Knowledge of mechanical measuring equipment techniques and the attendant mathematical requirements.

002. Ability to use precision measuring equipment including the techniques and setups required to obtain the measurement specified by the drawing.

003. Ability to use equipment such as surface plates, thread measuring equipment, optical comparators, scales, micrometers, vernier calipers and height gauges.

CS AUTOMATIC TEST AND MEASURING EQUIPMENT

001. Knowledge of automatic test and measuring equipment and the programming of such equipment.

002. Ability to use various automatic test and measuring equipment.

CT SCANNING ELECTRON MICROSCOPE

001. Knowledge of scanning electron microscope procedures and techniques.

002. Ability to use scanning electron microscopes as an inspection tool and the application of acceptance and rejection criteria and decision-making techniques.

CU INTERRELATED TECHNICAL DISCIPLINES

001. Knowledge of basic reliability and maintainability program requirements and demonstration/test (includes familiarity with MIL-STDS-785, 470, 471, and 781).

002. QA requirements as applied to: (1) nuclear hardened systems; and (b) QA requirements and tests that may be applied to systems or equipment that must be survivable or systems required to operate in an Electronic Warfare (E-W) environment.

003. QA applied to all AF Test and Evaluation program requirements such as those defined in AFR-80-14, where contractor is involved (such as DT&E).

KNOWLEDGES, SKILLS & ABILITIES

(KSAs)

5 MANAGEMENT DISCIPLINES

CU BASIC SUPERVISORY

001. Understanding of and ability to comply with civilian personnel policies and procedures.

002. Ability to establish standards of performance and evaluate employee performance to those standards.

003. Ability to conduct employment interviews and performance evaluation interviews.

004. Understanding of principles and policies of the EEO program and supervisory responsibilities related thereto.

005. Understanding of basic human relations concepts. (Reference Core and Systems Disciplines) and ability to apply.

006. Ability to communicate effectively. (Reference Core and System Disciplines).

007. Ability to direct and lead effective conferences, task force projects, and other group-participative tasks.

008. Ability to train, guide, direct and counsel employees in performance of their specific functions.

CV INTENSIVE SUPERVISORY/MANAGEMENT

001. Understanding of and ability to apply concepts of delegation of authority and responsibility, and tasks. (Task Management).

002. Understanding of objective setting concepts, and ability to carry through efficiently. (MBO/R).

003. Ability to understand and capacity to choose, ethical, logical and practical actions in relation to work. (Decision Making)

004. Ability to relate awarely, and creatively, with other workers, supervisors and subordinates.

005. Ability to motivate and influence employees both subordinate and superior.

006. Understanding of essential elements of effecting change, and the ability to make changes efficiently and smoothly.

007. Understanding of principles of time management and the ability to practice the concepts.

CW <u>ADVANCED MANAGEMENT CONCEPTS</u>

001. Understanding of program management organizational structures and analyses (functional, project, matrix, combinations).

002. Understanding of theory, organization and design of management systems, and the concept of viewing organizations as a total system.

003. Understanding of Management Information Theory, including the generation, organization, transformation, dissemination, codification, discrimination and economics of information.

004. Understanding of managerial applications of automated data systems with emphasis on management information systems.

005. Ability to make effective decisions based on relevant factors, including statistical theory and techniques, matrix charting, flexibility and originality in thinking, effective listening habits, etc.

006. Understanding of the relevance and significance of behavioral and social sciences regarding human organizational behavior.

007. Understanding of Productivity concepts and applications.

008. Understanding of the essential elements of leadership, including specific behavioral techniques, analysis of reactions of others, group dynamics, etc.

APPENDIX C

TABLES OF TASKS, RESPONSIBILITIES, AND TRAINING AREAS

TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
15. AFPRO or DCAS has responsibility to finalize preaward report and provide the report to the SPO.	II	II	I	I	---	---	III
18. Obtain general assistance and information on contractor's general capabilities in software areas from AFCMD. Augment AFCMD capability with SPO personnel during conduct of the preaward survey.	II	II	I	II	---	---	III
19. Establish channel of communication between SPO and AFCMD specific to new program/project.	I	I	I	I	---	---	I
21. Conference contract administration, quality assurance, engineering, and other AFCMD support as necessary to ensure flow of information to SPO.	I	II	II	II	---	---	II
22. Identify contract requirements for performance measurements.	I	I	---	III	---	---	---
23. Define the relationships and responsibilities of the SPO and AFPRO for the contract, including a memorandum of agreement, if appropriate.	I	I	I	I	I	I	I

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
24. General review of draft RFP for consideration of all components to ensure enforceability of contract requirements, proper subcontract or management, complete pricing requirements, and consideration of delegation of some monitoring efforts to AFPRO by SPO not already included in the RFP.	I	I	III	III	---	---	III
25. Review of the draft SOW to determine whether the management requirements, software development of criteria, procedures, referenced to CDRLs and specifications, and other consideration provide adequate means to properly monitor the contract.	I	I	III	III	---	---	III
26. Review of the draft CDRLs to determine the effectiveness of the DIDs and detailing of CDRLs provide adequate means to properly monitor the contract.	I	I	III	III	---	---	III

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
27. Review of the draft RFP to determine the reasonableness of the overall time duration; whether the delivery schedule for various plans supports their valid useability; identification of critical needs for long lead times; appropriateness of milestones and sequences; areas of high potential risk to schedule maintenance.	I	I	III	III	---	---	III
28. Review of the draft RFP to determine the presence of appropriate and current versions of standards and specifications and to check tailoring against requirements for valid administration of contract.	I	I	III	III	---	---	III
29. Prepare and review sample contract to ensure inclusion of necessary clauses and forms, deletion of unnecessary forms and clauses, proper cross references to MIL-specs included in SOW/CDRL, and other technical and clerical details.	I	I	III	III	---	---	III

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
48. Review contract as awarded to determine presence of software functional areas such as Work Breakdown Structure, Military Standards and Specifications, Contract Data Requirements Lists, Software Development Plan, Software Configuration Management Plan, Software Quality Assurance Plan.	I	I	III	II	---	---	II
49. Identify and document potential contract administration problems due to terms and conditions of the contract.	I	I	III	II	---	---	II
50. Insure that formal agreements (such as Memorandum of Agreement) exist between SPO and CAS identifying their specific duties and responsibilities concerning software.	I	I	I	I	---	---	I
51. Develop an internal CAS plan for assignment of manpower, test and review participation, inspection points, and communication responsibilities.	I	I	III	III	---	---	III

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
52. Identify and develop plan for desired contract administration activities for subcontracted software.	II	II	---	II	---	---	I
53. Optional SPO and CAS Postaward Orientation Conference of software concerns and responsibilities.	I	I	III	II	---	---	II
54. Ongoing evaluation and review of contractor's management system for software development.	I	I	III	III	---	---	III
55. Monitor critical subcontracts for flow-down of software contract requirements.	II	II	III	III	---	---	I
56. Monitor cost, schedule, and performance of software development tasks.	I	III	III	III	---	---	III
57. Resolve technical and management problems in software development process.	I	II	III	III	---	---	III
58. Review and evaluate software Engineering Change Proposals.	I	III	III	III	---	---	III

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

		ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
59.	Review contract modifications for any possible effects on monitoring software development.	I	II	III	III	---	---	III
60.	Monitor development and deliveries of software Contract Data Requirements List documents.	I	I	III	III	---	---	III
61.	Monitor and review compliance with automated test system standards and documentation.	I	II	III	---	---	---	---
62.	Audit contractor compliance with software requirements for configuration management and quality assurance.	I	I	---	---	---	---	---
63.	Review and report to SPO on contractor's technical planning and management of software development prior to each formal review.	I	III	---	---	---	---	---
64.	Follow up all action items designated by SPO as a result of each formal review.	I	I	---	---	---	---	---

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
67. Assess the contractor's system development tools and management and technical abilities to carry out the necessary development functions.	I	III	---	---	---	---	---
68. Review the Computer Software Configuration Item's functional, performance, data base, qualification, interface, and delivery requirements through the formal Software Specification Review.	I	II	---	---	---	---	---
69. Assess proceeding with contractor's designs for each Computer Software Configuration Item through the formal Preliminary Design Review.	I	I	---	---	---	---	---
71. Monitor software coding phase.	I	II	---	---	---	---	---
72. Monitor software integration and test.	I	II	---	---	---	---	---

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
76. Document the completeness of meeting requirements and specifications, and testing of each Computer Software Configuration Item and Software Product through formal Functional and Physical Configuration Audits.	II	I	---	---	---	---	---
77. Monitor planning and execution of Independent Verification and Validation.	I	II	---	---	---	---	---
78. Monitor development of automated tools used to prepare technical manuals, operating procedures, engineering drawings, etc., to assure compatibility with end-user systems.	I	II	---	---	---	---	---
79. Monitor assurance that configuration management and historical data systems are maintained.	I	II	---	---	---	---	---

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
80. Determine capability of maintenance procedures to human factors and equipment.	I	II	---	---	---	---	---
81. Evaluate training requirements for operation and support of software system.	I	II	---	---	---	---	---
82. Evaluate software system documentation to assure support of postacceptance maintenance.	I	III	---	---	---	---	---
83. Evaluate logistical support to system users for postacceptance maintenance.	I	III	---	---	---	---	---
84. Complete software acceptance procedures.	II	I	---	---	---	---	---

(Continuation)
TABLE OF RESPONSIBILITIES BY TASKS
WITH I or II RANKING IN TABLE FIVE
(See Table Four for meaning of Roman Numerals)

	ENGINEERING	QUALITY ASSURANCE	MANUFACTURING	CONTRACTING	PROGRAM MANAGEMENT SUPPORT	PROPERTY MANAGEMENT	SUBCONTRACTING
85. Review the Configuration Management Plan for postdevelopment software support.	I	III	---	---	---	---	---
86. Review the Software Quality Assurance Plan for postdevelopment software support.	I	I	---	---	---	---	---
87. Review Software Test Plans for postdevelopment software support.	I	I	---	---	---	---	---
88. Determine that proposed engineering changes, deficiencies, and latest defects in software are corrected by the contractor and the Configuration Management Plan adjusted accordingly.	I	II	---	---	---	---	---
89. Establish a Configuration Management Plan for postdevelopment software support.	I	II	---	---	---	---	---

TABLE OF TRAINING TOPIC AREAS FOR ENGINEERING TASKS	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST OEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM OESIGN ANO COOING	2167 AND OTHER REGS ANO STANDARDS
22. Identify contract requirements for performance measurements.		X				X		X
24. General review of draft RFP for consideration of all components to ensure enforceability of contract requirements, proper subcontract or management, complete pricing requirements, and consideration of delegation of some monitoring efforts to AFPRO by SPO not already included in the RFP.	X	X	X	X	X	X	X	X
25. Review of the draft SOW to determine whether the management requirements, software development of criteria, procedures, referenced to CDRLs and specifications, and other consid-eration provide adequate means to properly monitor the contract.		X		X		X		X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND CODING	2167 AND OTHER REGS AND STANDARDS
26. Review of the draft CDRLs to determine the effectiveness of the DIDs and detailing of CDRLs provide adequate means to properly monitor the contract.		X		X		X		X
27. Review of the draft RFP to determine the reasonableness of the overall time duration; whether the delivery schedule for various plans supports their valid useability; identification of critical needs for long lead times; appropriateness of milestones and sequences; areas of high potential risk to schedule maintenance.	X	X		X				X
28. Review of the draft RFP to determine the presence of appropriate and current versions of standards and specifications and to check tailoring against requirements for valid administration of contract.		X	X	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND CODING	2167 AND OTHER REGS AND STANDARDS
29. Prepare and review sample contract to ensure inclusion of necessary clauses and forms, deletion of unnecessary forms and clauses, proper cross references to MIL-specs included in SOW/CDRL, and other technical and clerical details.	X	X	X	X	X	X	X	X
48. Review contract as awarded to determine presence of software functional areas such as Work Breakdown Structure, Military Standards and Specifications, Contract Data Requirements Lists, Software Development Plan, Software Configuration Management Plan, Software Quality Assurance Plan.		X		X	X			X
49. Identify and document potential contract administration problems due to terms and conditions of the contract.	X	X	X	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM OEVELOPMENT TECHNIQUES	SOFTWARE TEST OEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM OESIGN ANO COOING	2167 AND OTHER REGS ANO STANOAROS
51. Develop an internal CAS plan for assignment of manpower, test and review participation, inspection points, and communication responsi-bilities.		X		X	X	X		X
54. Ongoing evaluation and review of contractor's management system for software development.				X	X			X
55. Monitor critical subcontracts for flow-down of software contract requirements.		X		X	X			X
56. Monitor cost, schedule, and perform-ance of software development tasks.	X	X		X				X
57. Resolve technical and management problems in software development process.				X	X			X
58. Review and evaluate software Engineering Change Proposals.	X	X		X	X			X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL ANO QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM OEVELOPMENT TECHNIQUES	SOFTWARE TEST OEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM OESIGN ANO COOING	2167 ANO OTHER REGS ANO STANOAROS
59. Review contract modifications for any possible effects on monitoring software development.	X	X		X	X			X
60. Monitor development and deliveries of software Contract Data Requirements List documents.	X	X		X	X			X
61. Monitor and review compliance with automated test system standards and documentation.		X		X	X			X
62. Audit contractor compliance with software requirements for configuration management and quality assurance.				X	X			X
63. Review and report to SPO on contractor's technical planning and management of software development prior to each formal review.		X	X	X	X	X		X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND CODING	2167 AND OTHER REGS AND STANDARDS
64. Follow up all action items designated by SPO as a result of each formal review.	X	X	X	X	X	X	X	X
67. Assess the contractor's system development tools and management and technical abilities to carry out the necessary development functions.	X	X	X	X	X	X	X	X
68. Review the Computer Software Configuration Item's functional, performance, data base, qualification, interface, and delivery requirements through the formal Software Specification Review.	X	X	X	X	X	X		X
69. Assess proceeding with contractor's designs for each Computer Software Configuration Item through the formal Preliminary Design Review.	X	X	X	X	X	X		X
71. Monitor software coding phase.	X			X			X	X

(Continuation) TABLE OF TRAINING TOPIC AREAS FOR ENGINEERING TASKS	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND CODING	2167 AND OTHER REGS AND STANDARDS
72. Monitor software integration and test.			X	X	X	X		X
76. Document the completeness of meeting requirements and specifications, and testing of each Computer Software Configuration Item and Software Product through formal Functional and Physical Configuration Audits.		X	X	X	X	X		X
77. Monitor planning and execution of Independent Verification and Valid-ation.		X	X	X	X	X		X
78. Monitor development of automated tools used to prepare technical manuals, operating procedures, engineering drawings, etc., to assure compatibility with end-user systems.	X	X			X			X
79. Monitor assurance that configuration management and historical data systems are maintained.				X	X			X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND CODING	2167 AND OTHER REGS AND STANDARDS
80. Determine capability of maintenance procedures to human factors and equipment.	X	X		X	X			X
81. Evaluate training requirements for operation and support of software system.		X	X	X	X			X
82. Evaluate software system document-ation to assure support of post-acceptance maintenance.				X	X			X
83. Evaluate logistical support to system users for postacceptance maintenance.	X			X	X			X
84. Complete software acceptance pro-cedures.	X	X	X	X	X	X	X	X
85. Review the Configuration Management Plan for postdevelopment software support.				X	X		X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR ENGINEERING TASKS

	TECHNICAL COST ASPECTS OF SOFTWARE PLANNING	TECHNICAL AND QUALITY FACTORS IN REQUIREMENTS ANALYSIS	SOFTWARE SPECIFICATION TECHNIQUES	ENGINEERING MANAGEMENT IN THE SOFTWARE LIFE CYCLE	SOFTWARE SYSTEM DEVELOPMENT TECHNIQUES	SOFTWARE TEST DEVELOPMENT TECHNIQUES	SOFTWARE PROGRAM DESIGN AND COOING	2167 AND OTHER REGS AND STANDARDS
86. Review the Software Quality Assurance Plan for postdevelopment software support.		X		X	X	X	X	X
87. Review Software Test Plans for post-development software support.				X		X		X
88. Determine that proposed engineering changes, deficiencies, and latest defects in software are corrected by the contractor and the Configuration Management Plan adjusted accordingly.				X	X	X		X
89. Establish a Configuration Management Plan for postdevelopment software support.	X	X	X	X	X	X		X

TABLE OF TRAINING TOPIC AREAS FOR QUALITY ASSURANCE	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
22. Identify contract requirements for performance measurements.	X	X	X	X	X
24. General review of draft RFP for consideration of all components to ensure enforceability of contract requirements, proper subcontract or management, complete pricing requirements, and consideration of delegation of some monitoring efforts to AFPRO by SPO not already included in the RFP.	X	X	X	X	X
25. Review of the draft SOW to determine whether the management requirements, software development of criteria, procedures, referenced to CDRLs and specifications, and other consid-eration provide adequate means to properly monitor the contract.	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT & QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
26. Review of the draft CDRLs to determine the effectiveness of the DIDs and detailing of CDRLs provide adequate means to properly monitor the contract.	X	X	X	X	X
27. Review of the draft RFP to determine the reasonableness of the overall time duration; whether the delivery schedule for various plans supports their valid useability; identification of critical needs for long lead times; appropriateness of milestones and sequences; areas of high potential risk to schedule maintenance.	X	X	X	X	X
28. Review of the draft RFP to determine the presence of appropriate and current versions of standards and specifications and to check tailoring against requirements for valid administration of contract.	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

		SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
29.	Prepare and review sample contract to ensure inclusion of necessary clauses and forms, deletion of unnecessary forms and clauses, proper cross references to MIL-specs included in SOW/CDRL, and other technical and clerical details.	X			X	X
51.	Develop an internal CAS plan for assignment of manpower, test and review participation, inspection points, and communication responsibilities.		X	X	X	X
54.	Ongoing evaluation and review of contractor's management system for software development.				X	X
55.	Monitor critical subcontracts for flow-down of software contract requirements.	X	X	X	X	X
56.	Monitor cost, schedule, and performance of software development tasks.	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
57. Resolve technical and management problems in software development process.	X			X	X
59. Review contract modifications for any possible effects on monitoring software development.	X	X	X	X	X
60. Monitor development and deliveries of software Contract Data Requirements List documents.	X	X	X	X	X
61. Monitor and review compliance with automated test system standards and documentation.	X	X	X	X	X
62. Audit contractor compliance with software requirements for configur-ation management and quality assurance.			X		X
64. Follow up all action items designated by SPO as a result of each formal review.	X	X	X	X	X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
68. Review the Computer Software Configuration Item's functional, performance, data base, qualification, interface, and delivery requirements through the formal Software Specification Review.	X	X	X	X	X
69. Assess proceeding with contractor's designs for each Computer Software Configuration Item through the formal Preliminary Design Review.	X	X	X	X	X
71. Monitor software coding phase.				X	X
72. Monitor software integration and test.		X			X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
76. Document the completeness of meeting requirements and specifications, and testing of each Computer Software Configuration Item and Software Product through formal Functional and Physical Configuration Audits.	X	X	X	X	X
77. Monitor planning and execution of Independent Verification and Validation.	X	X			X
78. Monitor development of automated tools used to prepare technical manuals, operating procedures, engineering drawings, etc., to assure compatibility with end-user systems.	X			X	X
79. Monitor assurance that configuration management and historical data systems are maintained.	X		X		X

(Continuation)
TABLE OF TRAINING TOPIC AREAS
FOR QUALITY ASSURANCE

	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
80. Determine capability of maintenance procedures to human factors and equipment.	X			X	X
81. Evaluate training requirements for operation and support of software system.	X			X	X
84. Complete software acceptance procedures.	X	X	X	X	X
86. Review the Software Quality Assurance Plan for postdevelopment software support.	X				X
87. Review Software Test Plans for post-development software support.	X	X			X
88. Determine that proposed engineering changes, deficiencies, and latest defects in software are corrected by the contractor and the Configuration Management Plan adjusted accordingly.	X		X		X

(Continuation) TABLE OF TRAINING TOPIC AREAS FOR QUALITY ASSURANCE	SOFTWARE QUALITY ASSURANCE PLANS AND DEVELOPMENT	SOFTWARE QUALITY ASSURANCE IN TEST PLAN AUDITING AND TEST EXECUTION AUDITING	SOFTWARE CONFIGURATION MANAGEMENT AUDITING	SOFTWARE DEVELOPMENT QUALITY ASSURANCE STANDARDS	MIL-STD-2167 AND OTHER REGS, ETC.
89. Establish a Configuration Management Plan for postdevelopment software support.	X		X		X

APPENDIX D

INDIVIDUALS INTERVIEWED IN THE STUDY

Lt. Arnold
Information Systems
B.M.C., Norton AFB

Don Brautigam, Industrial Engineer
MCCR Rep., Production Operations Division
Rockwell-Anaheim AFPRO

Lt. Crawford
Information Systems and Computer Resources
B.M.C., Norton AFB

Tom Ferris
QA Specialist, QAX
Rockwell-Anaheim AFPRO

Bernard Katz
Aerospace Corporation
DSP Program Office
Space Division

Maj. Mitchell
Information Systems
B.M.C., Norton AFB

Myron Pidhany, Electrical Engineer
MCCR Focal Point, EPP
Rockwell-Anaheim AFPRO

Shiguru Shozi
QA Manager for Configuration Mgmt.
DSP Program Office
Space Division

Janice Smith
Deputy Director, Mission Control
Mil-Star Program Office
Space Division

Leona Solar
ACO/TMD, Contracts Division
Rockwell-Anaheim AFPRO

Lt. Col. G. Stojanowsky
Information Systems and Computer Resources
B.M.C., Norton AFB

Mark L'Ecuyer
Mfg. Engineer/MCCR Focal Point for PD
AVCO AFPRO

Captain Jeffrey Ford
MCCR Focal Point
Lockheed AFPRO

Gerald Conroy
Chief, Pricing
Software Evaluation Team Member
Lockheed AFPRO

Captain Ruben Lopez
Chief, Subcontract Division
Software Evaluation Team Member
Lockheed AFPRO

Don Lewis
Industrial Engineer
Software Evaluation Team Member
Lockheed AFPRO

Colonel Bob Peters
EP Division Chief
Lockheed AFPRO

Captain Chuck Davidson
Computer Resource Acquisition Engineer
Eglin AFB

Major Bill Miller
Systems and Integration Branch Chief/EN
General Dynamics AFPRO

Gary Gaston
Electronics Engineer/Program Manager
General Dynamics AFPRO

Jim McQuaid
QA Software/Electronics Specialist
General Dynamics AFPRO

Sam Horstman
QA Technical Section Chief
General Dynamics AFPRO

Bill Turner
QA Software Specialist
General Dynamics AFPRO

Lynn Jones
QA Software Specialist/Focal Point
General Dynamics AFPRO

Captain Terri Fox Daeke
SPO Software Support
Hanscomb AFB

Steve O'Shaughnessy
Mitre Corporation
SPO Software Manager/Peace Shield

Lieutenant John Gann
MCCR Software Focal Point
AVCO AFPRO

James Wheeler
QA Software Specialist
AVCO AFPRO

Muffy Staffieri
MCCR Software Focal Point for Subcontracts
AVCO AFPRO

Maj. Stroud
Information Systems
B.M.C., Norton AFB

Darrah Whitlock
QA Specialist, Plans & Eval. Branch
Rockwell-Anaheim AFPRO

Lt. Col. Barry Prins
HQ/AFCMD
Kirtland AFB

Stan Brown
HQ/AFCMD
Kirtland AFB

Mike Bates
HQ/AFCMD
Kirtland AFB

Col. Martin
HQ/AFCMD
Kirtland AFB

Col. Williams
HQ/AFCMD
Kirtland AFB

Lt. Col. Ed Stevens
Space Division

Donna Mazzanti
HQ/AFCMD
Kirtland AFB

Dennis Drouilland
HQ/AFCMD
Kirtland AFB

Philip Babel
ASD
Wright-Patterson AFB

Capt. Dennis Smith
AFBRMC
Wright-Patterson AFB

Lowell Simon
Martin-Marietta AFPRO
Denver

Al Chmara
Martin-Marietta AFPRO
Denver

Andy Serino
Martin-Marietta AFPRO
Denver

Ken Hackett
Martin-Marietta AFPRO
Denver

Maj Hutchinson
HQ/AFCMD
Kirtland AFB