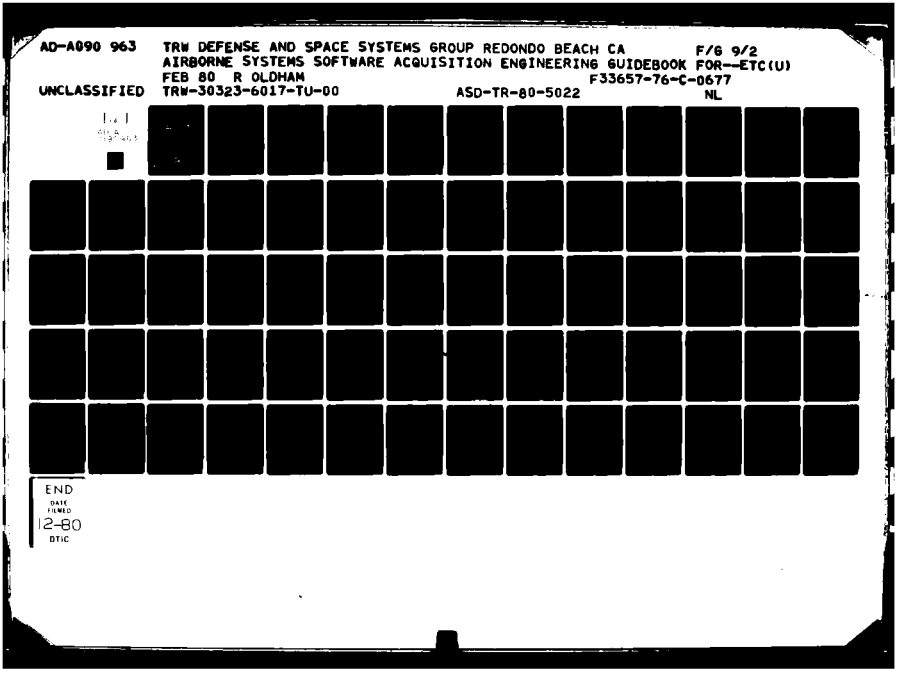
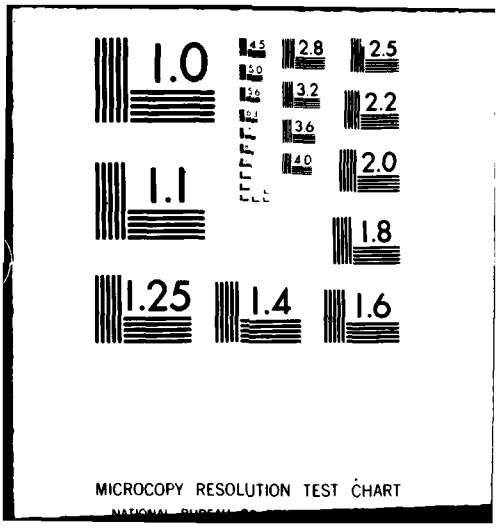

AD-A090 963
TRW DEFENSE AND SPACE SYSTEMS GROUP REDONDO BEACH CA
F/G 9/2
AIRBORNE SYSTEMS SOFTWARE ACQUISITION ENGINEERING GUIDEBOOK FOR--ETC(U)
FEB 80 R OLDHAM
F33657-76-C-0677
UNCLASSIFIED
TRW-30323-6017-TU-00
ASD-TR-80-5022
NL
END
DATE FILMED
12-80
DTIC


1.0
1.1
1.25
1.4
1.6
1.8
2.0
2.2
2.5
2.8
3.2
3.6
4.0
4.5
5.0
5.6
MICROCOPY RESOLUTION TEST CHART

ASD-TR-80-5022

(1)

Airborne Systems Software Acquisition Engineering Guidebook for SOFTWARE DEVELOPMENT PLANNING AND CONTROL

FEBRUARY 1980

PREPARED FOR
DEPUTY FOR ENGINEERING
AERONAUTICAL SYSTEMS DIVISION
WRIGHT-PATTERSON AFB, OH 45433

PREPARED BY
TRW DEFENSE AND SPACE SYSTEMS GROUP
ONE SPACE PARK
REDONDO BEACH, CA 90278

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the Office of Public Affairs (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public, including foreign nations.

This technical report has been reviewed and is approved for publication.

JOHN M. HOEFERLIN, Project Engineer
Information Engineering Division

RICHARD J. SYLVESTER
ASD Computer Resources Focal Point

FOR THE COMMANDER

ROBERT P. LAVOIE, Colonel, USAF
Director of Avionics Engineering
Deputy for Engineering

"If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization please notify ASD/ENAIA, W-PAFB, OH 45433 to help us maintain a current mailing list".

Copies of this report should not be returned unless return is required by security considerations, contractual obligations, or notice on a specific document.

AIR FORCE/56780/30 July 1980 — 350

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE	READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER	2. GOVT ACCESSION NO. AD-A090 963 / 3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) (6) Airborne Systems Software Acquisition Engineering Guidebook for Software Development Planning and Control	5. TYPE OF REPORT & PERIOD COVERED (9) Final Rept.
	6. PERFORMING ORG. REPORT NUMBER TRW-30323-6017-TU-00
7. AUTHOR(s) (10) R. Oldham (12) 67 (14) TRW	8. CONTRACT OR GRANT NUMBER(s) (15) F33657-76-C-0677
9. PERFORMING ORGANIZATION NAME AND ADDRESS TRW Defense and Space Systems Group, One Space Park, Redondo Beach, Ca 90278 (11) Feb 80	10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE64740F (16) Project 2238
11. CONTROLLING OFFICE NAME AND ADDRESS HQ ASD/ENAI, Wright-Patterson AFB, OH 45433	12. REPORT DATE February 1980
	13. NUMBER OF PAGES 72
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office) (18) ASD (19) TR-80-5022	15. SECURITY CLASS. (of this report) Unclassified
	15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Approved for public release, distribution unlimited

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

18. SUPPLEMENTARY NOTES

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

Plans	Software
Program Management	Software Development
Program Control	Software Acquisition
Planning	Guidebook
Program Office	Systems Software

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

This report is one of a series of guidebooks whose purpose is to assist Air Force Program Office and Engineering personnel in the acquisition and engineering of airborne systems software. This guidebook discusses the documents, methods and procedures for use in planning, monitoring, controlling and reporting weapon system acquisitions involving software development. It emphasizes the importance of planning and discusses the dynamics of the planning process along with the associated planning

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

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

409637

Block 20 Continued

documents. It describes the role, use and relationship of the Program Management Plan, Procurement Plan, Test and Evaluation Master Plan, Computer Program Development Plan, Configuration Management Plan, Quality Assurance Plan, Computer Resources Integrated Support Plan and the Operational/ Support Configuration Management Procedures. It stresses that plans are the basis for monitoring and controlling software development programs.

PREFACE

This guidebook is one of a series of guidebooks intended to assist Air Force Program Office and Engineering personnel in software acquisition engineering for airborne systems. The contents of the guidebooks will be revised periodically to reflect changes in software acquisition policies and practices and feedback from users.

This guidebook was prepared under the direction of the Aeronautical Systems Division, Deputy for Engineering (ASD/EN) in coordination with the Space Division, Deputy for Acquisition Management (SD/AQM).

The entire series of Software Acquisition Engineering Guidebooks (Airborne Systems) is listed below along with ASD Technical Report numbers and NTIS accession numbers where available.

Regulations, Specifications and Standards	ASD-TR-78-6	ADA058428
Reviews and Audits	ASD-TR-78-7	ADA058429
Software Quality Assurance	ASD-TR-78-8	ADA059068
Configuration Management	ASD-TR-79-5024	ADA076542
Computer Program Documentation Requirements	ASD-TR-79-5025	ADA076543
Statements of Work and Requests for Proposal	ASD-TR-79-5026	ADA076544
Requirements Analysis and Specification	ASD-TR-79-5027	
Verification, Validation and Certification	ASD-TR-79-5028	
Microprocessors and Firmware	ASD-TR-80-5021	
Software Development Planning and Control	ASD-TR-80-5022	
Software Testing and Evaluation	ASD-TR-80-5023	
* Contracting for Software Acquisition	ASD-TR-80-5024	

* Software Cost Analysis and Estimating	ASD-TR-80-5025
* Supportable Airborne Software	ASD-TR-80-5026
* Software Development and Support Facilities	ASD-TR-80-5027
* SAE Guidebooks - Application and Use	ASD-TR-80-5028

* These Guidebooks Available Fall 1980.

CONTENTS

CONTENTS (Concluded)

ILLUSTRATIONS

TABLE

ABBREVIATIONS AND ACRONYMS

AFM	Air Force Manual
AFR	Air Force Regulation
AFPRO	Air Force Plant Representative Office
AFTEC	Air Force Test and Evaluation Center
AISF	Avionics Integration Support Facility
AMSDL	Acquisition Management Systems and Data Requirements Control List
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CM	Configuration Management
CMP	Configuration Management Plan
CPC	Computer Program Component
CPCI	Computer Program Configuration Item
CPDP	Computer Program Development Plan
CRISP	Computer Resources Integrated Support Plan
CRWG	Computer Resources Working Group
C/SCSC	Cost/Schedule Control Systems Criteria
C/SSR	Cost/Schedule Status Report
DAR	Defense Acquisition Regulation
DCAS	Defense Contract Administration Services
DCP	Decision Coordinating Paper
DID	Data Item Description
DOD	Department of Defense
DODD	Department of Defense Directive
DSARC	Defense System Acquisition Review Council
DT&E	Development Test and Evaluation
FYDP	Five Year Defense Plan
GFE	Government Furnished Equipment
HOL	Higher Order Language
IOT&E	Initial Operational Test and Evaluation
IV&V	Independent Verification and Validation
OS	Operating System
O/SCMP	Operational/Support Configuration Management Procedures

ABBREVIATIONS AND ACRONYMS (Concluded)

OT&E	Operational Test and Evaluation
PDR	Preliminary Design Review
PERT	Program Evaluation Review Technique
PM	Program Management
PMD	Program Management Directive
PMP	Program Management Plan
PMRT	Program Management Responsibility Transfer
PO	Program Office
PP	Procurement Plan
QAP	Quality Assurance Plan
R&D	Research and Development
SDR	System Design Review
SRR	Software Requirements Review
TDM	Technical Direction Meeting
TEMP	Test and Evaluation Master Plan
TEOA	Test and Evaluation Objectives Annex
TIM	Technical Interchange Meeting
TRR	Test Readiness Review
V&V	Verification and Validation
WBS	Work Breakdown Structure

1. INTRODUCTION

1.1 PURPOSE AND SCOPE

The purpose of this guidebook is to identify documentations, methods, and procedures for use in planning, monitoring, controlling, and reporting software development.

Extensive documentation exists on software development planning and status; however, it is the goal of this guidebook to reduce volumes of data to a more concise, useful format in the context of avionics and space-borne software development efforts. The procedures and methods described herein help to identify the types of information pertinent to a project and how to use them in emphasizing initial planning and determining status of the software development.

1.2 RELATIONSHIPS

1.2.1 Life Cycle Relationships

Software development planning begins with initial focus on producing a system involving computer control or utilization and continues until the system is completely developed. This guidebook primarily applies to those computer programs developed within the normal weapon system life cycle as shown in Figure 1-1. However, certain software acquisitions may occur within only one life cycle phase (of the system) while others may span several phases. Monitoring and control procedures must be adjusted accordingly.

1.2.2 Relationship to Other Guidebooks

The degree of adherence to methods and procedures outlined in the other guidebooks directly correlates to the software development progress. Proper configuration management or proper quality control have intangible payoffs that obviously affect development progress. Absence of such disciplines causes a severely detrimental effect on progress yet, to the uninformed, enforcement of these disciplines seems to yield little except increased costs. This guidebook supplements those emphases made in other series guidebooks.

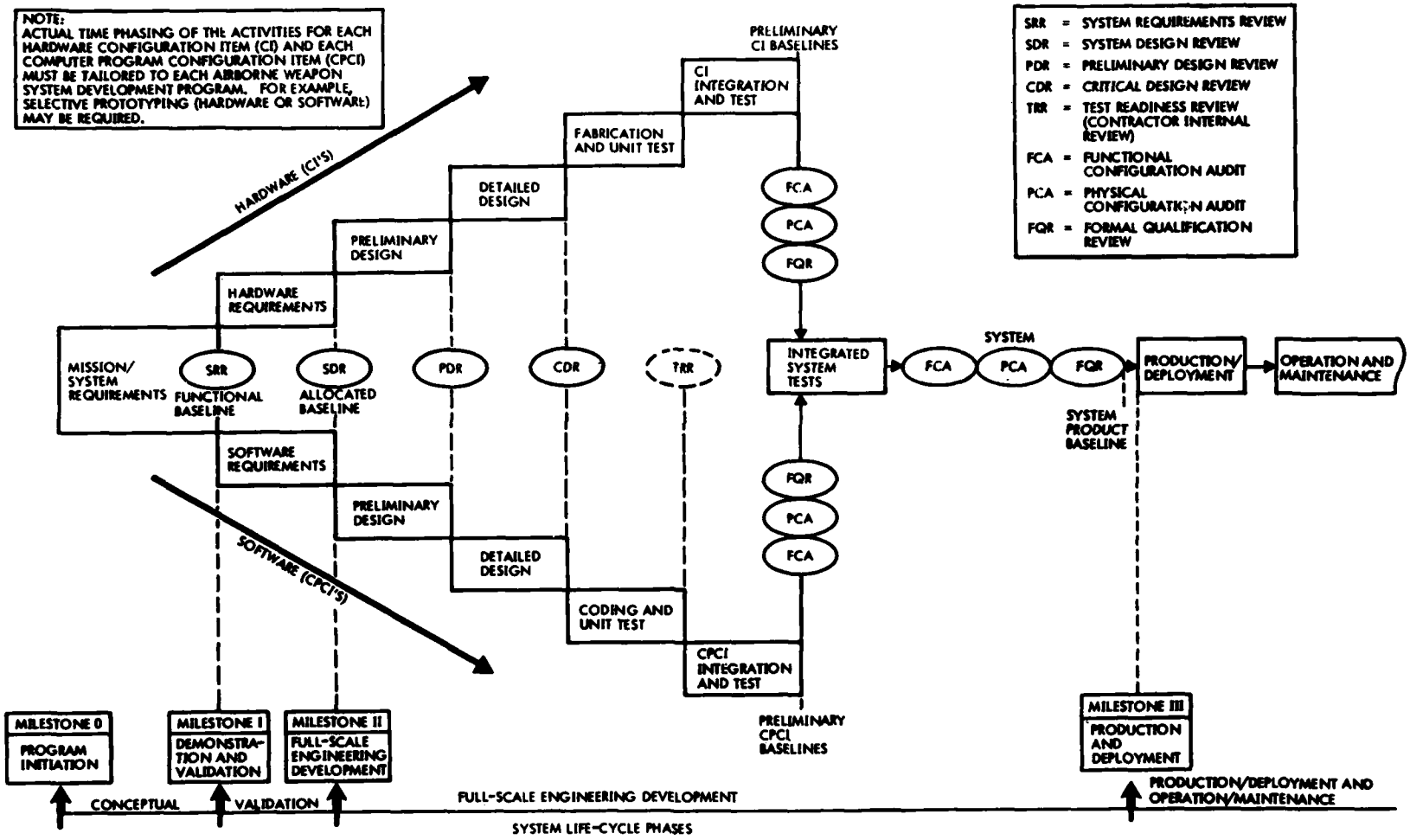


Figure 1-1. Idealized System Life Cycle

1.3 CONTENTS OF THIS GUIDEBOOK

1.3.1 Section 1: Introduction

Provides background information on the use of software development planning and status as they apply to the system life cycle and to other guidebooks published in this series.

1.3.2 Section 2: Relevant Documents

References the government regulations and standards which relate to software development planning and status reporting.

1.3.3 Section 3: General Guidelines for Software Development Planning and Control

Provides general guidance to the Program Office engineers and managers in the planning for software development. It further addresses monitoring, controlling, and reporting software development progress, techniques, and methodology currently in use, factors which affect development progress and associated risks.

1.3.4 Section 4: Implementing, Monitoring, and Reporting Development Status

Provides guidance on software development monitoring, controlling, and reporting. It provides discussion on milestones and schedules, reviews, and development status reporting.

1.3.5 Appendices A Through E

Provides additional information on planning documents and the evaluation of them.

1.3.6 Appendix F. Annotated Bibliography

Lists a number of references that provide information on various subjects related to software development, status, and test.

2. RELEVANT DOCUMENTS

The following documents are important sources of information relevant to software development planning and control. Additional sources of background information related to software development and control are provided in Appendix D.

DODD 4120.3	Defense Standardization Program
DODD 5000.1	Major System Acquisition
DODD 5000.2	Major System Acquisition Process
DODD 5000.11	Data Elements and Data Codes Standardization Procedures
DODD 5000.29	Management of Computer Resources in Major Defense Systems
AFM 26-1	Manpower Policies and Procedures
AFR 74-1	Quality Assurance
AFR 80-14	Test and Evaluation
AFR 300-10	Computer Programming Languages
AFR 310-1	Management of Contractor Data
AFR 800-2	Program Management
AFR 800-3	Engineering for Defense Systems
AFR 800-8	Integrated Logistics Support (ILS) Program for Systems and Equipment
AFR 800-11	Life Cycle Cost Management Program
AFR 800-12	Acquisition of Support Equipment
AFR 800-14	Management of Computer Resources in Major Defense Systems (Volumes I and II)
AFR 800-28	Air Force Policy in Avionics Acquisition and Support
AFSCM 375-7	Configuration Management for Computer Programs
AFSCR 800-1	Command Review of Systems Acquisition Programs and Test Resources

DAR 1-2102	Procurement Planning
MIL-STD 483	Configuration Management
MIL-STD 490	Specification Practices
MIL-STD 499A	Engineering Management
MIL-STD 1521A	Technical Reviews and Audits for Systems, Equipment, and Computer Programs
MIL-S 52779A	Quality Assurance
SAMSO STANDARD 73-3	Standard Engineering Practices for Computer Software Design and Development

DAR 1-2102	Procurement Planning
MIL-STD 483	Configuration Management
MIL-STD 490	Specification Practices
MIL-STD 499A	Engineering Management
MIL-STD 1521A	Technical Reviews and Audits for Systems, Equipment, and Computer Programs
MIL-S 52779A	Quality Assurance
SAMSO STANDARD 73-3	Standard Engineering Practices for Computer Software Design and Development

3. GENERAL GUIDELINES FOR SOFTWARE DEVELOPMENT PLANNING AND CONTROL

Air Force Regulation 800-2 requires each acquisition program to be managed by a single person known as the Program Manager (PM). This guidebook provides information to the PM or his representatives for planning and controlling software development pertinent to his acquisition. Control of software development is possible only to a level equivalent to the planning level associated with the software development. Top level project planning will enable only top level project control even though detailed control procedures (e.g., software control) are attempted. Since successful development control is highly dependent on adequate development planning, this section emphasizes the planning documents which facilitate software development allowing subsequent development control. Table 3-1 shows the relationship between various planning documents discussed in this section.

3.1 SOFTWARE ACQUISITION MANAGEMENT PLANS

Planning is a prerequisite to efficiently accomplishing any sophisticated task. Since it is nearly impossible to devise a plan with 100% accuracy and efficiency, initial planning will only approximate the actual situation. Experienced managers lay out a planned course of action based upon current data and, as the data changes, change the plan accordingly; however, a change is not adopted unless some "improved" data are perceived. This section assists in initially laying out a software project within a larger System Acquisition Program. The topics discussed represent coverage of the prime concerns in initial planning yet, by no means, cover all planning aspects.

3.1.1 Procurement Plan

One of the major initial tasks in any program is to develop an acquisition strategy for the total system. Technical, business, and management areas are addressed to provide a basis for integration of these areas to achieve program objectives. The strategy is expanded and refined as the program progresses. Schedules and funding plans are prepared to accomodate areas of uncertainty and risk. Usually, the system requirements

Table 3-1. Software Development Planning Documents

Document	Usual Developer	Type		Requirement Documents
		Mgmt	Tech	
Program Management Plan	Program Office	X		AFR 800-2
Procurement Plan	Program Office	X		DAR 1-2102
Test and Evaluation Master Plan	Program Office	X		AFR 80-14
Computer Program Development Plan	Contractor	X	X	AFR 800-14
Configuration Management Plan	Contractor	X		MIL-STD 483 Appendix I
Quality Assurance Plan	Contractor	X		AFR 74-1 and MIL-S 52779A
Computer Resources Integrated Support Plan	CRWG	X	X	AFR 800-14
Operational/Support Configuration Management Procedures	User/ Supporter	X		AFR 800-14

are defined only in operational terms, software requirements are vaguely stated, and management control is not yet formalized. However, for budgeting purposes, in-depth estimates to the DOD funding cycle are necessary. Although project uncertainty exists, the Procurement Plan (PP) outlines an initial course of action which will influence the entire system development.

3.1.1.1 Software Considerations in the Procurement Plan

The basic objective of the Procurement Plan is to facilitate attainment of the procurement objectives. A PP is prepared concurrently with the request for program funding and provides a matrix for the integration and coordination of personnel engaged in the determination of requirements, development of a technical data package, funding, contracting, and contract administration. Software is usually intertwined into all of these activities and can provide a substantial influence on developments costs. At a minimum, software related questions such as the following should be addressed

1. Will the software be a separate procurement from the rest of the system?
2. Will the software effort involve a source selection process?
3. How will selection criteria be established?
4. Who will govern the source selection and publish its procedures and proceedings?
5. Will the software be accomplished by multiple contractors?
6. Will the procurement involve independent verification and validation?
7. How will the lead contractor be established?
8. Will the Air Force organically develop any portions of the software?
9. What type of contract(s) is required?
10. Are data rights expected to be a factor?
11. What technical and/or schedule risks are evident?

Appendix A contains a list of information required by DAR 1-2102 to be included in the Procurement Plan.

3.1.1.2 Software Development/Visibility Through Use of Procurement Plan

Direct control of software development is somewhat ineffective when using only those items contained within the PP; however, at this point, the acquisition method of developing software has already been determined. Technical and schedule risks have been estimated or flagged as problem areas so management adjustments can be made accordingly. In summary, the PP has defined the acquisition strategy at a top level so control can be exerted at a commensurate level.

3.1.2 Program Management Plan

Air Force Regulation 800-2 requires the Program Manager to prepare a Program Management Plan (PMP) tailored to the needs of the program. The PMP is a directive to all participating commands and, although the direct responsibility of the Program Manager, is jointly prepared and coordinated with these commands. Updates to the PMP should be made whenever significant program changes occur or the data contained are obsolete.

3.1.2.1 Software Considerations of the Program Management Plan

The PMP is developed and issued by the Program Manager to show how the tasks described in the Program Management Directive (PMD) will be accomplished. It includes complete planning for the acquisition management of the computer resources. Specifically, AFR 800-14 Volume II requires coverage of the following, when applicable:

a. Operational and support concepts for computer programs.

b. Identification of technical and managerial expertise allocated to the program office.

c. Systems Engineering approach to be followed.

d. Coverage of appropriate hardware tradeoffs.

e. Standardization and commonality considerations.

f. Requirements for computer program and data rights consistent with the planned operational and support concepts.

g. Master schedule of major milestones.

h. Identification of required interfaces between this system and other systems.

i. Requirements for growth capability.

j. Requirements for acquisition and support of documentation.

k. Requirements for simulation, integration, and other facilities necessary to support computer programs.

l. Configuration management concepts.

m. Program Management Responsibility Transfer criteria.

n. Preparation of a Computer Resources Integrated Support Plan (CRISP).

Once established, the PMP serves as the basic overall planning document for guiding management and technical exchanges below the Air Staff level. Command responsibilities are established in conjunction with projected milestones enabling intra-Command planning to proceed at the next lower level. The PMP should be kept current through regular updates, or if necessary, to reflect actual program changes.

The following sequence of steps generally describes the generation of a PMP for a system, and in addition, points out some of the software unique concerns.

- Step 1 - Identify a single focal point. Whether the Program Manager personally spearheads the PMP development or delegates this responsibility, it is important that a single focal point be designated for continuity and coordination purposes.

- Step 2 - Identify any constraints. A compressed schedule, inadequate funds, use of a particular contractor, or an intense operational need are the kinds of constraints which drive programs. These can lead to eliminating and/or overlapping phases in the system life cycle. Constant management attention is necessary to counterbalance any of these constraints.

- Step 3 - Scope the project. It is necessary to scope any task (especially software) to adequately plan its development. A reasonable margin for error should be allowed in the scoping and can be included in both cost and schedule estimates. Con-

sidering any identified constraints, provide a generalized estimate of the job broken out by desired category (hardware, software, Computer Program Configuration Item (CPCI), or milestone). Use experience factors gained from previous successful programs of comparable complexity realizing that cost and/or schedule revisions will be likely as improved data are received. Consider interfaces with other systems and the relative stability of those systems. The results of this step should broadly scope the task, by category, and will serve as the initial planning estimates for the PMP. During the early phases of avionics projects, scoping of the subsystems may be a combined hardware/software estimate. Detailed contractor trades will ultimately define the hardware/software dividing line.

- Step 4 - Conceive an overall project approach. Operational needs have provided the basis for system requirements; these requirements should be used to specify a sequence of steps by which the system will be developed, for example:

    a. Hardware and software to be developed in parallel by one contractor.
    b. System will consist of four CPCI's.
    c. Joint test program using AFTEC, user, and supporting commands.
    d. Organic support during operational use.

- Step 5 - Develop a Master Schedule. Combining Steps 3 and 4 should provide a milestone schedule. If risks are substantial, then factor these in as management judgments.

- Step 6 - Identify resources. Standard personnel and equipment productivity factors applied to Step 5 should yield an indication of the recources necessary to accomplish the development. Forecast the source and date of procuring these resources. Identify expected tasks and schedules for the participating commands.

3.1.2.2 Development Visibility/Control Through Use of the Program Management Plan

The Program Management Plan primarily provides management control of the system development including software. Top level approaches to system engineering, schedule control, configuration management concept, and methods of engineering and status data acquisition are defined in the PMP. The PMP further outlines the areas of responsibility for all Air Force and contractor participants. Additionally, the document outlines the method by which system and/or software tradeoffs may be accomplished. The PMP schedules establish estimated dates for which the control documents such as the Computer Program Development Plan (CPDP) and the Configuration Management Plan (CMP) will be produced. In summary, completion of the PMP allows the orderly commencement of the development effort, and implementation of the PMP allows continued orderly development.

3.1.3 Test and Evaluation Master Plan

As early as possible, management should conceptualize and scope the testing to be accomplished. The test section of the Program Management Plan contains the generalized test objectives to be fulfilled; however, an overall test and evaluation plan to identify and integrate the test effort and schedules is also needed. AFR 80-14 defines such a plan and labels it the Test and Evaluation Master Plan (TEMP). The TEMP should be generated prior to full-scale engineering development and coordinated with AFTEC and other major commands as appropriate. All changes to the TEMP should require formal approval and should be accompanied by rationale for the changes. The document must be kept current for all key test and evaluation decision points.

3.1.3.1 Software Considerations of the Test and Evaluation Master Plan

The TEMP will identify all system testing to be accomplished. Software testing (such as exercises, simulations, etc.) must be conceived and integrated into the overall plan. Each test effort and its associated schedule must be listed. All test support agencies must be contacted with a statement of expected test participation and support. The type of

tests and what is to be tested must be identified. A description of management and control procedures related to testing and evaluation must be provided. Figure 3-2 represents a hierarchical flow for the various test documents.

3.1.3.2 Development Visibility/Control Through Use of the Test and Evaluation Master Plan

The Test and Evaluation Master Plan (TEMP) should insure that management responsibilities and control mechanisms have been outlined for any necessary software testing as well as system testing. This document will indicate any test schedule or integration conflicts and thus the conflicts may be resolved early in the development cycle. Early indications of problem areas, whether of technical origin or schedule/integration origin, serve to minimize development costs. The TEMP outlines the overall approach to testing and thus provides a baseline against which project management may control the test portion of software development.

3.1.4 Computer Resources Integrated Support Plan

Section 3.3.1 of this guidebook specifically addresses the Computer Resources Integrated Support Plan (CRISP) document and its preparation. This section focuses on the usefulness of the CRISP to early project planning.

Assurance of adequate support of computer resources after management responsibility has transferred to the supporting command is the main purpose of the CRISP. Early project planning must include provision for acquiring adequate documents, computer equipment, and computer programs necessary to provide support. Definition and possible procurement of these equipments, facilities, and software are the responsibility of the Program Office (to the extent specified in the PMD) and must be factored into early planning. Many times a support capability will evolve during the software development. Planning should allow transfer of such a capability to the supporting command prior to program management responsibility transfer. The CRISP is an appropriate document for addressing this or similar items.

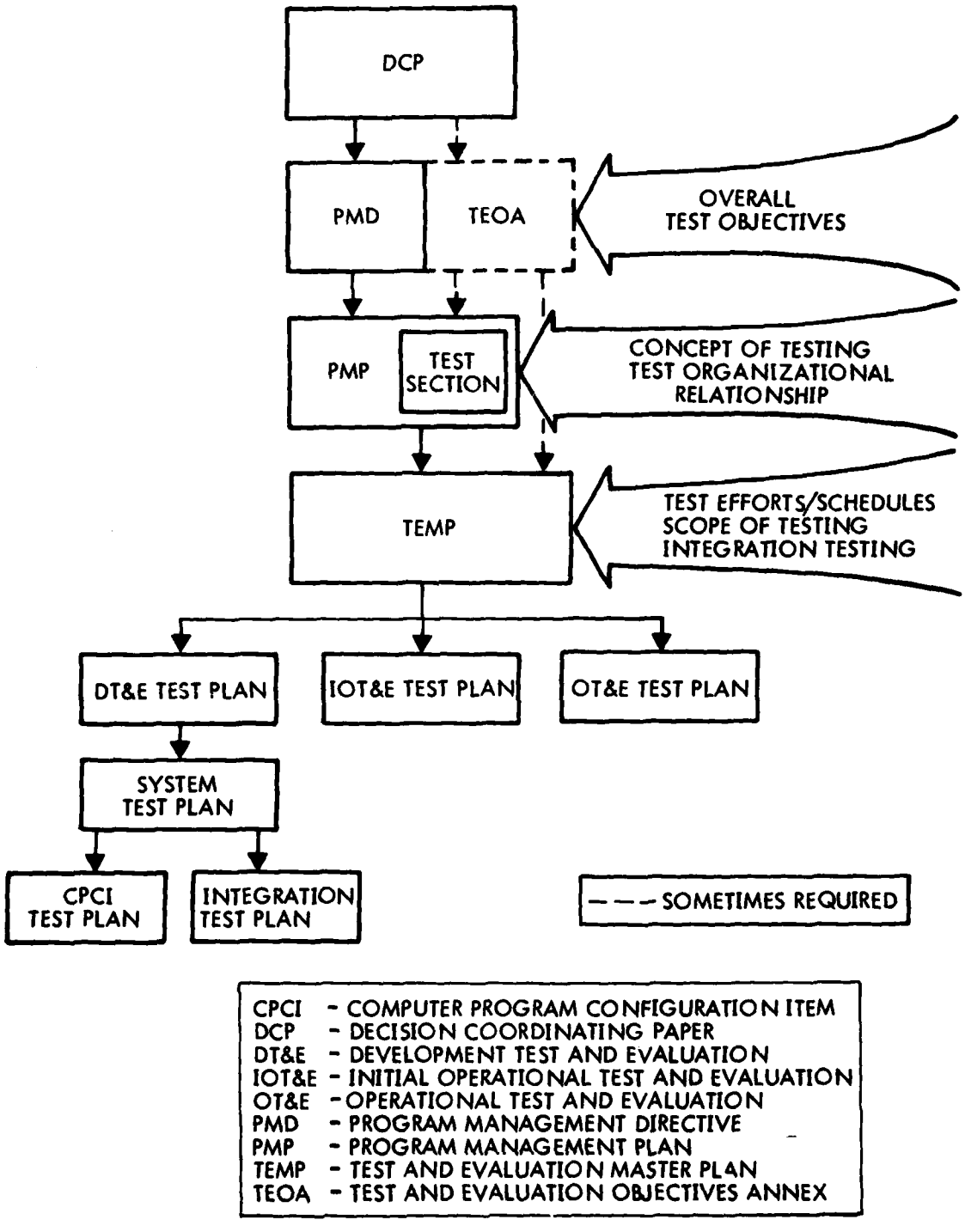


Figure 3-2. Test Plans Hierarchy

3.2 SOFTWARE DEVELOPMENT PLANS

Section 3.1 addressed the plans which outline the management control of an acquisition involving software development. This section covers the controls, disciplines, and tests that pertain directly to development of the software. The development plans must interface and complement the management control plans, but are normally contractor generated in coordination with the Air Force implementing agency.

3.2.1 Computer Program Development Plan

Attention must be given to the actual controls and mechanisms necessary to generate software programs. The Computer Program Development Plan (CPDP) is commonly used to provide this attention. The CPDP should be submitted as a part of the precontractual response to the Request for Proposal. Submission of this information should enable the government agency to gain a broad insight of the bidder's capability to produce the software. Even though a CPDP cannot be prepared completely as part of a contractor's proposal because the system definition has not progressed sufficiently, the fundamental approach to computer program development should be documented and the CPDP updated at some later time. Specific topics to be contained within the CPDP are listed in AFR 800-14 and Appendix B of this document. Data Item Description (DID) DI-S-30567A contains a further expanded list of topics to be addressed by the CPDP and the DID is included in its entirety in Appendix D.

Some additional topics which should be considered are:

1. What is the order of module development?
2. What are the interdependencies of the modules?
3. Are simulations or emulations necessary?
4. How are user manuals and positional handbooks to be evaluated?
5. Are the detailed integration and tests of related hardware elements and modules adequately planned?

Certain software developments require establishment of development facilities to house required equipment and personnel. Specific planning attention must be devoted to acquiring any associated long-lead items,

site construction, and/or concept of operation. In particular, developments which use Government Furnished Equipment (GFE) or government controlled buildings require advanced coordination and planning.

Another important planning/control consideration is the amount and types of software documentation to be produced. Documentation is needed during development to track progress and provide information for management visibility and decision making. Subsequently, it is needed to enable proper life cycle support and maintenance of computer resources. The CPDP should describe how the developing documentation will be generated.

During and subsequent to software development, the use of software will uncover discrepancies and desired improvements. A mechanism must be instituted to allow orderly update. The Configuration Management Plan (CMP) addresses the management and administrative control of such updates; however, the CPDP should address the engineering/technical mechanisms. In particular, it should insure that a change mechanism exists, feedback and interaction will occur among various system and component design and development activities, and an efficient approach to additional testing is defined.

3.2.1.1 Development Visibility/Control Through Use of the Computer Program Development Plan

The CPDP describes the development methods and procedures which the contractor will use in developing the software. Monitoring the contractor development progress against the published CPDP will allow appropriate Air Force control and visibility. Air Force technical and management control procedures should be aligned to interact with contractor procedures specified in the CPDP.

3.2.2 Configuration Management Plan

Configuration Management (CM) is a discipline applying technical and administrative direction and surveillance to identify, document, control, and change a system. A Configuration Management Plan is the scheme by which the CM will be implemented. In the plan, the contractor will describe how a system will be initially baselined, how the system will be documented and controlled from that baseline, what the approval channels (internal and the Air Force) are, and the concept for making changes

to the baseline. Completion of the CMP is important because it forces the contractor to plan each of these functions. Implementing the plan enhances the probability of adequate software developed in a more efficient manner. The CMP should contain the specifics of configuration identification, configuration control, and configuration status accounting. Internal contractor procedures, prior to formal Air Force baselining are of particular importance, since this may be the only document covering this aspect. More coverage of CM is contained within the Configuration Management Guidebook in this series.

3.2.3 Quality Assurance Plan

AFR 74-1 defines Quality Assurance (QA) as a planned and systematic pattern of all actions necessary to provide adequate confidence that material, data, supplies, and services conform to established technical requirements and achieve satisfactory performance. In a large or complex software development within a weapon system acquisition, it may be advantageous to formally apply the provisions of MIL-S 52779A, "Software QA Program Requirements". Where this is done, a Software Quality Assurance Plan (QAP), separate from the System QAP is highly desirable and should be required by the CDRL. For acquisitions which cannot use MIL-S 52779A, the standard still serves as a valuable reference source for aligning the overall QA program and for tailoring of the QAP Data Item Descriptions to be more responsive to software QA requirements. The SAE Quality Assurance Guidebook provides explicit coverage on the content and intent of a QA plan. Quality Assurance is stressed here because of its positive affect on status monitoring/control. A comprehensive Software QAP increases Program Office visibility through periodic QA reports and more effective interface with AFPRO/DCAS and contractor quality personnel. Enhanced visibility coupled with a reduced need to monitor details, such as coding practices and standards, make a Software QAP a significant leverage item in achieving program office control effectiveness. The QA plan should be commenced prior to contract award so that a preliminary version can be used as an input for the source selection.

3.3 SOFTWARE SUPPORT PLANS

Computer programs delivered at the conclusion of the development

phase of the system life cycle remain evolutionary throughout the useful life of the overall system. Consequently, some method of supporting the software must be created. The more complex the software, the more sophisticated the support must be, both in terms of equipment and personnel expertise. It is the intent of this section to stimulate thought and planning to enable the software support capability to be acquired and implemented.

3.3.1 Computer Resources Integrated Support Plan

Earlier coverage of the Program Management Plan in this guidebook highlighted the responsibility of the Program Manager to see that a CRISP is written. The PM accomplishes this by forming a Computer Resources Working Group (CRWG) under an approved charter. Like any other plan, the CRISP is subject to change as its input data changes. The CRISP is a "living document" because of the extensive and constant change it undergoes. Planning for support of computer resources is commenced at approximately the same time as planning for the development of the computer resources. Thus the CRWG is a forum for assuring that issues such as conflicts between the operational and support concepts, support responsibilities, and long lead procurements are addressed and resolved. This guidebook will not attempt to resolve these issues, but, rather to lead through a sequence of steps to facilitate adequate planning and recognition of the problems such that timely decisions can be made. Appendix C of this document contains a list of required CRISP items.

3.3.1.1 Development Visibility/Control Through Use of a Computer Resources Integrated Support Plan

The CRISP, when formally approved and implemented by the using, implementing, and supporting commands, should insure that sufficient capability is available at Program Management Responsibility Transfer (PMRT) to allow operational and maintenance support of computer resources. All maintenance/diagnostic software should have been developed and tested for adequacy. If facilities are necessary to provide hardware or software simulations, these should be complete and operational. The control gained through use of a CRISP is that no support equipment, documentation, or programs are omitted which would jeopardize software support.

An additional concern related to the CRISP is the testing of the supporting software for the "basic system" software. In particular, complex systems which require an Avionics Integration Support Facility (AISF) can contain sophisticated software within the support capability itself. Typically, the support system must: enable the operational system to think it is operating in its normal environment, enable the operational system to be corrected or changed, provide a scenario to exercise the operational system, provide diagnostic capability, generate any necessary simulations, exercise models, or emulations, and provide analytic and data recording capabilities. A test for the support system must be written and the CRISP is the document which flags the requirement.

3.3.2 Operational/Support Configuration Management Procedures

Preparation of an Operational/Support Configuration Management Procedures (O/SCMP) is the responsibility of the using and supporting commands. This document, which is heavily dependent upon the CRISP, should be generated near the end of the development phase prior to PMRT. Coverage of the O/SCMP is given in this guidebook to assist in its generation and because it is not explicitly addressed in other guidebooks of this series.

The O/SCMP contains Configuration Management procedures to be used during the operational life of a system. Any complex system using software will undergo software changes during its lifetime, and in particular, avionics systems; thus the CM discipline is important even after development completion. As a minimum, the O/SCMP should incorporate the provisions of change control and the following

a. The relationship of all commands involved.
b. The reporting of problems.
c. The method of processing deficiency reports or modifications.
d. Approval authority for changes.
e. Status accounting procedures and responsibilities.
f. The handling of emergency changes.
g. Methods for distribution of CPCI's and changes thereto.
h. Situations where system turnover preceeds PMRT.

Generally, the delaying factor in generation of the document is a lack of agreement between the using and support commands over the procedures to use in reporting, processing, and correcting problems. The O/SCMP simply records, in procedural format, the concepts by which system deficiencies are identified, reported, and corrected.

3.4 FACTORS AFFECTING SOFTWARE DEVELOPMENT STATUS

This section emphasizes those factors which will impact, either positively or negatively, the actual development progress. It is understood that certain of these factors themselves may be beyond the realm of management influence, but their impact to the software development must still be understood and considered.

3.4.1 Requirements Stability/Traceability

More discipline and structure are being adopted and applied to embedded computer systems each year. The government approach to this uses MIL-STD 483/490 to specify the documentation of software requirements. Type A specifications containing system requirements cause the generation of Type B-5 specifications (design-to requirements) followed by C-5 specifications (code-to requirements). Each is at a lower level of detail than the former and is dependent upon the quality of the forerunner documents. At each level, a set of requirements serves as the input and a design is produced as the output. This design becomes the requirements set for the designer at the next level down. Using this "top down" approach, the cascading effect of a system requirement change is apparent. The degree to which the system requirements change (stability) directly affects the generation of design and code. (The underlying fact is that Type A, B-5, and C-5 specifications are all technical requirements. There should have been an earlier interpretation and translation of "operational" requirements into technical requirements). Consider the case where a user is uncertain of his detailed operational requirements. Uncertainty at the operational level can only be amplified when translated into technical requirements. The resulting requirements specification will be either excessively vague or will not resemble a solution to the real user need. Also consider the case where a user changes his operational requirement, then look at the "ripple" effect the change has depending

upon whether the Type A, B-5, and/or C-5 specifications have been generated. Software development progress is that rate at which validated software programs are produced; if the requirements (at any level) change, development progress is impeded. If requirements continually change, development progress becomes negative.

In addition to the changing requirements, there is the problem of referencing an operational requirement all the way into a set of computer programs (traceability). This reference is essential to ensure that the user is getting a product to meet his needs. The interpretation/translation referred to earlier also affects the traceability of requirements. Assuming the user is not convinced his need is being met, there will be a reinterpretation, retranslation, redesign, restatement of requirement, or retest of the software. Any of these will impede software development progress.

In summary, the more stable and explicit the requirements, the more traceable and producible the software. On the other hand, any complex software requirement cannot be completely stabilized prior to development. Program Managers must make tradeoff decisions in this area to enhance the probability of program success. For example, if requirements are constantly undergoing change, one possible approach would be to "freeze" the requirements on an interim basis to allow design progress against a specific baseline, then incorporate a "block" change.

3.4.2 Project Management

"There are more opportunities for improving software productivity and quality in the area of management than anywhere else. The difference between software project successes and failures has most often been traced to good or poor practices in software management". [1]

In many cases, the most serious software development pitfall is the inability of management to recognize the technical significance of certain management decisions. As an example, the relaxation of configuration control enforcement to allow quicker documentation completion can assist

[1] Dr. Barry Boehm "Software Engineering" TRW Software Series.

in meeting the documentation milestone, but can produce an unuseable document. This appears to be a common occurrence. Testing areas also are quite commonly compromised without full realization of the consequences.

Even the management structure can influence development progress. In general, the simpler, more direct control organizations function more efficiently in software development; however, the degree of risk of this type organization correlates to the experience level of those personnel in key positions.

Certain projects are attempted with multi-agency (multi-command or multi-contractor) interests and personnel directly involved in the software development. These arrangements have a tendency toward committee management. Ideally, the committee approach should more fairly represent parochial agency interests. Realistically, decisions are made slowly and resulting delays and compromises can seriously affect the usefulness of the entire system. In projects of this sort, it is suggested that a very narrow focal point of management control be established with "near dictatorial" authority for final decisions.

3.4.3 Project Complexity

At the most fundamental level, software complexity is a composite of only two basic considerations - control complexity requirements and data manipulation requirements. Control complexity might be further factored into timing, accuracy, and interface requirements. How and to what extent these contribute and interact to yield some overall complexity is currently one of the most active areas in software research. Such efforts as cost and error prediction models are examples. For the present however, the normal method of estimating software complexity must rely on comparison to previously developed similar systems, plus a certain amount of intuition. Section 3.4 is intended to help form at least a part of this intuitive estimate.

3.4.4 System Interfaces

In some respects, the complexity of a system is related to the number of interfaces the system has with other systems. Avionics programs, such as that of a central computer aboard an airplane, interface with many other systems. Information flow through some of the interfaces is two-way, while others are only one-way. In any case, the processing sequence and capacity are more severely taxed by the complex systems (many interfaces) than the simpler systems. It naturally follows that development of the more complex programs proceeds slower than the simple ones.

3.4.5 Data Base Management

Any large or complex software system is likely to include extensive amounts of data. Organization of the data into various storage areas, in particular as unreferenced data (files) or cross referenced data (data base), requires vastly different considerations in designing the scheme to manipulate and use the data. If the computer program to be developed requires a sophisticated data management scheme, development will require more time than a simpler system.

3.4.6 Constraints

Several common system constraints are discussed in this section to indicate the kinds of impact to software development that they may cause. Quite often, projects are attempted with compressed development schedules, processor limitations, memory limitations, inadequate peripherals and tools, inadequate facility support, inadequate manning, or inadequate expertise. Any of these can be detrimental to development progress even if management is aware of them and otherwise adequate methods and procedures are used.

Compressed schedules have a tendency to encourage planning curtailment and a relaxation of engineering controls. In other words, the project becomes milestone driven instead of quality, end-product driven. As the pressure of meeting the compressed schedule increases, more and more compromise of planning and engineering controls become evident. The danger of not adequately implementing these management functions is evident and the results are negative. A school of thought exists which says,

"If you are behind schedule, add more manpower." This is analogous to attempting to put out an oil fire with water. Each person added must become acquainted with the system as well as his assigned task. This, when coupled with the hurry up mode, only compounds the management control problems.

Processor and memory limitations are somewhat related in their effects on development progress. Violation of either limit requires some workaround to allow an apparently improved processor or larger memory. Workarounds, by their nature, are difficult to document and are usually done so inadequately. More programming (design and code) is required to use roll-in, roll-out schemes to allow more room in memory. These added programs especially aggravate system timing (sort of a Program Evaluation Review Technique (PERT) consideration for the system elements) and increase design complexity. Processor limitations are sometimes helped by rewriting programs to run faster. This usually means the code is more interdependent, complex, and susceptible to error. Trying to "shoehorn" more code into a full memory or make a processor "work faster" will be detrimental to the basic software development progress. Project specifications should require spare processor capability and memory proportional to the technical risk of the software. Trade studies during validation phase are the appropriate vehicle to address the quantification of the spare capability.

Large software developments usually require capable processing centers equipped with high speed printers, display terminals, tape units, discs, etc. Additionally, software tools such as trace, dump, sort routines, etc. are extensively used. Occasionally, a development is attempted which has a scarcity of peripherals, tools, or even the basic facilities to support the development. This results in some negative development impact while workarounds are developed or substitutes are used. Such considerations as remote locations of coders from the processing center, restricted hours of operation, classified processing requirements, excessive downtime, etc., can delay the completion of programs or modules on which others are dependent. All of these scarcities can cause development impacts of various magnitudes.

The range of operating systems available today is enormous, but all fit into the category of batch-processing and/or interactive. The interactive system allows the programmer, through use of a display or terminal, to communicate directly with the computer while the batch-processing system isolates the programmer from the computer. Batch-processing systems are most often found in computing centers where program development is considered a minor part of the overall activity, and efficient production runs are considered a major part. Interactive systems are often found in centers where program development is the primary activity and where the nature of the problems being solved dictate programmer/computer interaction.

One of the most crucial constraints to developing software relates to the adequacy of manning and the expertise of personnel. It is very risky to rely upon too few people, but there is also a significant management problem in using too many people. Attempting to use new or uninformed personnel means that a learning process must occur prior to their being productive. For complex systems, the learning curve can be up to one year. Schedules should take into consideration the estimated productivity rate of the various developers. Unfortunately, no magic key exists which enables the manning of particular tasks at particular levels with particular skills to provide efficient software development. The general rule is to assign more experienced and capable personnel to design related tasks and integration and less knowledgable people to lesser tasks. Development progress relates in an intangible way to personnel competency, experience, and manning levels. Development progress is maximized when these areas are maximized.

3.4.7 Degree of Modularity

A program module can be defined as a logically self-contained and discrete part of a larger program. It accepts well-defined inputs, uses well-defined processing, and produces outputs of well-defined content. Modular design should break complex tasks into smaller and simpler subtasks which enhance the probability of writing more correct programs. This implies a significant increase of effort during the preliminary design phase of software production to achieve payoff during code and test

phases. The degree of modularity is the extent to which a large set of programs can be conceived as made up of a composition of smaller program elements. Largely modular programs lend themselves to simpler design and less interactions between program parts, thus simplifying the overall program.

3.4.8 Programming Languages, Standards and Practices

The applicability of the language(s) to the desired software development and the discipline of using approved programming practices both have positive effects upon development progress. The incorporation of standards for programming will also save time because it allows a more orderly, understandable progression through these design, code, and test activities. Poor practices or standards omissions often result in reaccomplishment to provide a basis of understanding for computer program maintenance and further enhancement.

3.4.9 Uniqueness

Development progress of software, which is totally unlike any previously generated program, is slower than "variations on a familiar theme". Care should be exercised to avoid pushing beyond the state-of-the-art and in areas where few have previously been. Technical risk goes up proportionate to the deviation from previously developed program designs. New or one of a kind hardware systems which require software development are also prone to be associated with lengthy development schedules.

3.4.10 Security

The coverage of security in this section is not intended to reflect guidance additional to DOD and Air Force regulations but to emphasize the additional impacts of developing software which contains classified programs or data. There are three types of security considerations which will affect development progress.

Production of software containing classified data means that physical security procedures must be implemented, to limit access to the facilities to only those appropriately cleared people who have a need-to-know. Computer facilities and procedures must be established which restrict unauthorized compromise or exposure of classified data.

A second consideration is associated with the actual generation of the data. Some method of controlled data acquisition (even if it's only classified mail) must be defined and implemented. Appropriate classification of actual "generated" data must be established in accordance with DOD and Air Force regulations.

The third consideration is control of access to the classified data base. Programmers or users of sensitive data must be assured that unauthorized personnel cannot intentionally nor inadvertently access the classified data. Protective software and/or procedure must be implemented to preclude this activity.

3.5 STATUS QUANTIFICATION

Several attempts at quantifying software development progress have been made, some of which have resulted in models, formulas, procedures, etc. The problem of exact status is a complicated one in which a truly generalized coverage is, at this point, unknown. Assuming that it is possible to forecast the approximate number of instructions to make up a software module, when a certain percentage of the code is done, a certain quantified portion of the module is done. However, the module isn't yet tested, nor is it integrated with other modules and tested. If one could go through a module code generation only once, test the results with no rewrite, and be assured the module would properly interface with others, then completion of a percentage of the code would equate to a specific percent of completion. Experience indicates that some degree of rewrite is highly probable. There will be module interface problems which could perturbate basic design, and there is likely to be some testing which requires retest. This is not to imply that counting lines of code is a useless exercise, but simply that an exact status will not be indicated. If the module in question represents 30 percent of the total task and 50 percent has been coded and tested, then you are somewhere below 15 percent project complete. At least there is some indication of upper bound.

Another common approach to quantification of status is by "functional threads." By this theory, a function starts in some module and threads its way through one or more additional modules. The inputs to the func-

tional thread are supplied and the reaction of the functional thread is checked. When the results test o.k., this functional thread is completed. If you assume this thread makes up some percentage of the total effort, then upon its completion, you are that percent complete of the total effort. Unfortunately, there are few (if any) functional threads which are completely independent of all other threads. Therefore, any influence or feedback of the other threads on the one just completed is discounted in indicating status. Like the module sizing theory, the thread theory will give some indication of development status but not an exact indication.

Note that both of the theories discussed in this section indicate a reliance upon modularly designed software. An unmodularized design would not lend itself to realistic indications using either theory.

4. MONITORING AND CONTROLLING DEVELOPMENT STATUS

The discussion in Section 3 emphasized the necessity for proper planning of a software development. This chapter emphasizes the need for controlling against the plan and the mechanisms considered are

- Milestones and Schedules
- Reviews and Meetings
- Monitoring Development Testing
- Contractor Progress Reports
- Financial Performance Reports
- Use of IV&V Contractor

4.1 MILESTONES AND SCHEDULES

DODD 5000.1 identifies four key milestones for major System Acquisitions as being of interest to the Secretary of Defense and the Defense Systems Acquisition Review Council (DSARC). These are: Program Initiation, Milestone 0; Demonstration and Validation, Milestone I; Full Scale Engineering and Development, Milestone II; and Production and Deployment, Milestone III. The DOD is interested in monitoring any project at this level. From that level downward, there is a cascading effect of more milestones at increasing frequency depending upon the level of visibility the Air Force, the Command, the Program Manager, the system engineer, the module designer, etc., needs. At the lower levels, the relevant milestones may approach almost a daily basis, but, in any case, they are designed to provide assurance that each level of tasks can be made to fit the critical milestones of higher level schedules. As discussed in Section 3, exact status of software development is seldom known. This results in detailed software milestones being projected with less precision than might be desirable. However, these milestones are not unuseable if viewed from the proper perspective (If an auto driver misjudges the distance remaining to a faraway location by a hundred feet, it is virtually meaningless, but if he misjudges the car immediately in front of him by that amount, it could be disastrous). The point is that

the milestone increments one needs are a function of not only the next critical decision point, but also the level of confidence with which the status is known. Lower confidence should result in more milestones, even if they tend to be artificial.

4.2 REVIEWS AND MEETINGS

This section addresses milestones in the context of formal reviews required by the contract and informal meetings which normally occur during the software development process. The formal milestones to be addressed (System Requirements Review, System Design Review, Preliminary Design Review, Critical Design Review, and sometimes, Test Readiness Review), are well defined, highly visible events upon which the entire system development has been based. They are therefore critical points and must be approached with a high confidence level. The informal meetings can be viewed as the preliminary milestones which occur as often as necessary to insure the integrity of the major milestones. These may be called Technical Interchange Meetings (TIM), Management Strategy Meetings, etc. They may be periodically scheduled or de facto meetings called by the Program Office or contractor, but their importance in maintaining program office visibility and control cannot be overemphasized.

Events leading up to major reviews and criteria for passage are discussed in detail in the SAE Reviews and Audits Guidebook and are briefly outlined below.

4.2.1 System Requirements Review

MIL-STD 1521A indicates the System Requirements Review (SRR) may be held at any time during system conceptual and/or validation phases but normally after the accomplishment of functional analysis and preliminary requirements allocation. Its purpose is to determine initial direction and progress of the contractor's System Engineering and Management effort and his convergence upon an optimum and complete configuration. The total System Engineering and Management activity and its output are reviewed for responsiveness to the Statement of Work and system requirements. The SRR emphasizes weapon system level requirements; however, where significant software development is involved it is recommended that

a separate system level software review be conducted, either independently or as a formal portion of an SRR. The purpose of the additional software review is to arrive at a joint Contractor and Air Force understanding of the provisions stated as software requirements. Analysis of all software requirements on an individual and composite basis is necessary prior to the software review. Techniques and tools used to perform the analysis should be explained, all identified problem areas discussed, and remedial actions outlined. The end result of the software review is a conceptual agreement on the software requirements which provides the basis for development of the software requirements specification(s).

4.2.2 System Design Review

MIL-STD 1521A indicates the System Design Review (SDR) shall be conducted to evaluate the optimization, traceability, correlation, completeness, and the risk of the allocated requirements in fulfilling the functional configuration baseline. The SDR represents the first technical review of a contractor's progress. Principally, this review is to prove the contractor's understanding of the requirements and for the Air Force to evaluate the contractor's system design approach. Specific computer programming requirements topics to be addressed are listed in Appendix B, Paragraphs 20.3.21, 20.3.2m, 20.3.10k, and 20.3.12.1 through 20.3.12.13 of MIL-STD 1521A.

4.2.3 Preliminary Design Review

MIL-STD 1521A indicates the Preliminary Design Review (PDR) shall be a formal technical review of the basic design approach for a Configuration Item (CI) or for a functionally related group of CI's. Specifically, availability of the Part I specification and accomplishment of the preliminary design are necessary prerequisites of the PDR. During this review, the contractor should demonstrate he is ready to proceed with the detailed design. Problem areas and possible solutions should be presented by the contractor. Specific computer program configuration items to be considered are addressed in Appendix C, Paragraphs 30.3.2.3, 30.3.13.4, and 30.2.2(a) through (k) of MIL-STD 1521A.

4.2.4 Critical Design Review

MIL-STD 1521A indicates the Critical Design Review (CDR) shall be conducted on each CI prior to fabrication/production design release to insure design solutions in the draft Part II specification satisfy performance requirements stated in the Part I specification. The contractor should present his design approach through detailed flowcharts and briefings to the Air Force at the proper level of detail as implied by MIL-STD 1521A, Appendix D. The CDR is a review of the design prior to release to the coders.

4.2.5 Test Readiness Review

The Test Readiness Review (TRR) is an informal review commonly used by the development contractor to review development test results and evaluate preparations for qualification and/or acceptance testing. Completion of this review ascertains the adequacy, traceability, and completion of accomplished development testing for planned qualification/acceptance testing plus the TRR evaluates the testing procedures, tools/facilities, personnel, configuration control procedures, etc., to be used in the quality or acceptance tests. Although not defined in MIL-STD 1521A, this review has proven extremely useful for programs involving complex or extended testing programs. Formalizing the TRR by including it in the Contract Statement of Work and other contractual documents should be seriously considered.

4.3 DEVELOPMENT TESTING

In addition to formal and informal reviews, a very useful method for gauging development progress is to monitor software development testing, (i.e., lower levels of testing than CPCI level). Typically, the contract will require demonstrations and formal acceptance tests at the CPCI level, but for informative purposes tests at much lower levels need to be witnessed on a sample basis. Reports resulting from these tests should be described in the Computer Program Development Plan and available to the Air Force if necessary through the data accession technique (see SAE Documentation Requirements Guidebook, section 3.3.4.2a.)

The lowest levels of development testing begin with individual programmers to insure that programmed code is valid and contributes to some requirement. Testing at the CPC and module level can also be accomplished followed by integration testing of multiple modules. Finally, testing of each CPCI can be accomplished followed by integration testing of multiple CPCI's.

Each level of testing must be accompanied by a comparable level of test requirements. In other words, if the project is contracted at a CPCI level of testing, but development testing is to be accomplished at a lower level, then test requirements must be defined for a commensurate lower level; as an example, testing a "unit" of software must be accomplished via "unit test" requirements.

Subsequent to satisfactory testing of low level units of software, the tested units must be placed under configuration control. The more frequent testing (at lower levels) requires a greater demand on configuration control.

Testing at any level is likely to be more thorough if performed by a group other than the software developing group. A mechanism for correction of discovered deficiencies must be implemented to allow correction of the deficiencies.

In summary, development testing provides good visibility into the actual development progress. The assistance gained is directly proportional to the quality of planning preceding the software unit definitions and their interfaces.

4.3.1 Test Report

The test report details the results of the executed test. It will identify the purpose and nature of test, will describe in detail any deviations from the test specification or procedures, and will compare the test results with the expected output.

4.4 CONTRACTOR PROGRESS REPORTS

Another technique for monitoring development status is the formal report prepared by the contractor. These may include:

- Periodic status reports
- Analytical reports
- Test reports
- Meeting reports

There are multiple formats available for each of these and more in the AMSDL. However, in the case of formal reports, a minimum number, carefully chosen and tailored, if necessary, is the best approach. They tend to be costly not only in dollars, but in use of human resources both for the contractor and the Program Office. Proper procedure would be to require those reports that supply the data required for visibility, are written in a useful, clear format, and are generated at a time in the project to allow timely decisions and/or forecast future development progress.

A periodic report is submitted at regular intervals and usually addresses milestones, accomplishments, future goals, and problem areas. Analytical reports are submitted as a result of some special activity completion and usually identify study results, modeling, or simulations. Test reports follow the completion of the appropriate testing activity and usually indicate the degree of requirements verification, data collected, and analysis conducted. Meeting reports describe the discussions held during a meeting and any guidance, agreement, or conclusion reached.

4.5 FINANCIAL PERFORMANCE REPORTING

Overall, schedule performance can be viewed in dollar terms by comparing budgets for completed work to budgets for scheduled work. The best place to measure accomplishment is at the level where work is performed. Summaries of Work Breakdown Structure (WBS) elements can provide meaningful data which justify management adjustments. If the software development can be divided into WBS elements, then actual development can be related to budgeted development. The management system which produces performance reports must be reasonably well disciplined to be effective. Arbitrary transfers of budget from one task to another, for example, can destroy the significance of reported values.

The Cost/Schedule Control Systems Criteria (C/SCSC) established many of the characteristics and disciplines required of an effective

performance measurement system. These criteria do not impose any specific management technique or methodology, but represent basic principles applicable to most management systems. Failure to meet a criterion generally indicates a weakness in a Cost/Schedule Control System. While compliance with the C/SCSC is mandatory on major defense programs, the criteria also provide a useful checklist for evaluating programs of any size.

The Cost/Schedule Status Report (C/SSR) provides for reporting the key data elements described by WBS elements and narrative comments explaining cost and schedule variances which exceed specified thresholds.

4.6 INDEPENDENT CONTRACTOR

Independent Verification and Validation (IV&V) is often used as a management tool for controlling software development. To illustrate the thoroughness of development status coverage, the following reports normally associated with IV&V are listed:

- Risk and criticality study reports
- Requirements analysis reports
- Design analysis reports
- Code analysis reports/data reduction
- V&V test plans and procedures
- Automated testing and analysis reports
- Independently generated scripts and scenarios
- Interface analysis reports
- Model and simulation documentation and user reports
- Error analysis and trouble reports
- Independent evaluation of developer's test program
- Reports of special studies
- Validated requirements data base
- System Performance Analysis Report

The philosophy behind these V&V products is to establish a pipeline through which information is quickly passed to management with formal follow-up procedures to ensure the problem and analysis reports include the disposition and status of all issues affecting the development.

5. GOVERNMENT REPORTING

The requirements for the Program Manager to report project status are frequent and may be assessed at varied levels within the Air Force structure. If a software problem or assessment is to be included as a portion of the status then the responsible software development manager should be aware of the expectations of these status requirements. Typically, the status data are contained in a few viewgraphs which are orally presented by the Program Manager or his appointed representative. AFSCR 800-1 defines three levels of status reports: Command Assessment Review (CAR) to be given at Air Force command level, Program Assessment Review (PAR) to be given at Air Staff levels, and Secretary Program Review (SPR) to be given at Air Force Secretarial levels. Specific guidance relative to viewgraph format and topics required is defined in "Revision No. 3, Format and Instructions for SPR, PAR, and CAR." This document published in April 1979, is supplemented by several amplifying messages and explanatory data. Both AFSCR 800-2 and "Revision No. 3" should be consulted prior to completing any status submittals on software development.

APPENDIX A

PROCUREMENT PLAN TOPICS

The following outline is offered as a guide for topics to be addressed by a Procurement Plan (PAR 1-2102 contains this guide and also authorizes its retailoring to more specifically address a particular program procurement).

1. A description of the program, item or system
2. Program Funding (R&D and Production) including a summary of monies in the FYDP-Budget Submissions
3. Delivery Requirements, both R&D and Production Contracts
4. Applicability of a Decision Coordinating Paper or Program Memorandum Defense System Acquisition Review Council (DSARC) or Internal Service Reviews
5. Background and Procurement History
6. Discussion of Program Risk, including Technical, Cost, and Schedule Risk
7. Integrated Logistics Support Planning Concept
8. Application of Design to Cost
9. Application of Life Cycle Cost
10. Reliability and Maintainability Objective
11. Test and Evaluation Approach
12. Approval for Operational Use
13. Government Furnished Material/Facilities/Component Breakout
14. Application of Should Cost
15. Milestone Chart Attachment Depicting the Objectives of the Acquisition
16. Milestones for Updating the Procurement Plan
17. Identification of Participants in the PP Preparation
18. Procurement Approach for each Proposed Contract

APPENDIX B

COMPUTER PROGRAM DEVELOPMENT PLAN TOPICS

The CPDP identifies the actions needed to develop and deliver computer program configuration items and necessary support resources. Specifically, AFR 800-14 requires the plan to address the following

a. The organization, responsibilities and structure of the group(s) that will be designing, producing, and testing all computer programs.

b. The management and technical controls that will be used during the development, including controls for insuring that all performance and design requirements have been implemented.

c. The methodology for insuring satisfactory design and testing, including quality assurance.

d. The development schedule for each computer program configuration item and proposed program milestone review points.

e. The procedure for monitoring and reporting the status of computer program development.

f. The resources required to support the development and test of computer programs. Special simulation, data reduction or utility tools that are planned for use in the development of computer programs should be identified.

g. The general procedures for reporting, monitoring, and resolving computer program errors and deficiencies during development and testing.

h. The methods and procedures for collecting, analyzing, monitoring, and reporting on the timing of time critical computer programs.

i. The management of computer program development masters, data bases, and associated documentation including its relationship to the configuration management.

j. Guidelines and checkpoints for insuring future computer program growth, modularity, and ease of modification.

k. The approach for developing computer program documentation.

l. Training requirements and associated equipment for the deployment phase.

m. Engineering practices include standards, conventions, procedures, rules for program design; program structures and conventions; display and logic standards; input/output signal standards; and other disciplines affecting development.

n. Security controls and requirements.

o. Simulation techniques and tasks.

APPENDIX C

COMPUTER RESOURCES INTEGRATED SUPPORT PLAN ITEMS

The Computer Resources Integrated Support Plan (CRISP) identifies organizational relationships and responsibilities for the management and technical support of computer resources. It functions during the full-scale development phase to identify computer resources necessary to support computer programs after transfer of Program Management Responsibility and system Turnover (PMRT). The CRISP continues to function after PMRT as the basic agreement between the supporting and using commands for management and support of computer resources. AFR 800-14 requires the following items to be included, as applicable

a. Offices of primary responsibility and management focal points for support of computer resources and the channels of communication among organizations.

b. Planning for configuration management of computer programs including the assignment of configuration control responsibilities during the deployment phase. This planning will reflect the operational and support concepts for the system.

c. Responsibilities for composite system integrity include

   1. Computer storage utilization

   2. Computer program operating time and priorities

   3. Computer program interface techniques

   4. Computer program baseline integrity

   5. Utilization of computer modules and peripherals

d. Documentation required to support each type of computer program.

e. Responsibility for funding, scheduling, and system integration.

f. Qualified personnel required for supporting computer equipment and computer programs together with training requirements.

g. Computer equipment and devices required to facilitate computer program changes, including acquisition responsibilities.

h. Computer programs required to support computer equipment and other computer programs including acquisition responsibilities.

i. Verification and Validation (V&V) of computer programs.

j. Plans to establish and operate necessary support facilities. Common and existing facilities will be used whenever practicable. The size and scope of the support facility will be based on work load predictions.

k. Provisions for the transfer of program management responsibility.

l. Provisions for system/equipment turnover.

APPENDIX D

DATA ITEM DESCRIPTION	2. IDENTIFICATION NO(S)	
	AGENCY	NUMBER
1. TITLE Computer Program Development Plan (CPDP)	USAF	DI-S-30567A

3. DESCRIPTION/PURPOSE

The CPDP is a document in which the contractor describes his specific detailed plan for the management and development of all of the computer programs and associated documentation that he needs for the completion of the contract; this may include support software such as simulations and analysis tools, as well as operational programs and automatic test equipment programs. The plan may be used by the procuring activity both to assess and approve the contractor's approach and methods for computer program (continued - see page 2)

4. APPROVAL DATE

22 Feb 1978

5. OFFICE OF PRIMARY RESPONSIBILITY

AFSC

6. DOC REQUIRED

8. APPROVAL LIMITATION

7. APPLICATION/INTERRELATIONSHIP

This Data Item Description applies to the computer resources portion of system development and acquisition and can be utilized during the validation and subsequent phases of the DOD system acquisition cycle. A CPDP may be obtained pre-contractually in the bidders' proposals and may be a product of the validation contract or acquired during the full scale engineering development contract. The plan may be maintained to reflect current status of the requirements. The CPDP is intended to complement other contractual management plans which address disciplines such as systems engineering, configuration management, and test and evaluation. Details from these related contractual management plans or other deliverable documents may be incorporated by reference, but the references must be specific enough to indicate precisely the elements of the plan. Supersedes DI-S-30567.

9. REFERENCES (Mandatory as cited in block 10)

MIL-STD-483 (USAF)
MIL-STD-881

MCSL NUMBER(S)

10. PREPARATION INSTRUCTIONS

10.1 The plan shall be in contractor format unless a specific format is specified in the Contract Data Requirements List. The following items shall be provided as a minimum. Where contractor format is used, a cross reference between the CPDP and the following items will be included.

a. Requirements Assessment Summary. This section of the CPDP shall summarize or specifically reference documents describing the contractor's understanding and assessment of the completeness and clarity of:

(1) The stated computer resource requirements,

(2) The definition of the hardware and software configuration items and their interfaces,

(3) The current functional allocation,

(4) Required margins and budgets for critical items such as:

(a) Memory
(b) Execution time
(c) Support products,

and shall include:

DD FORM 1664

3. Description/Purpose (Continued)

development and to assist in monitoring and evaluating the contractor's efforts during development and test of the products defined by the contract. The CPDP addresses, summarizes or references the management issues in a single document.

10. Preparation Instruction (Continued)

(1) A summary of high risk and uncertainty issues.

(a) Technical
(b) Cost
(c) Schedule

(2) Identification of proprietary computer resources and other used-as-is or modified-and-used computer resources to be used during the contract including subcontracted resources.

(3) Approaches with backup alternatives to minimize or overcome the risk and uncertainty issues.

(4) The risk involving long-lead items and their status.

(5) Trade and optimization studies yet to be conducted and their relationship to the computer resources.

(6) The rationale for the selection of the computer programming language(s) if not specified or if different from that specified.

(7) Identification of subcontract items and who will be performing the subcontract work.

b. Project Objectives. The CPDP shall provide: A statement of the contractor's objectives and goals; For each Computer Program Configuration Item (CPCI), per MIL-STD-483, the planned allocation of its functions and interfaces; For each CPCI, budget goals for timing memory, interface, and channel capacity (including redefinition, when required, based on the contractor's approach) and margins for growth in these capacities both during development and future operation; The relative importance of such software characteristics as reliability, maintainability, testability, efficiency, portability, interoperability, other factors from technical approach that effect the CPDP, etc.

DI-S-30567A (continued)
Preparation instructions (continued)

c. Work Definition. The CPDP shall provide: The work tasks required to achieve the objectives of the computer program development effort; Identification of the necessary development steps such as analysis, design, coding, checkout, integration, test, acceptance and delivery for each CPCI and major supporting computer resources and their relationship to the contractor's Work Breakdown Structure (WBS), per MIL-STD-881, including required WBS deviations; Tasks associated with support functions (such as documentation, configuration management, data management, management reviews, quality assurance, etc.) and their relationship to other contractual tasks; Tasks involving integration and test of CPCIs, including such factors as separate test configurations, system tests, and operational tests where applicable; Major development steps for all identified Computer Program Components (CPCs) in each CPCI, and all elements of the required computer resources which support the products.

d. Work Schedule. The CPDP shall provide: A time schedule of the above work elements, based on the contract master schedule, indicating initiation, intermediate (e.g., availability of draft and final copies of formal and informal documentation) and completion times for all CPCIs and time/performance critical Computer Program Components (CPCs) including formal and informal milestones, reviews, audits, key meetings, documentation releases, etc.

e. Activity Network. The CPDP shall provide: An Activity Network (e.g., PERT) compatible with the Work Schedule of computer program development efforts, including interface activities (such as hardware development) that can impact schedule and an identification of critical path or near critical path elements; To the extent possible, an identification of those tasks and activity paths that might be most strongly influenced by changes in program requirements.

f. Organization. The CPDP shall provide: A delineation of the contractor's organizational structures, authorities, responsibilities, interfaces, skill requirements and lines of communications necessary to manage and execute the scheduled activities to assure proper task completion; The relationships assumed between the contractor and independent or redundant verification and validation groups, other contractors, subcontractors, the procuring agency, the support agency, and the user agency; Identification of skill requirements and key skilled managers and employees by name.

g. Resource Allocation. The CPDP shall provide: An allocation of facilities, laboratories, computer time, test equipments, and other relevant resources against the organizational structures, work elements, schedules; Identification of project-peculiar resources required such as special purpose hardware and computer programs, government-furnished items, special data, etc.; List of items which may impact resources such as high risk development items, special security requirements, subcontractor control, etc.; Presentation of the methods to be used to assure compatibility of the procured systems with the intended operational physical facilities.

h. Engineering Standards. The CPDP shall provide: A definition or identification of software engineering practices as they apply to each CPCI and computer resource support product, how these practices will be maintained, and how their use will be assured; Examples of engineering practices include: standards, conventions, procedures, and rules for program design, structures; display and logic standards; input/output standards; guidelines for program subdivision, coding techniques, and programming languages; data base standards and other disciplines affecting development.

i. Design Assurance Techniques. The CPDP shall provide: A definition of the techniques used for design analysis and control to assure completeness, validity traceability to requirements, testability, and compliance with standards and practices; The approach to preliminary and critical design reviews such as incremental or multiple reviews; A definition of the design evaluation techniques and an identification of the purpose, application, and validity of the tools used such as computer resource support products, simulations, prototypes, mathematical models, and emulations; An identification of verification and validation procedures and aids and how they will be qualified and used during the entire software development process; A statement of the plan for defining, managing, controlling, and reporting the status of budgets on timing and memory, and technical interfaces and interface margins (such as I/O channel capacities and protocols), including guidelines and checkpoints for insuring future computer program growth capability, modularity, and ease of modification.

j. Detailed Design, Coding, and Checkout. The CPDP shall provide: A definition of the procedures, steps and documents associated with detailed design, coding, checkout, review and acceptance of the individual modules comprising CPCs and CPCIs including the funtions performed, the handling of the internal and external interfaces, and the control of parameters like memory allocations, operating sequences, data base requirements and test requirements; Identification of computer programs, equipment and devices required to support system computer hardware and to facilitate software changes, including diagnostic software for all computer resources; Criteria and the mechanism for acceptance of modules for integration and test.

k. Development Test and Evaluation. The CPDP shall provide: A presentation of the integration and test philosophy and approach for CPCs and CPCIs, how this philosophy is applied in the design and scheduling, and how the approach leads to the Preliminary and Formal Qualification Tests; A definition of the interfaces and responsibilities (such as training) among test groups and other project participants; The resource and schedule impact of independent verification and validation activities.

DI-S-30567A (continued)
Preparation Instructions (continued)

l. System Test and Evaluation. The CPDP shall provide: Procedures for the coordination and support of CPCIs and other computer resources during formal system testing (and during Operational Test and Evaluation as applicable), including necessary training and transfer of data.

m. Anomaly Control. The CPDP shall provide: The methods or system by which computer software anomalies are detected and documented during all test and evaluation, corrections are provided, and configuration control is maintained, including organization responsibilities; The data to be gathered to enable assessment of the reliability of the computer resources, reporting periods, data sources, and method to establishing when each anomaly (i.e., design, coding, test, etc.) was detected and corrected.

n. Management Controls. The CPDP shall provide: Relationships between the management controls of the CPDP and other applicable management plans such as: the Configuration Management Plan, security plan, and the Systems Engineering Management Plan (SEMP); The means and criteria for management assessment and control of the development process, including subcontracts; for example, mechanisms for initiating management actions when status dictates deviation from plan, such as resource reallocation, schedule slippage, cost increases, or performance degradation.

o. Documentation. The CPDP shall provide: Any new software documentation tools or a description of the contractor documentation practice including techniques intended for use on the subject contract; Identification of the specific documentation alternatives and the necessary changes to the content and format of the specified documentation requirements, including, as a minimum, a description of the equivalency features between the specified documentation requirements and the proposed alternative(s); A description of the procedures which will be used to keep the informal documentation current; A description of the approach to computer program in-line commenting.

p. Configuration Management. The CPDP shall provide: The special aspects of computer software configuration management not addressed in the overall Configuration Management Plan to include the organizational placement, change, deviation, waiver authorization and control, internal engineering change board, configuration identification, computer program library control, the control or changes to any non-deliverable computer programs, the handling of interface control between the hardware and the software of the system, maintenance of independent master card/decks/tapes/discs, control of development versions of CPCs and CPCIs, handling of source and object code, control over test environments, relationship to programming standards, and relationship to the Quality Assurance function.

DI-S-30567A (continued)
Preparation Instructions (continued)

q. Vendor/GFE Computer Resources. The CPDP shall provide: The procedures for qualifying and documenting the present adequacy and the adequacy for future use of vendor-supplied or GFE computer resources (hardware and software) and the means, such as testing, for accommodating revision of vendor-supplied computer resources.

r. Support Resources for the Deployment Phase. The CPDP shall provide: The recommended support philosophy and the resource requirements for use after the full-scale engineering development phase. This section of the CPDP shall summarize or specifically reference the plan for the transfer of computer resources including support software and tools to the appropriate Air Force agencies.

APPENDIX E

SOFTWARE ACQUISITION STANDARDS, METHODOLOGIES, AND TECHNIQUES

Generally speaking, there is considerable overlap in definition of the three terms - standards, methodologies, and techniques - which apply to software acquisition. For purposes of discussion within this guidebook, the following definitions are given:

- Standards - That set of published rules, conventions, or principles accepted as generalized guidance in software acquisition.
- Methodologies - That set of applied concepts used in generating design and code for software acquisition.
- Techniques - That set of procedural implementation used in applying software acquisition methodologies.

STANDARDS

Standards of time, distance, and speed are commonly used everyday without the user being fully aware of the fact. The importance of standards is so meaningful that a United States Bureau of Standards exists to publish and safeguard those master reference devices. Yet, when the term "Programming Standards," for example, is mentioned there is little recognition of its true meaning. When properly used, standards simplify an individual programmer's job by eliminating the need for tradeoff studies and by making automated tools available. Standards also establish a uniform basis for groups of programmers to work together with a minimum amount of confusion and misunderstanding. The Air Force has published several standards in an attempt to establish a common basis of communication and understanding in software acquisition. This guidebook does not attempt to address each of them for they are too numerous, however, the following discussion is offered to encourage use of standards to a practical extent.

- Programming Practices and Standards

Explicit coding practices and standards are very useful controls for producing software programs and to assure proper implementation. The degree of usage is likely to be proportional to the degree of management emphasis for adherence to these practices. Adherence to standards will make the software far more intelligible and should reduce the time required for debugging, testing, and changing the software. Documentation is more easily relatable to "clean" software than to software which was developed without regard to standards. Coding practices, standards, and their enforcement can be used as schedule status factors. Standards can be written for a particular program as well as those more universal ones published by the government. The following list of available standards is offered for consideration in your project

- Activity Networks - The requirement that a chart be drawn showing all the activities to be completed in the project, as well as the interactions among activities, provides an excellent tool for ensuring the completeness of a project plan and the validity of the schedule.
- Phase Review - Formal procedures for obtaining management and technical approval at the end of a task. Avoid false starts and permit timely decision making.
- Configuration Management - Constructing a baseline specification and establishing a control procedure to authorize changes to the baseline are necessary in the control of a large system.
- Machine Room Procedures - Better service can be obtained from a central computer facility if all users follow standard procedures for job submission, priority assignment, and usage forecasting.
- Top Down Programming - Direct the programmers to design their program by first defining files and major component structure and then adding calls to more detailed subelements.
- Structured Programming - By following valid rules, programmers tend to generate software that is easier to read and test.
- Using Library Programs - Considerable new programming effort can be avoided by multiple programmer use of existing programs.

- Languages

Not all phrases are as explicitly described in English as they are in Latin (the converse is true for other phrases). Similarly, not all computer languages have the same degree of applicability to all computer tasks. The selection of the implementation program language(s) should be viewed with regard to all phases of development including test and maintenance. The applicability of a computer language to the task at hand plus the degree of adherence to that language direcly affect programmer progress in generating software. Other factors that influence progress are: language version, translator availability, compiler stability, efficiency of generated code, and compiler/assembler code apportionment.

Higher Order Languages (HOL) are generally regarded as easier to use, however, they sometimes occupy more room in the computer memory than assembler language. Certain programming tasks lend themselves to more favorable usage of HOL than other tasks do. It is wise to study the system in-depth and decide upon the type of language to use prior to any contractual agreements.

METHODOLOGIES

Methodology is a term that, when applied to software development, attempts to describe a more orderly approach to generating software in modern times as compared to earlier software efforts. Sophistication of software programs has evolved to the extent it is no longer efficient (or in some cases, possible) to generate software with an unplanned (unstructured ?) approach. Methodologies attempt to more logically design, code, and test the software. Although methodologies are not a panacea for software development, they do offer a more disciplined approach which results in improvements to program quality, manageability, and productivity. Descriptions of methodologies are sometimes overlapping and vague. The intent of this section is to acquaint the reader with certain current concepts of the "how to's" of computer program development.

- Top Down/Bottom Up Design

The terms top down and bottom up have been widely used in recent years in association with software development. Primarily top down begins first with the establishment of a control architecture along with interface and data requirements. Subsequently, lower level functions are designed, and the process continues until all required functions and sub-functions, interfaces, and data requirements have been designed. Top down is analogous to commencing at the tip of an Air Force organizational chart and working downward through the organizational tiers. Bottom up is simply the reverse of top down (start at the lowest organizational level and work upward). Software designers in particular have used the top down approach and through successive refinements are able to produce more independent "hunks" of software (modules). Both top down and bottom up are used in the implementation phase (code, integrate, and test) of development; however, only top down is successful in the design. One word of caution in using bottom up is the design must have been completed prior to commencing botton up implementation or else there is a risk the module will not efficiently apply to the overall design. Frequently, coding commences prior to design solidification, therefore causing considerable recode.

The advantages of top down implementation are: integration is continuous and fully tested prior to proceeding to the next level down, software quality is enhanced through earlier detection and elimination of design problems and coding errors, and the need for writing drivers for module testing is eliminated. Few advantages are attributed to bottom up implementation although certain specialized programs, such as the device handlers of an operating system, are more reasonably accomplished by this approach.

Figure E-1 is included to identify design techniques which are consistently recommended plus some which are favored under certain circumstances.

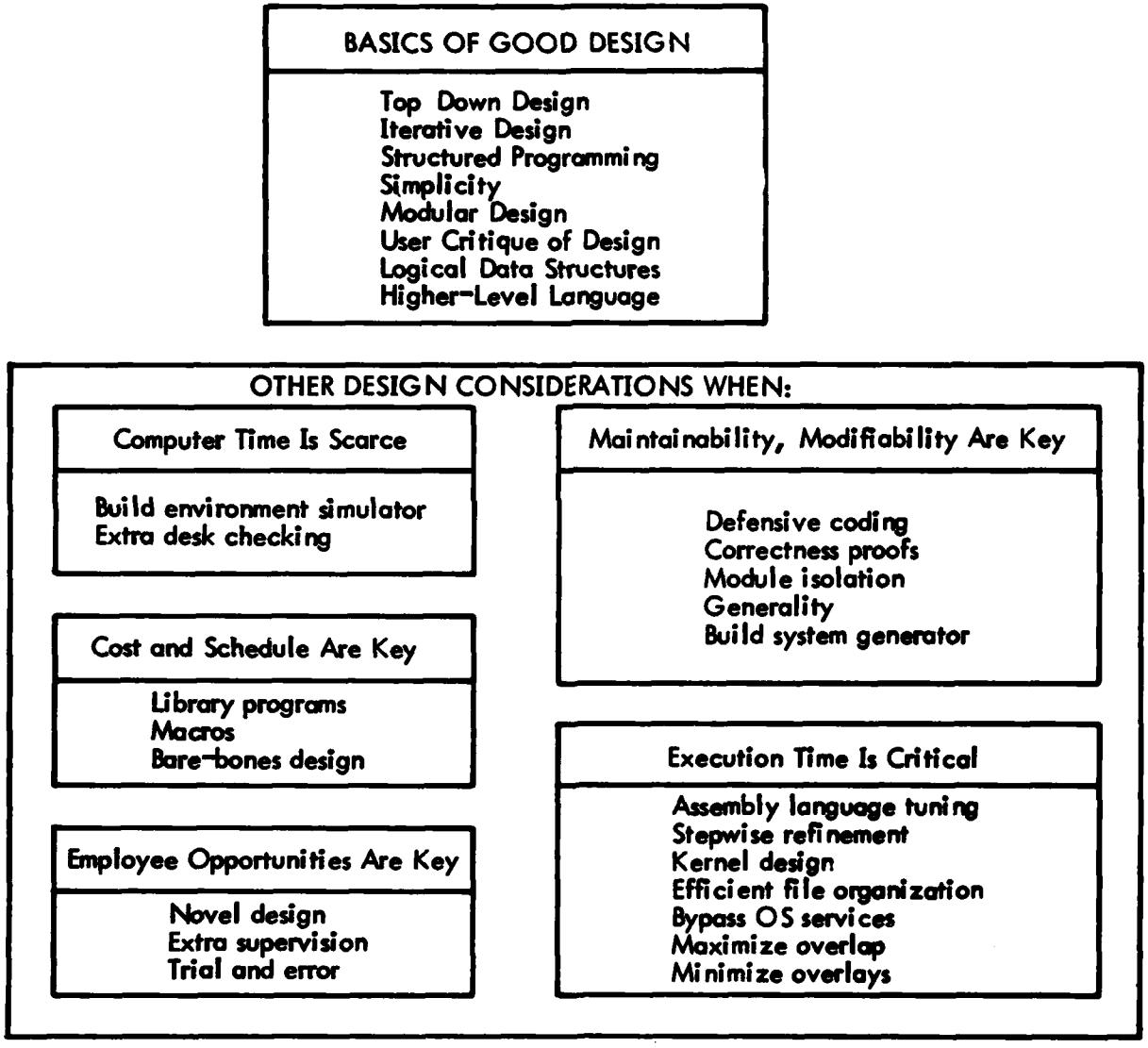


Figure E-1. Design Considerations

- Modularity

As previously mentioned in this guidebook, modularity is the partitioning of software into smaller pieces with well-defined inputs, well-defined processing, and well-defined outputs. The closer to an independent "box" these smaller pieces become the more modular the system. Modularization provides a systematic method for isolating problems and minimizes the effects of a module change upon other modules. A predictable module is one that operates identically the same each time the same input is supplied. The advantage of predictability to management visibility is obvious. In summary, modularity breaks complexity into a group of simpler pieces and enhances development and management visibility.

- Structured Design

Structured design combines both top down design and modularity. Its main emphasis is to break functions into modules such that the input, processing, and output of each module approximate a functional requirement. Those programs whose functional requirements are easily and clearly defined are most promising for using this approach.

TECHNIQUES

Techniques associated with software development are generally conceived to be of more detail than methodologies. An example to illustrate is; all pilots use the method of rudder and stick controls to approach the runway in crosswind conditions, yet some pilots use the technique of cross controlling while others use the technique of crabbing. Simply stated, a technique is the procedure by which a methodology is applied or implemented.

- Verification and Validation

It is possible to apply the technique of Verification and Validation (V&V) through use of an independent contractor or through the developing contractor. In either case, the process of V&V should be initiated long before the completion of the first pass through the development to take advantage of early identification of problems and their corrections. V&V is an attempt at determining the validity of computer programs but a

complete check of all variables and combinations is expensive and time consuming. Usually V& V encompasses software exercises, code checks, functional checks, requirements traceability, and other analytical considerations which give an indication of program quality. The process is most efficiently applied through use of a group that is independent from the design/code groups because it forces incremental proof of the validity of the developed programs.

- Organic and Inorganic Development

AFM 26-1 requires a study to be conducted that determines whether organic development will be accomplished or not. Subsequent to the determination, the actual task is undertaken. Most tasks are desirable for organic (in-house) development because of the residual expertise and experience left within the Air Force subsequent to the development. The demand for expertise levels and personnel resources sometimes exceed those resources available and thus the efforts are contracted to industry. If the tasks are contracted, then a mechanism for transferring needed expertise and system knowledge from the contractor to the Air Force must be identified and enforced. Quite often, a complex system is developed and the contractor and/or the Air Force do not make adequate allowance for knowledge transfer thus endangering long term support and operation of the system.

- Other Techniques

Sometimes it is desirable to break the development into more than one contract. This assigns the integration role to the Air Force or an integrating contractor as opposed to a prime contractor. Integration requires technical competence and should only be undertaken organically if the computer programs are almost independent of each other and a very low system risk is envisioned.

Seldom is it desirable to break the software away entirely from the hardware. Systems are growing more and more complex and the software plays such an integral role in the overall system that breaking hardware and software apart is impractical. The current tendency is toward "systems" considerations instead of "hardware" and "software" for development and support.

APPENDIX F

ANNOTATED BIBLIOGRAPHY

Aron, J. D., "The Program Development Process," 1974, Addison-Wesley Publishing Co., Reading, Mass.

Boehm, B.W., "Software Engineering," October 1976, TRW-SS-76-08.

This paper provides a definition of the term "software engineering" and a current state of the art and likely trends in the field. The paper is oriented primarily toward discussing the domain of applicability of techniques (where and when they work) rather than how they work in detail.

Horowitz, Ellis, editor, 1975, Addison-Wesley Publishing Co., Reading, Mass.

A consolidation of papers held in a 1974 seminar entitled "Modern Techniques for the Design and Construction of Reliable Software" held at the University of Southern California. Topics covered are: The High Cost of Software, Requirements Analysis, Design, Coding and Testing, Management and an extensive annotated bibliography.

Mullin, F. J., "Software Test Management," October 1977, TRW-SS-77-05.

This paper discusses the management of a modest to large software test effort. A large emphasis is placed upon planning as the key to success in developing and testing software.

Nelson, Eldred C., "Software Reliability," November 1975, TRW-SS-75-05.

This paper addresses the reliability of software as a major problem for computer applications and in computer based hardware systems. It discusses management practices, data collection and analysis, software development tools, and software test tools commonly used in developing reliable software.

Ralston, Anthony and C. L. Meek, eds., Encyclopedia of Computer Science, Van Nostrand Reinhold Co., New York, NY.

This book consolidates the inputs of some 200 leading authorities on every aspect of the field of computer science. The encyclopedia is also a dictionary, a glossary, and a handbook supplying answers to most any computer science question.

Weinwurm, George F., ed., On the Management of Computer Programming, Auerback Publishers Inc., New York, NY, 1970.

This book contains articles by several leading authorities on the problems, research, and forecast on management of computer programming. The book is an excellent overview of the 1970 computer programming management situation.

*U.S.Government Printing Office: 1980 — 657-084/27