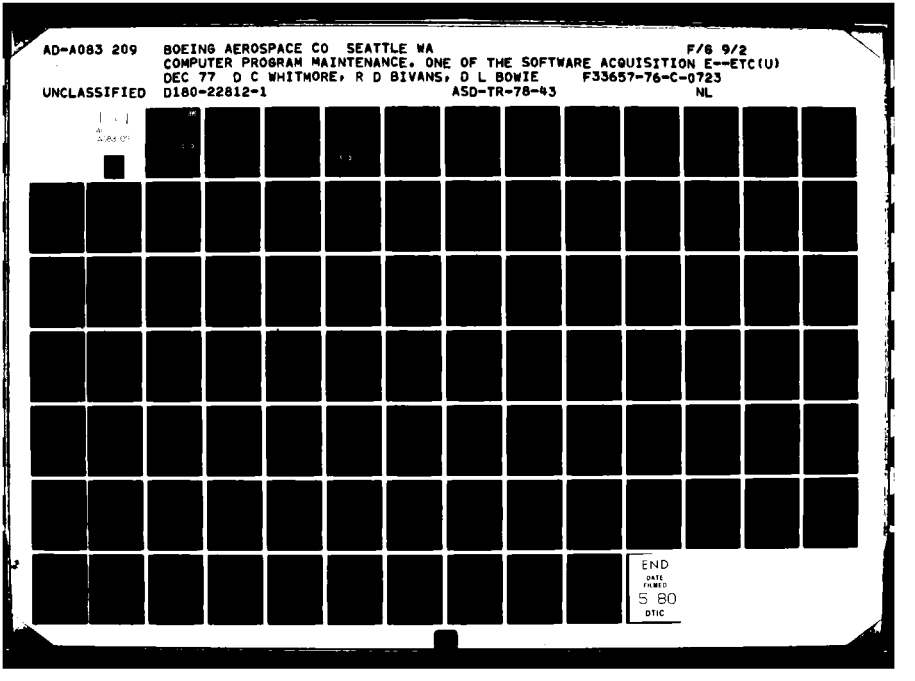
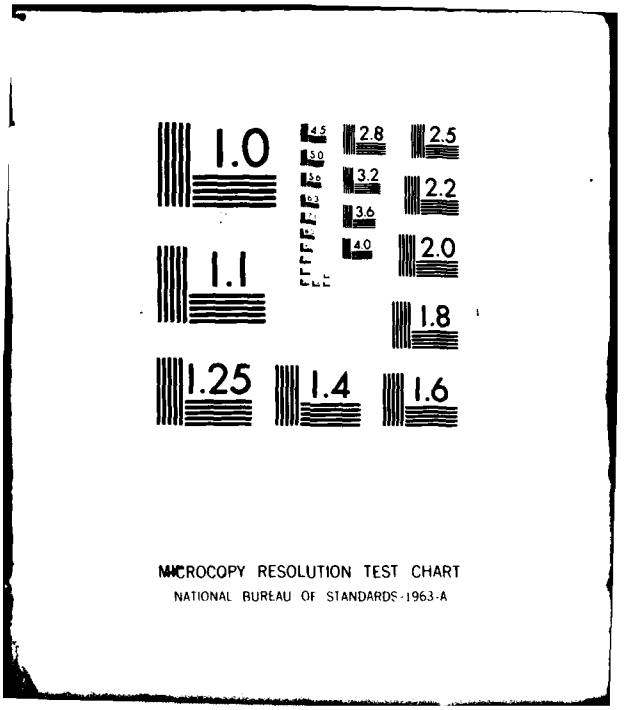

AD-A083 209 BOEING AEROSPACE CO SEATTLE WA F/6 9/2
COMPUTER PROGRAM MAINTENANCE. ONE OF THE SOFTWARE ACQUISITION E--ETC(U)
DEC 77 D C WHITMORE, R D BIVANS, D L BOWIE F33657-76-C-0723
UNCLASSIFIED D180-22812-1 ASD-TR-78-43 NL
END
DATE FILMED
5 80
DTIC


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
6.3
MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

ASD-TR-78-43

(2) LEVEL II

NW

COMPUTER PROGRAM MAINTENANCE
One of the Software Acquisition Engineering Guidebook Series

DIRECTORATE OF EQUIPMENT ENGINEERING
DEPUTY FOR ENGINEERING

DECEMBER 1977

TECHNICAL REPORT ASD-TR-78-43
Final Report

Approved for public release; distribution unlimited.

AERONAUTICAL SYSTEMS DIVISION
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433

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 Information Office (OI) 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.

Richard W. Ittelson

RICHARD W. ITTELSON,
Technical Advisor
Directorate of Equipment Engineering

Richard J. Sylvester

RICHARD J. SYLVESTER,
ASD Weapon Systems Computer Resource
Focal Point
Deputy for Engineering

FOR THE COMMANDER

John S. Kubin

JOHN S. KUBIN, Colonel, USAF
Director, Equipment 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 _____,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/56780/27 March 1980 — 400

(12) 92

UNCLASSIFIED

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

(19) REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
(18) 1. REPORT NUMBER ASD-TR-78-43	2. GOVT ACCESSION NO.	3. RECIPIENT'S CATALOG NUMBER
(6) 4. TITLE *(and Subtitle)* COMPUTER PROGRAM MAINTENANCE % One of the Software Acquisition Engineering Guidebook Series.		(9) 5. TYPE OF REPORT & PERIOD COVERED Final rept's
		(14) 6. PERFORMING ORG. REPORT NUMBER D180-22812-1
(10) 7. AUTHOR(s) D. C. Whitmore M. P. Kress R. D. Bivans D. L. Bowie		(15) 8. CONTRACT OR GRANT NUMBER(s) F33657-76-C-0723
9. PERFORMING ORGANIZATION NAME AND ADDRESS Boeing Aerospace Company PO Box 3999 Seattle, Washington 98124		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS PE64740F (16) Project 2238
11. CONTROLLING OFFICE NAME AND ADDRESS HQ ASD/ENE Wright-Patterson AFB, Oh 45433		(11) 12. REPORT DATE Dec [redacted] 77 13. NUMBER OF PAGES 100
14. MONITORING AGENCY NAME & ADDRESS*(if different from Controlling Office)*		15. SECURITY CLASS. *(of this report)* UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT *(of this Report)*

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)*

Software Acquisition, Acquisition Engineering, Software Maintenance, Computer Program Maintenance, Software Change Management, Software Change Control, Software Certification, Software Interim Contractor Support, Software Life Cycle Costs, Support 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 Personnel and other USAF acquisition engineers in the acquisition engineering of software for Automatic Test Equipment and Training Simulators. This guidebook describes the software maintenance life cycle, including maintainability, maintenance tasks and required maintenance resources.

DD FORM 1 JAN 73 1473

UNCLASSIFIED

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

059610

FOREWORD

This guidebook was prepared as part of the Software Acquisition Engineering Guidebooks contract, F33657-76-C-0723. It describes the software maintenance life cycle; maintainability attributes; detailed planning and maintenance tasks; and required resources. Responsibilities of the software acquisition engineer and development contractor are identified. The ground systems under specific consideration are training simulators and automatic test equipment.

This guidebook is one of a series intended to assist the Air Force Program Office and engineering personnel in software acquisition engineering for automatic test equipment and training simulators. Titles of other guidebooks in the series are listed in the introduction. These guidebooks will be revised periodically to reflect changes in software acquisition policies and feedback from users.

This guidebook reflects an interpretation of DOD directives, regulations and specifications which were current at the time of guidebook authorship. Since subsequent changes to the command media may invalidate such interpretations, the reader should also consult applicable government documents representing authorized software acquisition engineering processes. This guidebook contains alternative recommendations concerning methods for cost-effective software acquisition. The intent is that the reader determine the degree of applicability of any alternative based on specific requirements of the software acquisition with which he is concerned. Hence, the guidebook should only be implemented as advisory rather than as mandatory or directive in nature.

This guidebook was prepared by the Boeing Aerospace Company.

This Software Acquisition Engineering Guidebook is one of a series prepared for Aeronautical Systems Division, Air Force Systems Command, Wright-Patterson AFB OH 45433. Inquiries regarding guidebook content should be sent to ASD/ENE, Wright-Patterson AFB OH 45433. The following list presents the technical report numbers and titles of the entire Software Acquisition Engineering Guidebook Series. Additional copies of this guidebook or any other in the series may be ordered from the Defense Documentation Center, Cameron Station, Alexandria VA 22314.

ASD-TR-78-43,	Computer Program Maintenance
ASD-TR-78-44,	Software Cost Measuring and Reporting
ASD-TR-78-45,	Requirements Specification
ASD-TR-78-46,	Computer Program Documentation Requirements
ASD-TR-78-47,	Software Quality Assurance
ASD-TR-78-48,	Software Configuration Management
ASD-TR-78-49,	Measuring and Reporting Software Status
ASD-TR-78-50,	Contracting for Software Acquisition
ASD-TR-79-5042,	Statements of Work (SOW) and Requests for Proposal (RFP)
ASD-TR-79-5043,	Reviews and Audits
ASD-TR-79-5044,	Verification, Validation and Certification
ASD-TR-79-5045,	Microprocessors and Firmware
ASD-TR-79-5046,	Software Development and Maintenance Facilities
ASD-TR-79-5047,	Software Systems Engineering
ASD-TR-79-5048,	Software Engineering (SAE) Guidebooks Application and Use

TABLE OF CONTENTS

SECTION	TITLE	Page
1.0	INTRODUCTION	1
	1.1 Purpose	1
	1.2 Scope	1
	1.3 TS and ATE Overview	2
	1.3.1 TS System Characteristics	2
	1.3.2 ATE System Characteristics	2
	1.4 Guidebook Organization	5
2.0	APPLICABLE DOCUMENTS	7
3.0	SOFTWARE MAINTENANCE OVERVIEW	9
	3.1 Software Maintenance and Maintainability	9
	3.2 Maintenance Life Cycle	14
	3.3 Software Maintenance Tasks	17
	3.4 Unique Considerations for ATE	23
	3.5 Unique Considerations for TS	25
4.0	PLANNING FOR SOFTWARE MAINTENANCE	27
	4.1 Software Maintenance Planning Process	27
	4.1.1 Maintenance Analysis and Planning	27
	4.1.2 Life Cycle Cost Trades	28
	4.1.3 Resource Planning	30
	4.1.4 Organization Responsibilities and Interfaces	32
	4.1.5 Formal Reviews of Maintenance Plans	32
	4.2 Maintenance Planning Documentation	33
	4.3 ATE and TS Variants	34
	4.3.1 ATE Software	34
	4.3.2 TS Software	37
5.0	SOFTWARE MAINTENANCE ACTIVITY	39
	5.1 TS Software Maintenance	39
	5.1.1 Introduction	39
	5.1.2 Processing of Authorized Software Changes	42
	5.1.3 Specific Software Maintenance Tasks	42
	5.1.4 Pitfalls to Avoid in TS Software Maintenance	47

TABLE OF CONTENTS (Cont.)

SECTION	TITLE	Page
	5.2 ATE Software Maintenance	49
	5.2.1 Introduction	49
	5.2.2 Major Areas of ATE Software	49
	5.2.3 Attributes of Maintainable ATE Software	49
	5.2.4 Examples and Sources of Changes to ATE Software	50
	5.2.5 Development and Maintenance of ATE Test Software	51
	5.2.6 Debugging Test Software	60
	5.2.7 Control of ATE Software	61
	5.2.8 Pitfalls to Avoid During ATE Software Maintenance	61
	5.3 Change Management	61
	5.3.1 Change Management Definition	62
	5.3.2 Baseline Management	62
	5.3.3 Support and Control Software	63
	5.3.4 Change Management Reports, Documents	64
	5.3.5 Change Control	65
	5.3.6 Configuration Accounting	65
	5.3.7 Software Change Control Board	65
	5.3.8 Responsibilities of USAF and Contractors	67
	5.3.9 Maintainability of Source Program Code and Documentation	67
	5.3.10 Typical TS Change Management System	67
6.0	SPECIAL REQUIREMENTS FOR SOFTWARE MAINTENANCE	69
	6.1 Maintenance Support Environment	69
	6.1.1 Computer Aids to Software Maintenance	69
	6.1.2 Types of Computer Aids	69
	6.2 Program Management Transfer and System Turnover	70
	6.2.1 Major Elements of Transfer/Turnover Planning	71
	6.2.2 Transfer/Turnover Planning Process	71
	6.2.3 Interim Contractor Support and Life Cycle Costs	73
	6.3 Software Maintenance Training	73
7.0	BIBLIOGRAPHY	77
8.0	MATRIX: GUIDEBOOK TOPICS VERSUS GOVERNMENT DOCUMENTS	79
9.0	GLOSSARY OF TERMS	83
10.0	ABBREVIATIONS AND ACRONYMS	87
11.0	SUBJECT INDEX	89

LIST OF FIGURES

FIGURE	TITLE	Page
1.3-1	Typical Crew Training Simulator	3
1.3-2	Typical ATE Configuration	4
3.2-1	Software Maintenance Cost Escalation	15
3.2-2	Software Maintenance Life Cycle	16
3.3-1	Software Maintenance Process	22
4.1-1	Maintenance Resource Planning Considerations	31
5.1-1	Typical Software Change Request Flow	43
5.1-2	Static Verification Program Operational Flow	46
5.1-3	Off Line and Real Time Processing Facilities	48
5.2-1	ATE Test Software Requirements	53
5.2-2	ATE Test Software Concept Definition - Part 1	54
5.2-3	ATE Test Software Concept Definition - Part 2	55
5.2-4	ATE Test Software Generation	57
5.2-5	ATE Test Software Integration/Validation	58
5.2-6	ATE Test Software System Organization	59
5.3-1	Typical Class I Change Processing Flow	66
6.2-1	Transfer/Turnover Schematic	72
6.2-2	Life Cycle Cost Comparison for ICS Policy Decision	74
8.0-1	Guidebook Topics Versus Government Documentation	80

LIST OF TABLES

TABLE	TITLE	Page
3.1-1	Software Maintenance Factors	11
3.1-2	Attributes of Maintainable Software	12
3.1-3	Programmer Errors by Source	14
3.2-1	Software Maintenance Checklist for Technical Reviews	18
4.1-1	TS Complexity Options	29
4.2-1	Data Item Descriptions (DID) Applicable to Software Maintenance	35
5.1-1	Typical TS Software Changes	41

Section 1.0 INTRODUCTION

Maintenance of computer programs involves any activity which alters previously developed software. Such changes result from error removal, operating improvements, and changes in system requirements. The software maintenance activity is often thought to begin primarily after the computer program is delivered to the ultimate user. However, maintenance planning begins early in the conceptual phase and maintenance procedures are activated early in software development. Also, software maintenance is not limited to modifying program code; it encompasses all attendant documentation.

The cost of software maintenance is a major consideration in system engineering and it is a large component of weapon system life-cycle costs. Thus, software maintenance is an important factor in system cost-effectiveness design trades and careful attention must be paid to both the scheme for software maintenance and the acquisition of maintainable software. The particular systems with which this guidebook is concerned are Automatic Test Equipment (ATE) and Training Simulators (TS).

1.1 PURPOSE

The primary purpose of this guidebook is to assist AF engineering personnel directly responsible for TS and ATE software acquisition to ensure the capability for the long-term maintenance activities for this software are successfully monitored and performed. The guidebook should also be helpful to Air Force managers responsible for the procurement of the total TS or ATE systems.

1.2 SCOPE

This is one of a series of guidebooks related to the Software Acquisition Engineering (SAE) process for TS and ATE ground-based system. Other SAE guidebook titles are listed below:

- Software Cost Measuring and Reporting
- Contracting for Software Acquisition
- Statement of Work (SOW) and Requests for Proposal (RFP)
- Regulations, Specification and Standards
- Measuring and Reporting Software Status
- Computer Program Documentation Requirements
- Software Quality Assurance
- Verification
- Validation and Certification
- Requirements Specification
- Configuration Management
- Reviews and Audits
- Management Reporting by the Software Director
- Software Configuration Management

The subject of software maintenance cannot be adequately treated without reference to many of the topics listed above e.g.; Computer Program Documentation and Configuration Management. Thus, salient information from these other subject areas is included or referenced in this guidebook. The principal emphasis in this guidebook is on maintenance planning, maintenance activities and special requirements (e.g.: facilities, maintenance tools etc.).

Maintenance of ATE and TS software is also treated in the other guidebooks. For example, the cost of software maintenance is discussed in the Software Cost Measuring and Reporting guidebook. The scope of this guidebook is consequently limited to exclude redundant detail covered in other guidebooks. This guidebook, however, provides the basic resource material on maintenance planning and maintenance process, with particular regard to TS and ATE systems.

1.3 TS AND ATE OVERVIEW

The purpose of this section is to provide a brief sketch of TS and ATE system characteristics, including the function of the software associated with each.

1.3.1 TS System Characteristics

The TS system is a combination of specialized hardware, computing equipment, and software designed to provide a synthetic flight and/or tactics environment in which aircrews learn, develop and improve the skills associated with individual and coordinated tasks in specific mission situations. Visual, aural, and/or motion systems may be included. Figure 1.3-1 depicts a typical TS which employs digital processing capability.

The computer system, integral to the crew training simulator, can consist of one or more general purpose computers. The computing hardware operates with floating point arithmetic and sufficient bit capacity to provide efficient use of a simulator High Order Language (HOL).

When a multi-processor/multi-computer system is used, it must be designed such that computers can operate simultaneously and are controlled/synchronized by a single program (supervisor/ executive). The executive directs program execution and regulates priorities.

The simulator (1) accepts control inputs from the trainee (via crew station controls) or from the instructor operation station; (2) performs a real-time solution of the simulator mathematical model; and (3) provides output responses necessary to accurately represent the static and dynamic behavior of the real world system (within specified tolerance and performance criteria).

Since TS simulators consist of interdependent hardware and software, a joint hardware/software development effort is required. As the complexity of TS increases, simulation software continues to grow in complexity, size, and cost. Software costs can and do exceed computer hardware costs in many cases. Therefore, it is imperative that the simulation software acquisition engineering process be subjected to formal system engineering planning and discipline to ensure cost-effective procurement.

1.3.2 ATE System Characteristics

ATE is defined as that ground support system which performs vigorous system tests with minimum manual intervention. ATE is used in place of manual devices because it is more cost-effective, provides required repeatability, or repair of the item being tested requires the speed which only an automatic tester can achieve (e.g., the complex initialization routine that is normally required prior to testing a digital unit).

Figure 1.3-2 shows the typical components of an ATE system. Note that there are both hardware and software elements involved. Most of the elements shown in the figure will be found in the majority of ATE systems, although the packaging and interface design may vary between specific systems.

The controls and displays section consists of the computer peripheral devices such as control panels, magnetic tape cassettes or disks, a cathode ray tube (CRT), keyboard, and small printer. The computer (normally a minicomputer), as controlled by software, operates the peripheral devices; switches test stimuli on and off; and measures responses of the Unit Under Test (UUT) (comparing to predetermined values). The operator maintains supervisory control of the testing process through the peripherals. However, his interaction is usually minimal since, by definition, the automatic test feature was selected in preference to an operator-controlled test system.

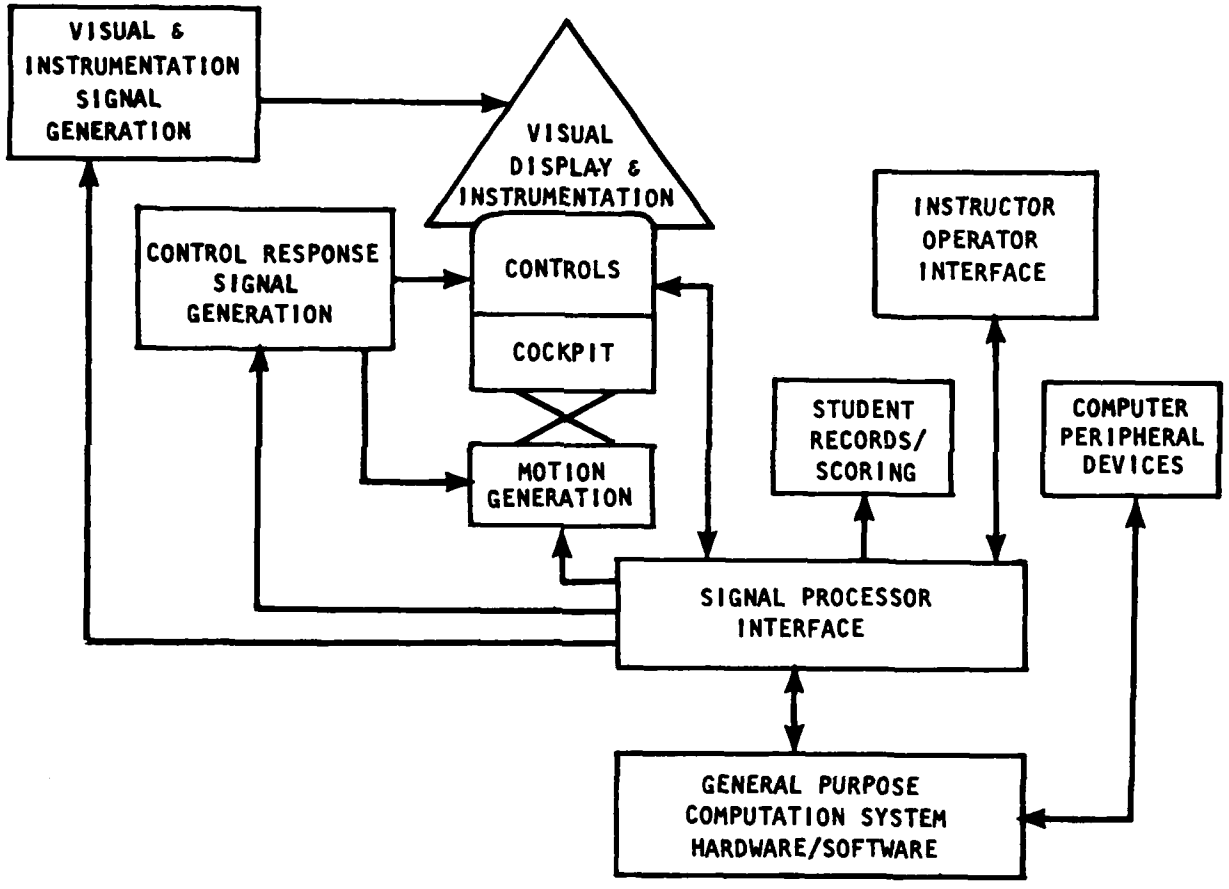


Figure 1.3-1. Typical Crew Training Simulator

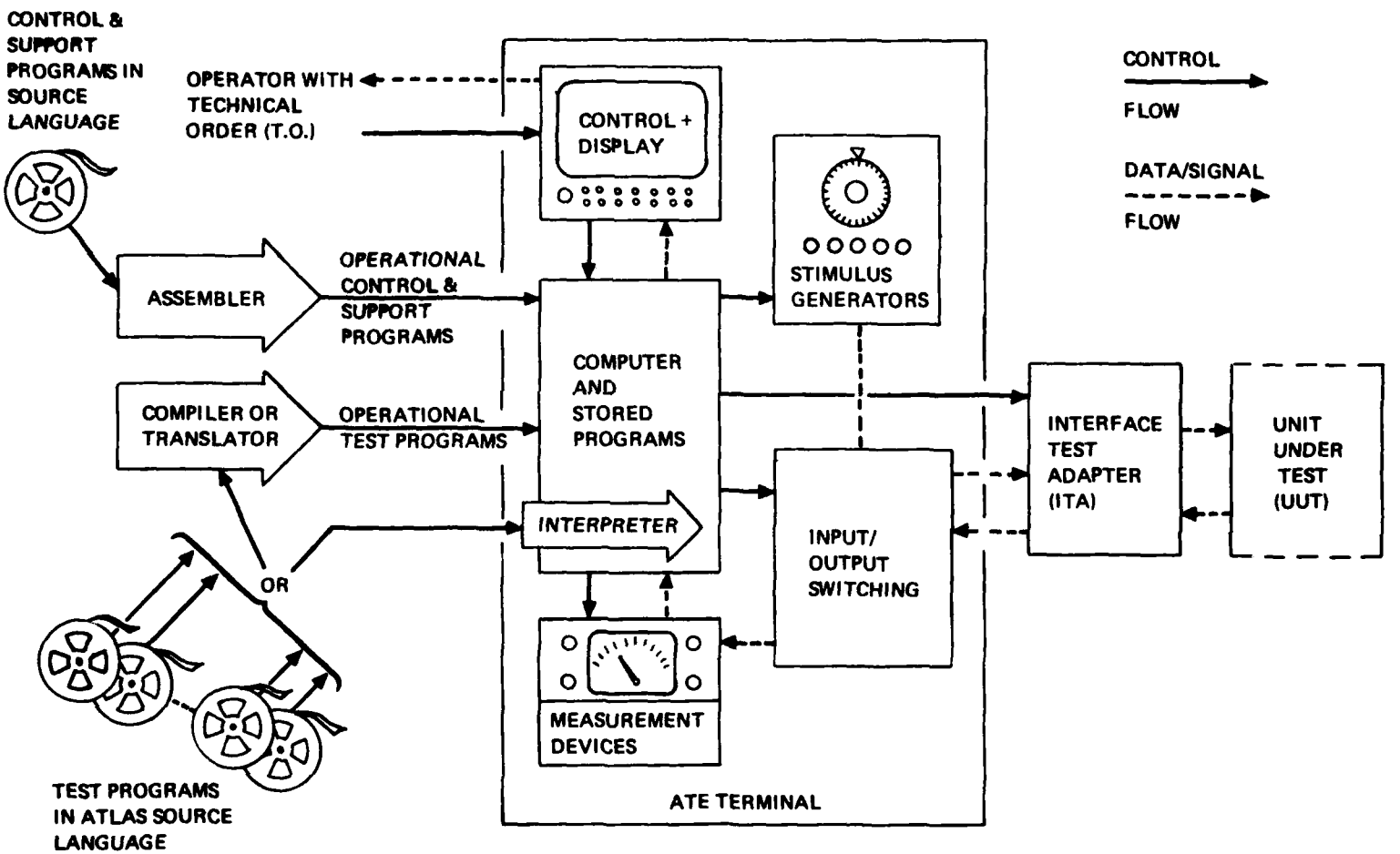


Figure 1.3-2. Typi ATE Configuration

ATE is normally designed to accommodate testing several different articles of system equipment (normally one at a time). The maintenance level being supported by the ATE is determined by logistics systems analysis.

The importance of the software portion of the ATE system should not be minimized since both the application of the test stimuli and the measurement of the result are achieved via software. Arbitrary function generation and complicated wave analysis can also be accomplished by software and is becoming more prevalent in the ATE systems. The cost of ATE software is a significant component of total ATE costs and design trades can be performed to minimize ATE lifecycle costs.

1.4 GUIDEBOOK ORGANIZATION

Section 1.0 of this guidebook contains introductory material about the guidebook, including guidebook purpose and scope, and the guidebook's relationship to the other SAE guidebooks. It provides a brief description of typical ATE and TS systems and describes the organization and use of the guidebook.

Section 2.0 is a list of key government documents that were referenced in the preparation of this guidebook. Sections 3.0 through 6.0 contain acquisition guidelines relative to computer program maintenance; both those which are common to most systems and those which are unique to TS and ATE systems. Section 3.0 provides an overall perspective on the subject of computer program maintenance, including identification of specific maintenance activities. Major elements in planning for software maintenance are also described in Section 3.0, as well as an introduction to the unique considerations for ATE and TS systems.

Section 4.0 expands on the topic of planning for software maintenance - especially those activities recommended to occur early in system development. Specific planning tasks and documentation are described, including major cost trades. Section 5.0 concerns the actual activity of software maintenance on ATE and TS systems, with particular emphasis on change management. Section 6.0 contains a series of discussions on various special requirements for software maintenance; e.g., organization and facilities, program turnover, and maintenance training.

Section 7.0 is a bibliography of documents that are generally applicable to the subject of computer program maintenance. This section is an expansion of Section 2.0, referenced documentation and the listed documents augment the material covered in this guidebook. Section 8.0 provides a matrix tabulation for the cross reference relationship between guidebook topics and corresponding government documents. Sections 9.0 and 10.0 contain, respectively, a glossary of selected terms used in the guidebook and the expansion of all abbreviations and acronyms used in the guidebook. Section 11.0 is a detailed subject index indicating which guidebook paragraphs address the specific topics.

Section 2.0 APPLICABLE DOCUMENTS

The following documents bear directly on the topic of computer program maintenance for ATE and TS software:

DOD 5000.29, Management of Computer Resources in Major Defense Systems, 26 April 1976

AFR 800-14 Vol. II, Acquisition and Support Procedures for Computer Resources in Systems, 26 September 1975

MIL-STD-483, Configuration Management Practices for Systems, Equipment, Munitions, and Computer Programs, 1 June 1971

MIL-STD-83468, Digital Computational System for Real-Time Training Simulators, 12 December 1975

AFLC Regulation 66-37, Management of Automated Test Systems, 24 October 1975

AFM 50-2, Instructional System Development

AFP 50-58, Handbook for Designers of Instructional Systems, Vol. 1-5

AFP 66-14, Equipment Maintenance Policies, Objectives, and Responsibilities

AFR 57-4, Retrofit Configuration Changes

AFR 800-19, System or Equipment Turnover

AFR 800-11, Life Cycle Costing

AFR 800-4, Transfer of Program Management Responsibility

MIL STD-470, Maintainability Program Requirements

MIL STD-1521A, Technical Reviews and Audits for Systems, Equipment, and Computer Programs

AFR 65-3, Configuration Management

AFR 800-8, Integrated Logistics Support (ILS) Program for Systems and Equipment

AFR 800-12, Acquisition of Support Equipment

AFR 800-21, Interim Contractor Support for Systems and Equipment

MIL STD-471, Maintainability Verification/Demonstration/Evaluation

Applicable Data Item Descriptions (DID's) are listed in Table 4.2-1.

Section 3.0 SOFTWARE MAINTENANCE OVERVIEW

This section constitutes an overview of software maintenance considerations in ATE and TS system acquisition. The overview includes:

a. A discussion of what is meant by software maintenance and how maintainable software can be developed (paragraph 3.1).

b. A description of the software maintenance life cycle - from early planning and analysis through system operational deployment (paragraph 3.2).

c. An identification of the specific tasks involved in various types of software maintenance (paragraph 3.3).

d. An introduction to major differences between ATE and TS that have unique impact on software maintenance (paragraphs 3.4 and 3.5).

This overview provides a context and working language for the descriptive sections that follow: Maintenance Planning (Section 4.0); Maintenance Activity (Section 5.0); and Special Requirements (Section 6.0).

Before proceeding with the overview, the concept of "software maintenance" is defined. Software maintenance may mean different things to different persons, but a generally accepted idea is that it pertains to any activity that alters previously developed software (coding, documentation, data base). This simple definition avoids the complication of describing what kinds of software changes are included under "maintenance" or having to specify when software maintenance occurs during the system life cycle. It encompasses all types of software changes, regardless of cause or scope, and it applies throughout the system life cycle; from the first time a set of established code is changed or corrected during initial development until the software is finally deactivated from operational deployment.

Although software maintenance is often thought to begin primarily after the software has been delivered to the ultimate user, there are good reasons to consider software maintenance starting very early in system development:

a. Once software development is placed under change control, software maintenance procedures must be fully activated.

b. Software maintenance is a fundamental factor in life cycle cost trade analyses performed in early system concept formulation.

c. Planning for software maintenance (organization, facilities, tools, skills, etc.) begins before the System Requirements Review (SRR).

d. Development of maintainable software is an important consideration in deriving software requirements and software maintainability should be demonstrated during system development.

3.1 SOFTWARE MAINTENANCE AND MAINTAINABILITY

The subject of software maintenance, as regards software acquisition engineering, has two major aspects:

a. Maintenance of software

b. Development of maintainable software

The first aspect concerns how software is to be changed (corrected, updated, etc., including plans for maintenance facilities, equipment, personnel, etc.). The second aspect concerns how software can be designed and developed so that it is easily maintained. Both aspects of

software maintenance include all three components of software: (1) computer program code, (2) program documentation, and (3) initialization data base. Program firmware is also included under software maintenance, as it does not belong generically to computer hardware in the maintenance sense (i.e., changes to program code).

The distribution between "maintenance of software" and "maintainable software" is clarified in Table 3.1-1 by listing the factors or considerations associated with the two maintenance aspects. Maintenance of software is described by a list of planning elements and by the sequence of tasks to accomplish a typical software change. The activity of developing maintainable software is described by a list of major considerations/objectives.

Maintenance of software, the first aspect, is discussed at length in Sections 4.0 through 6.0. The development of maintainable software, primarily the responsibility of the system contractor, is discussed in the balance of this section. Computer aids to software maintenance (related to maintainability but also a resource for software maintenance) are discussed in paragraph 6.1.

Although the development contractor has principal responsibility for developing maintainable software, the Air Force has an important role in defining maintainability requirements, planning resources for enhancing maintainability, and monitoring contractor efforts to provide software maintainability. Since the Air Force will usually acquire organic software maintenance capability, it has substantial interest in assuring that maintainable software is developed.

Maintainability is a somewhat relative term, however, it is possible to differentiate between good and poor software maintainability. That is, the characteristics of software that enhance or thwart easy maintenance have been identified by various studies and surveys. For example, a 1971 study (Reference 8, Bibliography) contracted by the Electronic Systems Division, AFSC, included an analysis of factors that inhibit the effectiveness of maintenance programmers. One researcher concluded that software is often not configured to harmonize with human traits. For instance, the programmer may have difficulty in finding information in the code and documentation which is relevant to solution of a specific maintenance problem. One solution is to provide conceptual groupings in the code and in the documentation which are coherent with each other.

The problem of software maintainability is closely allied with sources of programmer errors. A study by Youngs, reported in Reference 6, Bibliography, produced an analysis of error sources. A statistical breakdown is shown in Table 3.1-3. These error sources underline the importance of good coding structure and programming style. Reference 6 also has an extensive compilation of language-type errors for different languages.

The Air Force roles in acquiring maintainable software (identified earlier) can be facilitated by use of a checklist on "Attributes of Maintainable Software." There is presently no industry standard for such a list, however, the checklist provided in Table 3.1-2 includes attributes that are frequently mentioned in the literature. Improvements to the list may become apparent through consistent use.

Procurement of maintainable software requires that system requirements be prepared and then produced software be evaluated against those specific requirements. Otherwise, software maintainability is left to the whim of the development contractor. Improved methods for specifying and measuring software maintainability are needed. Reference 1, Bibliography, discusses this problem and

Table 3.1-1. Software Maintenance Factors

MAINTENANCE OF SOFTWARE	Planning	• Life cycle costs trades • System design maintenance requirements • Maintenance policy/procedures • Maintenance support environment (facilities, org, resources)
	Maintenance Tasks (Typical Update)	• Definition of required change; impact assessment • Change approval and control • Program structure analysis; modification design • Code and debug • Verification, validation, and certification • Data base and documentation update
DEVELOPMENT OF MAINTAINABLE SOFTWARE	• Maintainability criteria and quantitative requirements • Specific design and documentation standards/guidelines • Computer aids to software maintenance • Maintainability assessment at PDR and CDR • Maintainability demonstration; maintenance records analysis	

Table 3.1-2. Attributes of Maintainable Software (Sheet 1 of 2)

_____	MODULARITY. Good modularity provides:
_____	(a) Input/output separation from computation
_____	(b) Singular functions and limited size modules
_____	(c) Simple interfaces between modules
_____	ORGANIZATION. The program design and documentation are similarly organized and provide:
_____	(a) Top down structure
_____	(b) Conceptual groupings
_____	(c) Straightforward, easy to understand design
_____	(d) Common data base
_____	(e) Common symbology and cross-referencing between documentation and source code
_____	LANGUAGE. The programming launguage and its use is matched to user experience and provides:
_____	(a) Meaningful syntax with respect to intent of reverenced entities
_____	(b) Use of HOL
_____	(c) Limited restrictions and exceptions
_____	(d) Compactness between references and referent
_____	(e) Transferability between computers
_____	NOTATION. The notation style provides:
_____	(a) Frequent comments and brief identification of conceptual groups
_____	(b) Use of parentheses to clarify order of computation
_____	(c) Variable names that give mnemonic clues to meaning of variable and avoid duplication or confusion

Table 3.1-2. Attributes of Maintainable Software (Sheet 2 of 2)

_____	EXPLANATION. The documentation provides:	
_____	(a)	Clear statement of top-level and derived requirements behind software design
_____	(b)	Concise discussion of purpose and strategy for a module or interface organization
_____	(c)	Good cross-referencing for all media

MIL-STD-471 addresses to maintainability verification, demonstration, and evaluation.

Table 3.1-3. Programmer Errors by Source

	% of all Errors
"Assignment", accounting for	27
"Allocation", accounting for	15
"Iteration", accounting for	9
"I/O except formatting", accounting for	7
"I/O formatting", accounting for	6
"Parameter/subscript list", accounting for	5
"Conditional execution", accounting for	5
"Vertical delimiter", accounting for	4

3.2 MAINTENANCE LIFE CYCLE

As mentioned in the previous section, it is appropriate to consider software maintenance activity as starting early in systems development. The point was made that planning for software maintenance begins even before the Systems Requirements Review (SRR). The importance of early planning and the progression of software maintenance activities over the system life cycle are principal topics covered in this section. A checklist of software maintenance factors to be considered at formal reviews (SSR, System Design Review (SDR), Preliminary Design Review (PDR) and Critical Design Review CDR) is also included in this section.

Early planning for software maintenance has several objectives, for example:

a. Contribute to preliminary design concepts, e.g., via life-cycle cost trades. These trades, in turn, impact software maintenance policies and procedures.

b. Establish design requirements, e.g., for maintainable software.

c. Specify resources and policy/procedures for software maintenance throughout development and deployment phases.

The importance of good planning (including studies and analyses) can hardly be overstressed because of the high leverage that such planning has on long-term costs. Figure 3.2-1 illustrates, on a relative basis, the exponential rise in software maintenance costs over the course of system development and operation. Software modification or error correction becomes so costly in later phases that high penalties are paid for poor maintenance during early development. One reason for the high cost of late maintenance is the "domino effect" of changes propagating through an integrated software system. This "domino effect" can cause software maintenance programmers to contemplate a complete redesign rather than attempting major modifications. In other words, it is sometimes less costly to start over than it is to modify an old design.

Software maintenance planning and activities are not performed in a vacuum; they are accomplished in concert with other system development/operations activities. An illustration of the maintenance life cycle is provided in Figure 3.2-2, in which software maintenance activites are shown as a function of system phase and along with interrelated activities. Also, identified in Figure 3.2-2 are principal documents that can be impacted by software maintenance considerations

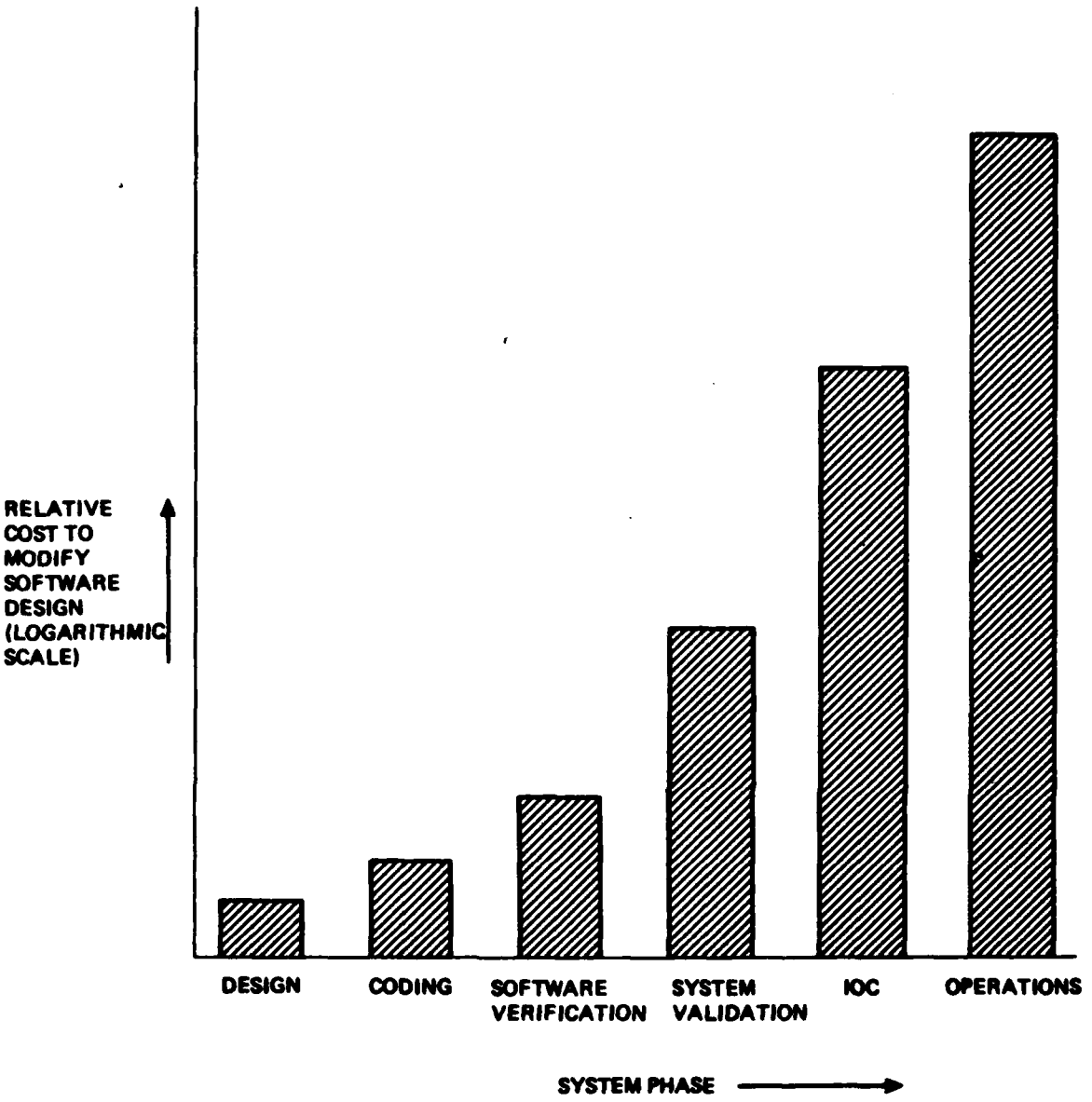


Figure 3.2-1. Software Maintenance Cost Escalation

PROGRAM ACTIVITY (PHASE)	CONCEPTUAL	VALIDATION	FULL SCALE DEVELOPMENT		DEPLOYMENT / OPERATIONS
PRINCIPAL SOFTWARE MAINTENANCE ACTIVITIES	• MAINTENANCE ENGINEERING ANALYSIS • LIFE CYCLE COST TRADES • SYSTEM DEVELOPMENT REQUIREMENTS • MAINTENANCE RESOURCE PLANNING	• SYSTEM DESIGN TRADES • DESIGN RQMTS SPECIFICATIONS • DETAILED MAINT. PLANNING • PROPOSAL EVALUATION • DEVELOPMENT MANAGEMENT POLICY & PROCEDURES	• SOFTWARE CORRECTIONS • MAJOR MODIFICATIONS; VV&C • DOCUMENTATION UPDATE • MANUALS PREPARATION • MAINTENANCE TRAINING • MAINTAINABILITY EVALUATION • MAINTENANCE PLANNING UPDATE • MAINTENANCE TESTING & DEMONSTRATION		• MAINTENANCE RESOURCE IMPROVEMENTS • SOFTWARE CORRECTIONS • MAJOR MODIFICATIONS • DOCUMENTATION UPDATE • DATA BASE UPDATE • VV&C • MAINTENANCE TRAINING
MAJOR REVIEWS & AUDITS	SRR	SDR	PDR CDR	FCA FQR PCA	IOC
MAINTENANCE-RELATED DOCUMENTS	ROC DCP WS SPEC SOW/RFP CDRL CRISP	PART I SPEC CPDP CMP TEST PLANS TECH PROPOSALS CRISP	DRAFT. PART II SPEC INTERFACE DESIGN DESCRIP CONFIGURATION INDEX CHANGE STATUS LIST PROGRAMMER MANUAL USERS MANUAL FINAL PART II SPEC	VDD SCN TEST PROCEDURES TEST REPORTS ECP CRISP	PART II SPEC PROGRAMMER MANUAL USERS MANUAL VDD INTERFACE DESCRIPTION CHANGE STATUS LIST CONFIGURATION INDEX SCN

Figure 3.2-2. Software Maintenance Life Cycle

or are closely allied with maintenance planning and activities. This chart provides an overview of the software maintenance role in the system acquisition process and it lists specific activities directly related to software maintenance. The planning activities, especially during the conceptual and validation phases, are described in Section 4.0. The processes of TS and ATE software maintenance, both during the following full-scale development, are described in Section 5.0.

Figure 3.2-2 also notes the major reviews and audits associated with system acquisition. The role of software maintenance is further specified in Table 3.2-1 which is checklist designed to be used at SRR, SDR, PDR, and CDR. This checklist was derived from MIL-STD-1521A and should prove useful in assuring that software maintenance is given appropriate and timely attention.

3.3 SOFTWARE MAINTENANCE TASKS

The process of performing software maintenance on TS and ATE systems is described in Section 5.0. However, the basic tasks involved in software maintenance are introduced in this section.

Software maintenance has been defined as any activity that alters previously developed software and software is said to include program code, documentation, and data bases. The specific procedures followed in software maintenance depend on both the nature of change to be incorporated and on the particular maintenance environment (resources, organizational responsibilities, phase of system development/deployment, etc.). Aside from specific variations in procedure, a general sequence of tasks is as depicted in Figure 3.3-1. This generalized process involves the following:

a. Change requirement. The need for a software modification is defined. Causes for change include:

(1) Coding errors detected from tests

(2) Module design problems

(3) New system requirements; e.g., capabilities; missions

(4) Operating improvements; e.g., timing

(5) Interface problems between modules, components, CPCI's and/or system hardware.

b. Change analysis and impact assessment. Alternative solutions for implementing the required change are identified and evaluated. Impact of change on other software and hardware components is assessed.

c. Change approval and planning. The proposed change is approved by a change control authority; e.g., change board. Plans for the modification are specified in terms of resources (equipment, personnel, etc.), schedules, and policy (e.g., selection of control procedures, required testing, etc.).

d. Change control procedures. The specific procedures and organizational responsibilities for controlling the modification are implemented.

e. Modification design. Design changes based on program structure analysis, are specified and documented.

f. Code and checkout. New or revised modules/components are coded and then tested in a stand-alone mode. Coding is debugged as required.

g. Verification, validation, and certification (V,V&C). The modified CPCI is verified and then validated when integrated with the full hardware/software system. If the system had been previously certified, then it would be recertified. All documentation, data bases, etc.,

Table 3.2-1. Software Maintenance Checklist for Technical Reviews (Sheet 1 of 4)

SYSTEM REQUIREMENTS REVIEW (SRR)

_____ Does the Integrated Logistics Support Analysis (LSA) adequately reflect software maintenance requirements? Do the LSA data provide sufficient input to define design requirements?

_____ Are the reliability and maintainability analyses adequate for establishing software design requirements and software maintenance planning?

_____ Do the life cycle cost analyses include system design/ maintenance trades?

_____ Are configuration management plans sufficiently complete to assure quality software development and maintenance?

_____ Is software maintenance adequately considered in system requirements trade-off analyses?

SYSTEM DESIGN REVIEW (SDR)

_____ Is software maintainability considered for mitigating technical program risk?

_____ Is the contribution of software maintenance to program concepts; unit costs, equipment and facility requirements; skill requirements identified?

_____ Do trade studies include:

_____ (a) Operation design vs maintenance design

_____ (b) Automated vs manual operation

_____ (c) Built In Test Equipment (BITE) vs separate support equipment

_____ (d) Life cycle costs vs programming language

_____ (e) Life cycle costs vs system performance

_____ (f) Functional breakdown between hardware, computer programs, firmware and personnel/procedures

_____ Is the software maintenance support concept fully specified?

Table 3.2-1. Software Maintenance Checklist for Technical Reviews (Sheet 2 of 4)

_____	Are requirements adequately defined for:
_____	(a) System design to support software maintenance
_____	(b) Design of maintenance equipment
_____	(c) Software maintenance facilities
_____	(d) Software maintenance personnel and training
_____	Is particular attention given to:
_____	(a) Corrective and preventive maintenance requirements
_____	(b) Special equipment, tools, material requirements
_____	(c) Maintenance analysis compatibility with weapon deployment constraints
_____	(d) Maintenance procedures for different types of software maintenance
_____	Are all computer program Configuration Items (CI) identified? (Compilers, simulators, etc.) Are software maintenance requirements specified for each type?
_____	Is design and planning for software maintenance based on size and operating characteristics of the computer programs?
_____	Are all programming languages identified that will be used in the system and is the impact on maintenance described for each language?

PRELIMINARY DESIGN REVIEW (PDR)

_____	If off-the-shelf software is included, are maintainability requirements adequately considered?
_____	Has maintenance of the hierarchical structures of the computer programs been given consideration?
_____	Are calibration requirements satisfactorily met in software design?
_____	Have trade-offs made prior to SDR been updated and verified?

Table 3.2-1. Software Maintenance Checklist for Technical Reviews (Sheet 3 of 4)

_____	Have all manuals and data been identified that are needed to support software maintenance?
_____	Is software design consistent with maintainability requirements specified at SDR?
_____	Has software designed maintainability been fully reviewed and documented:
_____	(a) Maintainability requirements in Computer Program Configuration Item (CPCI) Part I spec?
_____	(b) Provisions for diagnosing errors?
_____	(c) Maintainability design principles?
_____	(d) Maintenance demonstration plans?
_____	(e) Software change process?
_____	(f) Logistics supportability?
_____	Do system maintenance plans reflect software design?
_____	Are software maintenance plans sufficiently detailed for organizational responsibilities, equipment requirements, facilities, personnel, procedures, etc.
_____	Are plans for software maintenance data collection and analysis adequate?

CRITICAL DESIGN REVIEW (CDR

_____	Are maintainability characteristics of the detailed design adequate?
_____	Is the detailed design responsive to design vs logistics trade-offs?
_____	Have life cycle cost analyses been updated to reflect detailed design?
_____	What consideration is given to Government Furnished Equipment (GFE) support equipment? What problems exist for logistic support of support equipment?

Table 3.2-1. Software Maintenance Checklist for Technical Reviews (Sheet 4 of 4)

_____	Are calibration requirements adequate for detailed design?
_____	What debugging and diagnostic routines (resident and nonresident) are available for firmware?
_____	Do diagnostic programs (for software maintenance) comply with system maintenance concepts and specification requirements?
_____	How do recent predictions of quantitative maintainability compare with CPCI requirements?
_____	Is the detailed maintainability demonstration plan compatible with specified test requirements?

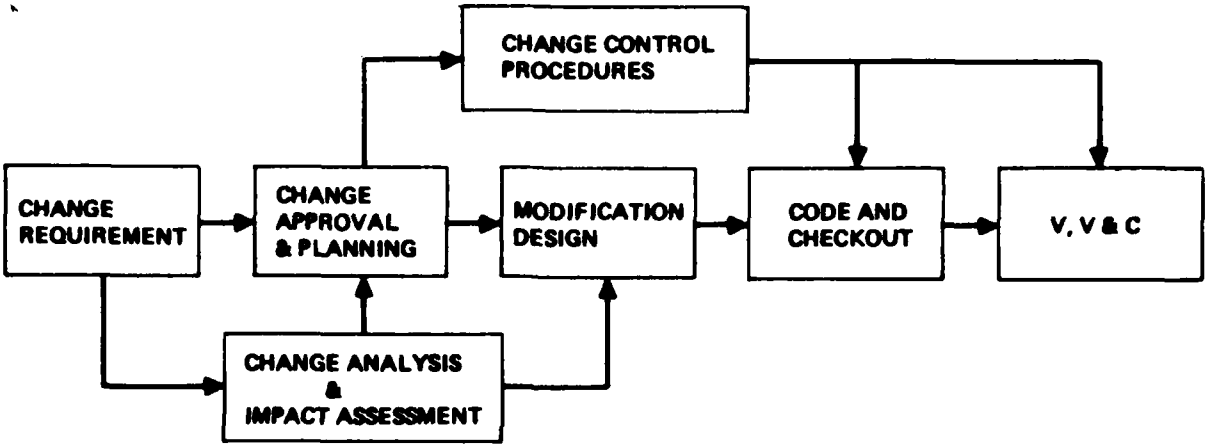


Figure 3.3-1. Software Maintenance Process

would be updated to satisfy Physical Configuration Audit (PCA) requirements. Documentation includes user guides, program descriptions, listings, data base descriptions, traceability matrices, test procedures, etc.

As mentioned above, there are specific exceptions to and interpretations of the general process shown in Figure 3.3-1. For example, during initial coding and checkout of a module (when system is still under development), change approval is not usually required unless it impacts the Part I specification. Change control procedures will also be rudimentary in early development, however, will be formalized when a module or component is brought under configuration control. The development contractor will institute "semiformal" procedures; e.g., discrepancy reports (DR), to keep software development under reasonable control without overburdening programmers with paperwork or discouraging needed changes. Multiple configuration versions will be allowed to accommodate dynamic modifications to still developing software. Program support libraries will be set up to facilitate controlled maintenance. The software maintenance procedures instituted for system development will evolve into standard procedures to be employed for maintenance of the deployed system. Some procedures utilized during development (e.g., program patches) will however, not be applicable to post-delivery maintenance.

The process of software maintenance introduced in this section is further elaborated in Section 5.0 with particular reference to TS and ATE systems. Computer aids to software maintenance are discussed in paragraph 6.1.

3.4 UNIQUE CONSIDERATIONS FOR ATE

Maintenance of ATE software is planned and conducted in basically the same fashion as other software. However, there are unique characteristics of ATE software and special problems in ATE acquisition that impact software maintenance. These unique considerations are summarized in the following itemization.

a. Control and support software are generally developed 1 to 2 years before test software (because test software must wait for UUT definition). When control, support, and test software are finally tested together, there are frequently incompatibilities which require a modification of the control/support software or a "workaround" in the Interface Test Adapter (ITA) hardware/software. The latter solution is often the most expedient but results in a different ITA for each UUT.

b. Control and support software are usually procured off-the-shelf from the ATE vendor but modified for the particular weapon system application. This fact has several implications for maintenance; including:

(1) If the specific software components have been used in extensive and varied applications, then most of the "bugs" will have been discovered and corrected. Modifications to that software may cause an early surge of errors, but these will likely dampen out quickly. If the software components are relatively new or have experienced limited application, then the software can still contain a number of undiscovered errors; some of which could cause unexpected delays and development costs.

(2) Documentation of off-the-shelf software may not be adequate for organic maintenance and also the ATE vendor may not wish to release rights to "proprietary" software. The procurement agency should be cognizant of these potential problems and clarify software delivery requirements at contracting time.

(3) Control and support software is seldom subject to major modification

during the deployment phase. Major changes are usually accommodated by augmentation or replacement with other off-the-shelf software.

c. Test software must be changed virtually everytime there is a design change in a UUT. Thus, changes are frequent and must be accomplished quickly. Support software is provided to facilitate rapid updates (e.g., editing routines). Consequently, test software maintenance is usually an organic system capability (either off-line or on-line). One pitfall is losing configuration control. For example, program listings and documentation can become quickly obsolete if formal change procedures are not implemented.

d. Self-test software at ATE stations can locate faults in end-to-end tests, but has limited diagnostic capability (e.g., the ATE computer may not be functioning properly to support self-test diagnostics). Self-test may include ATE station calibration.

e. Separate components of support software are provided for:

(1) Station self-test
(2) Test software updates
(3) ITA self-test
(4) Control/support software maintenance.

f. ATE software maintenance is performed by Air Force Logistics Command (AFLC) in facilities and resources associated with UUT maintenance. LSA's and Support Equipment Recommendation Data (SERD) are applicable to ATE software maintenance.

g. Changes to ITA interface software are infrequent, but there is usually unique ITA software for each UUT.

h. A major ATE pitfall which impacts ATE software maintenance is premature ATE station design before UUT Test Requirements Document (TRD) data are sufficiently detailed. Premature design results in numerous downstream changes.

i. It is common for ATE factory acceptance test procedures to be used for station self-test.

j. The dissimilarity between the factory tester and field (ATE station) tester causes UUT maintenance discrepancies. The trend is for the ATE station to be used for factory acceptance testing by the UUT vendor.

k. ATE growth potential, essential to software maintenance, is often ill-defined. A growth potential of, say 20%, is sometimes interpreted as providing 20% excess rack space. The problem of providing spare memory is being solved by the rapid expansion in memory associated with modern minicomputers and micro-processors.

l. There is a trade-off between doing test software maintenance on a host computer or at the ATE station. One problem with the host computer can be slow turn around on batch processing. One trend is toward use of a "program development station" which is much like the ATE station but without ITA and UUT hardware and test stimulus/measurement devices.

m. One ATE acquisition problem impacting software maintenance is the form and content of requirements specification and design documentation, including test requirements. While the control/support software are usually defined as CPCI's (and developed/tested accordingly), test software is often specified by TRD's and Technical Orders (T.O.).

n. ATE software costs can increase due to "noise" in ATE; instruments; ITA; or noise-sensitive UUT, resulting in testing errors. The software must then be modified by trial-and-error until an acceptable error rate is achieved. Usually active circuits must be installed

in the adapter to isolate the UUT, then software developed to test the adapter.

o. Detection and isolation diagnostics that are achievable is determined by the design of the UUT. If the UUT were designed for testability, the ATE and software would have a simple job. However, usually testability is required and, as a result, the ATE and software costs are out-of-sight needlessly.

3.5 UNIQUE CONSIDERATIONS FOR TS

Planning and procedures for TS software maintenance are very similar to that for most real-time software and TS software maintenance is less unique than, for example, ATE software. Even so, there are unique considerations for TS and these are summarized below.

a. Software maintenance requirements derive from, and are dependent on, the training program which the TS system supports. For example, the TS maintainability goal may be expressed quantitatively as "X" % mission success in supporting training for "Y" days per week and "Z" hours per day. This impacts how maintenance is performed (e.g., on-line or off-line) and how preventive and corrective maintenance are scheduled. High utilization of a TS makes on-line maintenance difficult, especially if extensive software modifications are to be thoroughly verified.

b. TS are generally maintained by the using command rather than AFLC.

c. Because of high utilization, TS calibration and preflight test must be performed rapidly; e.g., under software control. Software aids to TS maintenance are also desirable. Utilities for program modification include assembler, compiler link loader, source editor, file manipulation, debug and dump routines. Diagnostic programs can be provided for checkout of the following:

(1) Central Processing Unit (CPU), memory
(2) Peripherals
(3) Visual System
(4) Motion System
(5) Interfaces

d. The computing task allocation between firmware and software (i.e., between special purpose processors and programmable general purpose computers) is a significant TS design trade having long-term impact on life-cycle costs and software maintenance.

e. Portions of vendor-supplied TS software may be modified "off-the-shelf" programs and this can cause maintenance problems if such software is not fully documented. (See remarks in paragraph 3.4, item 2.)

f. Different TS software components may be maintained under different policies and procedures. For example, modifications to software modules interfacing with the data base supporting computer-generated imagery (CGI) may be contracted to the CGI system vendor, while other software modifications are implemented within the TS using command.

g. The magnitude of TS software maintenance is directly proportional to the number and complexity of special TS features; for example, instructor interactive software (including capability for malfunction insertion or deletion). Representative design feature trades are described in the Requirements Specification Guidebook.

h. TS system acquisition is often prone to design requirements being added or refined following Part I specification. Thus, software modification during TS development can be both frequent and substantial. Aggregating a number of changes under "block changes" can alleviate at least the frequency of configuration changes. TS software changes following system acceptance by the using command tend to be infrequent.

Section 4.0 PLANNING FOR SOFTWARE MAINTENANCE

ATE and TS are normally procured as integrated hardware/software systems from a single contractor. These ATE and TS systems must then be integrated into Air Force operations as a total system which then must be efficiently maintained. Paragraphs 4.1 thru 4.3 discuss the planning aspects to be considered in achieving this required maintainability. This planning begins in the conceptual phase as was indicated in Figure 3.2-2.

4.1 SOFTWARE MAINTENANCE PLANNING PROCESS

The maintainability of ATE and TS systems is of prime importance as it has a substantial effect on life cycle costs; thus life cycle costs are a major evaluation factor in selecting these systems. The following paragraphs provide an overall description of the software maintenance analysis and planning process specific to ATE and TS software, as well as discussions of life-cycle cost trade studies, resource planning, organization responsibilities and interfaces and formal reviews for software maintenance.

4.1.1 Maintenance Analysis and Planning

A maintenance engineering analysis of ATE and TS software should begin very early in the Conceptual Phase. Referring to the Software Maintenance Checklist in Table 3.2-1, the first item mentioned under SRR pertains to adequacy of the Integrated Logistics Support (ILS) analysis. An ILS plan is prepared, per AFR 800-8, under the direction (usually) of the supporting command, assisted by the using and implementing commands. This ILS plan, in turn, is based partly on the maintenance concept prepared under AFR 66-14. Development of the maintenance concept is normally led by the using command, assisted by the implementing and supporting commands.

The Integrated Logistic Support Plan (ILSP) is the Air Force blueprint for all detailed logistics planning, including software maintenance. The elements of the ILSP (AFR 800-8) are comprehensive and include:

a. Maintainability and Reliability Interface

b. Maintenance Planning

c. Support and Test Equipment

d. Supply Support

e. Transportation and Handling

f. Technical Data

g. Facilities

h. Personnel & Training

i. Logistic Support Resource Funds

j. Logistic Support Management Information

Although "maintenance planning" is singled out as only one element, it is affected by the other nine elements. The system development contractor also prepares a Development, Test and Engineering (DT&E) plan which is called Integrated Support Plan (ISP).

Proceeding to further detailed planning for software maintenance, a Computer Resources Integrated Support Plan (CRISP) is prepared by the Computer Resources Working Group (CRWG) with representatives from using, implementing and supporting commands. As stated in AFR 800-14, the "CRISP identifies organizational relationships and responsibilities for the management and technical support of computer resources." CRISP data, as identified in DID US-3914-ASD, carries software maintenance planning to a rather specific level, including such items as the following:

a. Devices to facilitate computer program changes

b. Assignment of configuration control responsibilities

c. Procedures for reviewing and publishing changes

d. Responsibilities for change scheduling and system integration

e. Personnel resources and training plans

f. Computer facilities

g. Predicted level of computer program changes; to size and scope the support facilities (based on expected types of software modifications, debug effort, and deficiency corrections)

The first draft of the CRISP is prepared in the conceptual phase but the CRWG is responsible for updating the CRISP throughout development so that it will be a viable plan following system delivery.

While the CRISP is especially applicable to post-delivery software maintenance, the development contractor will be the first to conduct software maintenance (as was discussed in Section 3.0). Thus, the contractor is required to prepare a Computer Program Development Plan (CPDP) with particular emphasis on software maintenance during DT&E. The CPDP, according to DID UDI-S-3911-ASD, should include, for example:

a. Procedures for reporting, monitoring, and resolving program errors and deficiencies

b. The contractor's approach to configuration management

c. Plans for on-site program maintenance during testing

d. Plan for insuring program growth, modularity, and ease of modification

e. Training requirements and equipment to support maintenance

The software maintenance planning process is considerably more detailed than described above, but major milestones have been identified. The software acquisition engineer and manager will find AFR 800-14 an invaluable guide to the process and how that process dovetails with other planning activities.

Because the software maintenance impact on life-cycle costs is so great, this planning activity is given special attention in the next paragraph.

4.1.2 Life-Cycle Cost Trades

Almost all the trade studies conducted in connection with ATE system procurement involve some maintainability considerations. This section highlights typical life-cycle cost trade studies which might affect maintainability of software. Three such studies for ATE systems are described, followed by two examples of TS studies. (Note: The reader is also referred to paragraphs 3.1, 4.5 and 5.3 of the Cost Measuring and Reporting Guidebook.)

4.1.2.1 Segmented Test Programs vs One Long Continuous Program. In this study a decision is made as to whether the UUT test programs should be designed, developed and configured in small, task-partitioned segments vs one continuous program. Segmentation, for example, has the advantages of adaptability to multiple UUT configurations, ease of change management, and ease of program development/maintenance through test-partitioning.

4.1.2.2 Manual Programs vs Automatic Test Program Generation (ATPG). Use of ATPG can be of significant value in certain ATE applications provided:

a. Fault models are clearly defined

b. Sufficient computer resources exist and are cost effective

c. ATPG run times are not prohibitive

Some disadvantages of ATPG usage might be:

a. Costly additional memory requirements to accommodate highly specialized ATPG software

b. ATPG run times which adversely affect throughput of the ATE system

c. Maintenance of special purpose software

d. Data in ATPG output format is less readable/changeable

Each application should be studied individually to determine if adoption of ATPG adversely affects software maintenance.

4.1.2.3 "Third Generation ATE" vs "Second Generation ATE". This decision is complex and requires special study. Third generation ATE concepts utilize software to transform a continuous signal into a discrete time signal by sampling at discrete time intervals. This, in theory, replaces the need for individual sampling at discrete time intervals. This, in theory, replaces the need for individual commercial instruments for performing such functions as pulse generation, wave form synthesis, AC signal source, and arbitrary function generation. Some advantages over the second generation approach are simplified maintainability of a single tester-replaceable unit rather than multiple commercial instruments, less cabinetry and improved self-test capability. Some disadvantages are the possibilities of sampled data acquisition errors, discrete-continuous signal conversion errors; processing speed limitations; and requires complex "unproven" software. From a software maintainability standpoint, therefore, this trade must be examined since a third generation system introduces another source of software errors.

4.1.2.4 Life-Cycle Cost Considerations. Typical life-cycle cost consideration which affect software maintainability for TS are:

a. System Complexity - This is probably the most significant system attribute impacting TS software maintainability. As system options are added, the software development and maintenance costs rise dramatically. For example, Table 4.1-1 shows a minimal flight simulator functional configuration vs a more sophisticated capability.

Table 4.1-1. TS Complexity Options

Basic Features	Advanced Feature Options
Cockpit displays	Cockpit displays and controls
Visual displays	Weapon system displays
Motion simulations	Flight control simulations
Audio simulations	Communications
Control loading	Flight instruments
	Navigation
	Audio simulations
	Instructor console control
	Initialization
	Malfunction insertion
	Procedure monitoring
	Results recording
	Flight conditions
	Trainee Control Consoles
	CRT displays, CGI

Each option exercised in the trainer system configuration selection will require a new and separate set of algorithms/ control routines integrated into the software set. Some features may require additional memory or possibly an additional processor. For example, in going to a CRT type display, a peripheral minicomputer may be needed to refresh the CRT image. Software to interface between the simulator computer and the minicomputer is therefore required, hence additional potential maintenance.

b. Disc-Operating System (DOS). Simulators which employ only core memory/mag tape systems have limited capability. The selection of DOS features in simulator design may add significant cost, but will afford useful maintenance features such as mass storage, file handlers, configuration control and debug aids, media conversion and memory dump/save routines all of which aid the job of maintaining controlled programs.

4.1.3 Resource Planning

Resource planning for software maintenance is a prime concern registered in numerous Department of Defense (DOD) documents governing ATE and TS procurements. The following DOD publications require that software maintainability be considered early in the conceptual phase of system acquisition:

DOD Directive 500.29 - Management of Computer Resources in Major Defense Systems

AF Reg 800-14 - Management of Computer Resources in Systems

AFR 574 - Retrofit Configuration Changes

AFR 65-3 - Configuration Management

These government regulations establish policy on management of computer resources (both hardware and software) from early acquisition phases through system development and deployment. In planning resources for software maintenance the procuring agency and the contractor must be cognizant of these regulations, the sub-specifications and planning documentation requirements stemming therefrom. Some of these plans are:

a. Program Management Plan (PMP)

b. Integrated Logistic Support Plan (ILSP)

c. Computer Resources Integrated Support Plan (CRISP)

d. Computer Program Development Plan (CPDP)

e. Configuration Management Plan (CMP)

Major resources for conducting software maintenance are identified in paragraph 6.1.

Since the degree of software maintenance required after development will vary with the nature of the system, the procuring agency should carefully consider anticipated needs so as not to over/ under specify maintenance resources and documentation. For example, since ATE systems require frequent software changes to accommodate UUT changes, careful planning for the acquisition of automated maintenance tools and related documentation, facilities, and personnel is important. On the other hand, over specifying the need for listings and flow diagrams which may seldom be used in the deployment phase can be a costly requirement. Tailoring a system's needs is accomplished through these plans and the DID's levied on the system contractor/supplier. Figure 3.2-2 shows when these maintenance planning documents are released/updated as a function of program phase. Figure 4.1-1 presents the resources required as a function of program phase. Though simplified, Figure

		SYSTEM PHASE			
		CONCEPTUAL	VALIDATION	FULL-SCALE DEVELOPMENT	DEPLOYMENT
RESOURCES	TRAINING	TRAINING SYSTEM REQUIREMENTS	TRAINING PLANS PREP FAMILIARIZATION PLANNING	TRAINING COURSE DEVELOPMENT FAMILIARIZATION COURSES QA/CM/ENG-COORDINATION	TRAINING SIMULATORS COURSE AND LESSON MATERIALS AUDIO-VISUAL AIDS
	PROCEDURES DOCUMENTATIONS RECORDS	SSSPECS STATEMENT OF WORK RFP DIDS CDRL's	PROCEDURES & DOCUMENTATION EXPERIENCE ON SIMILAR SYSTEMS	CHANGE CONTROL PROCED. MEDIA CONT. PROCED. CHANGE ACC'T RECORDS MEDIA CONVERSION RECORDS TEST/RETEST RECORDS	OPERATING MANUALS TECH ORDERS FAULT INSERTION DEMOS. FAILURE REPORTING STAT'S
	TOOLS/ FACILITIES	NO SPECIAL TOOLS OR FACILITIES REQUIRED	DEMONSTRATION TOOLS AND FACILITIES	PROG. DEV. STATION OFF LINE/ON LINE AUTOMATED AIDS SUPPORT SW LIBRARIES/MEDIA CONTROL	DEPOT MAINT. FAC. SINGLE VS. MULTIPLE SITE CONTR. VS. AF MAINT.
	PERSONNEL SKILLS	DESIGN ENGINEERS (UUT) TEST ENGINEERS CONFIG MGMT QUALITY ASSURANCE REL./MAINT.	DESIGN ENG. TEST ENG. CONFIG MGMT. QUALITY ASSURANCE REL/MAINT.	PROGRAMMERS (HOL) PROGRAMMERS (ASSG) CONFIG. MGMT. QA/REL INSTRUCTORS	PROGRAMMERS INSTRUCTORS AF TECH. CONTRACTOR SUPPORT STATISTICIANS
TASKS	MAINTENANCE RELATED TASKS	PLANNING LCC STUDIES/TRADES MAINT. ENG. ANAL.	REQ'MTS DEFINITION SOURCE SELECTION DETAILED PLANNING TRAINING PLANS C. M. PLANS QA PLANS	PREL'M DESIGN/DATA BASE PROG. STRUCTURE CODE/DEBUG/FINAL DES TEST/VERIF/VALIDATION CONFIG. CONTROL ACCEPTANCE MAINT. MANUAL PREP.	MAINTENANCE CREW TRAINING PROCEDURES IMPLEMENTATION FACILITIES DEVELOPMENT TRANSFER & TURNOVER SOFTWARE MAINTENANCE ACTIVITIES (SEE PARAGRAPH 3.3 AND SECTION 5.0)

Figure 4.1-1. Maintenance Resource Planning Considerations

4.1-1 implies that significant integration of facilities and documentation is required during the full-scale development phase. This is necessary to insure that automated tools used to control changes to program code are properly tracked, configured, retested and, if applicable, introduced into the appropriate design requirements documents. This integration is achieved through configuration management and quality assurance plans and procedures which assure that as built-tested, and delivered software meets the as-designed configuration.

4.1.4 Organization Responsibilities and Interfaces

Organizational relationships between the various implementing, supporting and using commands are extremely important to software maintenance. The CRWG, composed of representatives from all three commands, is responsible for insuring that agreements reached in the CRISP are reflected in Program Management Responsibility Transfer (PMRT) agreements. This group is concerned with organizing to achieve the following functions:

a. Organizational Responsibility Definition

b. Program Configuration Definition

c. Change Processing

d. Change approval authority and classification

e. Change accountability

f. Version recompilation, program patching

g. Library and Media Control

h. Program verification and acceptance

i. Change test and acceptance

j. Discrepancy documentation, tracking, corrective action

During the full-scale development phase, these functions are managed internally by the prime contractor and the selected ATE or TS system supplier. The CRISP should define the organizational transfer of these responsibilities at the end of the development phase. Following system delivery, ATE is maintained by AFLC, while TS is maintained by the using command. The development contractor will also usually support post-delivery maintenance at least initially.

4.1.5 Formal Reviews of Maintenance Plans

To insure that maintenance planning is properly documented, applicable organizations within the procuring agency/contractor should take a critical look at all maintenance-related documentation at the various formal design reviews. Figure 3.2-2 illustrates the major maintenance related documents which should be reviewed at SRR, SDR, PDR, CDR, Functional Configuration Audit (FCA), Physical Configuration Audit (PCA) and Initial Operational Capability (IOC) and Table 3.2-1 provides a checklist for major reviews. The organizational disciplines which should be represented at those reviews and typical concerns which they should register are presented below:

Maintainability

Adequate modularity of design
Efficient use of HOL's
Effective use of automated tools
Minimum program downtime
Adequate programmer skills
Software transportability
Adequate program development/maintenance stations
Clarity of manuals

Configuration Management

- Maintenance requirements definition clear
- CMP Plan generation/approval
- Change processing mechanics
- Change classification/approvals
- Automated program changes vs Version Description Document (VDD) updates
- Baseline program control
- Media definition/I.D. - tapes, discs, file names
- CPCI definition
- Programming Standards, Computer Program Library (CPL) procedures

Quality Assurance

- Verification of program baselines
- Change verification
- Retest of changes
- Change accountability
- Media control - storage/handling/environmental
- Labelling of Media
- Acceptance stamping of Media
- Test records
- Correlation of flow diagrams listings to versions
- Patch control - octal/hex vs recompilations
- Discrepancy reporting/corrective action
- Calibration software

Software Design

- Functional vs maintenance design
- Life cycle costs vs programming language
- Hardware vs software, i.e., Read-Only Memory (ROM), floating point, firmware
- Efficient modeling - fault detection
- Test program verification
- Use of analyzers
- Diagnostics/debug routines efficient?

4.2 MAINTENANCE PLANNING DOCUMENTATION

Maintenance documentation can be divided into two types - planning documents and implementing documents. Maintenance-related documents in these categories are listed below.

Planning

- CRISP - Computer Resources Integrated Support Plan
- PMP - Program Management Plan
- ILSP - Integrated Logistics Support Plan
- CDRL - Contract Data Requirements List
- DID - Data Item Descriptions
- CPDP - Computer Program Development Plan
- CMP - Configuration Management Plan

Implementing

- Part I Specification
- Part II Specification
- Interface Control Document
- Programmer's Manual
- Operators Manual
- Version Description Document (VDD)
- Configuration Index
- Specification Change Notice

The guidebook on documentation describes how these documents are related, where they are referenced and when they are released in the acquisition cycle. Most of the implementing documents are called up as deliverable documents in the RFP. The sequence of release is typically specified in the CPDP. The preparation of these documents should be a collective effort of all the concerned organizations identified in paragraph 4.1.5. Only after proposal evaluation and source/system selection can detailed maintenance requirements be defined and planned for. For example, should the

decision be made to select a system with an immature design or unproven computer or operating system, extensive maintenance should be anticipated. Similarly, if a system requires extensive use of specialty/proprietary software or unique HOL's, organic maintenance capability for the user command may be prohibitive and such problems should therefore be avoided. The detailed software maintenance approach, therefore should begin subsequent to source selection. DID's applicable to software maintenance are listed in Table 4.2-1. Selection of DID's for a specific system development is a primary responsibility of the software acquisition engineer. DID's can also be revised to tailor maintenance documentation to specific system requirements.

4.3 ATE and TS VARIANTS

ATE and TS software are sufficiently unique in many respects to warrant special maintenance planning considerations. Some of these special considerations are as follows:

4.3.1 ATE Software

a. ATE software is typically a blend of vendor and contractor developed software. The ATE supplier generates the control and support software, configured in his own format, while the contractor will develop the interface adapter software unique for each UUT. In planning for maintenance of the integrated package, the acquisition agent must consider:

(1) Adequate training of USAF programmers to insure sufficient understanding of the selected ATE vendor's operating system.

(2) Adequate facilities for software development, i.e., enough stations to accommodate software development schedules.

(3) Organizational relationships - the creation of some sort of vendor contractor working group available to work the program of integration of the two software products.

(4) Documentation - sufficient to insure clear definition of baseline configurations for both vendor and contractor's software.

(5) Change processing, problem resolution and statusing methods.

b. ATE software must accommodate multiple UUT configurations. The maintenance planning implication here is concerned with configuration control. As pointed out earlier, vendor-provided automatic configuration control aids will most likely be available for accomplishing code and version updates. The procuring agency should plan, therefore, for publication and control of master configuration logs, indices or catalogues which cross reference UUT numbers vs test program numbers, vs test-stations. Accounting is accomplished by dash number and version number as appropriate to define what UUT's may be tested by what programs on what stations using what adapters. These are the data items a master log or index must provide. This is a functional responsibility most appropriately assumed by a software configuration management group.

c. ATE software will require design inputs from a variety of sources. The UUT test requirements will originate with the UUT design organization. The UUT test program will be designed in all likelihood by a separate test group. The operating system design will probably remain the responsibility of the ATE vendor. Calibration and test software may be the responsibility of Quality Assurance (QA). The point is that the system must recognize the existence of these multiple design contributing factions and plan for the efficient inte-

Table 4.2-1. Data Item Descriptions (DID) Applicable to Software Maintenance (Sheet 1 of 2)

AFSCM/AFLCM 310-1/S-137	Calibration Requirement Summary
DI-3119A	Computer Program Development Specification
DI-A-3108	Configuration Management Plan
DI-A-5239	Computer Program Development Plan
DI-E-129/M	Computer Software/Computer Program/Computer Data Base Configuration Item(s)
DI-E-3106	Specification Maintenance Document
DI-E-3120A/MI	Computer Program Product Specification
DI-E-3121	Version Description Document
DI-E-3122	Configuration Index
DI-E-3123	Change Status Report (Computer Program)
DI-E-3127	Advance Change/Study Notice
DI-E-3128	Engineering Change Proposals
DI-E-3134	Specification Change Notice (Computer Program)
DI-E-3277	Training Equipment Computer Program Documentation
DI-H-3277/M3	Training Equipment Computer Program Documentation
DI-H-5070	System Maintenance Programs (Software)
DI-M-3410	Computer Program User's Guide
DI-M-3411	Computer Programming Manual
DI-M-5118	Computer Software Maintenance Manual
DI-R-3533	Reliability/Maintainability Program Plan
DI-R-3538	Reliability/Maintainability Demonstration Plan
DI-R-3544	Reliability/Maintainability Assessment and Demonstration Reports

Table 4.2-1. Data Item Descriptions (DID) Applicable to Software Maintenance (Sheet 2 of 2)

DI-R-3545	Reliability and Maintainability Allocations, Assessments, and Analysis Report
DI-S-6176	Ground Support Equipment Recommendations Data
DI-T-3703	Category I Test Plan/Procedures (Computer Programs)
DI-T-3717	Category I Test Report (Computer Program)
DI-T-3734	Test Requirements Document
UDI-E-695/EST	Computer Program Development Plan
UDI-E-3120B-ASD	Computer Program Protect Specification
UDI-M-3410A-ASD	User's Manual (Computer Program)
UDI-S3911/ASD	Computer Program Development Plan
UL-81-AQ	Logistics Support Analysis Data
US-3914-ASD	Computer Resources Integrated Support Data

gration of each organization's inputs. This objective is most effectively met by establishment of a software configuration management group which coordinates inputs, publishes formal configuration definition documents processes changes, tracks problems, manages source materials, code, support software listings flowdiagrams, etc. This group acts as a focal point for controlling software configuration. Its charter is to assure test programs are controlled at all times in accordance with approved plans and specifications, thereby freeing the programmer from this cumbersome but vital control task.

d. ATE software is subject to potentially frequent change during deployment due to UUT changes. Changes after deployment are obviously costly because of retrofit, retest, the potential for performance regression, increased red tape in the change processing cycle etc. ATE system growth, both in hardware and software, must therefore be provided for to minimize impact on testing of these undesirable changes. For example, a UUT change necessitating a new stimulus or measurement capability may require significant software redesign, additional memory or mini processor. System growth potential, should therefore be carefully considered. Active vs passive ITA is a design trade which will significantly impact software maintainability resource planning because adapters are a prime candidate for accommodating UUT configuration changes. It is much easier/cheaper to add test capability to a single UUT adapter than to program a change to retrofit a population of ATE stations in deployment across the nation. Adapters, however, as they become more complex, have the undesireable feature of destroying the commonality/interchangeability of the overall test station. A trade must be carefully studied.

4.3.2 TS Software

TS software, unlike ATE software, is less prone to standardization of design. For example, the central core of an ATE system can be designed and built independent of the application. ATE vendors can design this core to provide a standard set of stimuli and measurement signals.

An ATE central core composed of a central processor, memory, I/O and stimulus/measurement can be designed to provide and measure AC, DC, pulse generation, wave form generation over a somewhat standard range of amplitudes and frequencies, irrespective of application. TS systems, however, and associated operational software, must be designed around a given training mission where the unique software portion is a greater percentage of the total trainer software package than it is for an ATE software package. Standardization of TS software is therefore, more difficult. Some ease of maintenance can be achieved by modularity of design and maximum use of a HOL.

In planning for TS software maintenance, other considerations which programmers maintaining this software recommend are:

a. The languages used should be as machine independent as possible. To require operational (mission) program development on a single special configuration processor restricts software maintenance.

b. Standardization of arithmetic routines should be maximized. Sine, cosine, square root, etc., should be developed as separate standard routines, modularized and identified. This saves the redoing of such routines for each trainer mission scenario.

c. The support software should be designed to operate on the TS system processor, not some remote facility, such as a large general purpose computer. When programmers are required to resort to a remote facility for compilation, assembly, link edit, and load module generation, the efficiency of program maintenance drops.

d. The selection of a disc-operating system will aid in software maintenance by providing rapid assess mass storage. Program changes can be accomplished faster. The disc system provides more utility/maintenance routines useful in troubleshooting and fault diagnosis, configuration control and media conversion.

Section 5.0 SOFTWARE MAINTENANCE ACTIVITY

This section discusses how software is maintained in ATE and TS systems. The general process of software maintenance was introduced in paragraph 3.3 and unique considerations for ATE and TS were identified in paragraphs 3.4 and 3.5, respectively. Within this context, the particular "how to" maintenance tasks for TS and ATE software are now described. Alterations in software for weapon systems must be implemented under well defined controls and this aspect of software maintenance is given special attention in paragraph 5.3.

5.1 TS SOFTWARE MAINTENANCE

Maintenance of TS software has many similarities to maintenance of any real-time software. Also, much of the maintenance activity follows essentially the same process as development of new software, e.g.: design, coding, debugging, verification, and validation. Despite the similarities, TS do present unique constraints to software maintenance. TS uniqueness is described in paragraph 5.1.1 (Introduction). The sequence of events which indicates a software change is identified in paragraph 5.1.2 (Processing of Authorized Software Changes). Then, the TS software maintenance process is explained in paragraph 5.1.3 (Specific Software Maintenance Tasks). This process is described by a typical sequence of events for a contractor-implemented change authorized during system development. Similar steps could be followed by the using command in post-delivery maintenance. The particular type system chosen to exemplify a TS software change is the flight simulator. Potential problems in TS software maintenance are summarized in paragraph 5.1.4 (Pitfalls To Avoid).

5.1.1 Introduction

TS uniqueness for software maintenance is described in this section in terms of TS software components; foreground/background maintenance capability; and representative types of changes to TS software.

TS, more particularly flight simulators, are seldom developed by adapting previously developed (off-the-shelf) computing systems (hardware and software). However, to satisfy the unique requirements of a real-time TS system, previously developed software can sometimes be modified and used as CPCIs. TS support software generally includes:

a. Operating system (job control, centralized I/O, data management, etc.)
b. Assemblers/compilers
c. Utilities/support programs
d. I/O and math libraries
e. Diagnostic maintenance programs

TS software is developed to satisfy the requirements of a total simulator and its subsystems but is itself a contract end item. The software is part of the delivered end item equipment which for flight simulators supports the training for USAF flight personnel in the following mission tasks:

a. Take off
b. Maneuvering
c. Instrument approaches
d. Landing
e. Ground and emergency procedures

5.1.1.1 Major Areas of Flight Simulator Software. The four major areas of software for flight simulators are described below. This software will be delivered as a CPCI, with attendant documentation as specified in DIDs (C-133, H-110, etc.). A typical breakdown is as follows:

a. Flight Simulator Computer Programs (FSCP)
    (1) Executive

    (2) Airplane and flight subsystem math models

(3) Motion base and visual system drives

(4) Instructor-related control and display programs

b. Simulator Data Support Programs (SSP)

(1) Ground station data compiler
(2) Display page compiler
(3) Cycle time verification
(4) Math model verification

c. General Software Support

(1) Operating system, I/O libraries, math libraries
(2) Assembler, compilers
(3) Editor, utilities, debug
(4) Loaders

d. Simulator Maintenance and Test Programs (SMTP) (computer, interface and peripheral hardware diagnostics)

All TS software will be controlled through a configuration management system. Following CDR, control is maintained through use of the draft Part II specification, and a product configuration baseline will be established at the TS Formal Qualification Review (FQR) (see Figure 3.2-2).

5.1.1.2 Foreground/Background Computing Capability. Current TS computer technology often employs a foreground/background computing capability which permits certain utility type functions, e.g.: software maintenance to be accomplished simultaneously with real-time system training activities. High priority task processing is assigned to the real-time system with the remaining time assigned to background activities. A dedicated (complete) TS system will be required for portions of the checkout activities, however, system design should specify as many background functions (within timing, memory constraints) as is practicable.

5.1.1.3 Attributes of Maintainable TS Software. Attributes of maintainable TS software are very similar to those specified in paragraph 3.1 (see Table 3.1-III).

5.1.1.4 Typical TS Software Modifications. Approved changes to the air vehicles being simulated often necessitate a change to simulation software and this is the major source of TS software modifications. Examples of typical changes to a flight simulator which requires both hardware and software modifications are provided in Table 5.1-1.

Except for urgently required changes which may be expedited through the formal change process, it is desirable to accumulate changes and to make a modification block update to the TS system - both minor and major changes. The change will entail a thorough review by the appropriate engineering personnel and a data document revision will be made, if required, prior to any software change. It may also be necessary to obtain the assistance of the change originator if the change is not fully specified.

Modification block updates should be scheduled about once or twice yearly, except for urgent requirements.

5.1.1.5 Detailed Change Analysis. The maintenance organization will perform a detailed analysis of the approved change and recommend alternative solutions. The change may involve software only or a hardware-software interface. It may be advisable to process minor changes below the level of a formal change board.

Examples of minor changes are:

a. Coding errors (incorrect use of language)

b. Subroutine program errors (incorrect flow chart implementation)

Table 5.1-1. Typical TS Software Changes

Reason for Change	Software Change Description
Provide Dual Angle of Attack System	Revise software to drive angle of attack instrument as follows: (a) To drive two instruments (b) To modify compensation curve for meter movement
Change to Bleed Air Subsystem	Revise software Bleed Air Subsystem model to reflect operational changes: (a) Provide new variable(s) for display; scaling, variable change for new meters/display
APU Installation Revision	Revise software APU model to simulate new interval times and new delay sequences: (a) Constants in timer comparison (b) Generate pressure parameters and output valves (c) Load through circuit breakers in electrical program
Autopilot Engage Interlock Modification	Revise software Autopilot model to simulate new time delay gate test

c. Subroutine redesign (flow chart incorrect, new flow chart required)

Major modifications should always be submitted to and processed by a formal change board. These changes will be processed through software verification and system validation testing.

Examples of major changes are:

a. Efficiency or capability improvements

b. New major functions or new requirements

c. Restructuring software for timing or storage constraints

d. Major design errors

e. Database modifications (real-time programs usually reference a common database and thus, formal testing is required for any data base change)

An example of how a software change request would be administratively processed by a contractor during system development is contained in Figure 5.1-1. Change management is discussed in paragraph 5.3.

5.1.2 Processing of Authorized Software Changes.

If the software change requires implementation by a contractor, the AF shall prepare an Advance Change Study Notice (ACSN) and forward to the appropriate contractor(s). Similarly, if the contractor requires the services of a subcontractor, the change requirements will be submitted to the subcontractor and that subcontractor required to submit a Supplier Change Proposal (SCP) to the prime contractor. Following successful negotiations with the subcontractor, the prime contractor will submit an Engineering Change Proposal (ECP) to the AF in response to the ACSN. All cost, schedule and engineering data fully addressing requirements in the ACSN will be contained in the ECP. A typical series of events describing the flow from receipt of the ACSN to contract authorization is presented below:

a. Receipt of fully coordinated ACSN

b. Input from Engineering, Manufacturing, Materiel

c. Composite ECP available

d. Coordination meeting

e. Present to Change Board

f. Inputs to Finance - Engineering, Manufacturing, Materiel

g. Finance package to Contracts

h. Engineering package to Contracts

i. Management review

j. Submit to government

k. Final technical agreement

l. Government review complete

m. Start negotiation

n. Complete negotiation

o. Contract authorization

5.1.3 Specific Software Maintenance Tasks.

This description of the TS software maintenance process employs the example of a contractor's modification to a flight simulator during the development phase.

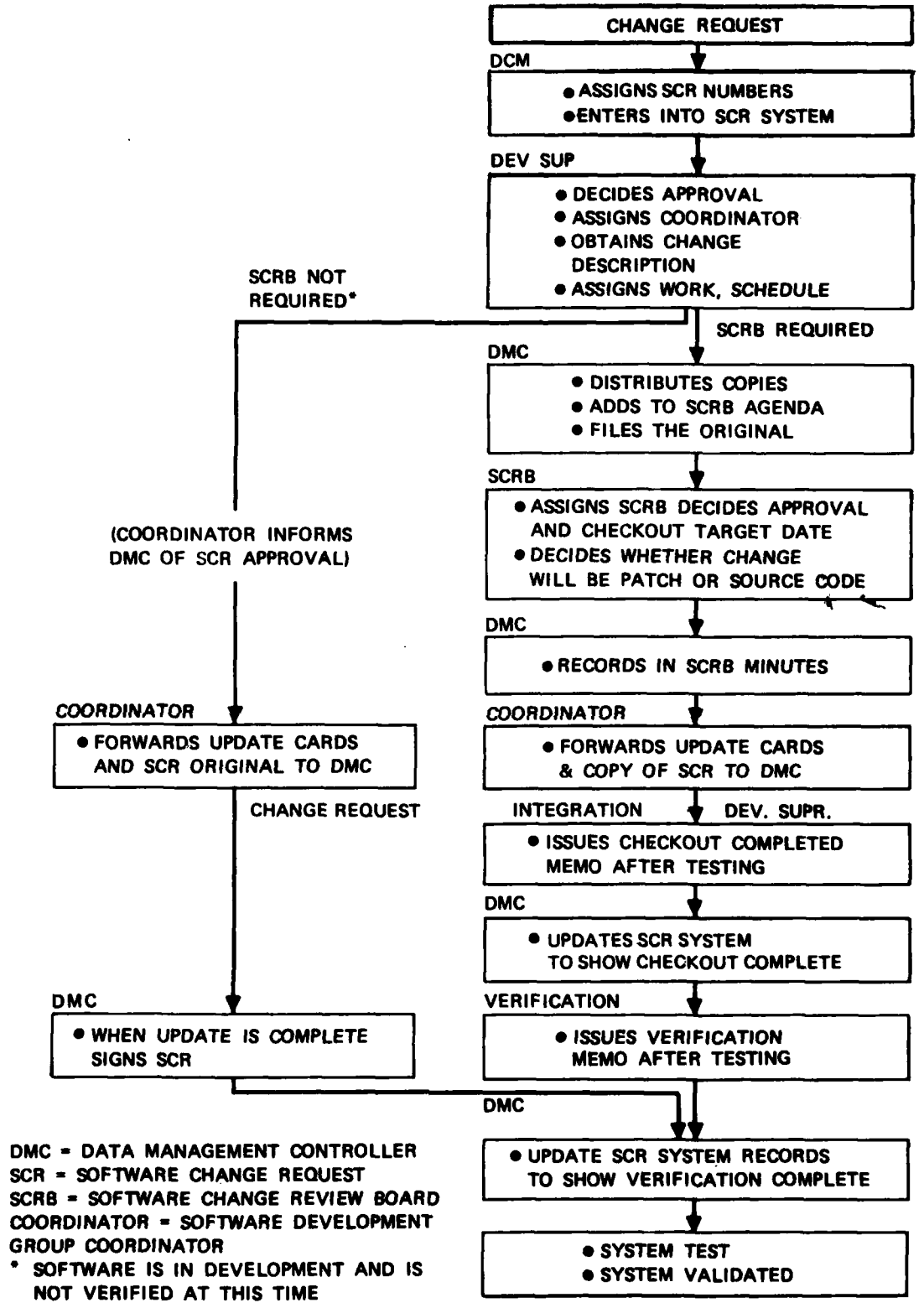


Figure 5.1-1. Typical Software Change Request Flow

5.1.3.1 Software Modification Design Task. The software modification design task (see Figure 3.3-1) is the first task in software modification. The initial activity in program design is a review of the Part I specification or change thereto to ensure that the modification meets the requirements. Task oriented flow charts are prepared or modified and data base descriptions are prepared. Interface discussions are held to define interface definitions and to look at the total interface picture. Coding specification contains calling sequences, input, output and processing requirements.

During program development, software management should maintain strict control of timing and sizing requirements throughout the TS project cycle. Evaluation of time and size parameters should be performed by specialist engineers who are skilled and experienced in simulation systems. Timing and sizing estimates are related to previous TS experiences and on analysis of relevant data provided by other engineering organizations.

Throughout the development phase the actual parameters are evaluated against the predicted values and displayed in a formal report published periodically (biweekly or monthly). This manner of providing visibility to management/technical personnel will show trends which can then be monitored and controlled.

5.1.3.2 Software Coding. New or revised coding must be accomplished using the programming language approved for that set of software modules. Many computing systems consist of both assembly language and HOL programs selected on the basis of real-time system timing constraints, core storage constraints, ease of checkout, and ease of maintenance considerations. Language selection should always be approved at the group leader level or higher level to ensure a correct selection. Programming personnel will review the language specifications with language consultants and lead engineer personnel to promote a thorough understanding. In addition, programming personnel should use standard routine interfaces. To facilitate this, set/use tables should be provided which identify, for each data item the routine which sets (computes) it and the routines which use it.

5.1.3.3 Software Subroutine Checkout. Computer program subroutine or module checkout is the first step of a series of tests which results in verified software and validated TS system. This may be accomplished by examining the subroutine trees of each computer program component and identifying the lowest level routines. The lowest level is checked out first, and successive levels are scheduled for checkout in ascending order. Checkout of higher level routines is considered incomplete until the routines interface correctly with lower level routines. Scheduling control is achieved by monitoring subroutine completion dates (which provides incremental integration status assessment) and by verifying checkout of each routine through reviewing test results.

5.1.3.3.1 Module Checkout. Given the modularity of program design, the stage is reached where stand alone module check-outs can begin. Testing procedures are often developed so that programs can be extensively debugged and modified in on-line fashion, by making computer time available to maintenance programmers within a real time test facility (desirable but not required) specially suited to simulation requirements.

This facility provides sufficient indicators, controls etc., such that any of the simulation programs can be fully tested in a real time environment. Extensive use is made of continuous core read out and print out facilities such that the program inputs and outputs are accurately monitored independently of the input/output simulation hardware. In this way, module program testing is carried out independently of the simulator computer configuration.

5.1.3.3.2 Software Component Checkout. Each computing program component will consist of a number of modules, e.g., engines can be broken down into:

a. Starter sections
b. Fuel control sections
c. 1st and 2nd Stage Turbine
d. Intake
e. Thrust Reversers
f. Instrumentation

In general, component checkout is the checkout of the highest level module in the component module tree. As with all modules, checks are made to verify each internal logic trail and the ability of the module to interface with lower level routines. Interfaces with other components are verified to the extent possible. Because components normally provide a functional capability or portion thereof, additional component tests are defined which thoroughly exercise the full functional capability of the component.

Component test plans are prepared by development personnel and reviewed by basic program design and integration personnel and management. Because of the complex test drivers that are often needed to drive functional tests, every effort is made to phase the integration of previously checked out components to allow their use as test drivers for downstream component checkout tests.

5.1.3.3.3 Software Static (non-real time) Verification. Verification programs shall be developed to provide a means of confirming the accuracy and fidelity of the programmed mathematical models. They operate in non-real time.

The verification programs generate pseudo inputs to the models whose output data is used to verify the accuracy of the mathematical model to the approved data within specified tolerances. The programs and procedures developed to accomplish this mathematical model verification are to be delivered as part of the software requirements. All data generated by the verification programs should be delivered also.

Descriptions should be provided prior to testing; identifying the procedure, the extent of model exercise for each software module, the input variables and the expected results in data output. Outputs shall be demonstrated to be within specified tolerances of the approved performance data to be provided. Test results and analysis (if any) shall be submitted in the form of test reports.

As an example, the following systems should undergo verification testing:

a. Aircraft systems
b. Navigation systems
c. Propulsion systems
d. Auxiliary Power Unit
e. Flight simulation
f. Environmental Systems
g. Radio Aids Facility

See Figure 5.1-2 for static verification program operational flow.

5.1.3.3.4 Software Component Integration. Component integration is the phase of computer program development when checked out components are integrated together to form a checked out computer program. Whereas ideally it is convenient to think of all components as being tested completely prior to integration, in practice, some component testing is done as part of the integration process in the step-by-step layout of the integration process. This layout is reviewed by program development personnel and iterated as necessary to arrive at a viable integration sequence. Following agreement on a layout sequence, integration personnel review CPCI Part I and II

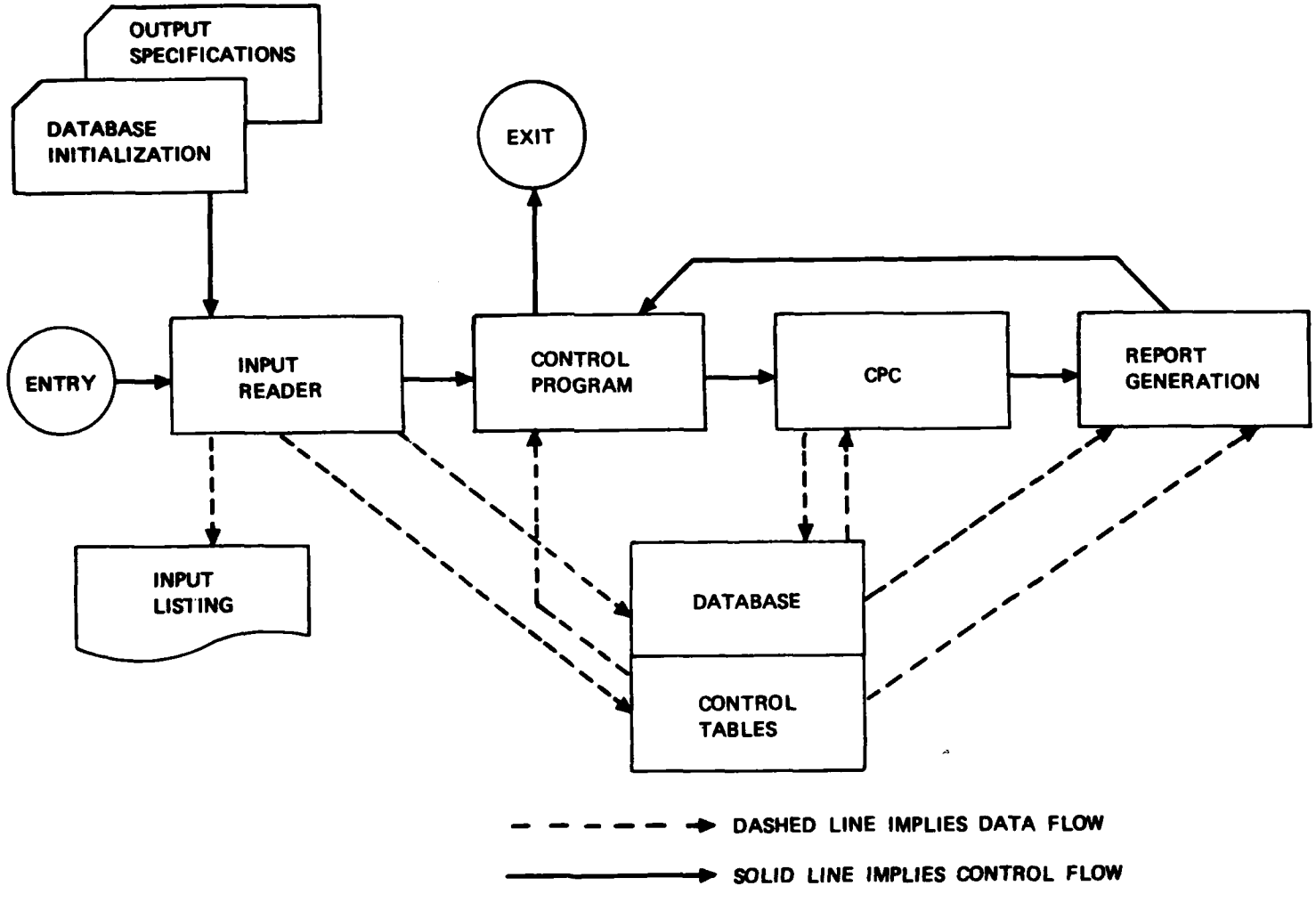


Figure 5.1-2. Static Verification Program Operational Flow

specifications and develop detailed test plans. These test plans are reviewed by basic program and development personnel and software program management. Following approval, test drivers and test data where required are prepared and placed under control. Tests are run as components become available.

Component integration is complete when all the components have been integrated, all defined tests have been completed, and the program has been released for verification testing.

5.1.3.4 Total System Integration/Test. Once the hardware has been commissioned, using software calibration programs as necessary, development of the complete simulation program will proceed using the cockpit and instructors hardware to exercise all the systems in a totally integrated manner. See Figure 5.1-3 which depicts off-line and real time processing facilities.

5.1.3.5 System Validation Test Plan-Demonstration-Report. The TS system validation test plan is a deliverable product to the USAF and establishes detailed qualifications requirements, criteria, general methods and overall planning for system validation. The test plan and reports are written to confirm, in accordance with Section 4 of the Part I specification that the requirements of Section 3 of the Part I specification are satisfied. Test personnel develop the test plan and software requirements and development personnel are requested to review the plan for consistency and adherence to the quality assurance and performance/design requirements of the Part I specification. Following USAF approval of the test plan, test procedures and support data are developed. The validation test is conducted with contractor test, software engineering, Quality Assurance and USAF Quality Assurance, TS and other USAF personnel as appropriate. Following completion of the test, a test report is written for delivery to the USAF. Following test report approval a new configuration baseline is established and the change (sometimes referred to as a modification block) is closed out.

5.1.3.6 Integration of Sub-Contract Software. A TS contractor may elect to subcontract a subsystem; e.g., the graphic display unit to provide a cost effective, reliable and maintainable TS system. Often the prime contractor will develop the display routines to meet the specific real time requirements, using supplier standard subroutines for character generation and standard mathematical subroutines. Software will undergo checkout and tests following equipment delivery and during the course of the graphics software development. Graphics display routines may be developed and verified independently of the main configuration and will be integrated with the TS real time system for in-house testing.

Following in-house testing, the graphics system will be integrated with the simulator hardware for final development and qualification. The graphics software will be subjected to the same integration and qualification testing as other areas of the TS real-time system.

5.1.4 Pitfalls to Avoid in TS Software Maintenance.

TS software integration-verification and system validation activities must be very carefully planned and monitored. If the change is confined to software only, the system need only be restored to the approved software configuration baseline and training resumed. However, hardware component modification poses a possible problem area as TS formal training activities must be continued and thus the hardware component baseline must be returned to that configuration when the software modification session is ended and training is to begin. It is advisable to schedule a short but comprehensive test to ensure that both hardware and software baselines are fully restored following a software modification session. There may well be a circumstance when a hardware modification cannot be removed and thus an intermediate baseline must be established. The appro-

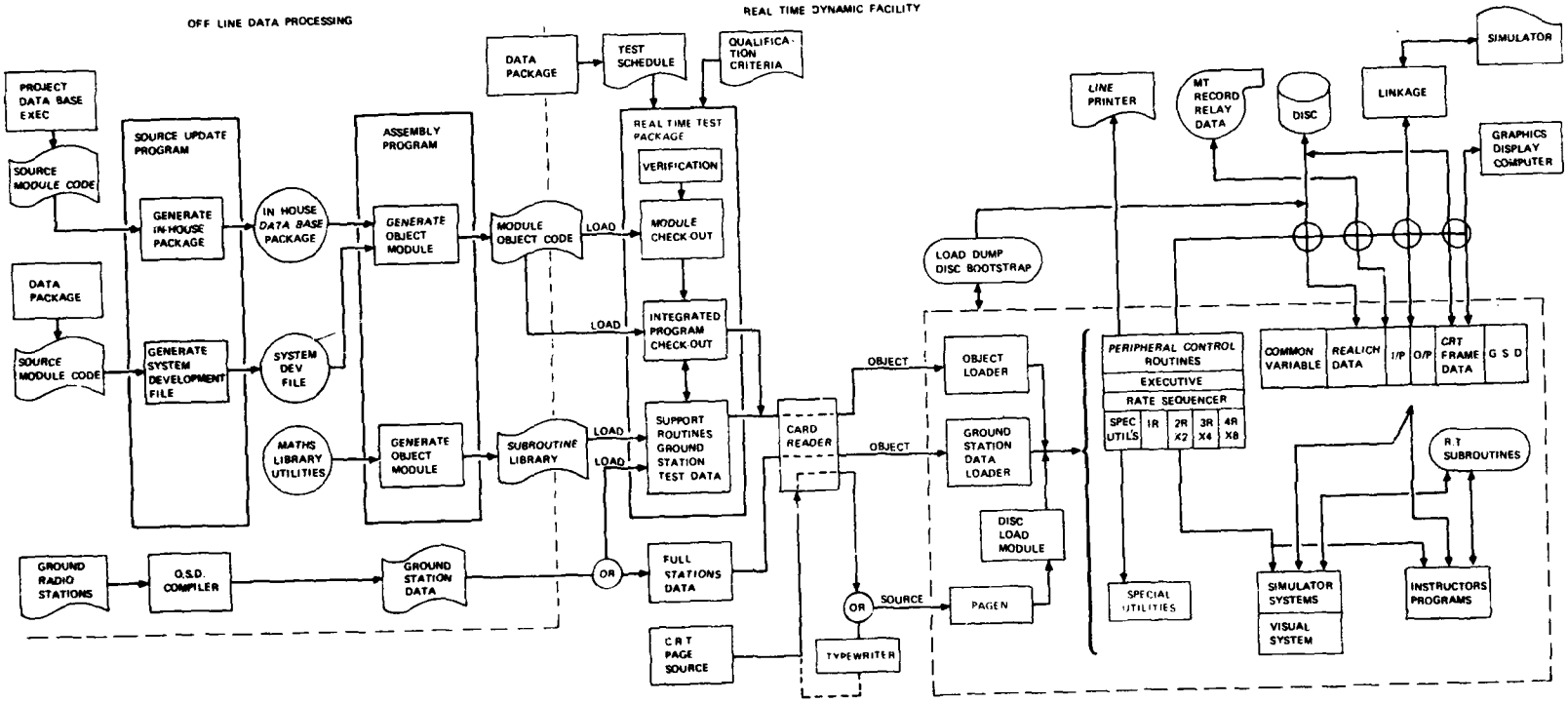


Figure 5.1-3. Off Line and Real Time Processing Facilities

priate level of USAF approval must be obtained for an intermediate hardware/ software baseline and special (temporary) documentation must be distributed to all affected personnel (Contractor and USAF). It is very important to create backup systems (cards, tapes, discs). Control of an intermediate baseline is sometimes more difficult than control of a formal baseline. Reconstruction can be costly.

During the planning for TS hardware/ software modifications, it would be important (if possible) to make provision to minimize the degradation of software maintenance capability when the TS system is being transitioned for incorporation of enhancements or deficiencies. Provision for a dual capability during the transition from one configuration to a subsequent configuration may be required. Another area of concern is the development of documentation (Part II Specification, User Manuals, Computer Programmers Manuals, etc.) in accordance with the CDRL and DID (C-133, H-110, H-111, etc.). Constant monitoring of contractors and subcontractors performance is required, especially with contractors who have not previously worked with a CDRL and DIDs. This monitoring will provide assurance that documentation will meet contract requirements.

5.2 ATE SOFTWARE MAINTENANCE

5.2.1 Introduction

ATE software is developed for (1) ATE station control, (2) language translation, (3) test programs to provide ATE diagnosis/fault isolation and (4) support software. An ATE complex in conjunction with adapters, test programs and procedures will be used for functional testing and fault isolation of printed circuit boards and other electrical/ electronic systems (UUTs).

Maintenance of ATE software is considered and specified as a requirement in the conceptual phase. The long term positive effect of procuring well designed complete maintenance software and maintainable software will normally offset any additional costs which accrue in the procurement of such software and the ATE system. The following sections will address major areas of ATE software, attributes of maintainable ATE software, sources and examples of software modifications, and a typical process to maintain test software. Pitfalls in ATE software maintenance are also noted.

5.2.2 Major Areas of ATE Software

The three general categories of ATE software are described in paragraph 4.1 of the Requirements Specifications Guidebook. These categories are control, support and test software and are defined in that section. Documentation of those three categories of software will be developed in accordance with standard data item descriptions (Table 4.2-1) or in accordance with good commercial practices. A complete and accurate USAF requirements specification and constant monitoring by the USAF during program reviews (PDR, CDR, etc.) will ensure the development of adequate documentation to maintain all elements of ATE software.

5.2.3 Attributes of Maintainable ATE Software

The attributes of maintainable ATE software listed below are limited to those unique to ATE software. The general attributes of maintainable software listed in Table 3.1-2 apply to ATE software.

a. Configuration control of all deliverable software (CPCIs) to include control, support and certified test software must be established and maintained through the life cycle. This includes

changes to the operating system, industry proposed changes to Abbreviated Test Language for All Systems (ATLAS), subprogram libraries, all of which are subject to the USAF change management system.

b. ATE test software must be written using a HOL, e.g., ATLAS (ARINC report 416-8).

c. Detection and isolation requirements for diagnostic maintenance programs must be specified in the requirements specifications and demonstrated during acceptance testing in accordance with acceptance test procedures.

5.2.4 Examples and Sources of Changes to ATE Software

This section identifies the specific components of ATE software that contribute the greatest workload to software maintenance. Workload in this case is an integration of the frequency and difficulty in making software changes. Areas of ATE code are rated from greatest contributors to least contributors of maintenance workload for each of the major sections of ATE software: test, control, and support. Examples of changes are also provided. Overall, the test software maintenance effort is usually an order of magnitude greater than for control and support software combined.

5.2.4.1 Changes to Control Software. As noted in paragraph 3.4, major changes to control software are usually accommodated by augmentation or replacement with other off-the-shelf software. Limitations in control software, e.g., to accommodate improved versions of ATLAS for digital UUT's, are generally overcome by "work-arounds." These mean, for example, writing test programs that switch back and forth between ATLAS and FORTRAN, then merging separately compiled code. If such work-arounds become too frequent or cumbersome, then a modification or update of the control software would be required.

Debugging of control software was discussed in paragraph 3.4. In addition to newly discovered program bugs, operational experience with modified control software (to a particular application) will likely reveal capability constraints. For example, the control software documentation may state that the software can handle up to five nested loops, but fail to mention a quantitative limit on instruction storage. Discovery of the constraint will likely cause a software update - at least a change in documentation (which must be controlled, published, circulated, etc.).

The principal elements of ATE control software are listed below in order of decreasing maintenance work (the first listed element is the greatest contributor to maintenance workload):

a. Test Manager
b. Operating System
c. Test Equipment Drivers
d. Peripheral Drivers

Program components within these major elements are listed and described in paragraph 4.1.1 of the Requirements Specification Guidebook. A typical software maintenance problem with the Test Manager is reconcilling data handling mismatches between interrupt processors and remote operating modes.

5.2.4.2 Changes to Support Software. ATE support software usually consists of several language processors (ATLAS, FORTRAN IV, Assembler, etc.) development aids (loader, editor, etc.) and test station aids (linkage editor, maintenance software, etc.) as depicted in Figure 4.1-2 of the Requirements Specification guidebook.

Support software, like control software, is usually developed 1 - 2 years before test software (paragraph 3.4) and integration of the three major program areas can reveal design incompatibilities. Also, relatively new or untested support

software may still contain design or coding errors. The principal elements of ATE support software are listed below in order of decreasing maintenance work:

a. Language Translators
b. Test Station Aids
c. Program Development Aids

Although ranked last, the Program Development Aids can become the number one contributor to maintenance workload if numerous problems are encountered with the ATPG or ATLAS flow chart processor (see paragraph 4.1.2 of Requirements Specification Guidebook). A typical maintenance problem for ATE support software is that caused by conversion problems between host-computer outputs and ATE station operations. (When a host is utilized for support tasks.)

5.2.4.3 Changes to Test Software. As mentioned earlier, test software is by far the largest contributor to ATE software maintenance workload. Paragraph 3.4 points out the major cause as being frequent changes to UUT's. The principal elements of ATE test software are listed below in order of decreasing maintenance work:

a. UUT Test Software
b. ITA Test Software
c. ATE Station Test Software

Components of these are identified and described in paragraph 4.1.3 of the Requirements Specification Guidebook.

UUT test software, the category of greatest concern to the ATE maintenance programmers, is composed of four components: (1) ITA/UUT Identification; (2) Safe-To-Turn-On; (3) End-To-End Performance; and (4) Diagnostic Fault Isolation. The last element, diagnostic software, is responsible for about 90% of all UUT test software maintenance. End-to-end test software accounts for most of the remaining maintenance.

A typical task in diagnostic software modification is the result of ATE failure to locate a UUT fault. The UUT may checkout OK by the tester even though it contains a faulty subassembly or circuit board component.

5.2.5 Development and Maintenance of ATE Test Software

The activities of software development and software maintenance have many parallels and this is especially true of ATE test software for a digital circuit card (using ATPG and guided probe methods). When the UUT is a digital circuit card and a change is made to that UUT, then the process of developing and modifying diagnostic software is virtually the same. This process is described in the next section and then a second example of ATE test software modification for a different type UUT is described in paragraph 5.2.5.2. These two examples of ATE software maintenance provide representative scenarios of modification activities.

5.2.5.1 Test Software Modification For Digital Circuit Boards. A typical overall method to develop and maintain verified test software for the testing of a digital circuit board on an automatic tester is described. This method is the planned and systematic pattern of all actions necessary to provide adequate confidence that the ATE test software will perform satisfactorily. This method involves: An orderly, systematic controlled process; periodic reviews to assure quality and conformity to standards; the documentation of concepts and decisions; a permanent file for each UUT; and the verification of the quality of the final product. This process consists of four major elements:

a. Requirements
b. Design
c. Generation
d. Integration and Validation

a. Requirements for test software development (Figure 5.2-1) are determined from the Secondary Replaceable Unit (SRU) characteristic sheets for various UUTs and from an ATE Test Software Guide. These requirements will assist in formulating standard operating procedures and standard ATE interface adapters.

The SRU characteristic data is obtained from the Primary Replaceable Unit (PRU)/SRU source data file by an engineer using manual means (i.e., visual analysis). The source data file for each UUT consists of a logic diagram and a hardware drawing, providing electrical and mechanical data respectively. This data informs the user of the UUT design constraints and helps to shape standard constraints for all vendors. The SRU characteristic data consists of: UUT identification, electrical type (digital, analog, hybrid, power, Radio Frequency (RF), and memory), complexity (number of Integrated Circuits (IC), transistors, etc.), component types (linear, memory, etc.), mechanical data (board type, coating types, etc.), connector data, test point access, power input and clock signals, signal inputs and outputs, board failure recommendations, required adjustments, references, functional description, and notes. The note section may advise of additional testing requirements and unique signals for a UUT. Additional circuitry is usually required for a UUT, such as a personality board or frequency dependent parts.

b. Design - Part I (Figure 5.2-2) is defined by the Basic Test Plan and Outline and is under constant review by a quality-assessing-committee. The Basic Test Plan and Outline for each UUT may contain the following items: the SRU Characteristic Data Sheet, the ATE Adapter Interface Specification, the Test Flow Diagram, the Decision and Concept Report, the fault detection and isolation quantities, and the Test Program Estimating and Status sheet. The Decision and Concept Report is derived from the SRU Characteristic Sheet and the Test Flow Diagram. The ATE Adapter Interface specification is also derived from the same two items; however, a limited number of adapter types will be used which will affect the specification. The Test Flow Diagram will be affected by the UUT design constraints, the pattern generation constraints, and the fault detection and isolation constraints, as determined by a Technical/Management Committee to be a standard for all UUTs.

The user, after reviewing the material in the PRU/SRU source data file for the UUT design constraint information will identify: redundant paths, wired logic, jumpers, direct input to output circuit paths, circuits not normally used (portions of multi-circuit IC packages, installed spare components), any circuits or components (shunt diodes, bypass capacitors, Electro-magnetic Interference/Pulse (EMI/EMP) filters) which if failed or degraded in certain modes will not adversely affect the circuit performance, any physical constraints (mainly dip clearance and test point accessibility), the reliability of components (a component with a high failure rate should be detectable, whereas one with a low failure rate, if undetectable, can be ignored), and special or unique test problems (Random-Access Memory (RAMS), ROMs, workarounds, and illegals). Finally, the user should identify fault detection and fault isolation quantities that are believed attainable after considering the aforementioned data. A Test Program Estimating and Status sheet will contain the above data and, if applicable, include reasons why the estimated quantities are less than the set standard.

Design Part 2 (Figure 5.2-3) is defined as the phase of coding a circuit modal deck from a circuit schematic, or processing the vendor supplied tape or listing, and automatically deriving stimulus

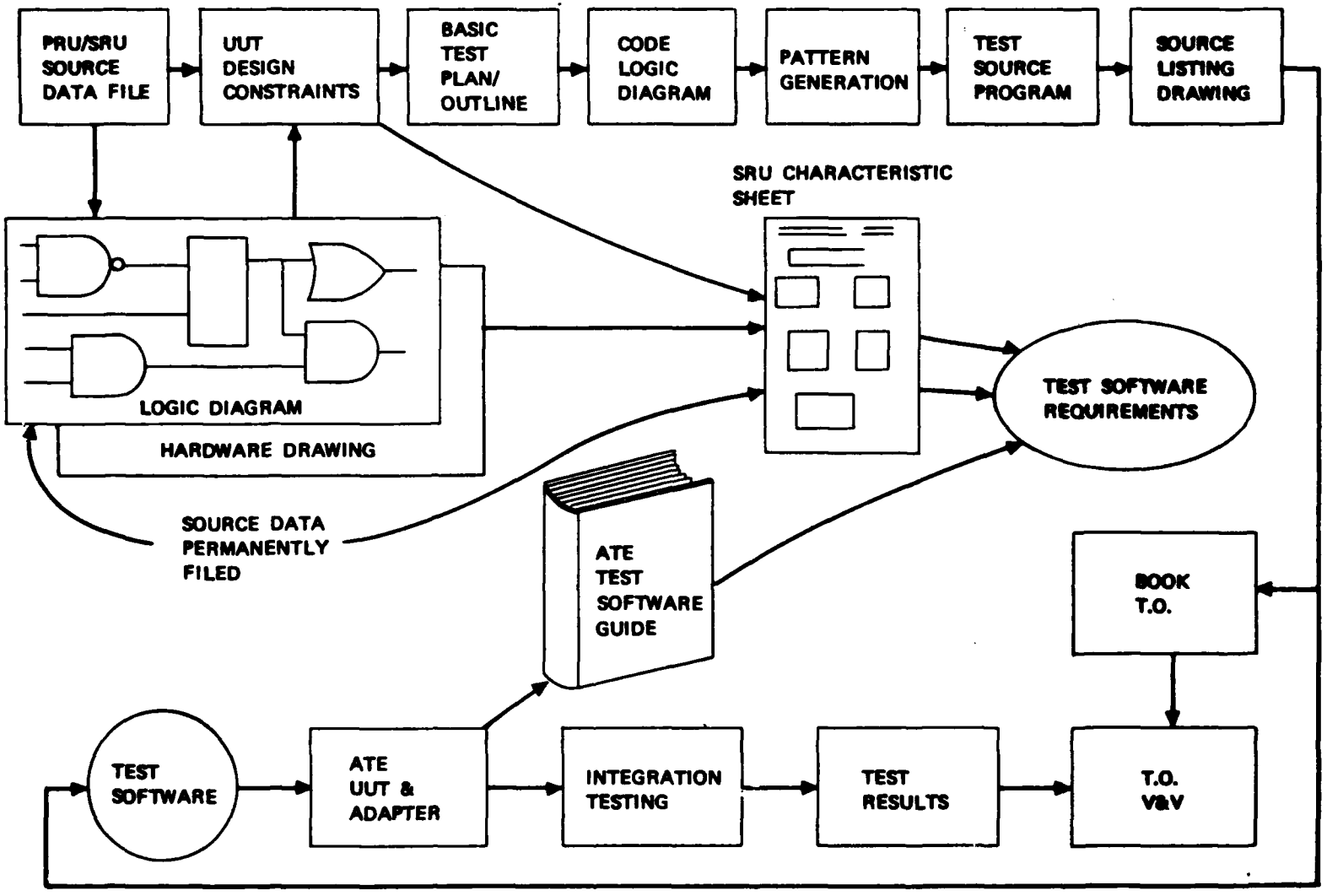


Figure 5.2-1. ATE Test Software Requirements

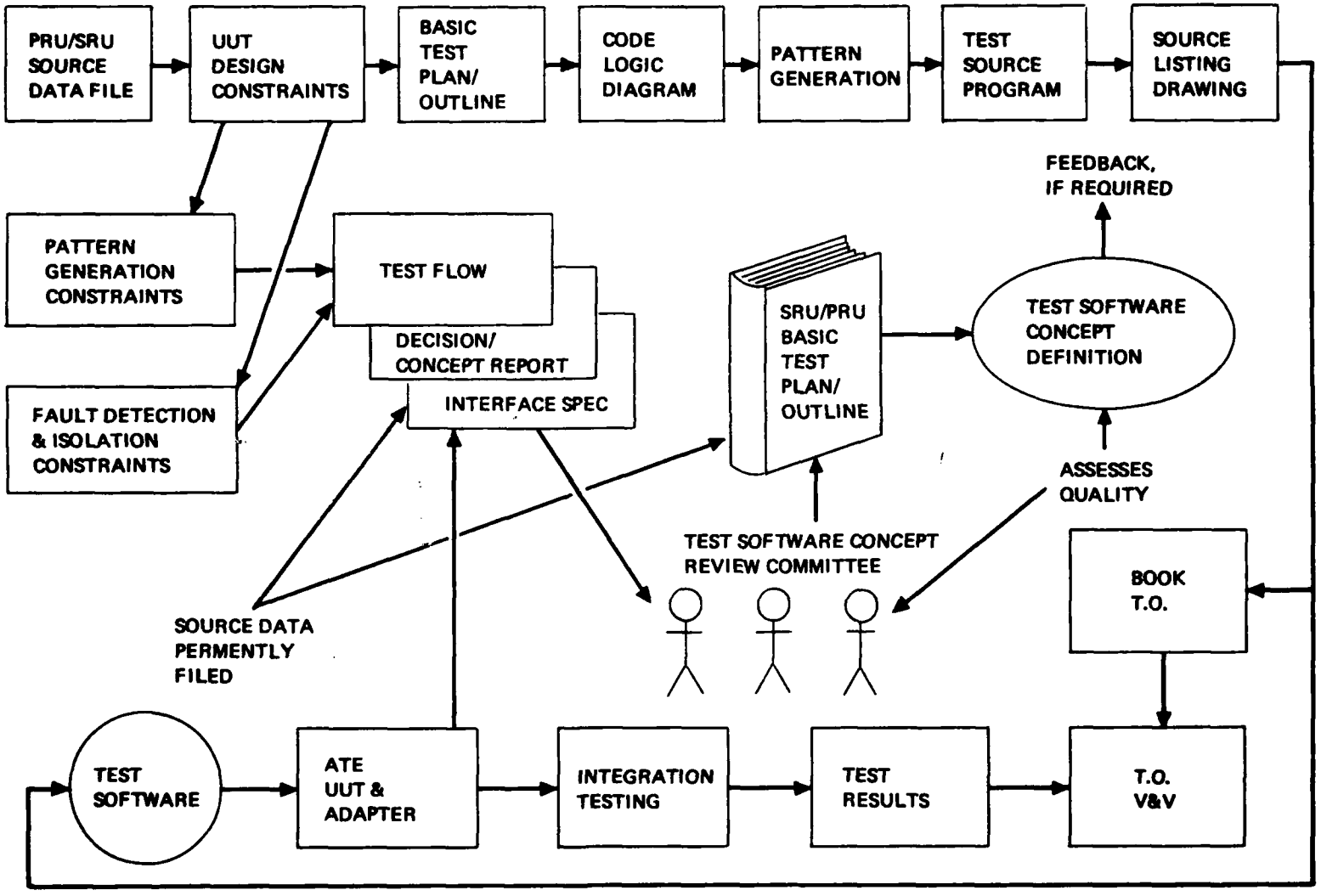


Figure 5.2-2. ATE Test Software Concept Definition - Part 1

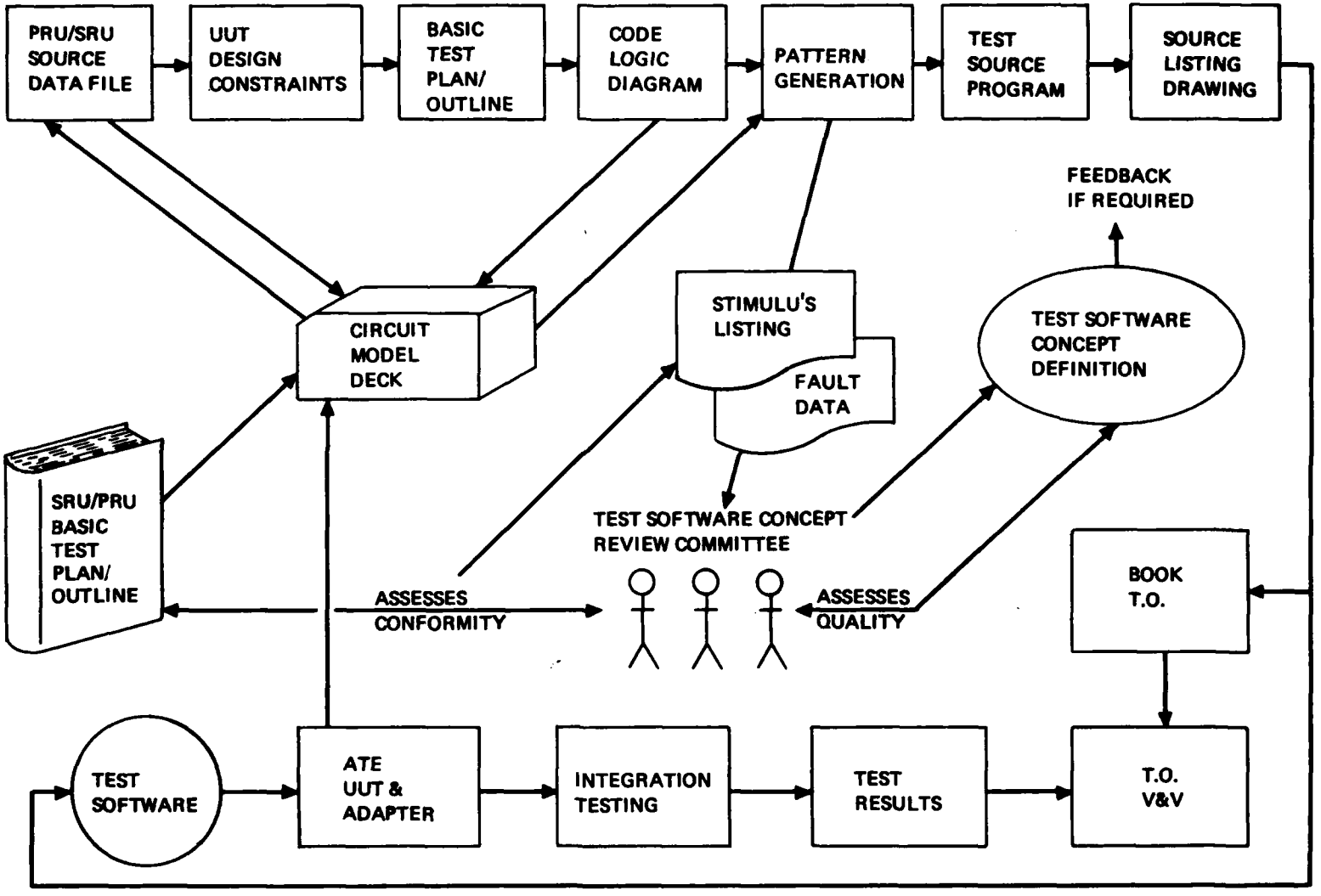


Figure 5.2-3. ATE Test Software Concept Definition - Part 2

and response patterns using ATPG (UUT is digital). After reviewing the Basic Test Plan and Outline material and the instruction manual for correct procedures on labeling, input modeling, and software options, the circuit can then be modeled from the logic diagram. If the circuit model data is available from another source, then a vendor to ATPG interface software program will be used to generate an ATPG usable input model (may require some modifications). The ATE-UUT interface adapter requirement for each vendor will determine whether a test point connector will be used as additional outputs (primary connectors will provide UUT inputs and outputs).

The ATPG system uses the circuit model, under user direction, to develop stimulus and response patterns, node state data, and usable fault isolation data (the guide probe isolation method may use a fault dictionary for some UUTs). If the stimulus and response data is also supplied by a vendor, then other vendor to ATPG interface methods (software or manual interpretation) can be used to obtain the required data.

The technical/management review committee will assess the conformity between the test plan guidelines and the ATPG generated data for a particular UUT.

c. Test Software Generation (Figure 5.2-4) involves the writing of an ATLAS program that tests the UUT and the transfer of information on the ATPG data tape containing the circuit description, the stimulus and response patterns, the node state data, and the fault dictionary, into an ATLAS compatible format for the ATE. An ATPG to ATLAS interface program, would be required as a data translator. A review of the Basic Test Plan and Outline will help determine voltage application and required statements for proper testing. This ATLAS data is stored on a new tape which will be duplicated for checkout purposes and later stored permanently. A source listing and drawing will be permanently filed as a document. A technical/management committee will assess the conformity of the Basic Test Plan and Outline against the product.

The subject of debugging the test software is addressed in paragraph 5.2.6.

d. The Integration and Validation (Figure 5.2-5) phase involves running the test tape on the tester with the associated UUT connected. The tester will apply the stimulus patterns to the UUT, measure the UUTs responses, compare those responses with the ATPG generated responses, and execute appropriate procedures to identify the failure if any responses are not what they should be. The guided probe, in conjunction with the fault isolation tables, if provided, will attempt to isolate an inserted fault on a UUT down to one component.

The Basic Test Plan and Outline will provide information relating to problems foreseen under this concept. If fault isolation is below standard, then the problem is analyzed, reported, and investigated by the appropriate group (digital, analog, etc.) for a solution. That solution is implemented for new data, verified on the tester as a correct fix and documented.

Finally, a problem close out report is filed. The ATE User's Guide (tester manual) will instruct the user on tester operations and on how to apply the probe for different component configurations on the UUT. The technical/management committee will assess the quality of the Test Report (permanently filed) and analyze the results for conformity with the expected results outlined in the Basic Test Plan and Outline.

5.2.5.2 Typical Test Software Update on Real-Time Disc Operations System. A typical real-time disc operations system consists of four major software components (see Figure 5.2-6).

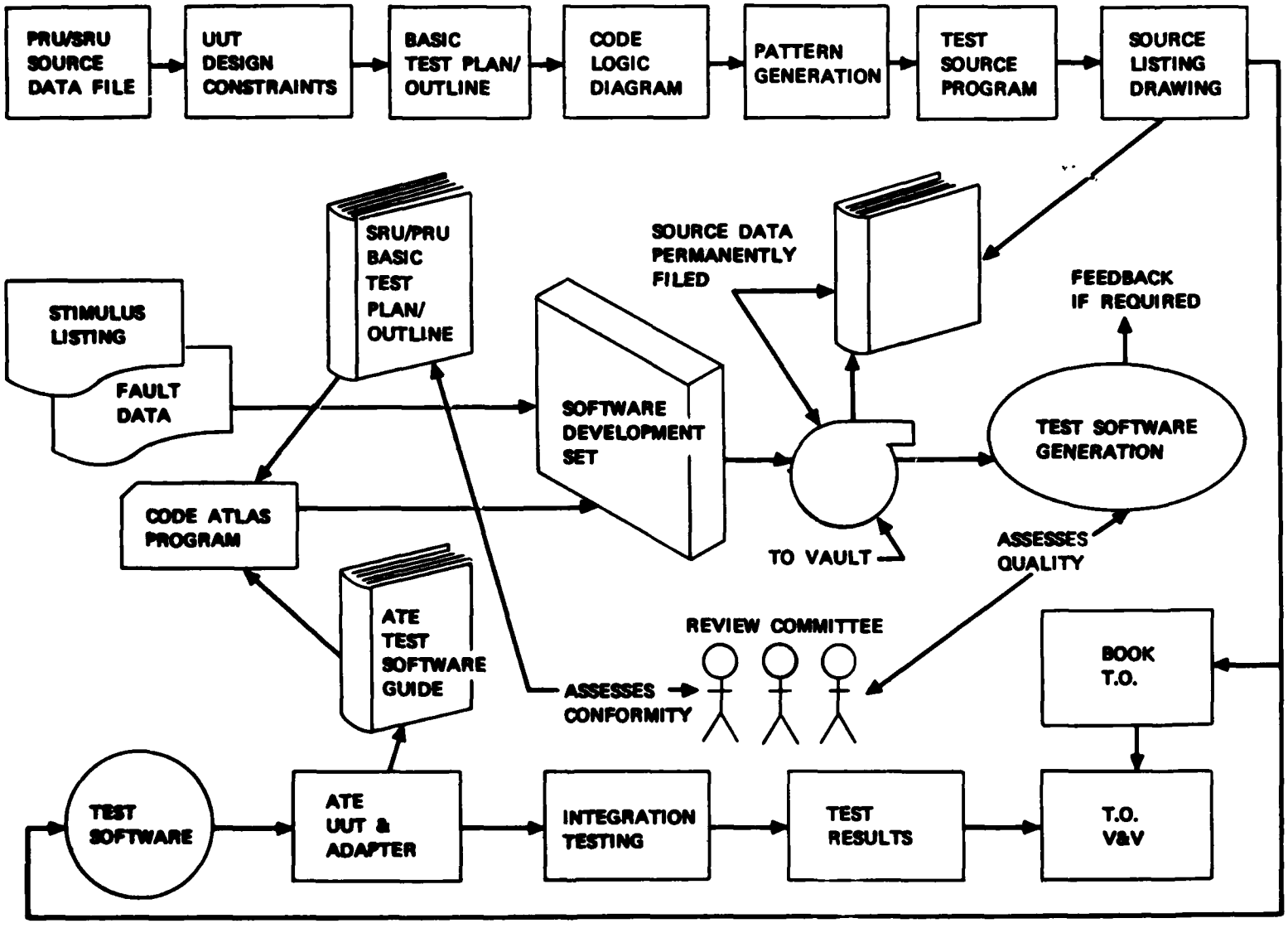


Figure 5.2-4. ATE Test Software Generation

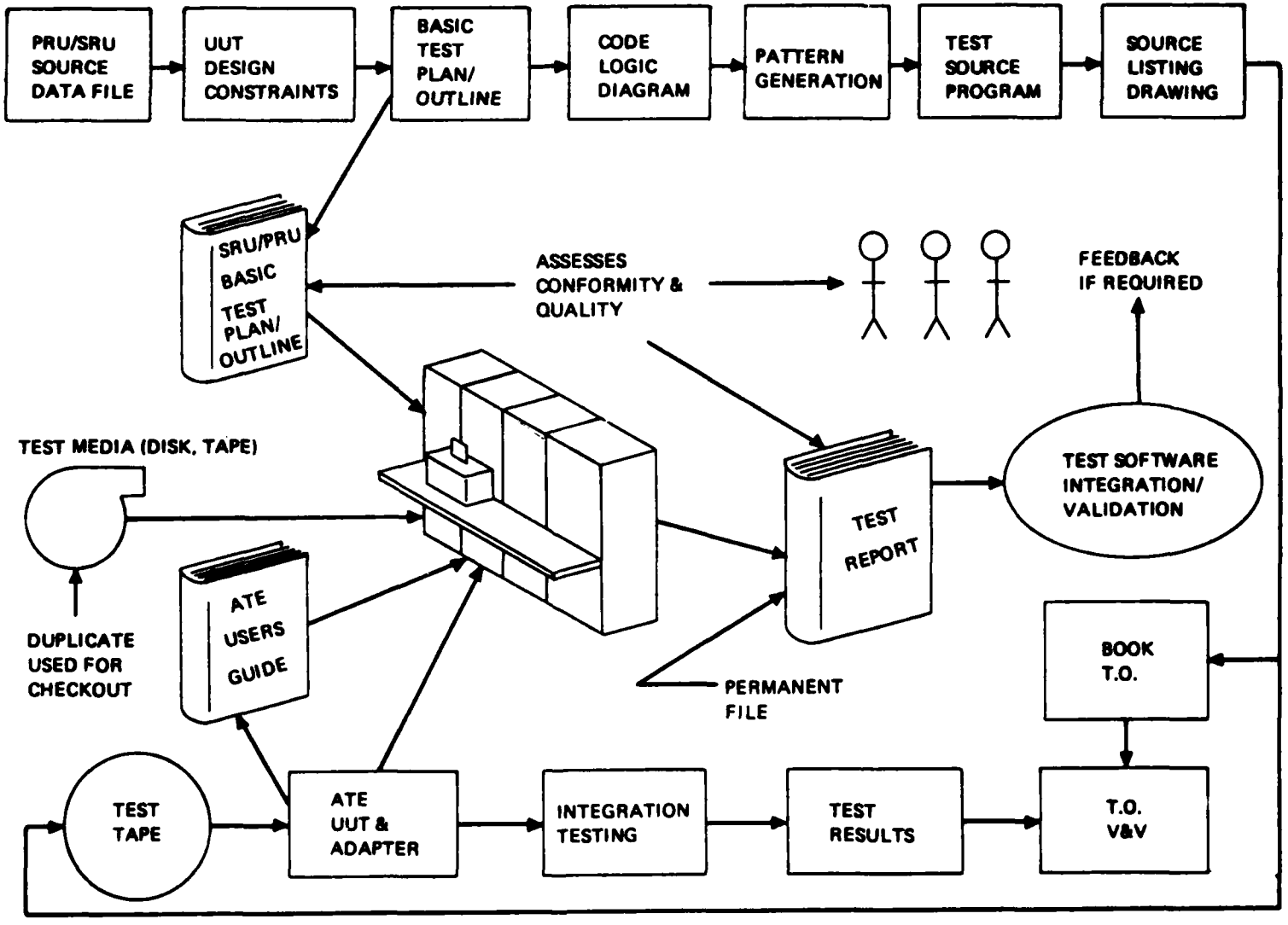


Figure 5.2-5. ATE Test Software Integration/Validation

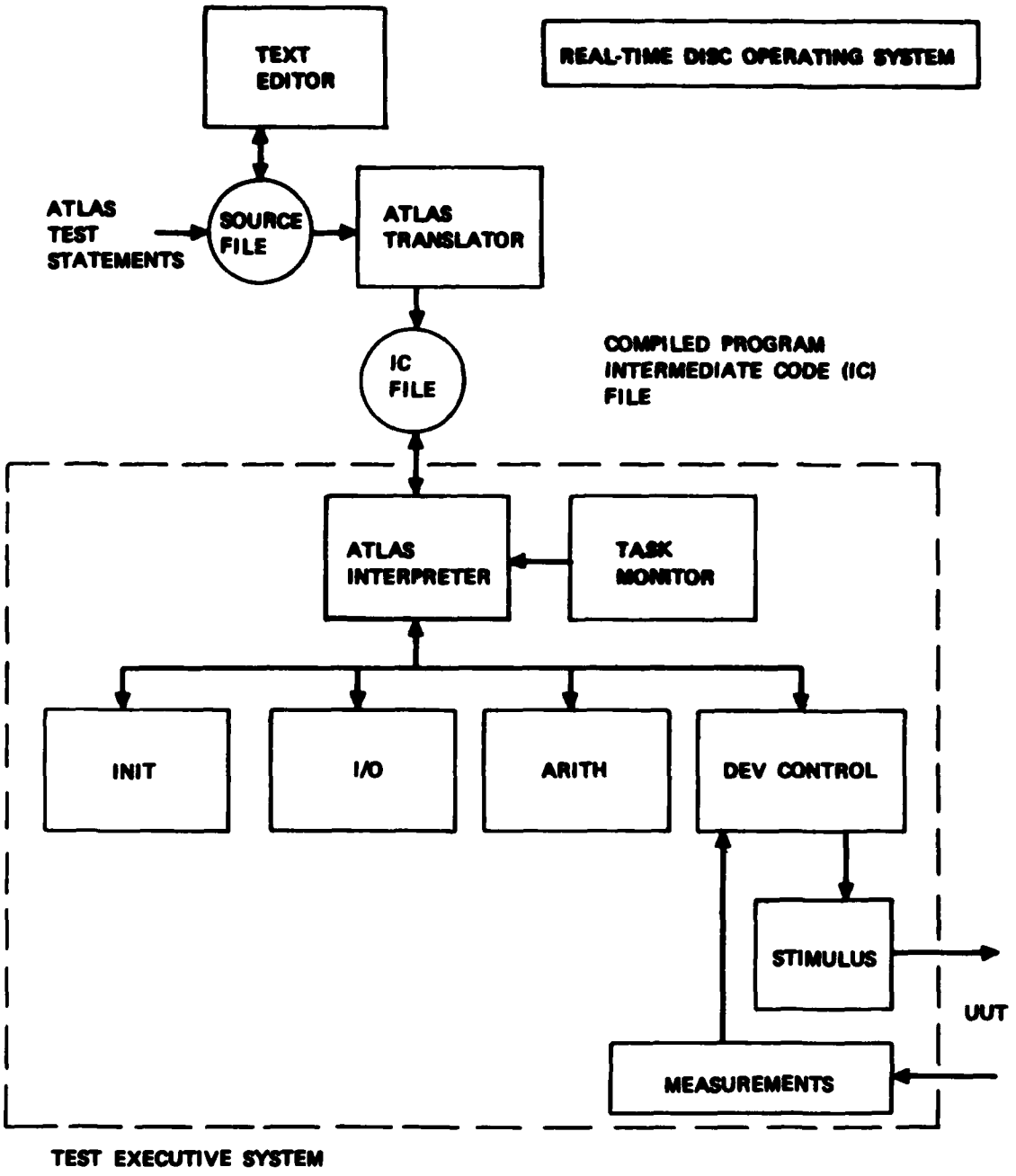


Figure 5.2-6. ATE Test Software System Organization

a. A text editor used to create and modify source programs (Test software source statements) written in the ATLAS language.

b. An ATLAS compiler/translator which accepts as its input, ATLAS source statements and generates object code as its output.

c. A test executive system which takes the code generated by the compiler and controls and executes the test sequence based upon these instructions.

d. An operating system, under which all of these programs execute.

The text editor typically performs the following operations to operate, modify or update ATLAS source statements.

a. Cursor control (e.g., move cursor right one character, move cursor to end of line, move cursor to end of text on screen, etc.)

b. Text insertion, deletion and transportation

c. Input and Output

d. Context Editing

The ATLAS compiler/translator accepts ATLAS source statements (as defined by ARINC Specification 416) and generates object code which is executed by the text executor system. The ATLAS compiler consists of a main segment which dispatches control to separate processors for each ATLAS verb type and a set of supporting routines. These routines perform basic compiler functions: lexical analysis, arithmetic expression analysis; symbol table manipulations and memory arrangement.

The ATLAS test executive system provides all run-time functions needed to run test programs in both an interactive mode for debugging and validation and also in a non-interactive mode for final programs.

The real-time disc operating system handles all input/output providing file/device independence at run time. It manages the disc memory, maintains program directories, and provides the scheduling and resource allocations in multitask environments.

Other successful ATE systems have been based on other architectures using interpreters (ATLAS or BASIC), conventional on-line compilers using a linking loader, or off-line compilers using an interpretive run-time system (VITAL).

5.2.6 Debugging Test Software

Regardless of the techniques used to generate the input stimuli and obtain the expected output responses, the test program must be debugged by testing with the UUT. Most new test programs have "bugs" and will cause "spurious" test results.

A common debugging problem for circuit board UUTs is obtaining a "known good" board. Boards that have been tested at the product level often still contains undetected faults because of the incompleteness of the product level tests. In addition, there may be designer errors that result in different responses between several good boards. Another problem results from the use of incorrect schematic diagrams.

The objective of the debugging procedure is to determine why actual test results differ from expected results and then changing the test software to obtain the desired results. The techniques used to generate the input stimuli will affect the debugging procedure.

Computer-aided stimuli generation and simulated output responses facilitate the test software checkout.

Simulated output responses will have known indeterminate conditions which can be surpressed prior to debugging. The simulation data aids the programmer in analyzing discrepancies between expected test results and actual results. Simulator diagnostics will rapidly isolate board faults if the problem's not a coding error.

Using two debugging methods, e.g. manual analysis and simulation techniques provides independent responses, so the physical boards do not have to be completely "known good." Other techniques use comparison tests with a "known" good board, but these tests must be verified using several good boards.

Debugging time can be reduced if program changes and test executions are conducted interactively. This provides immediate feedback on results of test software modifications.

5.2.7 Control of ATE Software

ATE control and support software will be controlled in much the same manner as changes to TS software discussed in paragraph 5.1. However, if the documentation of ATE control and support software consists of commercially developed documents, particular care should be taken to investigate contractor proprietary rights, guarantee clauses, etc., before modifications to this software are approved. All facets of the USAF change management system (see paragraph 5.3) will apply to the modification control and support software.

Test software will also be subject to the USAF change management system. In addition, when validated test software must be changed, adequate source listings, drawings, technical orders, etc., must themselves be verified and validated before the change is approved for implementation.

5.2.8 Pitfalls to Avoid During ATE Software Maintenance

ATE control and support software although developed in accordance with good commercial practices may still contain errors or deficiencies which will impact ATE software development. It will be important to develop a formal problem reporting system with the contractor which will address problems encountered and which will specify a timely response time for those causing severe impact. The contractor should also be responsible to test modified software for regression problems.

ATE test software, characteristically different from control and support software, will require very stringent test procedures to ensure that a software change achieves the intended purpose and does not create regression problems. Should a UUT successfully pass a formal test and be returned to the field as a fully tested replaceable unit and then fail as a result of an incorrect software change, serious problems may occur.

5.3 CHANGE MANAGEMENT

Configuration control of changes to ATE and TS software is essential and, in fact, can be a matter of life and death. Evidence exists of cases where uncontrolled modifications to ATE software have resulted in loss of life. In any event, poor management of software changes can be costly and have negative impact on system reliability and availability.

The subject of change management is given detailed coverage in the Configuration Management Guidebook. Basic concepts are introduced in this section with particular emphasis on the software maintenance activity during system acquisition. Before addressing that subject, however, the concept of a Central Maintenance Facility is discussed.

Software maintenance aids, i.e.: support programs and computers, can be provided such that program modifications can be made at virtually every installation of an ATE or TS system. Whether such decentralized software maintenance is advisable can be challenged on the basis of both cost-effectiveness and change management. It is recommended that LSA studies include detailed trades analyses if decentralized software maintenance is being contemplated. The alternative to decentralized maintenance is the Central Maintenance Facility in which all skills, techniques, resources, procedures, etc. are available to conduct efficient, systematic software modifications under close control. Once a change is fully implemented (coded, verified, validated, documented) then the change can be properly communicated to all ATE/ TS installations. Close control is more difficult under decentralized maintenance and also there is either (1) a duplication of skills, resources, etc., between locations, or (2) deficiencies in skills, resources at distributed locations. Thus, the Central Maintenance Facility is an attractive alternative.

5.3.1 Change Management Definition

Change management is the discipline which applies technical and administrative direction and surveillance to (1) properly identify, (2) control changes to, and (3) record and report change implementation status of the total configuration of the TS and ATE systems.

The essence of configuration identification is that every system/equipment or component thereof, will be identified by technical documentation as set forth in specifications, drawings and associated lists, and documents referenced therein. The recognition of this technical documentation, at a specific time, results in the establishment of a baseline. Once a baseline is established, this baseline plus approved changes thereto constitute the current configuration identification.

Change control provides the systematic evaluation, coordination, approval or disapproval, and implementation of all approved changes to the approved configuration identification.

Configuration accounting provides the recording and reporting of the configuration identification and change information that is needed to manage the configuration effectively.

The principal specifications for CPCI configuration identification are the Part I specification which establishes the computer program design requirements; the Part II specification which documents the complete and final design; and the interface specification which defines the requirements for interfaces external to the CPCI. As indicated in Section 4.0 of the Requirements Specifications guidebook, software for ATE UUTs depends on ATE/ ITA design and on the performance and diagnostic test requirements, which are often documented in the TRD. Further, this ATE test software is often controlled through the use of T.O.s. Formal change control is established for CPCI specifications when they are approved. Software change control is established for the other CPCI elements such as drawings and documents, up to the point when they are incorporated in a specification.

The principal drawings and documentation which pertain to configuration management of CPCIs include the following: interface control drawing, CPCI index, change status report, VDD manuals, and handbooks. The physical media are comprised of the tapes, decks, etc., which contain the computer programs and data.

5.3.2 Baseline Management

Under the baseline management concept, technical control points, or "baselines", or configuration central points, are established for computer program development and systematic evaluation,

coordination, and disposition of all proposed changes to these baselines is established and maintained. The CPCI parts I and II specifications shall be used to document the baselines for computer programs. This section discusses the baselines which are formally established for the development of computer programs: The allocated baseline and the product baseline. These baselines are documented in the part I specification and part II specification, respectively.

5.3.2.1 The Allocated Baseline. The allocated baseline is established by USAF approval of the part I specification which contains the performance, design and qualification requirements. The part I specification of the two part CPCI specification requirement is used to identify the design and testing activities of the project. Any change or addition to it must be submitted as a design requirements change and must be formally approved before the change can be implemented. The allocated baseline is controlled throughout the design development test and integration of the CPCI.

5.3.2.2 The Product Baseline. The part II specification becomes the documented product baseline. The part II specification shall be audited to determine that it adequately describes the fully assembled CPCI.

5.3.2.3 Baseline Control Documentation. The design requirements to be followed during CPCI development and the requirements for qualification shall be documented in the part I specification. After approval, this specification is designated as the allocated baseline. It shall govern the design, development, and testing of computer programs and serve as the baseline against which the impact of proposed performance and design changes are to be assessed.

The part I specification shall be placed under formal change control and configuration accounting upon USAF approval.

After the computer program components have been assembled to form the CPCI and subjected to preliminary qualification tests, the completed technical descriptions shall be documented by the part II specification. This specification shall contain the description of the overall design, the programming specifications, flow charts, and listings. After approval, this specification shall be established as the product baseline and shall serve as an instrument for use in diagnosing troubles, adapting the computer program to environmental and operating requirements of specific site locations, and for establishing minor or major changes to the computer program system.

The sections of the part II specification submitted for formal review shall be committed to software change control, as a minimum, at the time of those reviews.

Change control and accounting for both the allocated (part I) and product (part II) baseline specification shall be maintained throughout, and subsequent to, hardware/software integration.

5.3.3 Support and Control Software

Each support software program which in anyway affects development of the CPCI shall be placed under change control at the earliest appropriate point in the CPCI development. These include programs used to compile, assemble, update, or generate test inputs, etc.

Normally, the compiler/assembler (ATLAS, FORTRAN IV, Assembler) is the first support software used during CPCI development. It shall be placed under software change control no later than the start of TS computer program component (CPC) testing. In addition, testing and simulation programs used to support design and development testing and integrated system testing shall be designated for formal change control at the appropriate point.

5.3.4 Change Management Reports, Documents.

5.3.4.1 CPCI Index and Change Status Report. The computer program configuration item index is issued initially after completion of the preliminary design review and is updated and released following each major CPCI review and audit. The initial release of the CPCI index, plus the subsequent issues of change status reports, provide a progressive status of major events and schedules associated with the overall development of the CPCI.

5.3.4.2 Version Description Document. At the time of software/hardware integration testing, a VDD shall be prepared for each CPCI tape disk or other applicable media. After initial release of the CPCI tapes/disks, to either the customer or to integration testing, the VDD shall be used to control CPCI tape disk/revisions.

5.3.4.3 Manuals and Handbooks. Certain manuals and handbooks which are developed for CPCI projects shall be placed under software change control. The computer programmer's manual, which defines the rules and use of the compiler/assembler language, shall undergo software change control at the start of the coding effort. Also, the contents of the following documents shall be placed under software change control after completion of the PCA: The positional handbook, which defines the man/machine interface requirements, and the operator's or user manual, which is is used in running the programs and in operation of the computer system.

5.3.4.4 Interface Control. Interface control requires coordination of activities needed to assure that the CPCI and system characteristics are compatible. Interface control is established over the interfaces of the total computing system, of which the CPCI is an element. Certain of these system hardware, software, and inter-organizational design interfaces affect the CPCI design. The software group shall identify all such interfaces and establish administrative controls or liaison with them throughout the CPCI development process. Interface control for CPCIs shall be over the characteristics of the common boundaries between two or more CPCIs, between a CPCI and a hardware element, or for the design interfaces between a contractor and another contractor or agency.

Interface requirements shall be established concurrently with the development of the part I specification and shall be documented and released in an interface control specification (ICS). An ICS shall be developed for each interface which exists with the CPCI and both parties to the interface shall approve the specification. After USAF approval of the interface specification, all changes shall be controlled under formal change control procedures.

Concurrently with the CPCI design, detailed interface design drawings shall be developed for each of the interfaces identified. The detailed interface design shall be governed by the controlling interface specification. Each interface design shall be documented by an interface control drawing (ICD). After approval by both parties to the interface, the ICD shall undergo formal change control. Formal change control over the detailed interfaces shall occur no later than the CDR and shall continue throughout the duration of the CPCI project.

5.3.5 Change Control

The change control function provides the disciplined environment and administrative framework to control changes to the configuration baselines.

5.3.5.1 Change Classification. Changes to an established baseline are considered to be either of two types, class I or class II.

5.3.5.2 Class I Changes. Class I changes are those which, because of their criticality, require formal customer approval before the contractor can implement them. Changes shall be designated as Class I whenever one or more of the following is affected:

a. Operational capability, as specified in the baselined part I CPCI specification.

b. Contract price or schedule.

c. Systems equipment, computer programs, or facilities produced by another contractor, to the extent that the contractor must accomplish a change to maintain compatibility at the interface.

The ECP is used for the definition, explanation and coordination of all Class I changes. ECP content and preparation will be discussed in the Configuration Management Guidebook. Typical Class I change processing flow is depicted in Figure 5.3-1.

5.3.5.3 Class II Changes. Class II changes are those which the contractor may implement without prior approval by the USAF and at no additional cost to the USAF. Such changes may include the following:

a. Changes to correct editorial errors.

b. Changes to correct computer program errors.

c. Other changes of a minor nature, within categories specifically defined by the USAF.

5.3.6 Configuration Accounting

Configuration accounting is the reporting and documenting activities required for keeping track of the stauts of configurations at all times during the life of a system. For computer program items, configuration accounting is principally a matter of maintaining and reporting the status of the specification, associated documents and proposed changes. This function requires the data outputs delineated in the following paragraph.

5.3.6.1 Specification Change Notice (SCN). The SCN is the document which establishes corrections to and updating of the CPCI specifications. It is used to propose, transmit and record specification changes. A pre-determined number of SCNs which may be accumulated against a specification is usually established as a criteria for initiating a revision and reissuance of the specification. Other accounting records, indexes, VDD preparation, etc., will be discussed in the Configuration Management guidebook.

5.3.7 Software Change Control Board

The purpose of the Software Change Control Board is to evaluate software change requests for cost, schedule, design and other areas of impact, to approve or disapprove the change, and to issue a statement of commitment on the change. The Software Change Control Board conclusions are then summarized and distributed. Change processing procedures are established early in the project planning for identifying and implementing changes against the CPCI specifications and drawings. Change processing requires that the problem be identified and processing initiated on the appropriate change paper. An example of the flow of a software change request is presented in Figure 5.1-1. A change

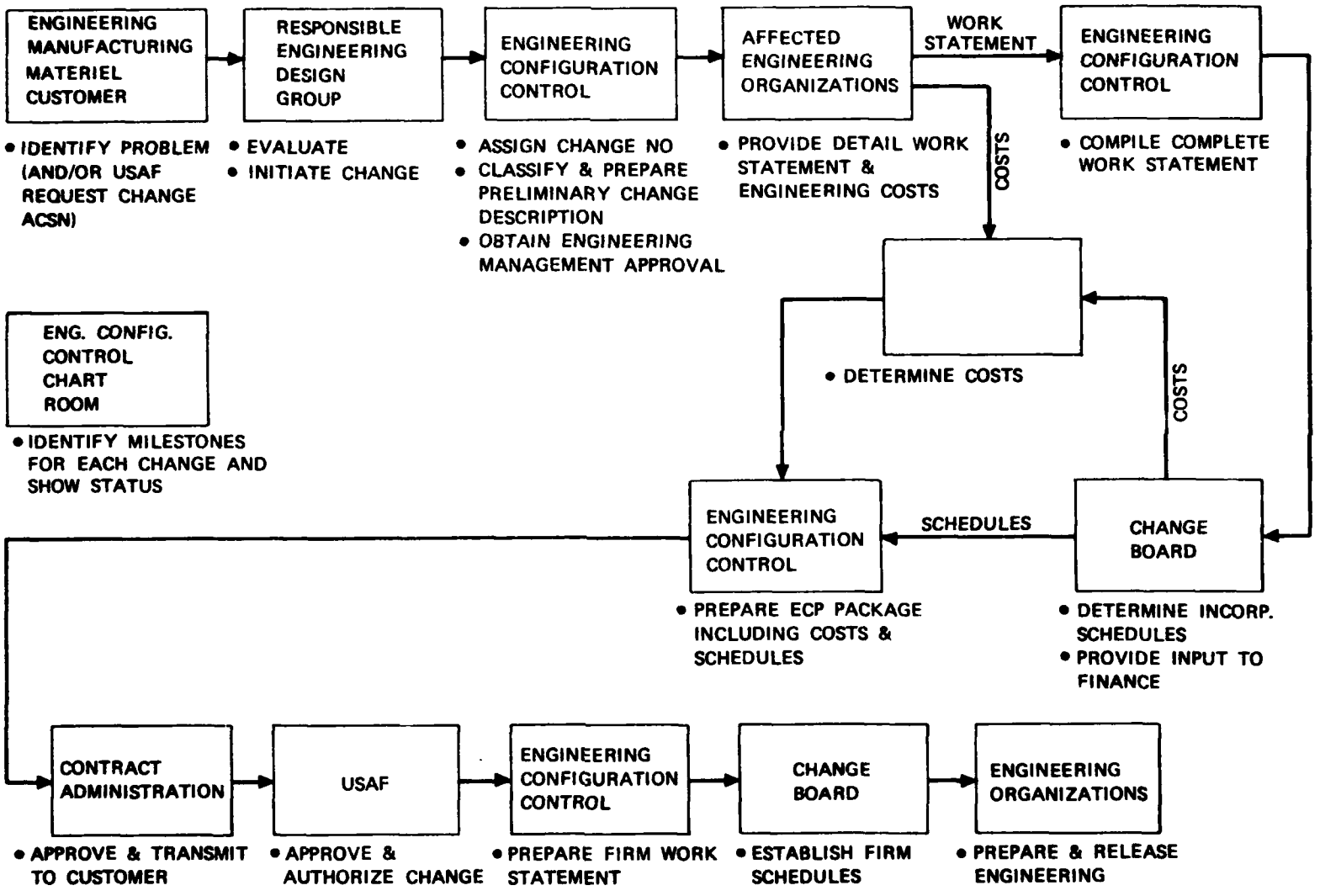


Figure 5.3-1. Typical Class I Change Processing Flow

request may be initiated by the software organizations, the USAF, or through a change in the interface with another software or hardware design organization. The change request is coordinated with affected organizations to determine the scope and extent of impact. Following this coordination effort the responses are compiled, documented and submitted to the Software Change Control Board.

5.3.8 Responsibility of USAF and Contractors

Large, complex TS or ATE software systems may well involve several USAF Commands, e.g., AFLC and Tactical Air Command (TAC), a prime contractor and one or more subcontractors. Because of the contractual aspects, both during the development and the maintenance phase, it is very important that changes be defined, developed and processed in accordance with documented change management procedures. It may well be that all of the above organizations have change boards to evaluate a particular change. Visibility should be provided to the USAF to define the purpose and methods employed by the other organizations.

5.3.9 Maintainability of Source Program Code and Documentation

TS and ATE software controls should require the timely maintenance of computer program source language code and the appropriate documentation (Part I, Part II, User Manual, T.O.'s, etc.) when software changes are implemented. Some software organizations permit the use of patches to object or executable programs during the checkout or debug phase. Although the use of patches may accelerate the debug phase, it is recommended that all patches be removed, source language corrections be recompiled and the newly generated object program be used in the acceptance test. Documentation should be prepared in draft form during the development of the change. All documentation should be finalized and distrubuted to the appropriate organization for review and approval before the change is closed.

5.3.10 Typical TS Change Management System

The change management system of a subcontractor will often be implemented at the stage in the real time software development when the tested software is integrated with the simulator hardware or earlier for individual software units on which testing is finalized.

The change management system will ensure that changes to software will result in the following:

a. Assessment of program timescale and cost penalties;

b. Approval by the designated authorities;

c. Production of the timescale plan for implementing the change, and monitoring of this timescale;

d. Identification of all documents affected by the change and monitoring of the resultant alterations;

e. Identification of verification software affected by the change and monitoring of the resultant alterations;

f. Records of all configuration changes.

In addition, for changes requiring contractor approval, a change order will be submitted to the contractor. Such changes are defined in general as follows:

a. Changes to documents forming the agreed Design Requirements Baseline, Contract Specifications, or Approved Acceptance Test Schedule.

b. Changes affecting financial liability.

c. Changes affecting major milestones in the program.

Section 6.0 SPECIAL REQUIREMENTS FOR SOFTWARE MAINTENANCE

The planning and conduct of software maintenance were discussed in Sections 4.0 and 5.0 but there are several topics relevant to software acquisition that warrant special consideration. These are (1) maintenance support environment; (2) program transfer and system turnover; and (3) software maintenance training. The significance of these topices to effective software maintenance is sometimes overlooked and consequently is highlighted in this section.

6.1 MAINTENANCE SUPPORT ENVIRONMENT

The support environment in which software maintenance is conducted can make the difference between responsive, low cost maintenance and difficult, costly maintenance attended by serious degradation in system reliability and availability. Elements of the software maintenance environment are:

a. Software maintenance budget

b. Skilled and trained personnel

c. Facilities (physical plant and furnishings)

d. Computing equipment and peripherals

e. Maintenance Organization structure and assignment of responsibilities

f. Contractor or vendor support

g. Computer program documentation

h. Technical data base

i. Special test equipment

j. Computer aids to software maintenance

k. Maintenance information analysis and dissemination

l. Equipment and supply support

m. Maintenance policy and procedures

n. Computer program design, re: maintainability (see Table 3.1-2 for attributes of maintainable software)

All elements are important and deficiencies in any element can result in poor maintenance capability. For example, availability of computers and necessary interface equipment to conduct software maintenance can be critical. Computer availability options are (1) dedicated computers; (2) dedicated time on computers; and (3) computer time sharing. The choice of option and degree of availability both impact software maintenance capability.

6.1.1 Computer Aids to Software Maintenance

Another critical element is computer aids to software maintenance. These range from simple utility programs, (ref: for taking memory dumps) to sophisticated automatic program restructuring and diagnostic routines. A wide variety of software aids to software maintenance is available and an important acquisition decision is to select appropriate aids to be procured with the software system. Software maintenance tools employed by the development contractor should also be a concern. Contractor propriatory methods and the availability of GFE software aids should also be considered.

6.1.2 Types of Computer Aids

Software tools for software maintenance can be classified under three general types: (1) utility modification aids; (2) static analysis aids; and (3) automatic program analyzers and aids.

Utility modification aids are general utilities that facilitate the process of error analysis and program modifications. These include:

a. Compilers

b. Assemblers

c. Linkage Editor

d. Loader

e. Media Converters

f. Text editor

g. File routines

h. Dump routines

i. Simulators

j. Recording routines

k. Configuration status routines (for Program Support Library)

l. Test Drivers

m. Program patch routine

n. Memory access and alteration routines

Static analysis aids provide tabulations for tracking data flow/use and module relationships. Such aids include:

a. Set/Use Matrix (cross reference analysis)

b. Common Data Pool (COMPOOL) listings

c. Data base analyzer (set/use/output)

d. Consistency analyzer (module interfaces)

e. Trace routines (machine and program status)

Automatic program analyzers and aids provide an array of tools to the maintenance programmers for error detection, program structure analysis, operating dynamics analysis, and other assists. These tools include:

a. Automatic Test Case Generators

b. Automatic Flow Charters

c. Automatic Program Execution Analysis

d. Coding Standards Auditor

e. Pre-compiler Instrumented Programs

f. Automatic Program Structuring

Numerous such tools are in use (some of which are proprietary) and several comparative studies have been made (References 2 and 5 Bibliography).

This brief introduction to computer aids for software maintenance indicates the depth of resources available to maintenance programmers. All resource elements, as listed at the beginning of paragraph 6.1, are to be specified in software maintenance plans, such as the *CRISP and CPDP. Specification of these* resources is of major interest to the software acquisition engineer.

6.2 PROGRAM MANAGEMENT TRANSFER AND SYSTEM TURNOVER

Perhaps the most difficult and vulnerable time period for software development and maintenance is that critical interval when systems are turned over, and management responsibility is transferred from the implementing command to the supporting command and/or using command. Usually this period is characterized by a still-fluid software configuration as error corrections and performance improvements are being incorporated. Computer program discrepancies are still being discovered during this period and previously known discrepancies are still in the process of being corrected. Thus the change environment,

e.g.: software maintenance workload, is much like that during full scale development. One major difference, however, is that an experienced contractor programming team is being phased out by an inexperienced Air Force programming team. Consequently, the planning and conduct of software maintenance during the transfer/turnover period is extremely critical.

The objective of rational planning for software maintenance in the transfer/turn-over period is to provide for an orderly and cost effective transition. Such planning can be accomplished, as will be evidenced in the sections which follow. Whether such planning succeeds depends on the diligence and vigilance of the software acquisition engineer.

6.2.1 Major Elements of Transfer/Turnover Planning

Principal planning for a smooth transition in software maintenance during the transfer/turnover period is contained in three documents:

a. PMRT Agreement

b. System/Equipment Turnover Agreement

c. Turnover Certificate

Planning documents having major input to the above three documents are the PMP, ILSP, CRISP, and CMP. These four planning documents are described in the guidebook on Computer Program Documentation Requirements.

The PMRT is prepared and updated by a Transfer Working Group (TWG) represented by the implementing, supporting, and using command. Development of the PMRT plans begins early in full-scale development phase and is effective until all residual tasks are completed (action items remaining at PMRT date).

The System/Equipment Turnover Agreement is prepared under policy established by AFR800-19 and provides, among other planning guidance, the special change procedures to be implemented between system turnover and program transfer.

The Turnover Certificate augments the System/Equipment Turnover Agreement by specifying responsibilities for software corrections, forecasted correction dates, etc.

Contents of these documents and their relationship to other planning documents are described in the next section.

6.2.2 Transfer/Turnover Planning Process

Criteria for program transfer and system turnover are established by the Program Management Plan (PMP), based primarily on:

a. Logistics support planning
b. Organic support policy
c. Interim contractor support policy

This relationship is shown in Figure 6.2-1 which presents a schematic of the transfer/turnover process. The schematic also shows (1) how the CRISP and CPM relate to the PMRT; (2) what regulations govern the process; and (3) principal contents of the PMRT Agreement, System/Equipment Turnover Agreement, and Turnover Certificate.

The PMRT accomplishes a number of purposes, but one paramount objective with respect to software maintenance is to establish special change procedures and phasing requirements during the critical interval between full contractor maintenance responsibility and full Air Force maintenance. This planning is influenced, of course, by the CRISP and CMP. The Computing Resources Working Group (CRWG) is responsible for assuring that provisions of the CRISP are adequately included in the PMRT.

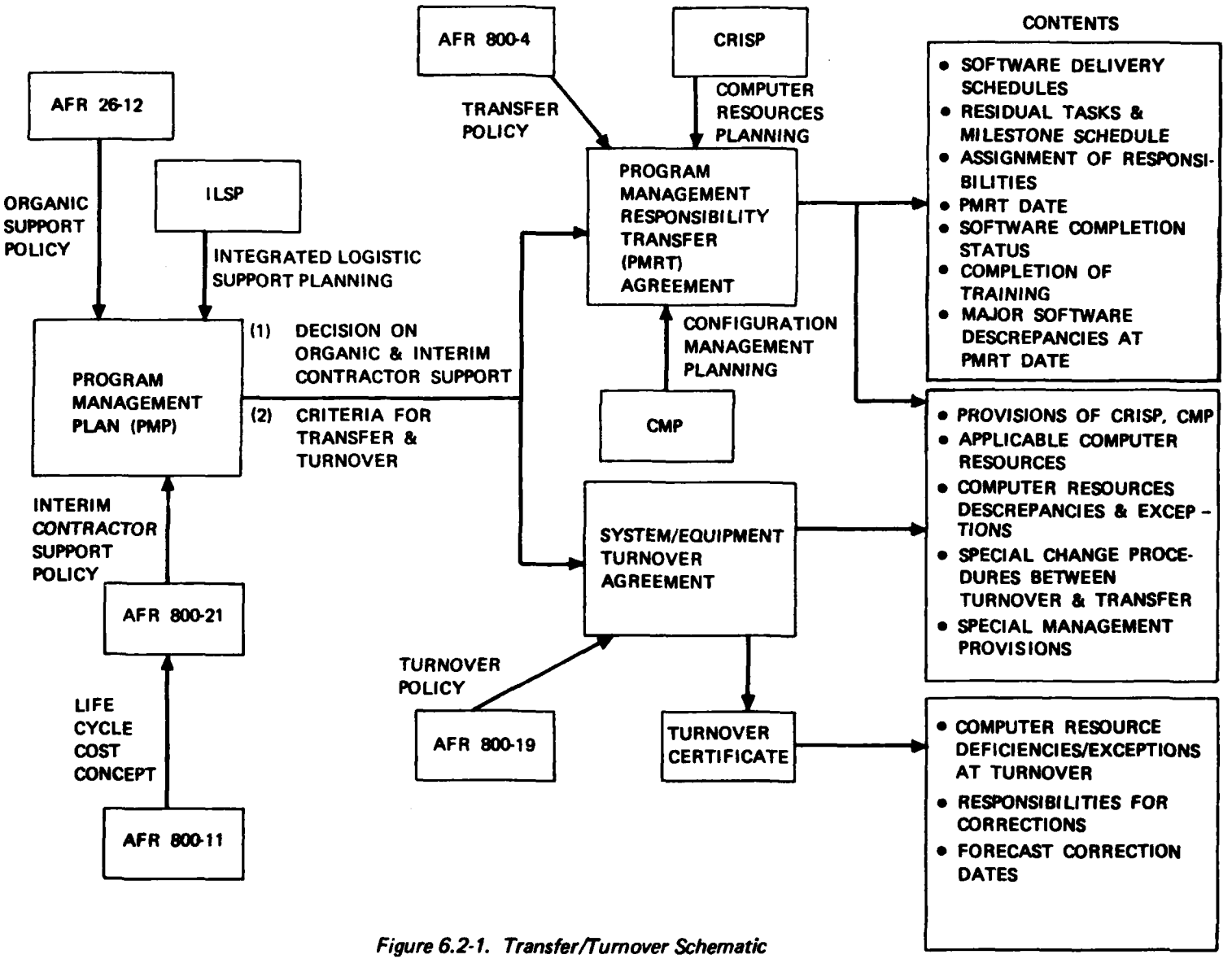


Figure 6.2-1. Transfer/Turnover Schematic

A key decision recorded in the PMRT is the scope and nature of contractor support of software maintenance to be provided in the interim period. (See Figure 6.2-1). The PMRT date is the official cut-off of all ECP's and the PMRT includes agreements for each and every transfer of system update or change modification. The decision on interim contractor support - and influenced by life cycle cost trends (Figure 6.2-1) - is discussed in the next section.

6.2.3 Interim Contractor Support and Life Cycle Costs

Policy governing interim contractor support (ICS) is contained in AFR800-21 (Figure 6.2-1) and the purpose of such policy is to:

a. Control and minimize capital investment in logistics support.

b. Allow time to refine logistics support requirements.

c. Assure resolution of technical problems.

d. Defer full organic maintenance until accpetable design stability is achieved.

Interim Support is normally provided during the period between first production system delivery and when full operational capability (FOC) is reached. The period can be extended, however, if justified through review, analysis, and experience. The decision to commit tasks to ICS is made during the full-scale development phase and specific planning is documented in the ILSP. The chronological sequence in the decision making process is described in AFR800-21. Planning for ICS is updated after test program results are analyzed, i.e.: before FQR.

A major input to ICS planning are life cycle cost trades, as shown in Figure 6.2-1. Life cycle cost concepts specified in AFR800-11 are applied to trades of organic versus contractor support (or mix of organic/contractor support). Cost analyses include, among other factors:

a. Salaries for logistics personnel
b. Spares and repair parts
c. Support and test equipment
d. Software modifications
e. Training and training aids
f. Military and contractor facilities
g. Technical data

Delta costs are computed as a function of fiscal year. Representative cost study results are depicted in Figure 6.2-2.

6.3 SOFTWARE MAINTENANCE TRAINING

Skilled and trained personnel is one of the major maintenance resource elements listed at the beginning of paragraph 6.1. Frequent reference is also made to training needs in planning and requirements documents. Yet, software maintenance training, as with training in general, tends to be an "also ran" when it comes to budget, administrative priority, and provision of necessary resources to develop and conduct a viable training program. Thus, it is not uncommon for the using and supporting commands to be ill-prepared to perform adequate maintenance until long after first production delivery. In this context, it is vital that the software acquisition engineer continue throughout system development to stress good planning and resources for software maintenance training.

Software maintenance training can occur by "osmosis" through the interface between contractor and Air Force programmers, or Air Force training can be systematically arranged through procurement of comprehensive training services from the development contractor. The latter method is recommended.

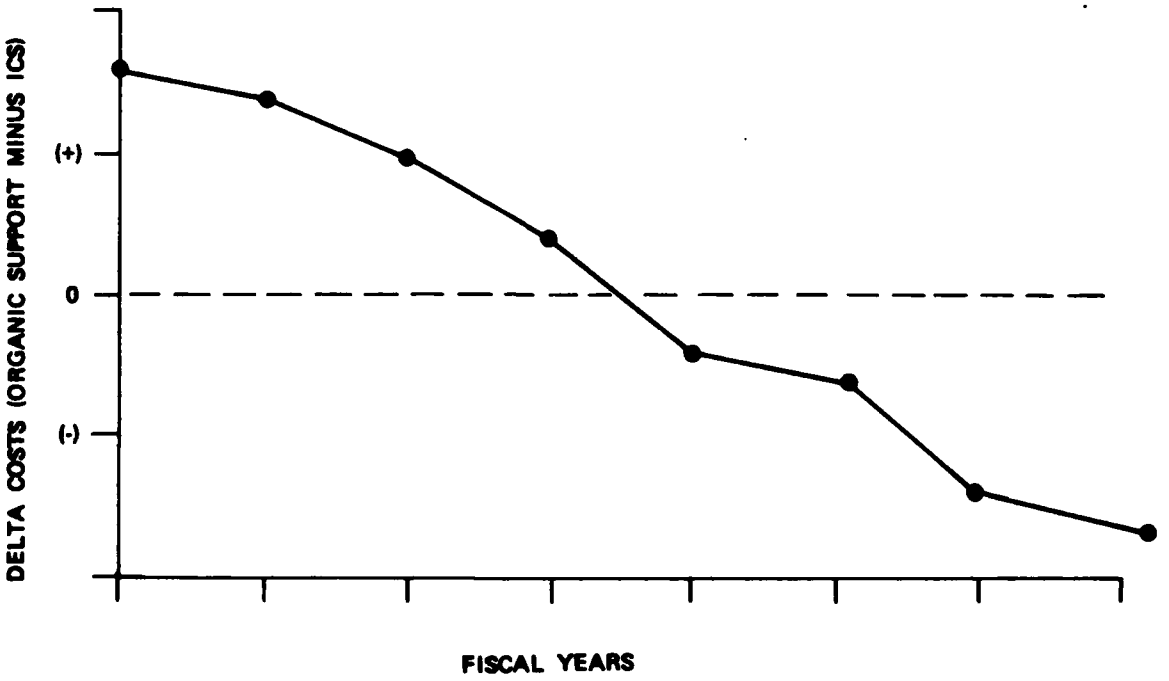


Figure 6.2-2. Life Cycle Cost Comparison for ICS Policy Decision

Comprehensive services for software maintenance training can be designed and implemented according to ISD concepts. Such services could include:

a. Analysis and specification of software maintenance training requirements

b. Planning for training system development

c. Design of training courses

d. Development of training resources

(1) Lesson materials - instructor and student

(2) Facilities, e.g.: system simulator; training station

(3) Training data base

(4) Audio-visual aids

e. Delivery and installation of training system

f. Implementation of training program

g. Analysis of training program effectiveness, program modifications/improvements.

A well-conceived and implemented training program will reduce transition costs and facilitate early organic capability.

Section 7.0 BIBLIOGRAPHY

1. Gilb, T., "The Measurement of Software Reliability and Maintainability", Computers and People, Page 16, September 1977

2. Stucki, L. G., "A Methodology For Producing Reliable Software", Volume II, March 1976

3. Stucki, L. G., "An Experiment in the Use of New Language Features and Automated Tools to Improve Software Quality", Boeing Computer Services, Inc., 1977

4. Fries, M. J., "Software Error Data Acquisition", The Boeing Company, February 1977

5. Brown, J. R., et al "Characteristics of Software Quality", TRW-National Bureau of Standards, December 28, 1973

6. Overton, R. K., et al "Research Toward Ways of Improving Software Maintenance", January 1973, ESD-TR-73-125

7. Overton, R. K., "Developments in Computer Aided Software Maintenance", September 1974, ESD-TR-74-307

8. Cirad, "A Study of Fundamental Factors Underlying Software Maintenance", December 1971, ESD-TR-72-121-Vol-1, 2

9. Peters, L. J., et al "Systematic Software Development and Maintenance", Boeing Computer Services, Inc., August 1976

10. ARINC 416-8 ARINC Specification 416-8, Abbreviated Test Language for Avionics Systems (ATLAS), Aeronautical Radio, Inc., June 1973

11. Omnicomp Handbook of Logic-Circuit Testing, Volume 1 "Techniques and Implementations" 1975

See also, References in Section 2.0.

Section 8.0 MATRIX: GUIDEBOOK TOPICS VERSUS GOVERNMENT DOCUMENTS

Figure 8.0-1 is a cross reference matrix showing the guidebook topics and government documents that address that topic. The government documents are identified as well as the sections, chapters, attachments, enclosures, appendicies, etc. in which the topics are found.

The elements in Figure 8.0-1 correspond to the sections in the government publication wherein the corresponding topic is discussed to the largest extent.

TOPICS \ GOVERNMENT PUBLICATION	DOD 5000.29	AFR 800-14 (VOL II)	MIL-STD-483	MIL-D-83468	AFR 800-4	MIL-STD-470	MIL-STD-1521A	AFR 800-8	AFR 800-21
LIFE CYCLE COSTS	VI.C.	3-7					10.3 20.3 30.2 40.1	2.d.	4c.(5)
ORGANIC MAINTENANCE		2-4(a) 3-7(a)			SUP. 1 2t.			ATTACH 1 PARA 3.	1(a) 3.
MAINTENANCE RESOURCES	V.C. V.D.	3-2 3-7 3-8		3.3 3.4.10		5.2 5.3 5.4	20.3 40.11	ATTACH 1 PARA 7, 9	3. 4.
MAINTENANCE TRADES		3-4 3-7				4.1 5.5	20.3 40.1	2.d. 3.	4c.(4) 4c.(5)
MAINTENANCE PLANNING	V.D.	CHAP 3				5.1 5.3	20.3 30.6 30.14 40.4	1.d. ATTACH 1 PARA 2	4c. 4d.
TESTING		CHAP 5		3.2 4.2 4.3	SUP. 1 SEC. B	5.11	30.6 40.5 70.2	ATTACH 1 PARA 3.	1i 4c.7.
CONFIGURATION MANAGEMENT	V.C.	CHAP 6	APP XIV		SUP. 1 SEC. B		10.3 60.3		

Figure 8.0-1. Guidebook Topics Versus Government Documentation (Sheet 1 of 2)

TOPICS \ GOVERNMENT PUBLICATION	DOD 5000.29	AFR 800-14 (VOL II)	MIL-STD-483	MIL-D-83468	AFR 800-4	MIL-STD-470	MIL-STD-1521A	AFR 800-8	AFR 800-21
MAINTENANCE DOCUMENTATION		CHAP 7	APP II APP VI APP VIII	3.4.8	SUP. 1	5.1 5.7	40.16 60.3	5.	
TURNOVER AND TRANSFER	V.D.	CHAP 9			1. 2. SUP. 1				1a. 3.
MAINTENANCE SUPPORT	V.E.	CHAP 10		3.2		5.3	20.3 30.2 40.1	2e.	2. 3. 4.
BASELINE MANAGEMENT		4-3	SEC 3		SUP. 1 2b(2)				
MAINTENANCE POLICY	V.A2. V.B.	CHAP 6 8-6 9-2 10-7, 8			2.	4.1		4.	1. 4.-10.
MAINTENANCE TRAINING	VI.C.	3-4c.(2) 3-6p. 3-8f. 3-9l. 4.7b.					20.3	ATTACH 1 PARA 8	3. 4.c. 8.
COMPUTER AIDS	V.F. 10-5			3.2 3.3.2		5.4	20.3		

Figure 8.0-1. Guidebook Topics Versus Government Documentation (Sheet 2 of 2)

Section 9.0 GLOSSARY OF TERMS

Acquisition Engineer - Military or civilian member of a SPO or an AFSC division who supports the activities of a SPO.

Allocated Baseline - The approved configuration item identification. It governs the development of selected configuration items that are part of a higher level specification, e.g., system specification. It is usually defined by the Computer Program Development Specification.

Baseline - An authorized documented technical description specifying an end item's functional and physical characteristics. It serves as the basis for configuration control and status accounting. It establishes an approved well-defined point of departure for control of future changes to system or equipment.

Certification - The test and evaluation of the complete computer program aimed at ensuring operational effectiveness and suitability with respect to mission requirements under operating conditions.

Computer-Generated Imagery - Visual scenes displayed in a TS in which the shape, position, and orientation of all displayed objects (lines, surfaces, volumes) are computed to correspond to the spatial relationship of observer to the simulated objects, e.g., pilot viewing a landing field. These computations are updated in real time and displayed on a CRT (or projection thereof) to simulate a dynamically changing visual scene.

Computer Program Configuration Items - A computer program or aggregate or related computer programs designated for configuration management. A CPCI may be a punched deck of cards, paper or magnetic tape or other media containing a sequence of instructions and data in a form suitable for insertion in a digital computer.

Configuration Control - A management discipline applying technical and administrative direction and surveillance to:

a. Identify and document the functional and physical characteristics of a configuration item

b. Control changes to those characteristics; and

c. Record and report change processing and implementation status

Control Software - Software used during execution of a test program which controls the nontesting operations of the ATE. This software is used to execute a test procedure but does not contain any of the stimuli or measurement parameters used in testing a UUT. Where test software and control software are combined in one inseparable program, that program will be treated as test software (AFLC 66-37).

Data Base - A collection of program code, tables, constants, interface elements and other data essential to the operation of a computer program or software subsystem.

High Order Language - Problem or system oriented code which can be automatically translated to machine language either directly or indirectly (through an assembly language step).

Host Computer - An off-line, general purpose, programmable computer which prepares data or code for a (target) system computer, e.g.: ATE station computer.

Interim Contractor Support - Contractor support of system operations, training, maintenance, capability improvements, etc., following system delivery. This support can be for all or part of a designated activity and is often contracted independently of a system development contract.

Life Cycle Costs - The sum of all costs to operate, maintain, and supply system functions for the full life of a system. This includes facilities, equipment, personnel, organic support services, system improvements, training, contractor support, etc.

Maintainable Software - Software (program code, documentation, data base) which has all the attributes that facilitate software maintenance. Such attributes include modularity, top down structure, conceptual groupings, etc.

Management Responsibility Transfer - The formal and complete transfer of all management responsibility for a system. This transfer occurs between the implementing command (procurement agency) and the supporting and/or using command.

Organic - A term used to designate a task performed by the Air Force rather than a contractor.

Product Baseline - The final approved configuration identification. It identifies the as designed and functionally tested computer program configuration. It is defined by the Computer Program Product Specification.

Quality Assurance - A planned and systematic pattern of all software-related actions necessary to provide adequate confidence that computer program configuration items or products conform to establish software technical requirements and that they achieve satisfactory performance.

Software - A combination of computer programs, documentation and computer data required to enable the computer equipment to perform computational or control functions and to enable program maintenance.

Software Maintenance - Any change to previously established software. Sources for such change are coding errors, module design problems, system interface problems, revised system requirements, and capability improvements.

Support Software - Auxiliary software used to aid in preparing, analyzing and maintaining other software. Support software is never used during the execution of a test program on a tester, although it may be resident either on-line or off-line. Included are assemblies, compilers, translators, loaders, design aids, test aids, etc. (AFLC 66-37).

System Life Cycle - The system acquisition life cycle consists of the following five major phases with major decision points:

a. Conceptual phase
b. Validation phase
c. Full-scale development phase
d. Production phase
e. Deployment phase

(AFR-800-14, Volume II)

Test Software - Programs which implement documented test requirements. There is a separate test program written for each distinct configuration of UUT (AFLC 66-37).

Top Down Structured Programs - Structured programs with the additional characteristics of the source code being logically, but not necessarily physically, segmented in a hierarchical manner and only dependent on code already written. Control of execution between segments is restricted to transfer between vertically adjacent hierarchical segments.

Validation - System validation is the integration and test of all hardware and software components to assure the complete system fulfills all system requirements. Note: "Validation" is also used in two other respects: (1) Validation Phase (between Conceptual Phase and Full Scale Development Phase), and (2) Software Validation - a component test checkout preceding software verification.

Verification - Computer program verification is the iterative process of continuously determining whether the product of each step of the computer program acquisition process fulfills all requirements levied by the previous step.

Section 10.0 ABBREVIATIONS AND ACRONYMS

ACSN	Advanced Change Study Notice
AFLC	Air Force Logistics Command
ASD	Aeronautical Systems Division
ATE	Automatic Test Equipment
ATLAS	Abbreviated Test Language for All Systems
ATPG	Automatic Test Program Generator
BITE	Built-In Test Equipment
CDR	Critical Design Review
CDRL	Contract Data Requirements List
CGI	Computer Generated Imagery
CI	Configuration Item
CMP	Configuration Management Plan
COMPOOL	Common Data Pool (Base)
CPC	Computer Program Component
CPCI	Computer Program Configuration Item
CPDP	Computer Program Development Plan
CPL	Computer Program Library
CPU	Central Processing Unit
CRISP	Computer Resources Integrated Support Plan
CRT	Cathode Ray Tube
CRWG	Computer Resources Working Group
DCP	Development Concept Paper
DID	Data Item Description
DOD	Department of Defense
DOS	Disc Operating System
DR	Discrepancy Report
DT&E	Development, Test, and Engineering
ECP	Engineering Change Proposal
EMI/EMP	Electro-magnetic Interference/ Pulse
FCA	Functional Configuration Audit
FOC	Full Operational Capability
FORTRAN	Formula Translator
FQR	Formal Qualification Review
FSCP	Flight Simulator Computer Programs
GFE	Government Furnished Equipment
HOL	Higher Order Language
IC	Integrated Circuit
ICD	Interface Control Drawing
ICS	Interface Control Specification
ICS	Interim Contractor Support
ILS	Integrated Logistics Support
ILSP	Integrated Logistics Support Plan
IOC	Initial Operational Capability
ISD	Instructural Systems Development
ISP	Integrated Support Plan
ITA	Interface Test Adapter
LCC	Life Cycle Cost

LRU	Line Replaceable Unit
LSA	Logistics Support Analysis
PCA	Physical Configuration Audit
PDR	Preliminary Design Review
PMP	Program Management Plan
PMRT	Program Management Responsibility Transfer
PRU	Primary Replaceable Unit
QA	Quality Assurance
SMTP	Simulator Maintenance and Test Programs
RAM	Random-Access Memory
RF	Radio Frequency
RFP	Request for Proposal
ROC	Required Operational Capability
ROM	Read-Only Memory
SAE	Software Acquisition Engineering
SCN	Specification Change Notice
SCP	Supplier Change Proposal
SDR	System Design Review
SERD	Support Equipment Recommendation Data
SOW	Statement of Work
SPO	System Program Office
SRR	System Requirements Review
SRU	Secondary Replaceable Unit
SSP	Simulator Data Support Program
TAC	Tactical Air Command
T.O.	Technical Order
TRD	Test Requirements Document
TS	Training Simulator
TWG	Transfer Working Group
USAF	United States Air Force
UUT	Unit Under Test
VDD	Version Description Document
VV&C	Verification, Validation, and Certification
WBS	Work Breakdown Structure

Section 11.0 SUBJECT INDEX

	PARAGRAPH
ATE Unique Consideration	3.0, 3.4, 4.3.1, 5.2.3, 5.2.5
Automatic Test Program Generation	4.1.2
Calibration	3.2, 3.4, 3.5, 4.1.5, 4.3.1
Central Maintenance Facility	5.3
Change Approval	3.3, 5.1, 5.1.2, 5.3.7
Change Board	3.3, 5.1.2, 5.3.7
Change Impact Assessment	3.2, 3.3
Change Management, Control	1.4, 3.0, 3.3, 3.5, 4.1.4, 5.3
Change Sources	3.1, 3.3, 5.1.1.4, 5.1.1.5, 5.2.4
Check List	3.2, 4.1.1, 4.1.5, 6.1, 6.3
Computer Aids	3.1, 3.3, 3.5, 4.1.3, 4.1.5, 6.1, 6.1.1, 6.1.2
Conceptual Phase	1.0, 3.2, 4.0, 4.1.1
Configuration Management	3.2, 4.1.1, 4.1.4, 4.1.5, 4.3.1, 5.1.1.1, 5.2.3, 5.3
Data Base	3.0, 3.2, 3.3, 6.1, 6.2.3
Corrective Maintenance	3.2, 3.5
Data Item Descriptions	2.0, 4.1.1, 4.1.3, 4.2, 5.1.1.1, 5.1.4
Debugging	3.2, 3.3, 3.4, 4.1.5, 5.1, 5.2.6
Deployment Phase	3.2
Diagnosis	3.2, 3.4, 3.5, 6.1.1
Documents and Documentation	1.0, 2.0, 3.2, 3.3, 3.4, 4.1.1, 4.1.3, 4.1.5, 4.2, 4.3.1, 5.2.2, 5.3.4, 5.3.9
Facilities	1.2, 1.4, 3.1, 3.2, 4.1.1, 4.1.3, 4.3.1, 6.1, 6.2.3
Firmware	3.1, 3.2, 3.5
Full Scale Development Phase	3.2, 4.1.4, 5.1.3.1
Growth Potential	3.4, 4.1.1, 4.3.1
High Order Language	1.3, 3.2, 4.1.5, 4.2

Life Cycle Costs	1.0, 3.2, 3.5, 4.1, 4.1.2, 4.1.5, 6.2.3
Logistics Support Analysis	1.3.2, 3.2
Maintainability Attributes	3.1, 4.1.5, 5.1.1.3, 5.2.3, 6.1
Maintainable Software	1.0, 3.0, 3.1, 3.2, 3.5, 4.1.1, 4.1.2, 4.1.5
Maintenance Activities	1.2, 1.4, 3.2, 3.3, 5.0, 5.1, 5.2.5
Maintenance Data Collection	3.2
Maintenance Demonstration	3.0, 4.1.5
Maintenance Equipment	3.2, 4.1.3, 6.1
Maintenance Planning	1.0, 1.2, 1.4, 3.2, 3.4, 3.5, 4.0, 4.1, 4.3.2, 6.2.1, 6.2.2
Maintenance Programmers	3.1, 6.1
Maintenance Resources	3.1, 3.2, 3.3, 4.1.1, 4.1.3, 6.1
Maintenance Tools	1.2, 3.2, 6.1, 6.2.3
Maintenance Training	1.4, 3.2, 4.1.1, 4.1.3, 4.3.1, 6.1, 6.2.3, 6.3
Modification Design	3.3
Modules, Modularity	3.2, 3.3, 4.1.1, 4.1.5, 4.3.2, 5.1.3
Organic Software Maintenance	3.1, 3.4, 4.2, 6.2.2, 6.3
Organizations	4.1.4, 4.3.1, 6.1
Pitfalls	3.4, 5.1.4, 5.2.8
Preventive Maintenance	3.2, 3.5
Programmer Skill	3.0, 3.2, 4.1.5, 6.1
Program Structure	3.2, 4.1.5
Program Support Library	3.3, 6.1.2
Program Transfer/System Turnover	6.0, 6.2
Proprietary Software	3.4, 4.2, 5.2.7
Reviews/Audits	3.2, 4.1.5, 5.2.2
Software Maintenance Costs	1.0, 1.2, 3.2, 6.1
Software Maintenance Life Cycle	3.0, 3.2
Software Maintenance Policy	3.2, 3.3, 6.1, 6.2
Software Maintenance Procedures	3.0, 3.3, 4.1.1, 6.1, 6.2.2
Software Maintenance Tasks	3.0, 3.2, 5.1, 5.1.3
Special Requirements	1.4, 3.0, 6.0

Testing	3.2, 3.3, 3.4, 4.1.1, 4.1.5, 5.1.3
Trades, Trade-offs	3.0, 3.2, 3.4, 4.1, 4.1.2, 6.2.3
TS Unique Considerations	3.0, 3.5, 4.3.2, 5.1, 5.3.10
Validation Phase	3.2
V, V, & C	3.3, 4.1.5

*U.S.Government Printing Office: 1980 — 657-084/530