

AFHRL-TP-89-87

AIR FORCE**AD-A221 809****HUMAN RESOURCES****ADVANCED ON-THE-JOB TRAINING SYSTEM:
SOFTWARE TEST PLAN**

Douglas Aircraft Company
A Division of McDonnell Douglas Corporation
2450 South Peoria
Aurora, Colorado 80014

TRAINING SYSTEMS DIVISION
Brooks Air Force Base, Texas 78235-5601

May 1990

Interim Technical Paper for Period August 1985 - December 1989

Approved for public release; distribution is unlimited.

LABORATORY

**AIR FORCE SYSTEMS COMMAND
BROOKS AIR FORCE BASE, TEXAS 78235-5601**

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely Government-related procurement, the United States Government incurs no responsibility or any obligation whatsoever. The fact that the Government may have formulated or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication, or otherwise in any manner construed, as licensing the holder, or any other person or corporation; or as conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

The Public Affairs Office has reviewed this paper, and it is releasable to the National Technical Information Service, where it will be available to the general public, including foreign nationals.

This paper has been reviewed and is approved for publication.

HENDRICK W. RUCK, Technical Advisor
Training Systems Division

RODGER D. BALLENTINE, Colonel, USAF
Chief, Training Systems Division

This technical paper is printed as received and has not been edited by the AFHRL Technical Editing staff. The opinions expressed herein represent those of the author and not necessarily those of the United States Air Force.

REPORT DOCUMENTATION PAGEForm Approved
OMB No. 0704-0188

Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.

1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE May 1990	3. REPORT TYPE AND DATES COVERED Interim - August 1985 to December 1989	
4. TITLE AND SUBTITLE Advanced On-the-job Training System: Software Test Plan			5. FUNDING NUMBERS C - F33615-84-C-0059 PE - 63227F PR - 2557 TA - 00 WU - 02	
6. AUTHOR(S)				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Douglas Aircraft Company A Division of McDonnell Douglas Corporation 2450 South Peoria Aurora, Colorado 80014			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) Training Systems Division Air Force Human Resources Laboratory Brooks Air Force Base, Texas 78235-5601			10. SPONSORING/MONITORING AGENCY REPORT NUMBER AFHRL-TP-89-87	
11. SUPPLEMENTARY NOTES				
12a. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution is unlimited.			12b. DISTRIBUTION CODE	
13. ABSTRACT (Maximum 200 words) The Software Test Plan for the Advanced On-the-job Training System (AOTS) establishes the plan for testing the Computer Program Configuration Items (CPCI) in the AOTS Computer Support Subsystem, Software Component. The Plan describes AOTS Development Phase (Phase II) software testing which was divided into two phases: formal and informal testing. The levels of testing described in this document equate with the test types described in the quality assurance provisions paragraphs in the AOTS Development Specifications for the Management CPCI, Evaluation CPCI, and Support System CPCI.				
14. SUBJECT TERMS advanced on-the-job training system computer program configuration items software test plan			15. NUMBER OF PAGES 53	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UL	

May 1990

ADVANCED ON-THE-JOB TRAINING SYSTEM:
SOFTWARE TEST PLAN

Douglas Aircraft Company
A Division of McDonnell Douglas Corporation
2450 South Peoria
Aurora, Colorado 80014

TRAINING SYSTEMS DIVISION
Brooks Air Force Base, Texas 78235-5601



Accession For	
NTIS GRA&I	☒
DTIC TAB	☒
Unannounced	☐
Justification	
By _____	
Distribution/	
Availability Codes	
Dist. as 1/or	
Dist	Special
A-1	

Reviewed and submitted for publication by

Jack L. Blackhurst, Major, USAF
Chief, Advanced On-the-Job Training System Program

This publication is primarily a working paper. It is published solely to document work performed.

SUMMARY

The Advanced On-the-job Training System (AOTS) was an Air Staff directed, AFHRL developed, prototype system which designed, developed, and tested a proof-of-concept prototype AOTS within the operational environment of selected work centers at Bergstrom AFB, Texas, and Ellington ANGB, Texas, from August 1985 through 31 July 1989. The AOTS Software Test Plan was prepared for the AOTS Software Development Team and establishes the plan for testing the Computer Program Configuration Items (CPCI) in the AOTS Computer Support Subsystem, Software Component. The plan describes AOTS Development Phase (Phase II) software testing which was divided into two phases: formal and informal testing. The levels of testing described in this document equate with the test types described in the quality assurance provisions paragraphs in the AOTS Development Specifications for the Management CPCI, Evaluation CPCI, and System Support CPCI. Computer Program Test and Evaluation as described in the Development Specifications is equivalent to unit level and integration level testing as described in the AOTS Software Test Plan.

PREFACE

This paper was developed by Douglas Aircraft Company, the AOTS development contractor, under Government Contract Number F33615-C-84-0059. The AFHRL Work Unit number for the contract was 2557-00-02. The primary office of responsibility for management of the project is the Air Force Human Resources Laboratory, Training Systems Division, and the Air Force AOTS project manager is Major Jack Blackhurst.

This document was prepared for the AOTS Software Development Team. It is not a CDRL item, however, it is referred to by the CPCI Development Specifications, Software Development Plan, and Master Test Plan. It is intended to be used both as a standalone document in low level software development testing, and in conjunction with the Master Test Plan in higher level requirement verification testing. The procedures described in this document will be updated as necessary.

This document is based on the provisions set forth in Data Item Description (DID) DI-MCCR-80014. This DID is applicable to DOD-STD-2167, which is not an AOTS applicable document. Some deviation is taken from this DID in Sections 3 and 4. In Section 3, the terminology Computer Program Component (CPC) was substituted for Computer Software Component (CSC). CPC is consistent with the AOTS CPCI Development Specifications written according to the 21 March 1979 version of MIL-STD-483. CSC is terminology adopted at a later point in time and is used with DOD-STD-2167 documentation. In Section 4, this document divides formal testing into system and acceptance testing. The DID does not make this distinction.

DIDs DI-MCCR-80015, Software Test Description, and DI-MCCR-80016, Software Test Procedure, were not followed structurally in the Formal Test Procedure model (see Attachment C). The Formal Test Procedure contains the same information as these two DIDs describe, but in a format that makes formal test procedures consistent with informal test procedures. The same is true for DID DI-MCCR-80017, Software Test Report, and the Formal Test Report model presented in Attachment D.

The identification of AOTS formal test procedures in Section 4 is complete to the point of the current state of the phased software development effort. As detailed design proceeds on the next set of software, these tables will be updated accordingly.

ACKNOWLEDGEMENTS

Some of the material contained in this document has been adapted for use from the following manuals and books:

- MDAC-STL Software Engineering Practices Manual
- The Software Development Project, Planning & Management
Phillip Bruce and Sam M. Pederson
1982, John Wiley and Sons, Inc.
- Software System Testing and Quality Assurance
Boris Beizer
1984, Van Nostrand Reinhold Company
- Productive Software Test Management
Michael W. Evans
1984, John Wiley and Sons, Inc.

Table of Contents

Section		Page
1.	SCOPE	1
1.1	Identification	1
1.2	Purpose	1
1.3	Introduction	1
2.	REFERENCED DOCUMENTS	4
3.	PLANS FOR INFORMAL TESTING	5
3.1	Unit Test Planning	5
3.1.1	Unit Test Requirements	5
3.1.2	Unit Test Responsibilities	5
3.1.3	Unit Test Schedule	6
3.1.4	Unit Test Procedure Model	6
3.1.5	Unit Test Report Model	7
3.2	CPC Integration and Test Planning	8
3.2.1	CPC Integration and Test Requirements	9
3.2.2	CPC Integration and Test Responsibilities	9
3.2.3	CPC Integration Test Classes	10
3.2.4	CPC Integration and Test Schedules	10
3.2.5	Integration and Test Procedure Model	10
3.2.6	Integration and Test Report Model	12
3.3	Resources Required for Informal Testing	13
3.3.1	Facilities	13
3.3.2	Personnel	13
3.3.3	Hardware	13
3.3.4	Interfacing/Support Software	14
3.3.5	Source	14
3.3.6	Test Configuration	14
4.	PLANS FOR FORMAL TESTING	16
4.1	System Test Planning	16
4.1.1	System Test Requirements	16
4.1.2	System Test Responsibilities	16
4.1.3	System Test Schedule	17
4.1.4	System Test Procedure Model	17
4.1.5	System Test Report Model	18
4.2	Acceptance Test Planning	19
4.2.1	Acceptance Test Requirements	19
4.2.2	Acceptance Test Responsibilities	19
4.2.3	Acceptance Test Schedule	20
4.2.4	Acceptance Test Procedure Model	20
4.2.5	Acceptance Test Report Model	20

Table of Contents

Section		Page
4.3	Formal Test Classes	21
4.4	Formal Tests	21
4.4.1	Management CPCI Test Procedures	21
4.4.2	Evaluation CPCI Test Procedures	23
4.4.3	System Support CPCI Test Procedures	24
4.5	Formal Test Levels	26
4.6	Formal Test Summary	26
4.7	Formal Test Schedule(s)	26
4.8	Data Recording, Reduction, Analysis	26
4.9	Formal Test Reports	26
4.10	Resources Required for Formal Testing	26
4.10.1	Facilities	26
4.10.2	Personnel	26
4.10.3	Hardware	27
4.10.4	Interfacing/Support Software	27
4.10.5	Source	27
5.	TEST PLANNING ASSUMPTIONS AND CONSTRAINTS	28
6.	NOTES	28

FIGURES

Figure		
1.	AOTS Software Test Plan	2
2.	Hardware Configuration for Informal Testing ..	15

TABLES

Table		
1.	Management CPCI Test Procedures	22
2.	Evaluation CPCI Test Procedures	23
3.	System Support CPCI Test Procedures	25

ATTACHMENTS

Attachment		
A.	Unit and Integration Test Procedure Model	A-1
B.	Unit and Integration Test Report Model	B-1
C.	System and Acceptance Test Procedure Model	C-1
D.	System and Acceptance Test Report Model	D-1
E.	Acceptance Test Log	E-1
F.	System Test Procedure Example	F-1
G.	Acceptance Test Procedure Example	G-1

1. SCOPE

1.1 Identification. This Software Test Plan establishes the plan for testing the Computer Program Configuration Items (CPCIs) in the Advanced On-the-job Training System (AOTS) Computer Support Subsystem, Software Component. The CPCIs are identified as the Management CPCI as described in the development specification numbered 70S647411; the Evaluation CPCI, 70S647413; and the System Support CPCI, 70S647414.

1.2 Purpose. The purpose of the AOTS is to test a design concept that will apply automated support to increase the efficiency and effectiveness of the current Air Force On-the-Job Training (OJT) system. The Management CPCI will provide software support for the AOTS Management Subsystem, 70S647100. It will provide software to support the functions of identifying and managing training requirements for an Air Force Speciality (AFS); managing the airmen undergoing training in the AFS; and managing the resources required to train those airmen.

The Evaluation CPCI will provide software support for the AOTS Evaluation Subsystem, 70S647300. It will provide software to support the functions of developing and maintaining evaluation instrumentation, evaluating performance of tasks by airmen, performing training quality control functions, and evaluating the effectiveness of the AOTS as a system.

The System Support CPCI will be composed of the services required by the other CPCIs to interface with the Hardware Component, 70S647401. This CPCI will perform operating system functions, terminal communication and data base input/output, and will provide security functions to control access to the system.

1.3 Introduction. This plan describes AOTS Phase II software testing. Software testing is divided into two phases: informal and formal testing. Both of the testing phases are divided into two levels. Unit and integration level testing are done in the informal phase. System and acceptance level testing are done in the formal phase. Together, these four levels satisfy the Phase II testing requirements for the Software Component. Figure 1 depicts software testing phases and their levels. The following paragraphs detail each test phase accompanied by the two levels of testing within the phase.

It is appropriate to equate the levels of testing described in this document with the test types described in the quality assurance provisions paragraphs in the Development Specifications for the Management CPCI, Evaluation CPCI, and System Support CPCI. Computer Program Test and Evaluation as described in the Development Specifications is equivalent to unit level and integration level testing as described in this document.

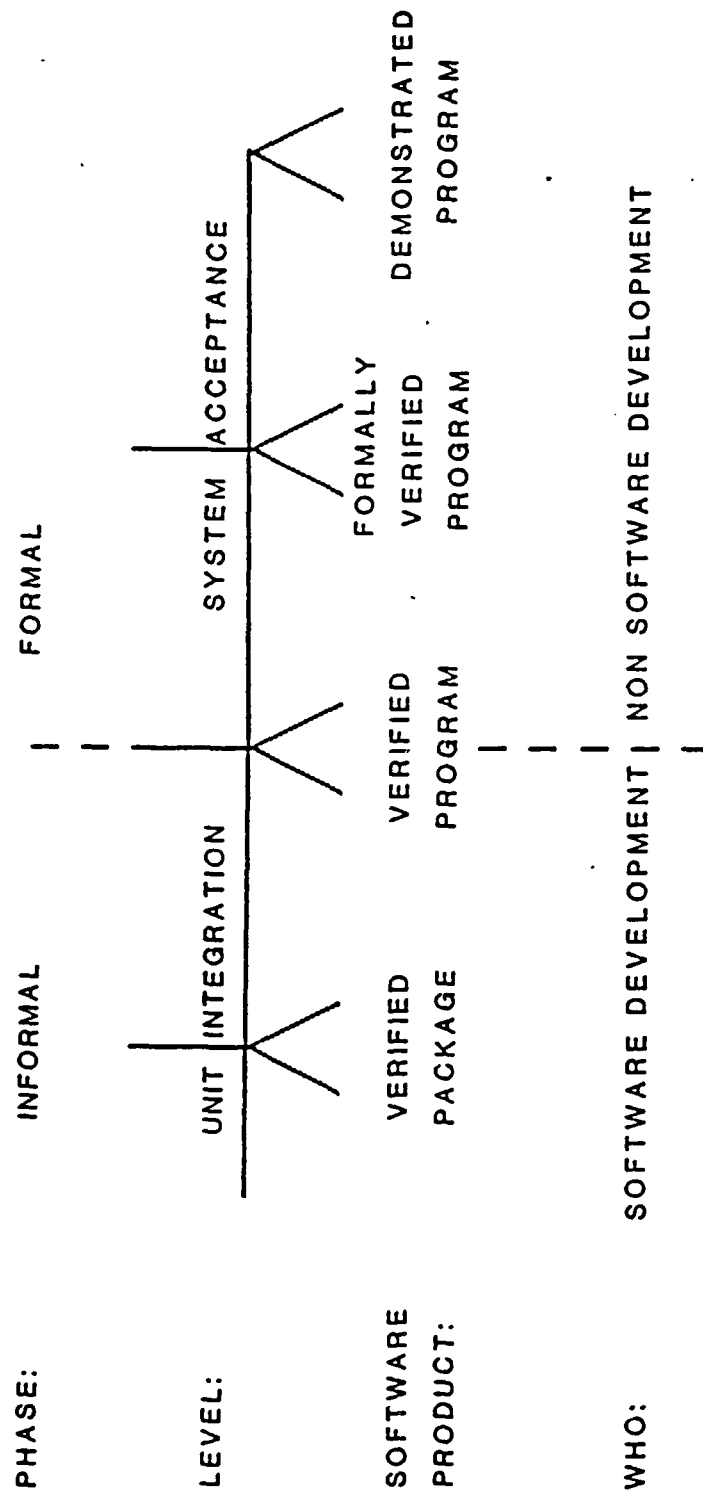


Figure 1. AOTS SOFTWARE TEST PLAN

Preliminary Qualification Tests as described in the Development Specifications is equivalent to system level testing as described in this document. Formal Qualification Tests as described in the Development Specifications is equivalent to acceptance level testing as described in this document.

2. REFERENCED DOCUMENTS. The following government documents are referenced in this plan. These documents were written by MDAC as part of system design and development under the AOTS contract.

70S647100	Prime Item Development Specification for the Management Subsystem of the AOTS
70S647300	Prime Item Development Specification for the Evaluation Subsystem of the AOTS
70S647401	Critical Item Development Specification for the Hardware Component of the AOTS
70S647411	AOTS Management CPCI Development Specification
70S647413	AOTS Evaluation CPCI Development Specification
70S647414	AOTS System Support CPCI Development Specification
	AOTS Software Development Plan
	AOTS Configuration Management Plan
	AOTS Master Test Plan

3. PLANS FOR INFORMAL TESTING. The informal phase of testing consists of unit level testing and integration level testing. It is informal because the testing takes place as the code is being developed and is planned and performed by the people doing that development. The expected result of informal testing is a verified program.

3.1 Unit Test Planning. The intent of unit testing is to verify that a program unit performs as specified in the respective CPCI Product Specification. For the purposes of AOTS software development, unit is defined as an Ada package.

3.1.1 Unit Test Requirements. Unit testing consists of two phases. The first phase consists of compiling the unit and correcting compilation errors. The second phase consists of testing the unit for adherence to the following general requirements:

- a. Conformation to specifications and requirements will be verified;
 - Check that the unit is necessary
 - Check function, logic, and computations
 - Check adherence to standards - justify exceptions
- b. Execution paths will be verified;
 - Execute every instruction at least once
 - Every decision should be executed at least once in each possible direction
- c. Data handling capability will be verified;
 - Check type and format of input data
 - Check input data at nominal, extreme, erroneous, and exceptional values
 - Check value, type, and format of output data
- d. Design extremes will be verified.
 - Check error detection and recovery

This second phase is accomplished by desk checking the unit and actual execution of the unit. The testing continues until all known errors have been eliminated and coded unit logic matches the design. Interfaces may be checked by using stubs to simulate other units.

3.1.2 Unit Test Responsibilities. The unit test is planned, performed, and controlled by the programmer responsible for the unit. Planning includes what the programmer intends to accomplish for a test, the inputs that are required, the outputs that are expected, and how the test is to be conducted. Paragraph 3.1.4 provides a model for putting this data together in to a unit test procedure. The programmer prepares a procedure, according to this model, that contains the tests to be conducted for each unit. This procedure should be reviewed at the unit code walkthru. The procedure is maintained in the Unit

Development Folder (UDF). Much of the testing at this level is very detailed and will not be conducted again in higher level tests. Thus, the record of informal testing and the corresponding test results will be kept in the UDF and preserved for reference purposes. A model test report is provided in paragraph 3.1.5 to assist in this.

Unit testing can uncover coding deficiencies. The programmer responsible for the unit is expected to correct the deficiencies and perform the unit test again.

Unit testing can also uncover design deficiencies. The programmer responsible for the unit should report the possibility of any such conditions to the CPCI team leader and, in turn, to the software manager. Together, they will determine the effect on the unit and released documentation. Correction of the deficiencies of this nature must be handled in accordance with the Configuration Management Plan.

During unit testing, the primary goal is to test the functions of the unit. Thus, stubs are used to simulate other units which the unit in test is targeted to interface with, but which are still under development. Data that the unit in test provides to another unit must be checked for accuracy. However, data provided by another unit to the unit in test could be simulated.

3.1.3 Unit Test Schedule. Initial unit testing takes place during the software development period for a unit. Thus the unit test schedule follows the software development schedule for a program. The general program development schedule is specified in the AOTS Software Development Plan. The detailed unit test schedules are kept with the test procedures in the UDFs.

3.1.4 Unit Test Procedure Model. Attachment A provides a model test procedure. This is to be used as a guide for the individual creating specific AOTS software unit test procedures. The following paragraphs describe the elements of the procedure and include instructions to be used by the author in making up the procedure.

Section I identifies the unit and the author of a test procedure. Both of these items, as well as the preparation date of the procedure, should be filled in. The test type should be identified as "unit". As the document is revised, the revision information, including author and date, should be filled in.

Section II is scheduling information for the performance of the unit test during development. Enter the date that the test is projected to start and the date it is expected to complete, as well as an estimate of the length of time to perform the procedure. These entries can be used for planning and scheduling purposes for this unit. The development hours entry is a record of how long this test procedure took to develop and is intended

to be used for future planning and schedule purposes.

Section III specifies the plan for resources necessary for conducting the unit test. Identify the source of the unit to be tested. Identify data base files, if any, that are necessary to test this unit. Indicate whether the files are new to this unit or if existing files and data can be used. Identify other units, if any, that are necessary to test this unit. Besides serving as a planning tool this provides a low level critical path, i.e., the actual versions of unit x and unit y have to be completed for you to test unit z. Identify hardware requirements for the test. If the normal AOTS hardware environment is to be used, then indicate this. If the unit requires hardware that has not been used in the AOTS test environment before, e.g. mark sense reader or X/Y input device, then list this equipment.

Section IV identifies debug information, should that be appropriate. Does the unit have special debugging code? If so, how is that feature activated?

Section V describes the specific functional testing to be performed on the unit. Identify the elements of the unit that are tested by this procedure. Identify the functions of the unit that are tested by this procedure. Identify execution paths in the unit that are not tested by this procedure. This may include fatal error conditions, nonexistent production data base files, permanent i/o errors, etc. Elaborate on the actual sequence for the test procedure: what inputs are necessary, what outputs are expected from that input, and what function does this input/output sequence check out (this is indicated with an ordinal cross reference to the above function list). If specific input from a file is necessary for the test, indicate what this data is. Similarly, record what the condition of particular files are as a result of this test. List what error handling is to be verified in this unit. If the unit uses data produced by another unit or produces data to be used by another unit, then this information should be recorded under "data compatibility". The heading "other testing" is to be used to describe testing that you feel should be performed but does not fall into one of the other classes. When writing the test procedure, be familiar with the contents of the Test Report and the Testing Check List, described in the following paragraph, so that minimum and maximum data ranges, as well as typical values and similar constraints can be tested. Remember to author the test plan in such a manner that other people, besides yourself, can perform the test. Be clear, complete, yet concise.

3.1.5 Unit Test Report Model. Attachment B is a model test report. It is to be used as a guide to report on the execution of unit tests and should be filled out by the individual performing the test during the process of, or immediately after, a unit test. The Test Report should be used as an aid in all

unit testing, with the completed test reports placed in the UDF. When unit testing is performed at several levels, i.e. testing each subprogram, or group of subprograms, followed by testing at the package level, the test reporting should follow that same plan. The following paragraphs describe the elements of the report.

Section I identifies the unit tested, the person that performed the test, and the date of the test. The same report form is used to document results for a unit level test and for an integration level test. Therefore mark the test level as appropriate. Fill in the environment that the unit was executed in. The time duration of the test can be used for future planning purposes in unit testing and should be entered as hours or minutes.

Section II is the reason the test was performed. Is the test being performed on a newly developed unit, is a unit being enhanced, or is the unit being tested because other units that it references were changed? Unit test reports are necessary not only for new units but also for maturing units.

Section III is an indication of the test results. Was the outcome expected or unexpected? If the outcome was not as expected, then a description of the problem or the action to be performed should be furnished, either in this section or in the following checklist. This area can also be used for notes or reminders if the test was successful.

Section IV is a worksheet to be used in conjunction with the test sequence section of the test procedure. It is to be used to note problems or to act as a placeholder while executing the test sequence.

Section V is a checklist for unit testing. It is to be used as a guide, as well as a reminder of what to test and what to look for while testing. The headings in the checklist represent the major software performance and requirement classifications to be checked in unit testing. Beneath each heading is a list of items to exercise or check. As items are tested, they are marked off or notes made for problem reminders.

3.2 CPC Integration and Test Planning. Computer Program Component (CPC) Integration consists of building a program by iteratively adding units, and, as the units are integrated, testing to ensure the resulting software matches that described in the Development and Product Specifications.

At each iteration, integration subelements are combined to form integration elements. This process begins with two or more functionally and logically related units which have passed unit testing. These units are integrated and the resultant integration element tested. Once fully tested, such elements become the subelements which are integrated to form larger elements. Thus, an integration subelement may be either a fully

tested unit or a fully integrated set of units and an integration element is a set of functionally and logically related subelements.

Ultimately, aggregates of integration elements form CPCs. For the purposes of AOTS software development, the terminology CPC and program are used interchangeably.

3.2.1 CPC Integration and Test Requirements. Many of the requirements for integration testing are the same as for unit testing. Other requirements are imposed to ensure that subelements are compatible and consistent.

Like unit testing, integration testing consists of two phases. The first phase consists of compiling, linking, and loading the integration element and correcting any resulting errors. The second phase consists of testing the unit for adherence to the following general requirements:

- a. Confirmation to specifications and requirements will be verified;
 - Check that the elements are necessary
 - Check function, logic, and computations
 - Check adherence to standards - justify exceptions
- b. Execution paths will be verified;
 - Execute every major path through the subelement
 - Every subelement should be called at least once by each possible calling subelement
- c. Data handling capability will be verified;
 - Check type and format of input data
 - Check input data at nominal, extreme, erroneous, and exceptional values
 - Check value, type, and format of output data
- d. Design extremes will be verified.
 - Check error detection and recovery
 - Check data compatibility between subelements

Finally, the testing must verify that the integration is forming the CPCs as they are described in the specifications. If the CPCs interface with other CPCs, then these interfaces must also be validated.

The second phase is accomplished by desk checking the element and actual execution of the element. The testing continues until all known errors have been eliminated and integrated element logic matches the design.

3.2.2 CPC Integration and Test Responsibilities. As in unit testing, the CPC integration test is planned, performed, and controlled by the programmer(s) responsible for the CPC. Planning includes what the programmer intends to accomplish for a test, the inputs that are required, the outputs that are expected, and how the test is to be conducted. Paragraph 3.2.5 provides a model for a CPC integration test procedure. The

programmer prepares a procedure, according to this model, that contains the tests to be conducted for the CPC. This procedure should be reviewed at the CPC walkthru. This test procedure is maintained in the UDF for the unit that is the linkable entity that results in a program. A record of test performance along with the corresponding test results should be preserved for reference purposes in the UDF. A model test report is provided in paragraph 3.2.6 to assist in this.

CPC integration testing can uncover coding deficiencies. The programmer(s) responsible for the CPC is expected to correct the deficiencies and perform the CPC integration test again. If the testing reveals problems with a unit outside of the CPC that the CPC is integrating with, then the programmer should report this to the CPCI Team Leader, and if the unit is outside of the particular CPCI, to the software manager. Resolution for the integration deficiency should be made as quickly as schedules allow so as to not hamper further testing.

CPC integration testing can also uncover design deficiencies. The programmer responsible for the CPC should report the possibility of any such conditions to the CPCI team leader and, in turn, to the software manager. Together, they will determine the effect on the CPC and released documentation. Correction of the deficiencies of this nature must be handled in accordance with the Configuration Management Plan.

Any stubs that were used in unit testing are replaced, in stages, by actual units until CPC integration testing is complete. The product of a completed CPC integration test is a complete CPC.

3.2.3 CPC Integration Test Classes. The test classes for CPC integration testing, and as appropriate for unit testing, include functional requirement testing, interface testing, terminal and device input/output testing, file input/output testing, and error handling testing. The model test procedure described in paragraph 3.2.5 and the checklist described in the model test report in paragraph 3.2.6 reflect these test classes.

3.2.4 CPC Integration and Test Schedules. Integration testing for a program takes place during the software development cycle for that program. Thus the integration test schedule follows the program development schedule. The general program development schedule is specified in the AOTS Software Development Plan. The detailed integration test schedules are kept with the test procedures in the UDFs.

3.2.5 Integration and Test Procedure Model. Attachment A is a model test procedure. The same model is used for the unit test procedure and the integration and test procedure, as much of required information is the same. As in the unit test procedure

description, this model is to be used as a guide for the individual preparing specific integration test procedures. The following paragraphs describe the elements of the procedure and include instructions to be used by the author in formulating the procedure.

Section I identifies the CPC to be integrated and the author of the test procedure. Each of these items, as well as the preparation date of the procedure, should be completed. The test type should be identified as "integration". As the document is revised, the revision information, including author and date, should be filled in.

Section II is scheduling information for the performance of the integration and test during development. Enter the date that the test is projected to start and the date it is expected to complete, as well as an estimate of the length of time to perform the procedure. These entries can be used for planning and scheduling purposes for this unit. The development hours entry is a record of how long this test procedure took to develop and is intended to be used for future planning and schedule purposes.

Section III specifies the plan for resources necessary for conducting the integration and test. Identify the source of the unit containing the subelements being integrated. Identify data base files, if any, that are necessary to test this unit. Identify people whose expertise or time is required. Identify other software, such as stubs, drivers, or test data generators, necessary to test this element. Identify unique hardware, i.e. hardware of limited availability or hardware not normally a part of the AOTS hardware environment, that is necessary to test this element. Unique hardware might include a logic analyzer. Limited availability hardware might include a mark sense reader or X/Y input device.

Section IV identifies debug information, should that be appropriate. Does the element have special debugging code? If so, how is that feature activated?

Section V describes the specific functional testing to be performed by the integration test. Identify the elements that are being integrated and tested by this procedure. Identify the functions of the unit that are tested by this procedure. Identify execution paths in the unit that are not tested by this procedure. This may include fatal error conditions, nonexistent production data base files, permanent i/o errors, etc. Elaborate on the actual sequence for the test procedure: what inputs are necessary, what outputs are expected from that input, and what function does this input/output sequence check out (this is indicated with an ordinal cross reference to the above function list). If specific input from a file is necessary for the test, indicate what this data is. Similarly, record what the condition of particular files are as a result of this test. List what error handling is to be verified in this unit. If the test uses

data produced by another test procedure or produces data to be used by another test procedure, then this information should be recorded under "data compatibility". The heading "other testing" is to be used to describe testing that you feel should be performed but does not fall into one of the other classes. When writing the test procedure, be familiar with the contents of the Test Report and its Testing Check List, described in the following paragraph, so that minimum and maximum data ranges, as well as typical values and similar constraints can be tested.

Section V of this test plan is much like Section V of the Unit Test Plan. However, the intent of the Specific Testing section is different between the two plans. In the Unit Test Plan the scope of the Specific Testing section deals with all instructions within the unit. In the Integration and Test Plan, the Specific Testing section is concerned with major paths through elements and all of the calling sequences of subelements being integrated.

3.2.6 Integration and Test Report Model. Attachment B is a model test report. As with the test procedure model, the same model for reports is used for unit testing and integration testing, as much of required information is the same. As described in the unit test report description, this model is to be used as a guide to report on the execution of integration tests and should be filled out by the individual performing the test during the process of, or immediately after, an integration test. The Test Report should be used as an aid in all integration and testing, with the completed test reports placed in the UDF. The following paragraphs describe the elements of the report.

Section I identifies the integration test being performed, the person that performed the test, and the date of the test. The same report form is used to document results for a unit level test and for an integration level test. Therefore mark the test level as appropriate. Fill in the environment that the test was executed in. The time duration of the test can be used for future planning purposes in integration testing and should be entered as hours or minutes.

Section II is the reason the test was performed. Is the test being performed on a newly developed element, is an element or subelement being enhanced, or is the element being tested because other elements that it references were changed? Integration and test reports are necessary not only for newly integrated elements but also for maturing elements.

Section III is an indication of the test results. Was the test outcome expected or unexpected? If the outcome was not as expected, then a description of the problem or the action to be performed should be furnished, either in this section or in the following checklist. This area can also be used for notes or

reminders if the test results were satisfactory.

Section IV is a worksheet to be used in conjunction with the test sequence section of the test procedure. It is to be used to note problems or to act as a placeholder while executing the test sequence.

Section V is a checklist for integration and testing. It is to be used as a guide, as well as a reminder of what to test and what to look for while testing. The headings in the checklist represent the major software performance and requirement classifications to be checked in unit testing. Beneath each heading is a list of items to exercise or check. As items are tested, they are marked off or notes made for problem reminders.

3.3 Resources Required for Informal Testing.

3.3.1 Facilities. The MDAC facilities at Building 428 Bergstrom AFB Austin, Texas, will be the location at which informal testing will be performed. The AFHRL facilities at Building 578 Brooks AFB San Antonio, Texas, will be the location where the host computer for AOTS resides. No classified information will be processed in conjunction with the AOTS Phase II informal tests.

3.3.2 Personnel. Personnel required for informal testing are members of the MDAC AOTS Software Development Organization. An in-depth knowledge of the requirements and design of the unit or CPC whose test is being planned or performed is mandatory for the personnel. Access to Bergstrom AFB is necessary for these personnel, however, security clearances are not necessary.

3.3.3 Hardware. Hardware to be used for informal testing is as follows:

- A. VAX 8600 located at Building 578, Brooks AFB;
- B. Zenith Z248 Personal Computers located at Building 428, Bergstrom AFB;
- C. Printers, of the following types, located in Building 428, Bergstrom AFB:
 - 1. Laser printers
 - 2. Color printers
 - 3. Dot matrix printers;
- D. Digitizing tablets, of the following types, located at Building 428, Bergstrom AFB:
 - 1. 11x11 digitizers
 - 2. 20x20 digitizers;
- E. Optical Mark Readers located at MDAC, Building 428, Bergstrom AFB;
- F. Communication lines and equipment to link the above hardware.

3.3.4 Interfacing/Support Software. Software required for informal testing is as follows:

A. VAX/VMS operating system, including the utilities:

1. Command Language
2. EDT
3. Linker
4. Debug
5. Run-time Library

B. DEC Ada Compiler

C. VAX-11 FORTRAN Compiler

D. VAX-11 MACRO Assembler

E. DEC Code Management System

F. MS-DOS, including the utilities

1. Command Language
2. Editor
3. Debug

G. Alsys Ada Compiler

H. Microsoft Macro Assembler

3.3.5 Source. The above resources are provided by AFHRL and MDAC in support of the AOTS contract.

3.3.6 Test Configuration. The test configuration is graphically portrayed in Figure 2. It consists of the hardware listed in paragraph 3.3.3, located at the facilities specified in paragraph 3.3.1. Note that the peripheral equipment layout in Building 428 at Bergstrom AFB shows all the different types of devices to be used in Phase III AOTS. A particular unit or integration test will use a subset of this equipment. However, the unit and integration tests, considered as a group, will use all of the devices.

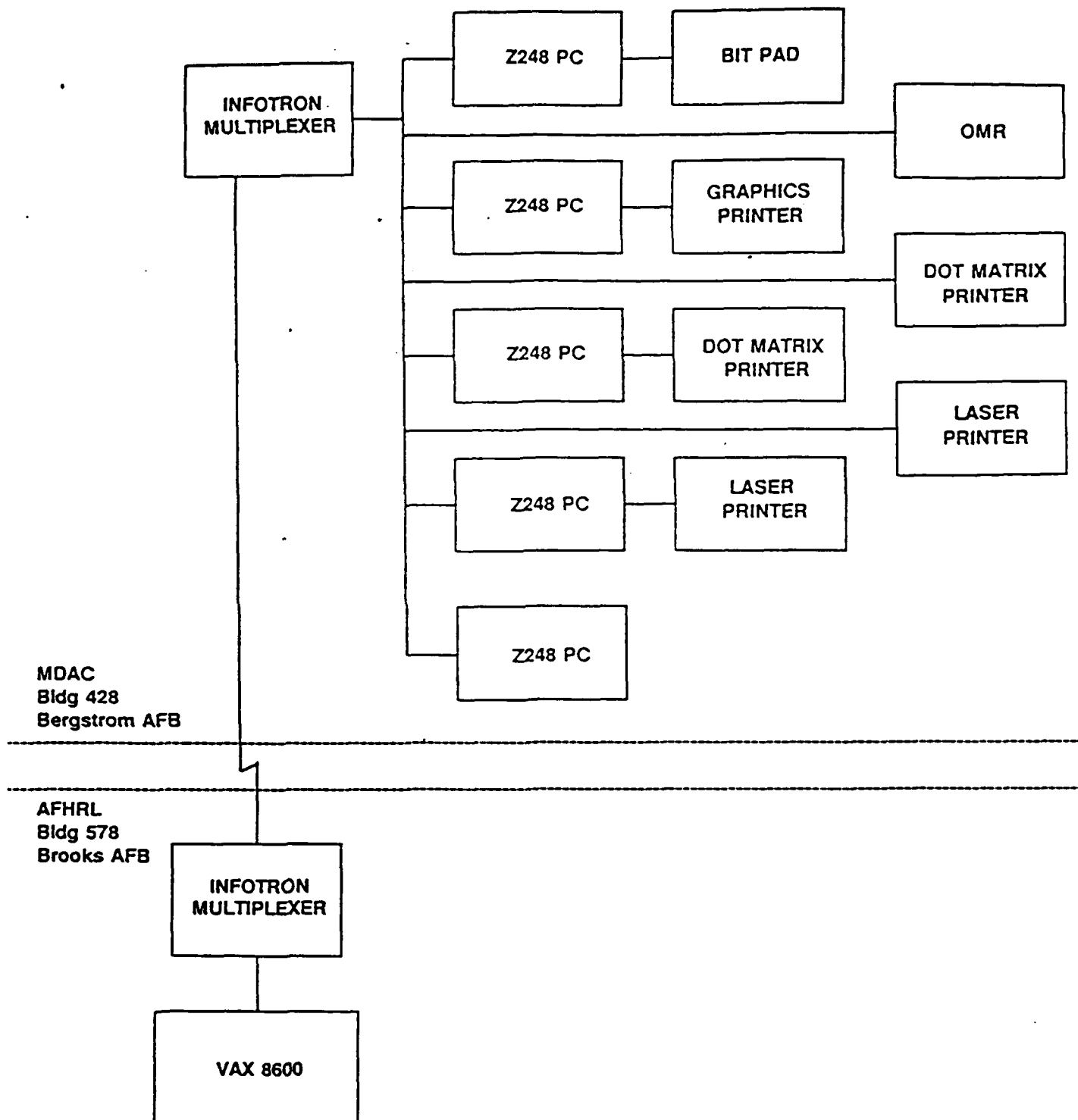


Figure 2. HARDWARE CONFIGURATION FOR INFORMAL TESTING

4. PLANS FOR FORMAL TESTING. The formal phase of testing consists of system level testing and acceptance level testing. The objective is verification that software meets performance and interface requirements and demonstration that the software meets the acceptance criteria. Formal testing is planned by the software development group but is performed by a group independent of the software developers. The expected product is a demonstratable and deliverable program.

Formal testing will be conducted in accordance with the phased software development approach for AOTS Phase II. A formal test procedure, at the system test level, will be prepared during the development period for that program. This test procedure will be performed and results recorded before the program is released for Air Force use. Thus, before the data development programs are released to the IST, they will go through the system testing cycle.

Formal test procedures, at the acceptance test level, will be performed and results recorded, in order for the Air Force to accept AOTS Phase II software.

Both system and acceptance test procedures can be used to satisfy test objectives in the AOTS Master Test Plan that require automated support.

4.1 System Test Planning. The intent of System Testing is to formally verify software performance and interface requirements. System level testing is performed on programs that have completed integration level testing.

4.1.1 System Test Requirements. System Testing consists of tests and analyses performed to confirm that the software satisfies all the requirements set forth in the AOTS CPCI Development Specifications. It verifies that the "as built" software conforms to these specifications. Formal test plans and procedures are written for System Testing. Analysis is performed for those requirements that are too expensive, in terms of time or resources, to verify by testing.

In order for a program to be ready for system testing, it must have successfully completed integration level testing and be placed under configuration control. The system tests are performed according to test procedures. Test results are documented, including any discrepancies found.

4.1.2 System Test Responsibilities. The system test procedures are prepared by the MDAC AOTS software development organization, in particular by the programmer(s) responsible for individual CPCs. This preparation is done during the software development cycle for a particular program. The test procedure should define in detail the actual activities required to perform the test. It should include a description of what should be

accomplished by the test, the inputs that are required, the outputs that are expected, and how the test is to be conducted. The programmer responsible for a CPC will ask that both the program and test procedure be placed under configuration management when the program has completed integration testing, thereby declaring the program is ready for system testing.

Conduct of the system tests will be carried out by the MDAC AOTS Instructional Technology Branch personnel. These individuals are different from the people that comprise the software development group, but they are very familiar with system and program requirements. The people who conduct a system test will also be responsible for recording the test results.

Monitoring of system tests will be performed by the MDAC AOTS Program Manager or designee. This person's responsibility will be to spot check that the system tests are being conducted according to procedures and that results are recorded.

The Air Force AOTS Program Manager, or representative, may witness system tests.

Results of the system tests are recorded in test reports. If the tests uncovered deficiencies, these are also recorded in the test reports. All deficiencies found in system testing are examined by the Software Technical Review Board. This board is comprised of the software manager and each CPCI team leader. The board decides the extent of each deficiency and schedules the problem to be fixed or directs the deficiency and its ramifications to the Configuration Control Board.

4.1.3 System Test Schedule. The system test schedule is specified in the AOTS Software Development Plan.

4.1.4 System Test Procedure Model. Attachment C provides a model formal test procedure. The model is to be used as a guide for the individual creating specific formal test procedures for AOTS software. Note that the model is very similar to the informal testing model discussed under unit level testing and integration level testing. Also note that this formal test procedure model is valid for both System Test Procedures and Acceptance Test Procedures. The following paragraphs describe the elements of the procedure and include instructions to be used by the author in making up the procedure.

Section I identifies the type of formal test procedure, full identification of the program to be tested, and the author of the test procedure. As the plan is revised, the revision author and date should be filled in.

Section II is scheduling information for the performance of the formal test. Enter a projection of the date when the test will start and how long will it take. Also enter the time necessary to develop the plan. This entry, as well as the projected test performance date and duration are for planning and

scheduling purposes.

Section III specifies the resources required for conducting the formal test. The software resource category identifies the program executable to be tested and other programs that are necessary for the test. Identify the data base files required to test this program. Indicate what the state of these files should be, e.g. whether these files are required to contain a previously determined set of data, whether they should be empty, etc. Some test procedures may require a known data base to predict known test results. This resource category should reflect that. Identify what other test procedures need to be run prior to this procedure. This is important when test procedures need to be run in a certain order. For example, it may be necessary to run the Tentative MTL Test Procedure prior to running the MTL Editor Test Procedure. Identify the hardware configuration for the test by listing the required hardware. If the tester needs special qualifications to run the test, enter this data under the personnel requirements category. An example of this might be that the tester should be familiar with one of the AOTS supported Air Force Specialities. When special resources are required, such as a video disk or a drawing, enter this information under the "other" category.

Section IV describes the test. A high level overview of the test should be entered to explain the test. The functions provided by the program that the test procedure is to check should be listed. If there are program functions not tested by the procedure, then these should be listed together with the identification of the test procedure in which they are tested.

Section V lists the actions necessary to perform the test. The test sequence listed here details the input expected from the user, what output can be expected, and a cross reference as to what program function that this action applies to. The listed test sequence should be as exact as possible, in order to produce test consistency.

Attachment F is an example of an AOTS Formal Test Procedure to perform system testing on the task information field editing functions in the MTL Editor. This is to serve as an example only and is not intended to be complete.

4.1.5 System Test Report Model. Attachment D is a model AOTS Formal Test Report. It is to be used as a guide to report on the execution of formal system tests and should be filled out by the individual performing the test during the process of, or immediately after, a system test.

Section I identifies the program tested and the person that did the test. The same report form is used to document results for a system level test and for an acceptance level test. Therefore mark the test level as appropriate. Fill in the test environment, as well as the date. The time duration of the test

can be used for future planning purposes in system testing and should be entered as hours or minutes. If there is a test witness then the identity of that individual should be recorded.

Section II is the reason the test was performed. Is the test being performed on a newly developed program, for a program that has been enhanced, or as part of a release of other enhancements?

Section III is an indication of the test results. Was the outcome expected or unexpected? If the outcome was not as expected, then a description of the problem or the action to be performed should be furnished, either in this section or in the following worksheet. This area can also be used for notes or reminders if the test was successful.

Section IV is a worksheet for system testing. It is to be used to keep notes of test progress, as well as to keep track of possible discrepancies or questions.

4.2 Acceptance Test Planning. The intent of acceptance testing is to demonstrate that the software satisfies the set of predetermined acceptance criteria.

4.2.1 Acceptance Test Requirements. Acceptance Testing consists of performing test procedures designed to confirm that the software satisfies all the requirements set forth in the AOTS CPCI Development Specifications. Acceptance testing must show that the software performs in the operational environment, as it did in test environment. This includes using actual external interfaces.

Generally acceptance testing is performed when a system is delivered. In the case of AOTS, acceptance testing will be performed during the latter part of Phase II. Test results are documented, including any discrepancies found. Some discrepancies are permissible. The Air Force and MDAC AOTS management must agree jointly upon the number and severity of permissible discrepancies.

4.2.2 Acceptance Test Responsibilities. The acceptance test procedures are prepared by the MDAC AOTS software development organization, in particular by the programmer(s) responsible for individual CPCs. This preparation is done after system testing has been performed on the CPC. The test procedure should define in detail the actual activities required to perform the test. It should include a description of what should be accomplished by the test, the inputs that are required, the outputs that are expected, and how the test is to be conducted. The acceptance test procedure is placed under configuration control after it is written and approved.

Conduct of the acceptance tests will be carried out by the MDAC AOTS Instructional Technology Branch personnel. These

individuals may be the same people that performed the system tests. They should be familiar with system and program requirements. The people who conduct an acceptance test will also be responsible for recording the test results.

Monitoring of acceptance tests will be performed by the MDAC AOTS Program Manager or designee. This person's responsibility will be to verify the acceptance tests are being conducted according to procedures and that results are recorded.

The Air Force AOTS Program Manager, or representative, will witness acceptance tests.

Results of the acceptance tests are recorded in test reports. If the tests uncover deficiencies, these are also recorded in the test reports. As in system testing, all deficiencies found in acceptance testing are examined by the Software Technical Review Board. The board decides the extent of each deficiency and schedules the problem to be fixed or directs the deficiency and its ramifications to the Configuration Control Board.

Execution of an acceptance test procedure with expected and agreed upon results, as well as the generation of the corresponding test report, should result in acceptance of the program by the Air Force AOTS Program Manager.

4.2.3 Acceptance Test Schedule. The acceptance test schedule is specified in the AOTS Software Development Plan.

4.2.4 Acceptance Test Procedure Model. Attachment C provides a model formal test procedure. The model is discussed in paragraph 4.1.4. Sections I, II, III, and IV of the test procedures should be filled out accordingly.

Section V of the acceptance test procedure has the same format as the system test procedure. This section, the test sequence, should list the actions necessary to perform the test. The test sequence includes the input expected from the user, what output can be expected, and a cross reference as to what program function that the actions apply to. However, the inputs and outputs listed in the acceptance test procedure are more descriptive than in the system test procedure. The reason for this is that since the acceptance test procedures can be used for demonstration purposes, the audience for the test procedure might not be familiar with the program being demonstrated.

Attachment G is an example of an AOTS Formal Test Procedure to perform acceptance testing on the MTL Editor. This is to serve as an example only and is not intended to be complete.

4.2.5 Acceptance Test Report Model. Attachment D is a model AOTS Formal Test Report. It is to be used as a guide to report on the execution of formal acceptance tests and should be filled out by the individual performing the test during the process of,

or immediately after, an acceptance test. The sections of the model are discussed further in paragraph 4.1.5.

4.3 Formal Test Classes. The test classes for formal testing include functional requirement testing, interface testing, user input/output testing, file input/output testing, error condition testing, timing constraints, and capacity testing. The first six of these classes are verified by each system and acceptance test procedure for which the class is appropriate. AOTS capacity testing is verified by its own test procedure.

4.4 Formal Tests. Each AOTS CPCI will have a series of formal tests, both at the system and acceptance test level, designed to verify each CPC and interface required by the CPCI. The tests will be structured according to the models described in paragraphs 4.1.4 and 4.2.4. The paragraphs below identify these tests for the initial design and development effort. Further tests will be identified during the secondary design effort with an update to this document prior to the Critical Design Review for the total software effort.

4.4.1 Management CPCI Test Procedures. The Management CPCI Test Procedures will consist of the tests listed in Table 1. With each entry is a paragraph reference to the Management CPCI Development Specification. This reference specifies the functions to be performed and the requirements to be satisfied by the program and therefore to be tested by the test procedure. The actual test procedure will describe the test, the functions the procedure is to check, and the inputs and expected outputs to exercise the functions. Other entries in the table specify the levels of testing to be performed for each program and the test methods involved to check each program.

Table 1. Management CPCI Test Procedures

Test #	Test Procedure Name	70S647411 Ref (1)	Test Level(2)	Test Method(3)
1	Tenative MTL	3.2.1.1.1	U,I,S,Ac	T,D
2	Final MTL	3.2.1.1.2	U,I,S,Ac	T,D
3	MTL Editor	3.2.1.1.3	U,I,S,Ac	T,D
4	Task Performance and Proficiency Editor	3.2.1.2.1	U,I,S,Ac	T,D
5	Generic Position Training Requirements Editor	3.2.1.4.1	U,I,S,Ac	T,D
6	Operational Position Training Requirements Editor	3.2.1.4.2	U,I,S,Ac	T,D
7	Other Training Requirements Editor	3.2.1.5	U,I,S,Ac	T,D
8	IMedit	3.2.3.1.1	U,I,S,Ac	T,D
9	IMQueue	3.2.3.3	U,I,S,Ac	T,D
10	MapEdit	3.2.3.3	U,I,S,Ac	T,D
11	Capacity Test	3.1.1	S,Ac	T,D
12-n	(4)			

Notes:

1. 70S647411 is the AOTS Management CPCI Development Specification. Items in this column are references to paragraphs in this specification.
2. Test Level Codes: U=Unit, I=Integration, S=System, Ac=Acceptance
3. Test Method Codes: A=Analysis, I=Inspection, D=Demonstration, T=Test, R=Review
4. The remainder of this table will be developed in accordance with the AOTS phased software development concept.

4.4.2 Evaluation CPCI Test Procedures. The Evaluation CPCI Test Procedures will consist of the tests listed in Table 2. With each entry is a paragraph reference to the Evaluation CPCI Development Specification. This reference specifies the functions to be performed and the requirements to be satisfied by the program and therefore to be tested by the test procedure. The actual test procedure will describe the test, the functions the procedure is to check, and the inputs and expected outputs to exercise the functions. Other entries in the table specify the levels of testing to be performed for each program and the test methods involved to check each program.

Table 2. Evaluation CPCI Test Procedures

Test #	Test Procedure Name	70S647413 Ref (1)	Test Level (2)	Test Method (3)
1	Behavioral Objectives Editor	3.2.1.1	U,I,S,Ac	T,D
2	Test Item Bank Editor	3.2.1.2	U,I,S,Ac	T,D
3	Test Editor	3.2.1.3	U,I,S,Ac	T,D
4	Strategy Manager	3.2.1.7	U,I,S,Ac	T,D
5	Graphics Editor	3.2.1.5	U,I,S,Ac	T,D
6	Capacity Test	3.1.1	U,I,S,Ac	T,D
7-n	(4)			

Notes:

1. 70S647413 is the AOTS Evaluation CPCI Development Specification. Items in this column are references to paragraphs in this specification.
2. Test Level Codes: U=Unit, I=Integration, S=System, Ac=Acceptance
3. Test Method Codes: A=Analysis, I=Inspection, D=Demonstration, T=Test, R=Review
4. The remainder of this table will be developed in accordance with the AOTS phased software development concept.

4.4.3 System Support CPCI Test Procedures. Many of the functional requirements of the System Support CPCI are considered tested as a result of the software development process and informal and formal testing of the Management and Evaluation CPCIs. Therefore these requirements need no separate test procedures. These requirements are listed in Table 3 without a test procedure number, but instead a reference to note 4. The System Support Test Procedures will test the remaining requirements of the System Support CPCI. The test procedures listed in Table 3 with a test procedure number identify these procedures. With each entry in Table 3 is a paragraph reference to the System Support CPCI Development Specification. This reference specifies the functions to be performed and the requirements to be satisfied by the program and therefore to be tested by the test procedure, when a formal test procedure is applicable. The actual test procedure will describe the test, the functions the procedure is to check, and the inputs and expected outputs to exercise the functions. Other entries in the table specify the levels of testing to be performed for each program and the test methods involved to check each program.

Table 3. System Support CPCI Test Procedures

Test #	Test Procedure Name	70S647414 Ref (1)	Test Level (2)	Test Method (3)
(4)	Operating System	3.2.1.1		D
(4)	Ada Compiler	3.2.1.2.1		D
(4)	Host Language Compiler	3.2.1.2.2		D
(4)	Text Editor	3.2.1.3		D
(4)	Virtual Machine Interface	3.2.2		D
(4)	Program Control	3.2.3.1		D
(4)	Inter-Process Communication	3.2.3.2		D
(4)	Data Management	3.2.3.3		D
(4)	Terminal Communication	3.2.3.4		D
(4)	Text Handling	3.2.3.5		D
(4)	Mathematical Services	3.2.3.6		D
(4)	AOTS Utilities	TBD		D
(4)	Operating System (Terminal)	3.2.5.1		D
(4)	Ada Compiler (Terminal)	3.2.5.2		D
1	Terminal	3.2.5.3	U,I,S,Ac	T,D
2	Capacity Test	3.1.1	S,Ac	T,D
3-n	(5)			

Notes:

1. 70S647414 is the AOTS System Support CPCI Development Specification. Items in this column are references to paragraphs in this specification.
2. Test Level Codes: U=Unit, I=Integration, S=System, Ac=Acceptance
3. Test Method Codes: A=Analysis, I=Inspection, D=Demonstration, T=Test, R=Review
4. These functions are demonstrated to be performing properly as a result of other system and acceptance test procedures using the functions.
5. The remainder of this table will be developed in accordance with the AOTS phased software development concept.

4.5 Formal Test Levels. The levels of formal testing for AOTS are system level testing and acceptance level testing. These are discussed in paragraphs 4.1 and 4.2, respectively.

4.6 Formal Test Summary. Identification of formal tests is provided in Tables 1, 2, and 3 above. Additionally, these tables identify test levels, test methods, and program requirements. Test classes are identified, and checked, as necessary in each test procedure, with the exception of capacity testing, which has its own formal test procedure.

4.7 Formal Test Schedule(s). Refer to paragraph 4.1.3 for the System Test Schedule and paragraph 4.2.3 for the Acceptance Test Schedule.

4.8 Data Recording, Reduction, Analysis. The specification of data to be recorded, reduced, or analyzed in order for programs to be properly verified will be detailed in the system and acceptance test procedures for those programs. When the results of the data recording, reduction, or analysis is on hard copy, then this hard copy will be stored with the test reports, which are described in paragraphs 4.1.5 and 4.2.5.

A test log shall be kept when performing acceptance test procedures. All significant events will be reported chronologically on the test log. A completed test log is stored with the test report. A model test log is contained in Appendix E.

4.9 Formal Test Reports. Formal test report models are described in paragraphs 4.1.5 and 4.2.5.

4.10 Resources Required for Formal Testing.

4.10.1 Facilities. The MDAC facilities at Building 428 and the AFHRL IST facilities at Building 1808, both at Bergstrom AFB Austin, Texas, will be the locations at which formal testing will be performed. The AFHRL facilities at Building 587 Brooks AFB San Antonio, Texas, will be the location where the host computer for AOTS resides. No classified information will be processed in conjunction with the AOTS Phase II formal tests.

4.10.2 Personnel. Personnel required for the preparation of formal test procedures are members of the MDAC AOTS Software Development Organization. An in-depth knowledge of the requirements and design of the CPC whose test is being planned is mandatory for the personnel.

Personnel required for conducting formal tests are members of the MDAC AOTS Instructional Technology Branch. The personnel represent the Management Subsystem, Evaluation Subsystem, and IST

Support group and must have the following qualifications:

- A. An in-depth knowledge of one or all subsystems;
- B. Familiarity with the user interface to automated functions for a subsystem;
- C. Understanding of data contents requirements for the automated functions.

Access to Bergstrom AFB is necessary for both groups of personnel, however, security clearances are not necessary.

4.10.3 Hardware. Hardware to be used for formal testing is as follows:

- A. VAX 8600 located at Building 578, Brooks AFB;
- B. Zenith Z248 Personal Computers located at Buildings 428 and 1808, Bergstrom AFB;
- C. Printers, of the following types, located in Buildings 428 and 1808, Bergstrom AFB:
 - 1. Laser printers
 - 2. Color printers
 - 3. Dot matrix printers;
- D. Digitizing tablets, of the following types, located at Buildings 428 and 1808, Bergstrom AFB;
 - 1. 11x11 digitizers
 - 2. 20x20 digitizers;
- E. Optical Mark Readers located in Buildings 428 and 1808, Bergstrom AFB;
- F. Communication lines and equipment to link the above hardware.

4.10.4 Interfacing/Support Software. Software required for formal testing is as follows:

- A. VAX/VMS operating system, including the utilities:
 - 1. Command Language
 - 2. EDT
 - 3. Linker
 - 4. Debug
 - 5. Run-time Library
- B. DEC Ada Compiler
- C. VAX-11 FORTRAN Compiler
- D. VAX-11 MACRO Assembler
- E. DEC Code Management System
- F. MS-DOS, including the utilities
 - 1. Command Language
 - 2. Editor
 - 3. Debug
- G. Alslys Ada Compiler
- H. Microsoft Macro Assembler

4.10.5 Source. The above resources are provided by AFHRL and MDAC in support of the AOTS contract.

5. TEST PLANNING ASSUMPTIONS AND CONSTRAINTS. To help assure valid system level and acceptance level testing, it is assumed that actual AOTS data, e.g. Master Task List data, Behavioral Objectives, Test Items, etc., will be available to exercise the code. Until such time that this actual data is available, simulated versions of the data must be used for testing purposes.

6. NOTES. Not Applicable.

Test Procedure

I. Identification

Author: _____
 Date Prepared: _____
 Unit ID _____
 CPC No: _____
 CM Id: _____
 Package Name: _____
 Test Type: Unit or Integration
 Revision History: _____

II. Scheduled Development Test Performance Dates

Projected Start: _____ Complete: _____
Development Hours: _____ Estimate to perform: _____

III. Resources

Test Source: _____
Data Base Files (N = New or unique, E = Existing)

Unit Test : Other Units (S = Stub, A = Actual)

Integration Test : Other personnel and software

Hardware:

Normal, or tested, AOTS hardware environment
or

Unique, or testing, hardware environment

Host: _____ Terminal: _____ Printer: _____
Mark Reader: _____ X/Y Device: _____ Other: _____

IV. Debug

Special debug code(Y/N): ____ How activated: _____

V. Specific Testing

Test Element(s):

1. List of
2. test
3. elements

Function(s) to test:

1. List of
2. functions
3. to test

Execution paths not tested:

1. List of
2. paths
3. not tested

Test Sequence:

<u>Functions</u>	<u>Inputs</u>	<u>Outputs</u>
function #	expected input	expected output
.	.	.
.	.	.
.	.	.

File Input: _____

File Output: _____

Error conditions: _____

Data Compatibility: _____

Other testing: _____

Test Report

I. Identification

Test Type: Unit or Integration

Unit ID

CPC No: _____

CM Id: _____

Package Name: _____

Environment: _____

Tester: _____

Date: _____

Time duration: _____

II. Reason for test (check one)

☐ Initial Development ☐ Revision/Enhancement ☐ Regression

III. Results (check one)

☐ Satisfactory results ☐ Action required

Action, problem description, or notes:

Date:
Page:

IV. Test Sequence Worksheet

Function Notes

This image shows a full page of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.

V. Testing Checklist

Functions

☐ Performs required functions
☐ Logic ☐ Computations ☐ Input/Output
☐ Relationship to other units
☐ Arguments ☐ Declarations
Action, problem, notes: _____

Execution paths

Unit: ☐ Instructions ☐ Decision branches ☐ Loops
Integration: ☐ Major paths ☐ Calling sequences
Action, problem, notes: _____

User input (N/A ☐, Verify ☐)

☐ Consistency (return key, etc)
☐ Field screen positions ☐ Start ☐ Stop
☐ Data validity - valid and invalid ranges
☐ Minimum (=, <, >) ☐ Maximum (=, >, <) ☐ Typical
☐ Exceptional (null input, etc.) ☐ Type ☐ Format
☐ Action keys
☐ Redisplay ☐ Help ☐ Exit ☐ Abort
☐ Response time
Action, problem, notes: _____

File input (N/A ☐, Verify ☐)

☐ Data validity - valid and invalid ranges
☐ Minimum (=, <, >) ☐ Maximum (=, >, <) ☐ Typical
☐ Exceptional (null input, etc.) ☐ Type ☐ Format
☐ I/O error conditions
Action, problem, notes: _____

Device input (N/A ☐, Verify ☐)

☐ Data validity - valid and invalid ranges
☐ Minimum (=, <, >) ☐ Maximum (=, >, <) ☐ Typical
☐ Exceptional (null input, etc.) ☐ Type ☐ Format
☐ I/O error conditions
Action, problem, notes: _____

Screen or report output (N/A__, Verify__)

(Include all menus, prompts, help sequences)

— Displays consistent
— With other __ displays in program __ programs
— Displays within boundary
— Display proportioned ok __ Too much data __ Too little data
— Constants/Literals display ok __ Spelling __ Understanding
— Variables display ok __ Minimum __ Maximum __ Typical
— Video enhancements (reverse video, underline, color)
— Useful __ Readable in various light conditions
— Display speed __ Too slow __ Too fast
— Erase __ Whole screen __ Partial
Action, problem, notes: _____

File output (N/A__, Verify__)

__ Data validity __ Type __ Format __ I/O error conditions
Action, problem, notes: _____

Device output (N/A__, Verify__)

__ Data validity __ Type __ Format __ I/O error conditions
Action, problem, notes: _____

Error conditions (N/A__, Verify__)

— Unit does not abort __ External interfaces not corrupted
— Data base not corrupted
Action, problem, notes: _____

Data Compatibility

__ Range __ Type __ Format __ Method of transfer
Action, problem, notes: _____

Other testing (N/A__, Verify__)

— _____
— _____
— _____
Action, problem, notes: _____

AOTS Formal Test Procedure

I. Identification

Test Type: System or Acceptance

CPCI: _____

Program: _____

Author: _____

Date Prepared: _____

Revision History: _____

II. Scheduled Test Performance Dates

Projected Start: _____ Duration: _____

Development Hours: _____

III. Resources

Software

Executable Program: _____

Other Programs: _____

Data Base Files: _____

Other Test Procedures: _____

Hardware Configuration: _____

Host: _____ Terminal: _____ Printer: _____

Mark Reader: _____ X/Y Device: _____ Other: _____

Personnel/Qualifications: _____

Other: _____

IV. Description

Function(s) to test:

1. list
2. of
3. functions
4. to
5. test

Function(s) not tested:

1. list
2. functions
3. tested elsewhere

V. Test Sequence

<u>Step</u> <u>seq#</u>	<u>Func</u> <u>fn#</u>	<u>Inputs</u> <u>expected input</u>	<u>Outputs</u> <u>expected output</u> <u>(any special acceptance criteria)</u>
:	:	:	:
:	:	:	:

AOTS Formal Test Report

I. Identification

Test Type: System or Acceptance

CPCI: _____

Program: _____ Revision: _____

Environment: _____

Tester: _____

Date: _____

Time duration: _____

Witness: _____

II. Reason for test (check one)

☐ Initial Development ☐ Revision/Enhancement ☐ Regression

III. Results (check one)

☐ Satisfactory results ☐ Action required

Action, problem description, or notes:

Date:
Page:

IV. Test Procedure Worksheet

Step	Note
------	------

This image shows a full page of white paper with horizontal blue or grey ruling lines. The lines are evenly spaced and run across the width of the page. There is no handwriting or other markings on the paper.

AOTS Acceptance Test Log

Page ____ of ____

I. Identification

CPCI: _____

Program: _____

Person Completing Log: _____

Date: _____

II. Event Log

Time	Event

AOTS Formal Test Procedure

I. Identification

Test Type: System
CPCI: Management CPCI
Program: MTL Editor
Author: J. Snordley
Date Prepared: 13 Jan 87
Revision History: NA

II. Scheduled Test Performance Dates

Projected Start: 15 Feb 87 Duration: 4 hrs
Development Hours: 8 hrs

III. Resources

Software

Executable Program: MTLEdit
Other Programs: NA
Data Base Files: MTL0001 MTL0001 MTL0003 MTL0004
TPP0001 TPP0002 TPP0003 TPP0004

Other Test Procedures: Tenative_MTL

Hardware Configuration:

Host: VAX 8600 Terminal: Z248
Personnel/Qualifications: Familiar with AOTS concepts,
MTL data, and at least 1 supported AFS
Other: OMC task list for an AFS

IV. Description. This test procedure will test all task information field editing functions of the MTL Editor.

Function(s) to test:

Create and change the following fields in a task:

1. Task statements
2. Task identification numbers
3. Source identifications
4. Speciality Training Standard identifications
5. User identification codes
6. Training material identifications and locations
7. Task certification before performance codes and frequencies
8. Task recertification requirement codes and frequencies
9. Common subtask requirement codes
10. Position task requirement codes
11. Task factors
 - A. Percent members performing
 - B. Difficulty
 - C. Training emphasis
12. Weapon system
13. Support equipment required

14. Priority codes

- A. Training
- B. Evaluation
- C. Development

15. Mandatory task training and performance requirement codes

Function(s) not tested:

Editting of the following fields is tested in Task Decomposition

Test Procedure:

1. Subtask identification codes with performance sequence
2. Task steps/performance sequence

V. Test Sequence

<u>Step</u>	<u>Func</u>	<u>Inputs</u>	<u>Outputs</u>
1.		type aots	AOTS logo and prompt for ID
2.		enter ID	Prompt for password
3.		enter password	If qualified user, logon menu
4.		execute MTL Editor	MTL Editor Menu
5.		enter create	prompt for AOTS task id
6.		enter task id	if new task, task information menu if existing task, error message
proceed through task information menu			
7.	1	enter task statement	statement is displayed
8.	2	enter task id number	id number is displayed
		.	
		.	
		.	
modify the task information fields			
101.	1	enter option to modify statement	task statement is displayed
102.	1	change task statement - try various positioning and editing sequences	updated task statement is displayed
103.	1	enter option to redisplay task statement	task statement is displayed as edited above
		.	
		.	
		.	

AOTS Formal Test Procedure

I. Identification

Test Type: Acceptance
CPCI: Management CPCI
Program: MTL Editor
Author: J. Snordley
Date Prepared: 15 Mar 88
Revision History: NA

II. Scheduled Test Performance Dates

Projected Start: 15 July 88 Duration: 2 hrs
Development Hours: 8 hrs

III. Resources

Software

Executable Program: MTLEdit
Other Programs: NA
Data Base Files: MTL0001 MTL0001 MTL0003 MTL0004
TPP0001 TPP0002 TPP0003 TPP0004

Other Test Procedures: Tenative_MTL

Hardware Configuration:

Host: VAX 8600 Terminal: Z248
Personnel/Qualifications: Familiar with AOTS concepts,
MTL data, and at least 1 supported AFS
Other: OMC task list for an AFS

IV. Description. This test procedure will demonstrate all functions of the MTL Editor that were specified in the Management CPCI Development Specification.

Function(s) to test:

1. Search for a task entry
2. Display, review, and print task and task information
3. Decompose task into subtasks and subtasks into steps
4. Group task information by catagory
5. Prioritize tasks
6. Perform task and task information editing functions.
7. Cross reference Task Performance and Proficiency Data to task
8. Update MTL

Function(s) not tested: NA

V. Test Sequence

<u>Step</u>	<u>Func</u>	<u>Inputs</u>	<u>Outputs</u>
1.		type aots	This is the common entry point for all AOTS users. The AOTS logo is displayed along with a prompt for user identification.
2.		enter ID	The user must establish valid identification with the system. After the user id is entered, a prompt for password is displayed.
3.		enter password	The password will not echo on the screen. If the user is valid, a logon menu listing the programs which the user is qualified for is displayed.
4.		enter the ordinal associated with the MTL Editor	This executes the MTL Editor. The main menu of the MTL Editor is displayed. Review this main menu.
5.	1	enter MTL search function by pressing the S key	A prompt is displayed that is requesting the search criteria.
6.	1	enter search criteria (search criteria will be definitized here)	A search is performed on the MTL. If a task satisfies the criteria, high level data from the task is displayed. Otherwise a message is displayed indicating no task can be found which meets the input criteria. (This search must be performed within a maximum of 60 seconds.)
7.	2	enter display option by pressing D key	Task information associated with the task found on the search above is displayed.
		.	
		.	
		.	