

WL-TR-93-1003

COMMON ADA PROGRAMMING SUPPORT
ENVIRONMENT (APSE) INTERFACE SET (CAIS)
IMPLEMENTATION VALIDATION CAPABILITY (CIVC2)

AD-A261 043



JAIRO FREYRE
DAVID REMKES
JEFFREY RAGSDALE
SOFTECH, INC.
1300 HERCULES, SUITE 105
HOUSTON TX 77058

JUN 1992

FINAL REPORT FOR 09/02/90-06/15/92

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.

93-04042



AVIONICS DIRECTORATE
WRIGHT LABORATORY
AIR FORCE SYSTEMS COMMAND
WRIGHT PATTERSON AFB OH 45433-7409

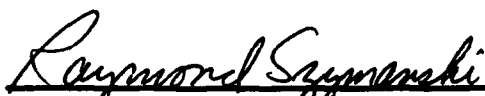
DTIC
ELECTE
FEB 26 1993
S E D

93 2 25 060

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.

THIS TECHNICAL REPORT HAS BEEN REVIEWED AND IS APPROVED FOR PUBLICATION.



RAYMOND SZYMANSKI
Program Manager



TIMOTHY G. KEARNS, Maj, USAF or
Chief
Readiness Technology Group



CHARLES H. KRUEGER, Chief
System Avionics Division
Avionics Directorate

DYIC QUALITY INSPECTED 1

Unannounced Justification		<input checked="" type="checkbox"/>
By _____ Distribution /		
Availability Codes		
Dist	Avail and / or Special	
A-1		

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 WL/AAAF, WRIGHT-PATTERSON AFB, OH 45433-6543 TO HELP 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.

REPORT DOCUMENTATION PAGE

Form 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 JUN 1992	3. REPORT TYPE AND DATES COVERED FINAL 09/02/90--06/15/92
----------------------------------	-----------------------------------	---

4. TITLE AND SUBTITLE COMMON ADA PROGRAMMING SUPPORT ENVIRONMENT (APSE) INTERFACE SET (CAIS) IMPLEMENTATION VALIDATION CAPABILITY (CIVC2)	5. FUNDING NUMBERS C F33615-87-C-1449 PE 63226 PR 2853 TA 01 WU 02
---	--

6. AUTHOR JAIRO FREYRE DAVID REMKES JEFFREY RAGSDALE	
--	--

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) SOFTECH, INC. 1300 HERCULES, SUITE 105 HOUSTON TX 77058	8. PERFORMING ORGANIZATION REPORT NUMBER
--	--

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AVIONICS DIRECTORATE WRIGHT LABORATORY AIR FORCE SYSTEMS COMMAND WL/AAAF, Attn: SZYMANSKI 513-2553947 WRIGHT-PATTERSON AFB OH 45433-7409	10. SPONSORING / MONITORING AGENCY REPORT NUMBER WL-TR-93-1003
--	--

11. SUPPLEMENTARY NOTES ADA JOINT PROGRAM OFFICE (SPONSOR) 1211 S. FERN ST ARLINGTON VA 22202

12a. DISTRIBUTION STATEMENT (If applicable to this report) APPROVED FOR PUBLIC RELEASE; DISTRIBUTION IS UNLIMITED.	12b. DISTRIBUTION CODE
--	------------------------

13. ABSTRACT (Maximum 200 words)

THIS REPORT SUMMARIZES THE ACCOMPLISHMENTS AND LESSONS LEARNED FROM PHASE III OF THE CIVC PROJECT. IT INCLUDES RECOMMENDATIONS IN THE EVENT THE CIVC TECHNOLOGY IS REINSTATED IN THE FUTURE AND SUGGESTIONS FOR TECHNOLOGY TRANSFER OF VALIDATION PRODUCTS TO OTHER AREAS OF SOFTWARE DEVELOPMENT.

14. SUBJECT TERMS CAIS; MIL-STD-1838A, VALIDATION, TESTING, ADA FRAMEWORKS, TEST CLASS, TEST CASE, TEST OBJECTIVES EXCEPTION PROCESSING, CAIS PRAGMATICS	15. NUMBER OF PAGES 29
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
--	---	--	---

TABLE OF CONTENTS

1. SCOPE.....	6
1.1 Introduction.....	6
1.2 Purpose	7
2. APPLICABLE DOCUMENTS.....	8
2.1 Government Documents.....	8
2.2 Non-Government Documents.....	8
3. PHASE III PRODUCTS AND METHODS.....	10
3.1 Examples of Phase III Products	10
3.1.1 Test Suite Entities.....	10
3.1.1.1 Test Objective.....	10
3.1.1.2 Test Scenario.....	10
3.1.1.3 Test Case.....	10
3.1.1.4 CAIS_Pragmatics Test.....	11
3.1.1.5 Static Semantics Test.....	11
3.1.2 Test Administrator.....	11
3.1.3 Framework.....	11
3.1.4 Automation and Tools.....	12
3.1.4.1 Test Objective Interface Selection Algorithm.....	12
3.1.4.1.1 Preparatory Analysis	13
3.1.4.1.2 Test Objective Selection.....	13
3.1.4.2 Exception/Condition Code Selection Algorithm	14
3.2 Phase III Test Suite Coverage Analysis.....	14
3.3 Software Productivity in Phase III	15
3.3.1 Beta Test Suite Prototype	17
3.4 Lessons Learned.....	18
3.4.1 Test Objective Implementation.....	18
3.4.2 Stand-alone Test Cases	18
3.4.3 Suspect Test Suite Errors.....	19
3.4.4 Multiple Stack Framework.....	19
3.4.5 WBS Granularity.....	19
4. CIVC PHASE IV MAINTENANCE.....	21
4.1 CIVC Phase IV Maintenance Activities.....	21

5. SUGGESTIONS FOR TECHNOLOGY TRANSFER.....18

5.1 Technology Transition.....22

5.1.1 Hypertext Technology.....22

5.1.1.1 Ada 9X.....23

5.1.1.2 Ada Compiler Evaluation Capability (ACEC).....23

TABLE OF CONTENTS (cont'd)

5.1.2 Test Selection.....24

5.1.2.1 ACEC Test Selection.....24

5.1.2.2 PCIS Test Selection25

6. SUMMARY.....26

APPENDIX A Notes.....A-1

A.1 AcronymsA-1

LIST OF TABLES

Table 3.3-1 Phase I Test Suite Software Productivity Estimates.....13

Table 3.3-2 Phase III Test Suite Software Productivity Estimates13

Table 3.3-3 Phase I Test Administrator Software Productivity Estimates.....14

Table 3.3-4 Phase III Test Suite Support Software Productivity Estimates14

1. SCOPE

1.1 Introduction

The CIVC Final Report provides a summary of the accomplishments and lessons learned from Phase III of the CIVC project, recommendations if CIVC technology is reinstated in the future, and suggestions for technology transfer of validation products to other areas of software development.

The four phases of the CIVC project are summarized as follows:

- Phase I produced a validation suite and test administration tool for CAIS implementation (DoD-1838) and associated Framework hypertext documentation product.
- Phase II was the maintenance activity for the Phase I software.
- Phase III produced a validation suite for CAIS-A (MIL-STD-1838A) implementations, a new test administration tool, and a new Framework hypertext documentation product.
- Phase IV has been identified as the maintenance activity for Phase III software, although no current plans exist to fund Phase IV activities.

CAIS and CAIS-A are military standards specifying common interface sets for Ada Programming Support Environments (APSEs). The interface sets (specified as Ada packages) provide environment and device management services for tools. For a more comprehensive discussion of the interface sets, see Munck88a and Oberndorf88.

The CIVC Final Report is divided into six sections. Section 1 addresses the scope of this document. Section 2 lists the applicable documents. Section 3 discusses the products of CIVC Phase III, its development methodology, CIVC software productivity, and lessons learned. Section 4 addresses CIVC validation technology in the future. Section 5 offers suggestions for technology transfer of CIVC technology to relevant programs within the government. Section 6 summarizes the accomplishments of CIVC Phase III.

1.2 Purpose

The purpose of the CIVC is to develop validation suites for implementations of the CAIS and CAIS-A . The validation suites will be applied to CAIS implementations in much the same way the Ada Compiler Validation Capability (ACVC) test suites are applied to Ada compilers. Validation is necessary to maintain the effectiveness of the standards.

1. SCOPE

1.1 Introduction

The CIVC Final Report provides a summary of the accomplishments and lessons learned from Phase III of the CIVC project, recommendations if CIVC technology is reinstated in the future, and suggestions for technology transfer of validation products to other areas of software development.

The four phases of the CIVC project are summarized as follows:

- Phase I produced a validation suite and test administration tool for CAIS implementation (DoD-1838) and associated Framework hypertext documentation product.
- Phase II was the maintenance activity for the Phase I software.
- Phase III produced a validation suite for CAIS-A (MIL-STD-1838A) implementations, a new test administration tool, and a new Framework hypertext documentation product.
- Phase IV has been identified as the maintenance activity for Phase III software, although no current plans exist to fund Phase IV activities.

CAIS and CAIS-A are military standards specifying common interface sets for Ada Programming Support Environments (APSEs). The interface sets (specified as Ada packages) provide environment and device management services for tools. For a more comprehensive discussion of the interface sets, see Munck88a and Oberndorf88.

The CIVC Final Report is divided into six sections. Section 1 addresses the scope of this document. Section 2 lists the applicable documents. Section 3 discusses the products of CIVC Phase III, its development methodology, CIVC software productivity, and lessons learned. Section 4 addresses CIVC validation technology in the future. Section 5 offers suggestions for technology transfer of CIVC technology to relevant programs within the government. Section 6 summarizes the accomplishments of CIVC Phase III.

1.2 Purpose

The purpose of the CIVC is to develop validation suites for implementations of the CAIS and CAIS-A . The validation suites will be applied to CAIS implementations in much the same way the Ada Compiler Validation Capability (ACVC) test suites are applied to Ada compilers. Validation is necessary to maintain the effectiveness of the standards.

2. APPLICABLE DOCUMENTS

2.1 Government Documents

- [AJPO88] Ada Joint Program Office, *Comparison of CAIS-A and PCTE+*, Joint study with Independent European Programme Group (IEPG TA-13), June 1988.
- [AJPO89] Ada Joint Program Office, *PCTE+ and CAIS-A Convergence*, Joint feasibility study with Independent European Programme Group (IEPG TA-13), November 1989.
- [DOD-STD-1838] Common APSE Interface Set (CAIS), 9 October 1986.
- [MIL-STD-1815A] Reference Manual for the Ada Programming Language, 17 February 1983.
- [MIL-STD-1838A] Common APSE Interface Set Revision A (CAIS-A), 6 April 1989.
- [MIL-STD-847B] Format Requirements for Scientific and Technical Reports Prepared by or for the Department of Defense, 7 November 1983.

2.2 Non-Government Documents

- [CIVC-IG] CIVC Implementor's Guide for the CAIS Implementation Validation Capability, CIVC-FINL-19-02, SofTech, Inc., 15 June 1992.
- [CIVC-SPS] Software Product Specification for the CIVC2 CSCI of the CAIS Implementation Validation Capability Project, CIVC-FINL-14-05, SofTech, Inc., 15 June 1992.
- [CIVC-SRS] Software Requirements Specification for the CIVC2 CSCI of the CAIS Implementation Validation Capability Project, CIVC-FINL-013-05A, SofTech, Inc., 1 May 1991.
- [CIVC-STR] Software Test Report for the CIVC2 CSCI of the CAIS Implementation Validation Capability Project, CIVC-FINL-16-02, SofTech, Inc., 30 June 1992.

- [Gutzmann90] Kurt Gutzmann, David Remkes, Jeff Ragsdale, Software Project Activity Network for Managing the Development and Testing Process, *SofTech Technical Report H90-1*, August 31, 1990.
- [HC-REF] HyperCard Reference, Claris Corporation, 1989-1990.
- [HC-SLG] HyperCard Scripting Language Guide, Claris Corporation, 1989-1990.
- [Hooi90] Robert Hooi, Mark Denson, A Taxonomic Method for Interface Set Validation, *CIVC Technical Report (working paper)*, 1990.
- [MAC-REF] Macintosh Reference, Apple Computer, Inc., 1990.
- [MAC-SYS] Macintosh System Software User's Guide, Version 6.0, Apple Computer, Inc., 1988.
- [Munck88a] Robert Munck, Patricia Oberndorf, Erhard Ploedereder, Richard Thall, An Overview of DOD-STD-1838A (proposed), The Common APSE Interface Set, Revision A, *CACM*, May 1988.
- [Munck88b] Robert Munck, Why Strong Typing was added to DOD-STD-1838A, The Common APSE Interface Set, *Proceedings of the Sixth Annual Conference on Ada Technology*, March, 1988.
- [Oberndorf88]. Patricia Oberndorf, The Common Ada Programming Support Environment (APSE) Interface Set (CAIS), *IEEE Transactions on Software Engineering*, October 1988, pp. 742-746
- [SEVWG89] Tim Lindquist *et al*, Issues and Strategies for Evaluation and Validation of CAIS-A Implementations, *SEVWG Working Paper*, Version 2.0, December 1989.
- [Woodcock90] Gary Woodcock, Automated Generation of Hypertext Documents, *CIVC Technical Report (working paper)*, 1990.

3. PHASE III PRODUCTS AND METHODS

3.1 Examples of Phase III Products

The CIVC2 development approach has yielded a set of products and methods that test conformance of a CAIS-A implementation to the specification as prescribed by MIL-STD-1838A. Refer to the CIVC-SPS for complete source code listings of all CIVC2 software products. Refer to CIVC-STR for complete conformance reports generated by CIVC2 test suite execution.

3.1.1 Test Suite Entities

The CIVC2 test suite comprises test objectives, scenarios for implementing the test objectives, the test cases which are implemented test objectives, a test for the package CAIS_Pragmatics, and a test for the static semantics and completeness of the CAIS-A implementation.

3.1.1.1 Test Objective

A test objective is a mapping to a subprogram interface identified in MIL-STD-1838A that defines a testable condition, action, or CAIS-A environment database configuration applicable to a CAIS-A implementation. For each CAIS-A interface selected as a test objective to implement, a normal and an exceptional processing mode test objective is created.

3.1.1.2 Test Scenario

A test scenario is a test case design consisting of a defined precondition, postcondition, and set of required CAIS-A interfaces necessary for demonstrating a particular aspect of a test objective. Scenario data are embedded as special comments in the test case source code, and are designated by the --% prefix. Specifying the precondition and postcondition processing independently for each test case minimizes the possibility of test case dependency failures in the test suite.

3.1.1.3 Test Case

The test case is an implementation of a test objective associated with a CAIS-A interface. The test case implements either a normal processing test objective or an exceptional processing test objective. Test cases are logically grouped into test classes based on their similar package dependency requirements. A normal processing mode test case is an implemented test objective based on the normal execution of a test case interface in which no exceptional condition is expected to be raised. An exceptional processing

mode test case is an implemented test objective based on the execution of a test case interface in which an exceptional condition is expected to be raised.

3.1.1.4 CAIS_Pragmatics Test

Verification of the CAIS Pragmatics Ada package is accomplished by a demonstration test on the CAIS-A host platform. The CAIS_Pragmatics_Test evaluates the constraints identified in the package CAIS_Pragmatics, and reports any errors to the log file PRAGMATICS_TEST.RPT. Implementation-defined and non-implementation-defined constants, in addition to the exceptions CAPACITY_ERROR and RESOURCE_ERROR, are evaluated by this test. As the CAIS_Pragmatics_Test executable does not link with the CAIS-A, it may be executed directly at the command line at any time after the installation procedures have been completed.

3.1.1.5 Static Semantics Test

This test product tests the static semantics and completeness of a CAIS-A implementation. This test requires inspection of the CIVC2 Ada library to determine if any of the wrapper units are missing for which test cases are developed. The wrappers are derived from the published version of the CAIS-A specification. Failure of any wrapper unit to successfully compile using the CAIS-A implementation's library specification files indicates a non-conformant element in the CAIS-A implementation.

3.1.2 Test Administrator

The Test Administrator (TA) provides the means for executing the test suite comprising all the test classes and associated test cases. The functional capabilities of the TA are summarized as follows:

- provides a user interface
- executes test class programs
- verifies the minimal set of CAIS-A functionality necessary to execute a test class
- collects the test results from each test class and present it in a single conformance report
- maintains the selection lists.

When the TA is started, it provides the user with a menu. The user can choose to modify the selection lists. The selection lists allow the user to define persistent groups of test classes to execute. After a selection list has been created, the user can instruct the TA to execute the test classes in the list. As the test classes are executing, the results can be displayed on the screen. As

each test class executes, the TA concatenates the test results file onto the conformance report for that selection list. The user may interrupt the execution of the selection list and processing will be terminated after the currently executing test class has completed.

3.1.3 Framework

The Framework is a Macintosh-based HyperCard^{®1} stack which allows the user to obtain information regarding the CIVC2 test suite. The Framework is not necessary to execute the Test Administrator or the Test Suite. Its primary function is to provide diagnostic information when a CAIS-A implementation does not successfully execute a CIVC2 test case. The Framework is an on-line hypertext-based documentation product that captures the relationships between MIL-STD-1838A text, Ada package specifications, test cases, test classes, CAIS Pragmatics test, exceptions, and keywords. Designed as an information network, the Framework allows the user intuitive access to CIVC2 data in a non-linear fashion.

3.1.4 Automation and Tools

More than 4000 potential test objectives have been identified through analysis of the CAIS-A interfaces. However, it is not feasible to implement the large number of potential test objectives given the cost and schedule constraints of the CIVC contract. Therefore, the 550 potential test objectives that provide maximum test benefits have been selected for implementation. To optimize the benefit from the testing process, two approaches to identifying the test objectives to be implemented as test cases have been utilized. Software tools have been developed to automate algorithms that support both the approaches for selecting objectives. These tools include an automatic test case code generator that outputs an Ada source code template for each test case, a program that identifies the test objectives to be implemented as test cases, and a program that identifies the appropriate exceptional processing conditions for maximizing coverage of exceptions. A brief discussion of these tools and the underlying selection algorithm follows. Refer to the CIVC-IG for specific information on test objective selection.

3.1.4.1 Test Objective Interface Selection Algorithm

The selection process assigns a weight value to each of the CAIS-A interfaces based on the four pre-determined selection criteria. The specification of the selection criteria and their relative importance has been jointly established by the contractor, government representatives, and CAIS-A implementors based on discussions from numerous Technical Interchange Meetings and CIVC

¹ HyperCard is a registered trademark of Apple Computer, Inc.

Working Group (CIVCWG) meetings. The 275 CAIS-A interfaces having the highest weight values are selected for test case development in both normal and exceptional processing mode.

3.1.4.1.1 Preparatory Analysis

The selection of test objectives from the list of potential test objectives requires an analysis of the CAIS-A interfaces as the initial step of the process. This analysis is based on 1) a set of selection criteria that is applied to all of the CAIS-A interfaces, and 2) a mapping of the CAIS-A interfaces to their possible exceptional conditions and exceptions. The selection criteria is based on the following activities, in order of importance:

- *Identify the CAIS-A interfaces having high implementation complexity.* Known as the *set_1_value*, if an interface is a member of this set, it receives a value of 2. Non-set members receive a value of 0.
- *Classify the CAIS-A interfaces into non-I/O, device-independent I/O (e.g., Text_IO), and hardware-dependent I/O package sets.* Known as the *set_2_value*, a non-I/O package interface receives a value of 2, a device-independent I/O package interface receives a value of 1, and a hardware-dependent I/O package interface receives a value of 0.
- *Classify the CAIS-A interfaces into active, transitory, and passive sets.* Known as the *set_3_value*, an active interface receives a value of 2, a transitory interface receives a value of 1, and a passive interface receives a value of 0. Refer to the CIVC-IG for more information on active, transitory, and passive interfaces.
- *Identify the closure sets of the CAIS-A packages.* Known as the *set_4_value*, each interface receives a value that corresponds to the number of library compilation units included in its context clause. Refer to the CIVC-IG for more information on closure sets.

3.1.4.1.2 Test Objective Selection

The application of the selection criteria will result in a potential test objective for each CAIS-A interface identified through domain analysis. Associated with each of the potential test objectives is a score known as its Figure of Merit (FOM). The Figure of Merit represents the relevancy of a particular interface based on the selection criteria, and provides the basis for determining which of the potential test objectives become formal test objectives. A raw FOM is obtained based on the following equation:

$$\text{Raw FOM} = \text{set_1_value} * 4 + \text{set_2_value} * 3 + \text{set_3_value} * 2 \\ + (\text{set_4_value}/\text{avg. closure}).$$

The raw FOM scores for all of the interfaces are then multiplied by a scaling factor to achieve a range of FOM scores having 100 as the maximum value. For example, if the highest raw FOM score is 20, then the scaling factor to be applied to all the raw FOM scores would be 5.

3.1.4.2 Exception/Condition Code Selection Algorithm

Selection of the exception/condition code for each test case interface's exceptional processing mode execution is aided by a modified best fit approach to optimize exception/condition code coverage. A variation of the classic bin packing algorithm, the steps of this technique are summarized as follows:

- *Assign a weight to each interface that is based on the number of exception/condition code pairs it may raise. The equation used to assign the weight is expressed as $W(I) = (K - N_e/c(I) + 1) / K$ where W is the weight of the interface, I is the interface, K is the maximum weight constant (100.0), and N_e/c is the number of exception/condition code pairs of the interface. Refer to the CIVC-IG for more detail on the exception/condition code selection process.*
- *Sort the weighted interfaces in descending order.*
- *Assign a mass weight to each condition code pair based on the weights of all its associated interfaces.*
- *Sort the exception condition code pairs by mass in ascending order.*
- *For each test objective interface, select the first exception/condition code pair in the sorted mass list for exceptional processing implementation, provided that the pair has not been previously chosen. If the pair is already selected, choose the next available pair not chosen. Note that most exception/condition code pairs are not unique to a particular interface, therefore, some pairs will be implemented more than once for different interfaces.*

3.2 Phase III Test Suite Coverage Analysis

The coverage of the Phase III test suite was as follows:

- 273 of the 496 principal CAIS-A interfaces (55%)

- 108 unique exception/condition code pairs of 231 possible (47%)
- 25 of 37 packages tested (68%)
- Package breakdown:
 - Packages having 100% coverage of interfaces:
 - CAIS_ATTRIBUTE_MONITOR_MANAGEMENT
 - CAIS_DEFINITION_MANAGEMENT
 - CAIS_FILE_NODE_MANAGEMENT
 - CAIS_IO_CONNECTION
 - CAIS_NODE_MANAGEMENT
 - CAIS_PROCESS_MANAGEMENT
 - CAIS_STRUCTURAL_NODE_MANAGEMENT
 - Packages having >50% coverage of interfaces:
 - CAIS_COMMON_IO
 - CAIS_TRANSLATION_MANAGEMENT
 - CAIS_FRAME_IO
 - CAIS_ACCESS_CONTROL_MANAGEMENT
 - CAIS_BOOLEAN_ATTRIBUTE
 - CAIS_TEXT_IO
 - CAIS_TRANSACTION_MANAGEMENT
 - CAIS_IMPORT_EXPORT
 - Packages having <50% coverage of interfaces:
 - CAIS_ATTRIBUTE_MANAGEMENT
 - CAIS_ATTRIBUTE_MANAGEMENT.FLOAT
 - CAIS_ATTRIBUTE_MANAGEMENT.INTEGER
 - CAIS_ATTRIBUTE_MANAGEMENT.ENUMERATION
 - CAIS_ATTRIBUTE_MANAGEMENT.IDENTIFIER
 - CAIS_ATTRIBUTE_MANAGEMENT.STRING
 - CAIS_SEQUENTIAL_IO
 - CAIS_DIRECT_IO
 - CAIS_LIST_MANAGEMENT
 - CAIS_STATUS_MANAGEMENT

3.3 Software Productivity in Phase III

Phase III software productivity level realized a significant increase relative to the Phase I software productivity level. Table 3.3-1 reflects the Phase I productivity estimates and Table 3.3-2 reflects the Phase III productivity estimates.

Table 3.3-1 Phase I Test Suite Software Productivity Estimates

Unit of Measure	Number produced	Productivity per month based on 24 man months
Test Cases	253	11
Lines of code	14,090	587
Ada statements	7,446	311
comments	4,506	188

Table 3.3-2 Phase III Test Suite Software Productivity Estimates

Unit of Measure	Number produced	Productivity per month based on 30 man months
Test Cases	550	18
Lines of code	102,501	3417
Ada statements	25,690	856
comments	35,363	1179

A number of factors contributed to the significant increase in number of lines of Ada code developed (582%). These items, in order of significance, are as follows:

- Automated test case code generator - This Ada development tool generated Ada source code modules for each of the test case modules. These modules served as templates which were completed by the developers.
- Design/development methodology - Phase III used a 1:1 ratio of test objectives to test cases and implemented each test objective into a test case prior to developing scenarios for subsequent test cases. Phase I did not maintain a 1:1 ratio of these products, and

also did not implement each test objective before developing additional objectives. Consequently, Phase I resulted in numerous test objectives developed that never became implemented as test cases.

A similar increase in productivity was also attained for development of the test suite support software (test administrator, report generator, etc.). Tables 3.3-3 and 3.3-4 show a comparison of these metrics.

Table 3.3-3 Phase I Test Administrator Software Productivity Estimates

Unit of Measure	Number produced	Productivity per month based on 27 man months
Lines of code	25,286	937
Ada Statements	13,402	496
Comments	9,122	338

Table 3.3-4 Phase III Test Suite Support Software Productivity Estimates

Unit of Measure	Number produced	Productivity per month based on 30 man months
Lines of code	45,088	1053
Ada Statements	16,189	540
Comments	10,794	360

For the CIVC2 test suite, the ratio of comments to statements is 137%; for the test suite support software, the ratio is 66%. The scenario comments account for the significant increase in test suite comments over the test suite support code comments.

3.3.1 Beta Test Suite Prototype

Subsequent to Phase I and II activities, but prior to the initiation of the CIVC2 development activities, an out-of-scope prototype development task was undertaken to port the CIVC1 test suite to a new suite targeted to the CAIS-A. The impetus for this activity was to provide immediate testing support to the CAIS-A development team at the SofTech San Diego office. Establishing early

rapport with the developers has resulted in corrections and improvements to the various CAIS-A implementation releases targeted to the VAX/VMS platforms. This relationship has continued throughout the CIVC2 program and has been one of the most significant yet unscheduled benefits to the CAIS-A program.

The beta development task required 18 man months to implement over a 6 month period. A total of 41 unique interfaces were implemented in both normal and exceptional processing modes. No specific metrics were gathered during that activity, but the products have been rolled over into the final CIVC2 test suite where their metrics are ultimately reflected.

3.4 Lessons Learned

The lessons learned by SofTech during the CIVC project are presented here. The lessons learned are used to improve the products and productivity for projects that plan to utilize CIVC technology.

3.4.1 Test Objective Implementation

The CIVC Phase I test suite did not implement test objectives, test scenarios, and test cases on a 1:1:1 basis. Additionally, no test objectives were implemented into test cases until all of the test objective scenarios had been designed for the complete test suite. Due to schedule constraints, this resulted in a significant surplus of designed test objectives which were never implemented.

For CIVC Phase III, each test objective had a single design scenario developed and implemented into a complete test case prior to initiating design of a new test objective. This resulted in all test objectives being implemented into test cases.

3.4.2 Stand-alone Test Cases

The CIVC Phase I test suite implemented all the test cases of a test class within a single compilation unit. While it is logical to group all of the test cases with similar dependency relationships into the same test class, it magnifies the problem of side effects and introduces dependency failure. Specifically, the precondition of a test case was dependent on the post-condition of the immediately preceding test case. If the preceding test case failed, the subsequent test cases were not executed because of the unexpected state of the CAIS-A node structure.

Design of the Phase III test suite eliminated this dependency failure problem by implementing each test case as a separate compilation unit, with the precondition and postcondition processing of each test encapsulated within the module. This design approach reduces the likelihood that a test case will

fail due to preceding test case execution. It is important to note that there is no guarantee that side effects will not occur based on this design approach. Section 3.4.3 addresses such issues.

3.4.3 Suspect Test Suite Errors

Errors that appear suspect may be the result of side effects of preceding test case that had executed with unexpected errors, which often results in corruption of the CAIS-A predefined node structure. In such instances, a subsequent modified execution should be run following the restoration of the frame.dat file (refer to CIVC-OG for file restoration steps). In the following execution, remove the test class that generates the initial unexpected error from the selection list. This will ensure that no side effects are caused by its execution. If the offending error is caused by a test case from the same test class in which other test cases receive suspect errors, it will be necessary to stub out the offending test case code.

3.4.4 Multiple Stack Framework

The CIVC Phase III Framework is a large hypertext-based stack that is approximately eight megabytes in size. As such, it may exhibit slow execution performance with respect to a few of its functional capabilities (e.g., keyword searches). For future stack development activities, it might be beneficial to examine the feasibility of a multiple stack Framework. Such a design might result in a product having better performance characteristics; however, the complex links established between objects may preclude the ability to develop an efficient multiple stack Framework.

3.4.5 WBS Granularity

The work breakdown structure (WBS) is a hierarchical listing of the units of work in a project. The WBS is used for tracking the amount of effort expended and for scheduling the units of work. Overly detailed WBSs introduce an unneeded overhead burden, while insufficiently detailed WBSs obscure cost elements of interest to the project manager.

A detailed work breakdown structure (WBS) is required for accurate cost tracking. Accurate historical cost information is the best basis for estimation of future costs of similar activities. The CIVCWG recommended in June, 1989 that SofTech implement a more detailed cost accounting system to enable more accurate prediction of Phase III development costs and productivity.

Development of a more detailed WBS for Phase III, however, was too specific in many instances and did not provide a general task identifier for each product to account for unforeseen activities. Occasionally, task activities would occur for which there was no identifier. Additionally, many of the

same task activities had different task identifiers due to the position of the person performing the task (e.g., software engineer versus systems consultant). For future reference, while it is ideal to provide detailed visibility into each task, the WBS should not be structured to the level of identifying the position of the person performing the task. Experience in Phase III has shown that often the person performing a specified task was not of the title designated for the activity. Also, staff profiles change throughout the development cycle, creating further problems identifying appropriate charge numbers. Finally, a number of general task identifiers should be created for each discrete development product to accommodate any unexpected activities related to the work.

4. CIVC PHASE IV MAINTENANCE

4.1 CIVC Phase IV Maintenance Activities

CIVC Phase IV is the designated maintenance activity phase for the CIVC Phase III product. As the Ada Joint Program Office has decided not to fund the Phase IV activity, there will be no maintenance support for the Phase III products (as well as the CAIS-targeted Phase I products). Users of the CIVC products should be advised that they will be required to support any maintenance or desired modifications to the products.

5. SUGGESTIONS FOR TECHNOLOGY TRANSFER

5.1 Technology Transition

During the course of the CIVC project, the program developers have successfully applied the emerging hypertext technology and intelligent test selection process to produce sophisticated on-line documentation systems and a test suite that yields optimum test coverage. Being a large scale software development project, CIVC shares many things in common with other software development programs. Namely, these items are 1) the ability to establish traceability between the various phase products, 2) a complete and easily accessible documentation system, and 3) a test plan that provides optimal test coverage when exhaustive test coverage is not feasible. Fortunately, with the advent of hypertext technology, the first two aforementioned items can readily be accommodated by creating hypertext stacks specific to the project. Regarding the third item above, designing and implementing a test plan that provides the user with a high degree of confidence in large-scale software products can be attained by applying intelligence to the test selection process.

5.1.1 Hypertext Technology

Hypertext systems are relatively new software products that support the development of unique information environments. These systems are developed by using very high level programming commands to store object information and also information on the relationships between objects in as much or as little detail is required. Completed hypertext applications are referred to as stacks. Users can execute, or launch, the stacks to quickly and easily access products and related information. Relationships between objects (and sub-objects) are established by creating sticky text links between any two pieces of data. The sticky text capability allows one to quickly access information in one object while viewing related information in another object. For this reason, it is especially useful for navigating through numerous volumes of data.

Extending the concept of links to another level, hypertext systems could support user transitions by linking the discrete task activities for each designer/developer/reviewer/tester (i.e., "What do I need to work with next?" problem is resolved).

The CIVC program has successfully implemented hypercard project stacks that assist the user in traceability analysis and documentation review. Additionally, as part of the CIVC Framework development activities, stack administration scripts have been created that automatically populate the stack

with object data and automatically regenerates stacks from data files. These script utilities can be used to support the development of hypercard framework stacks for the Ada 9X and ACEC programs, as well as other software development projects having immense documentation and traceability requirements.

5.1.1.1 Ada 9X

Using hypertext technology to support the Ada 9X program would provide invaluable assistance to both the developers and users of the new language specification. The traceability mechanisms inherent in hypertext systems would allow developers to link every aspect of Ada 9X to all related items in MIL-STD-1815A, in addition to any other objects, including itself. This complete traceability would be the most important feature that hypertext systems could provide to the Ada 9X program. Typical object entities for ACEC might include:

- Language Specifications:
 - 1815A Language Reference Manual
 - Ada 9X Language Reference Manual
- Related Documentation
 - Comparison information on the two language specifications
 - MIL-STD 1838A
 - MIL-STD 2167
- Vendor Data
 - Manufacturers supporting 9X compilers
 - Manufacturers supporting 1815A compilers
 - ACEC performance results

5.1.1.2 Ada Compiler Evaluation Capability (ACEC)

An ACEC Hypertext Framework product could easily provide on-line access to ACEC products and related documentation. Object entities for ACEC might include:

- Test Code / Test Results / Test Analyses

- Version Description Document
- ACEC Reader's Guide
- ACEC User's Guide
- Structural Data (Test Groups, Scenarios, etc.)
- Support Documentation:
 - 1815A Language Reference Manual
 - Compiler Vendor Data
 - Comparison Information

A hypertext system would be ideally suited for identifying specific ACEC group tests to apply rather than the complete test suite. For example, one could easily access Task Group performance tests for assessing pre-emptive scheduling needs. Other applications for hypertext systems could be an ACEC training course. An ACEC tutorial could be developed that uses a non-supervised, user-guided approach to understanding and using the ACEC test suite. The material content of the tutorial could address all phases (environment setup, performance testing, assessor testing, and analysis) of the ACEC. A captive scenario example could be developed that would guide the user through all aspects of the ACEC test suite

5.1.2 Test Selection

The test selection methodology employed by the CIVC developers has a high potential for reuse on other software development contracts. Although the test objective selection algorithm implemented for CIVC2 is specific to CAIS-A validation activities, its parameters can be modified to accommodate other programs. Additionally, because the approach supports weighting the various parameters, future test developers will be able to specify the critically of the selection factors. Refer to section 3.1.4.1 Test Objective Interface Selection Algorithm for detailed information on the behavioral characteristics of the selection process.

Two programs under the auspices of the AJPO that can benefit from the CIVC developments in test selection domain are the Ada Compiler Evaluation Capability (ACEC) and the Portable Common Interface Set (PCIS).

5.1.2.1 ACEC Test Selection

The ACEC program is an evaluation suite comparable to Europe's Ada Evaluation System (AES), in which vendors exercise the suite against their Ada compiler implementations to assess operational performance

characteristics. Three particular areas where the intelligent test selection could be applied to ACEC activities are:

- optimization software can be applied to current ACEC performance coverage to identify strong and weak areas
- optimization software can be used to strengthen weak areas in future phases of ACEC
- user-defined variable weight selection criteria for identifying prioritized list for future performance/assessor test implementations.

5.1.2.2 PCIS Test Selection

The PCIS program is based on the proposed merger of two interface standards, the European Computer Manufacturer's Association (ECMA) Portable Common Tools Environment (PCTE) and the DoD's CAIS-A. A critical design driver in the PCIS program is to support ~90% of the code developed for these two independent standards in the future PCIS implementation. Current thought on the design of PCIS appears to favor a dual-layered PCIS standard, with ECMA PCTE providing the direct host interface and CAIS-A providing indirect host interface via the ECMA PCTE. This approach would allow tools already hosted on PCTE to operate without modification, as they can bypass the CAIS-A layer. Additionally, tools requiring CAIS-A specific services could be developed without having to interface to the PCTE.

The CIVC2 test suite could easily be adapted to the CAIS-A layer of a PCIS implementation. The selection criteria could be modified to emphasize test coverage in areas deemed most critical. Supplemental test cases could be developed to support the additional ~10% functionality expected by the CAIS-A layer of PCIS. The PCTE layer of PCIS would also have the need for validation test activities, although selection methodology would be more applicable for reuse than the actual test cases for this layer.

6. SUMMARY

Phase III of CIVC produced the following products:

- Beta Test Suite Prototype
- Test Suite of 550 test cases
- CAIS_Pragmatics test
- Static Semantics test
- Test Administrator CAIS-A tool
- Validation suite development methodology
- Hypertext traceability framework

The coverage of the Phase III test was

- 55% of the principal CAIS-A interfaces
- 47% of the exception/condition code pairs
- 68% of the CAIS-A packages

Lessons learned were presented in the following areas:

- Test objective development process
- Test case design
- Test suite errors
- Framework stack design
- Work breakdown structure granularity

The Phase III test suite reached its target of delivering 550 test cases at an exceptional level of productivity.

APPENDIX A NOTES

A.1 Acronyms

APSE	Ada Programming Support Environment
CAIS	Common APSE Interface Set
CAIS-A	Common APSE Interface Set Revision A
CI	Configuration Item
CIVC	CAIS Implementation Validation Capability Phase 1
CIVC2	CAIS Implementation Validation Capability Version 2
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSU	Computer Software Unit
DOD	Department of Defense
TA	Test Administrator
VDD	Version Description Document