DTIC FILE COPY

(2)

AFWAL-TR-85-1016 VOL IV

AD-A206 394

# EVALUATION & VALIDATION (E&V)

## TEAM PUBLIC REPORT
## Volume IV

RAYMOND SZYMANSKI
E&V TEAM CHAIRMAN
AVIONICS LABORATORY
AIR FORCE WRIGHT AERONAUTICAL LABORATORIES
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

January 1989

Interim Technical Report for Period

30 June 1987 - 31 December 1988

**APPROVED FOR PUBLIC RELEASE**
**DISTRIBUTION UNLIMITED**

PREPARED FOR:

ADA* JOINT PROGRAM OFFICE

3D139 (FERN ST/C107) PENTAGON

WASHINGTON, D.C. 20301

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

Raymond Szymanski

Raymond Szymanski
Program Manager

Date 20 January 1989

FOR THE COMMANDER:

Charles H Krueger Jr

CHARLES H. KRUEGER, JR
Director
System Avionics Division
Avionics Laboratory

02 FEB [illegible]
Date

Unclassified
SECURITY CLASSIFICATION OF THIS PAGE

# REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

| | |
|---|---|
| 1a. REPORT SECURITY CLASSIFICATION<br>UNCLASSIFIED | 1b. RESTRICTIVE MARKINGS |
| 2a. SECURITY CLASSIFICATION AUTHORITY | 3. DISTRIBUTION/AVAILABILITY OF REPORT<br>Approved for Public Release;<br>Distribution Unlimited |
| 2b. DECLASSIFICATION/DOWNGRADING SCHEDULE | |
| 4. PERFORMING ORGANIZATION REPORT NUMBER(S)<br>AFWAL-TR-85-1016, Vol IV | 5. MONITORING ORGANIZATION REPORT NUMBER(S) |
| 6a. NAME OF PERFORMING ORGANIZATION<br>Air Force Wright<br>Aeronautical Laboratories; 6b. OFFICE SYMBOL *(If applicable)*<br>AFWAL/AAAF-3 | 7a. NAME OF MONITORING ORGANIZATION |
| 6c. ADDRESS *(City, State, and ZIP Code)*<br>Wright-Patterson Air Force Base,<br>OH 45433-6543 | 7b. ADDRESS *(City, State, and ZIP Code)* |
| 8a. NAME OF FUNDING/SPONSORING ORGANIZATION<br>Ada Joint Program Office; 8b. OFFICE SYMBOL *(If applicable)* | 9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER |
| 8c. ADDRESS *(City, State, and ZIP Code)*<br>3D139 (Fern Street/C107) Pentagon<br>Washington, D.C. 20301 | 10. SOURCE OF FUNDING NUMBERS<br>PROGRAM ELEMENT NO. 63226F; PROJECT NO. AJPO; TASK NO 28; WORK UNIT ACCESSION NO. 53 |

11. TITLE *(Include Security Classification)*
Evaluation and Validation (E&V) Team Public Report, Volume IV

12. PERSONAL AUTHOR(S)
Raymond Szymanski, E&V Team Chairman

| 13a. TYPE OF REPORT | 13b. TIME COVERED | 14. DATE OF REPORT *(Year, Month, Day)* | 15. PAGE COUNT |
|---|---|---|---|
| Interim | FROM ~~1 SEP 87~~ TO ~~30 NOV 88~~ | 1989 January | 184 |

16. SUPPLEMENTARY NOTATION

| 17. COSATI CODES | | | 18. SUBJECT TERMS *(Continue on reverse if necessary and identify by block number)* |
|---|---|---|---|
| FIELD | GROUP | SUB-GROUP | Ada, Evaluation, Validation, Common APSE Interface Set (CAIS). [illegible]<br>Ada Programming Support Environment |
| 09 | 02 | | |

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

The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components and to determine their conformance to applicable standards (e.g., DOD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) task is to provide a focal point for addressing the need by:

(1) Identifying and defining specific technology requirements;
(2) Developing selected elements of the required technology;
(3) Encouraging others to develop some elements; and
(4) Collecting information describing elements which already exist.

| | |
|---|---|
| 20. DISTRIBUTION/AVAILABILITY OF ABSTRACT<br>☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS | 21. ABSTRACT SECURITY CLASSIFICATION<br>Unclassified |
| 22a. NAME OF RESPONSIBLE INDIVIDUAL<br>RAYMOND SZYMANSKI | 22b. TELEPHONE *(Include Area Code)*<br>(513) 255-5648; 22c. OFFICE SYMBOL<br>AFWAL/AAAF-3 |

DD Form 1473, JUN 86 *Previous editions are obsolete.* SECURITY CLASSIFICATION OF THIS PAGE
Unclassified

## TABLE OF CONTENTS

SECTION 1 - Project Technical Summary . . . . . . . . . . . . 1

APPENDIX A - Ada Compiler Evaluation Capability (ACEC) Questions and Answers . . . . . . . . . . . . . . . A-1

APPENDIX B - Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Reference System Flyer . . . . . . . . . . . . . . . . . . . . . B-1

APPENDIX C - Evaluation and Validation (E&V) Reference System Presentation Slides and Narrative . . . . . C-1

APPENDIX D - Final Report: 1988 USAF-UES Summer Faculty Research Program (Mike Burlakoff) . . . . . . . . . D-1

APPENDIX E - Tools and Aids Document, Version 2.0 . . . . . . . E-1

APPENDIX F - Ada Environments Working Group Presentation . . . F-1

APPENDIX G - Minutes of the Evaluation and Validation (E&V) Team Meeting December 1987 . . . . . . . . . . . . G-1

APPENDIX H - Minutes of the E&V Team Meeting March 1988 . . . . H-1

APPENDIX I - Minutes of the E&V Team Meeting May 1988 . . . . . I-1

APPENDIX J - Minutes of the E&V Team Meeting September 1988 . . J-1

# SECTION 1

# PROJECT TECHNICAL SUMMARY

## 1.0 INTRODUCTION

This report is the fourth in a series of technical reports to be published by the Evaluation and Validation (E&V) Team. The purpose of the E&V Public Report is to provide an overview of the many technical accomplishments of the E&V Team during an appropriate time frame. This fourth report contains information resulting from E&V activities during September 1987 to December 1988 which is being made available for public review and comment. Contents of this report reflect an observation of the E&V Team progress during the specified time frame and should not be viewed as final representations of the technology being developed.

## 1.2 Background

In June 1983 the Ada Joint Program Office proposed the formation of the E&V Task and a tri-service APSE E&V Team, with the Air Force designated as lead service. In October 1983 the Air Force officially accepted responsibility as lead service on the E&V Task.

The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components and to determine their conformance to applicable standards (e.g., DOD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government sponsored, professional society sponsored, or private effort. The purpose of the Evaluation and Yalidation (E&V) Task is to provide a focal point for addressing the need by:

(1) Identifying and defining specific technology requirements,

(2) Developing selected elements of the required technology,

(3) Encouraging others to develop some elements, and

(4) Collecting information describing elements which already exist.

This information will be made available to DoD components, other government agencies, academic institutions, and industry.

## 1.3 E&V Meetings

E&V Team meetings are held on a quarterly basis. For the period covered by this report quarterly meetings were conducted on the following dates: 1-3 September 1987, 2-4 December 1987, 2-4 March 1988, 24-26 May 1988, and 7-9 September 1988.

## 1.4 E&V Team Organization

In order to coordinate all of the activities to be accomplished within the E&V Task, the E&V Team is partitioned into six working groups. The identification of these working groups, and their associated areas of responsibility, are delineated in the following sections. These working groups are subject to change during the life of the E&V Task. Each working group has a designated Chairperson and Vice-Chairperson. It is the responsibility of each working group Chairperson to coordinate the activities of the working group with the E&V Team Chairperson. In addition, each working group Chairperson is required to brief the status of the respective working group at every E&V Team meeting.

### 1.4.1 Directional Management Working Groups

#### 1.4.1.1 E&V Requirements Working Group (REQWG)

The REOWG is responsible for performing the following tasks:

- Maintain an E&V Requirements Document against which the E&V Reference Manual will be developed.

- Provide analysis of requirements in the E&V Requirements Document to determine their adequacy, completeness, traceability, testability, consistency, and feasibility.

- Identify issues which may impact the development of E&V technology.

- Provide recommendations for acquisition of E&V tools and aids through the development of an E&V Tools and Aids Document.

- Prepare position papers through the duration of the E&V Task which address issues on E&V requirements.

#### 1.4.1.2 E&V Standards Evaluation and Validation Working Group (SEVWG)

The SEVWG is responsible for performing the following tasks:

- Recommend specific areas of consideration for standards related to future evaluations and validations.

- Emphasize study on the CAIS.

- Review the development of the CAIS and identify areas of possible concern to E&V.

- Provide presentations to the E&V Team on the CAIS.

- Prepare position papers throughout the duration of the E&V Task which address particular aspects of the CAIS as relevant to E&V.

### 1.4.1.3 E&V Coordination Working Group (COORDWG)

The COORDWG is responsible for performing the following tasks:

- Develop a Technical Coordination Strategy Document which will:
  - identify related technical efforts;
  - identify relationships between the E&V Task and each of the related tasks;
  - identify areas of mutual benefit to the tasks;
  - identify impact of schedules;
  - identify level of coordination required;
  - identify issues which require resolution to the mutual benefit of
- Identify professional organizations which are technically related to the E&V effort.
- Develop a Public Coordination Strategy Document which provides an approach as to how such public coordination will be performed.
- Maintain and distribute a set of E&V viewgraphs and corresponding text to allow E&V Team members to present the status of the E&V Task at public meetings.
- Prepare E&V status reports for publication in related journals and newsletters dissemination at related conferences.
- Catalog all issues related to the E&V effort.
- Develop and maintain an E&V Project Reference List.

## 1.4.2 Technical Management Working Groups

### 1.4.2.1 E&V Ada Compiler Evaluation Capability Working Group (ACECWG)

The ACECWG is responsible for performing the following tasks:

- Provide a formal interface between the Ada community and the ACEC effort.
- Evaluate and critique aspects of the technical approach being employed on the ACEC effort.
- Evaluate and critique selected ACEC deliverables.
- Discuss and provide feedback on issues critical to the ACEC.

### 1.4.2.2 E&V CAIS Implementation Validation Capability Working Group (CIVCWG)

The E&V CIVCWG is responsible for performing the following tasks:

- Provide technical expertise to E&V chairman and team for review of CIVC contractors' products and activities.
- Provide to E&V chairman and CIVC project engineer recommendations regarding validation of CAIS.
- Coordinate regularly and closely with SEVWG concerning validation of DoD Standard 1838 implementations.

### 1.4.2.3 E&V Technology Classification Working Group (CLASSWG)

The CLASSWG is responsible for performing the following tasks:

- Serve as focal point for analysis of Reference System (Reference Manual and Guidebook).
- Solicit information and recommendations regarding E&V technology.
- Classify E&V technology.
- Aid in the technology transition of the Reference System.
- Delineate whole APSE issues.
- Recommend new areas of investigation.

## 1.5 Conclusion

This E&V Public Report is being made available by the E&V Team in order to solicit comments from those individuals who are not actively involved in the E&V Task. All comments should be addressed to:

Raymond Szymanski
AFWAL/AAAF-3
Wright-Patterson AFB, Ohio 45433-6543
(SZYMANSK@AJPO.SEI.CMU.EDU or
EV-TEAM@AJPO.SEI.CMU.EDU)

Ada Compiler Evaluation Capability (ACEC)
Questions and Answers
Version 1.0 --- 9/3/1988

## General Issues

1. What is the primary purpose of the ACEC?

The ACEC is designed to help Ada compiler users evaluate the performance of their compilers with respect to execution speed, compilation speed, and code expansion size.

2. Who is intended to be the primary user of the ACEC?

The ACEC test suite is being designed for use by Department of Defense Program Offices and their contractors. Other interested parties, however, may request the ACEC.

3. Will the ACEC be used by the Ada Validation Facilities to "include compiler efficiency in validating procedures"?

The House Appropriations Committee Report (100-681), dated June 10, 1988, directs the Ada Joint Program Office to "include compiler efficiency in validating procedures". The ACEC is currently under consideration for use in the validating procedures.

4. What are the current restrictions on distribution of the ACEC?

The ACEC will be distributed by the Data and Analysis Center for Software (DACS) in accordance with current rules and regulations guiding the export of computer software.

## Managerial Issues

1. Who is the prime contractor on the ACEC effort?

The prime contractor on the ACEC effort is Boeing Military Airplanes (BMA) in Wichita, KS.

2. When will the ACEC be ready for release?

The first version of the ACEC will be available for release in Aug 88.

3. How many versions of the ACEC are planned?

There will be 2 versions of the ACEC test suite. The first version will be released in Aug 88. After the first release, feedback from the user community will be gathered and improvements will be incorporated into the existing suite. The resultant second version will be released in May 89.

4. How can my organization obtain a copy?

It is currently planned that the Data and Analysis Center for Software (DACS) in Rome, NY will handle distribution of the ACEC test suite, tools, and guides. There will be a nominal fee for reproduction and distribution to be paid by the requesting organization. In order to obtain a copy of the ACEC from DACS, an organization must have a DLSC/Qualified Contractor Access List number assigned by the following agency:

United States/Canada Joint Certification Office
Defense Logistics Services Center
ATTN: DLSC-FBA
Federal Center
Battle Creek, MI 49017-3084
(800) 352-3572

An order form for DACS software can be obtained by writing or calling:

Data and Analysis Center for Software
RADC/COED
Bldg 101
Griffiss AFB, NY 13441-5700
Attn: Document Ordering
(315) 336-0937

5. What documentation is available for the ACEC user?

There are three documents which will be included with the ACEC test suite, the name and purpose of each is as follows:

ACEC User's Guide - This document will help the ACEC user set up and run (collect results) on the benchmarks. Lessons learned on previous systems will be included in applicable sections (in order to facilitate operation on other systems).

ACEC Reader's Guide - This document will help the ACEC user interpret the results of the test suite. Test suite content and theory behind timing loop and MEDIAN are discussed.

ACEC Version Description Document - This document will provide the ACEC user with descriptions of individual tests. A number of useful appendixes to aid the user in locating and using tests are also included.

6. What are the opportunities for user feedback into the effort?

Problem reports and improvement suggestions on the ACEC will be accepted after release has begun. The reports will be forwarded to the Air Force Wright Aeronautical Laboratories (AFWAL) with a carbon copy going to BMA. The exact problem report format and filing procedures will be discussed in the ACEC Reader's Guide and the ACEC User's Guide.

## Technical Issues

1. What are the functional requirements of the ACEC?

The functional requirements of the ACEC are as follows:

a. Compare the performance of several Ada compiler implementations. The benchmarks shall permit an analysis and comparison of systems' performance. The ACEC may also be used to observe and evaluate the performance of a single system.

b. Isolate the strong/weak points of a specific system, relative to others which have been tested. Weak points, once isolated, can be enhanced.

c. Determine that significant changes were made between releases of a specific compilation system.

d. Predict the performance of differing Ada design approaches.

All ACEC tests, tools, and guides will be geared toward supporting the objectives stated in these functional requirements.

2. What aspects of compiler performance will the ACEC cover?

The ACEC will directly measure execution time and code size for the benchmark under test. If desired, the ACEC user can construct a test harness to measure compilation speed (lines of code/sec). The MEDIAN analysis tool can be used to compare results which are numerically quantifiable. A classification taxonomy of ACEC tests is attached.

3. How portable will the ACEC be?

Although portability is a major design goal of the ACEC, it is recognized that this can only be achieved to a limited extent. While portability might be enhanced by restricting all tests to a subset of the Ada language, this was not deemed to be an acceptable alternative since many system dependent and optional features are needed for mission critical computer applications. In addition, there is no guarantee of consistency even for those language features which are tested under the ACVC, since portions of the LRM are subject to the implementor's interpretation.

From our experience porting the ACEC test suite over to different systems, the following factors were found to impact portability:

a. Compiler System Implementation Dependencies

EXAMPLES
-- Differing floating point representations.
-- Differing task scheduling implementations.
-- Differing package Calendar implementations.

b. Unsupported Compiler System Features

<u>EXAMPLES</u>
-- Unimplemented language features (LRM Chapter 13).
-- Unimplemented attributes (LRM Appendix A).
-- Unimplemented pragmas.
-- Limited target processor I/O capabilities.

c. Capacity limitations of a given compiler

<u>EXAMPLES</u>
-- Lines of source code in a file may be limited.
-- Stack space allocation may be limited.
-- Limited target processor memory space.

d. Compilation system implementation errors

<u>EXAMPLES</u>
-- A compilation system may have errors which are not addressed by ACVC tests in the current validation suite.

Despite these obstacles, steps have been taken to insure that the procedure necessary to port the ACEC between different compiler implementations is addressed in the ACEC guides. In addition, the effort required to accomplish a port will be minimized to the greatest extent possible.

4. What steps are being taken to insure portability?

In order to overcome the difficulties described in the previous response, the following paragraphs describe aspects of the approach being taken to enhance portability of the test suite:

a. The use of predefined types Integer and Float is avoided.
b. No database interface is used.
c. Only 16 bit integers are required.
d. Order dependences in package elaborations are avoided.
e. Tests which use system dependent or optional features are identified in the manuals. The manuals will deal with a number of different approaches to overcome specific observed problems.
f. System dependent code will be clearly identified in the source code and the Version Description Document.
g. Several approaches to performing a given measurement are provided with the test suite. For example, in order to measure the size of the code generated for the feature under test, it is necessary to either use Ada features which are potentially unsupported (like 'Address) or an assembly language function. Files which contain multiple approaches for measuring code size are available and can be inserted into each benchmark using the INCLUDE preprocessor.
h. A set of portable math routines is provided. System dependent math data (float) is isolated to a small package. This data is then used by the portable math routines.

5. What analysis tools are provided for use with the test suite?

The primary analysis tool is called MEDIAN. The purpose of MEDIAN is to statistically compare performance measurements collected from the execution of the ACEC test suite on several target systems. The inputs to the tool are either the time or size measurements produced by running the benchmarks. The FORMAT program prepares this raw data for input to MEDIAN.

The MEDIAN outputs are:

a. Matrix of raw timing (or size) measurements (system vs problem).
b. Histogram of residual values. (Residual values give an indication of whether the problem executed faster, slower or about as expected. This can be used to determine the language features a compiler produces the most efficient code for, relative to the other systems under test).
c. A summary of the statistical data (minimum, maximum, interquartiles, and median) for each problem (across systems under comparison).
d. A system factor for each compiler under test. This is a measure of how well a specific compiler has performed relative to the other systems under test.
e. A problem factor for each problem. This is a measure of the relative difficulty of each problem under test.

6. What approach is being taken to insure timing loop (execution time measurement) integrity?

Since the integrity of the timing loop code (in which the code under test will be embedded) determines the validity of the ACEC generated results, it is essential that the code under test be accurately measured. The steps taken to insure timing loop integrity fall into two categories:

i. To measure time accurately, four steps are taken:

1) As part of program initialization, the achievable clock resolution is computed using a software vernier. This lower bound on measurements is used to compute a minimum time to execute a test problem and ensure that errors due to quantization will be less than a pre-specified error tolerance. If the number of repetitions of a problem is not sufficient, the number of repetitions is increased.

2) As part of program initialization, the ACEC measures clock jitter (random variations in the clock) and computes a minimum time to require execution so that the effects of jitter should be less than error tolerances.

3) After the number of repetitions of a problem is computed, it is executed for several cycles with this number of repetitions. The variance observed is displayed.

4) Statistical confidence levels are computed and the number of cycles increased if not met. (Failures are flagged).

ii. Steps are taken to ensure that each problem is called so that it is accurately measurable. Test problem code:

1) Should not be dead.
2) Should not be susceptible to code motion outside of the timing loop.
3) Should follow the same path on each repetition.
4) Should not be unduly foldable.

7. Is the Operating System's computed CPU time or elapsed time measured? Why?

The ACEC provides the user the option of using either time. However, the default is elapsed time because:

a. Calendar package can be used, thereby enhancing portability.
b. Elapsed time is the appropriate metric for embedded targets.
c. Calls to measure CPU time are system dependent.
d. CPU time can be measured in separate ways on different systems.

Despite this, there are significant advantages to using CPU time on multiprogramming systems. In this case, the timing loop code can be modified to make a system dependent OS call to measure CPU time; the modified code can then be inserted into individual benchmark files using the INCLUDE preprocessor.

8. What types of Ada application benchmarks will the ACEC have?

The application profiles included in the ACEC will be representative of classical algorithms, use of Ada in (current) practice, and the ideal use of Ada in MCCR applications.

-- Classical benchmarks (Whetstone, Dhrystone, etc)
-- Communications Protocol
-- Aircraft Simulation
-- AI algorithm
-- Radar tracking algorithm
-- Kalman filter algorithm

Other application programs are currently being sought to improve the suite coverage.

9. Does the ACEC take the underlying processor performance into account? How?

The ACEC measures system performance - both target hardware and software factors contribute to overall performance.

10. How is code expansion size measured? Is this method portable?

The measurement of code size is an extremely system dependent function. Because no one approach will work on all systems, the basic approaches to measuring this attribute can be selected; they are as follows:

-- Use of labels with address attribute and unchecked conversion.
-- Write an assembly routine to return the calling address.

Once an approach is chosen, it can be inserted into the benchmark files using the INCLUDE preprocessor.

11. What documentation will be used to describe the ACEC development approach? Will this be available?

The following documents are being produced under the ACEC contract:

| | |
|---|---|
| Configuration Management Plan | (DI-E-3108/T) |
| Software Development Plan | (DI-MCCR-80030/T) |
| Software Requirements Spec | (DI-MCCR-80025/T) |
| Software Product Spec | (DI-MCCR-80029/T) |
| Software Test Plan | (DI-MCCR-80014/T) |
| Software Test Procedures | (DI-MCCR-80016/T) |
| Abstract of New Technology | (DI-A-3028B) |
| Final Technical Report | (DI-3591A/T) |

These reports will be available through the Defense Technical Information Center (DTIC). The date of availability is still to be determined.

12. What documents have served as input to the development of the ACEC?

The following documents served as a point of reference for the development of the ACEC:

-- IDA Report on Preliminary ACEC
-- SEI Report: Factors Causing Unexpected Variations in Ada Benchmarks
-- ARTEWG Report: Catalogue of Runtime Implementation Dependencies
-- ARTEWG Report: First Annual Survey of Mission-Critical Application Requirements
-- AFIT Thesis: Using Ada in the Real-time Avionics Environment: Issues and Conclusions

13. How is the integrity of an ACEC benchmark established?

Each ACEC benchmark is checked against six different compilation systems which include:

| | | |
|---|---|---|
| a. | DEC Ada | (Vax Target) |
| b. | TeleSoft Ada | (Vax Target) |
| c. | AIMS Ada | (1750A Target) |
| d. | TLD Ada | (1750A Target) |

| | | |
|---|---|---|
| e. | Harris Ada | (self-target) |
| f. | Verdix Ada | (68020 target) |

If the test fails to compile or run correctly due to incorrect coding or problem representation, then the test will be withdrawn from the suite and corrected.

14. What are some reasons for disqualifying a test on a particular system?

There are several potential reasons why a benchmark may not run on a specific compilation system; these reasons include:

-- Incorrect implementation of that language feature.
-- Incorrect coding of the benchmark.
-- Optimization into null (may disqualify test only for a particular system).
-- The system does not support the feature which is being tested.

If a user encounters a problem with a particular ACEC test, they are encouraged to follow the procedures for filing a problem report (outlined in the User's Guide and Reader's Guide). Problem reports are necessary for correcting deficiencies in the test suite and making suggested improvements.

15. How much work will be involved in setting up the test suite? To run it? To analyze the results?

The level of effort involved in setting up the test suite is dependent on the maturity of the compilation system and the services provided by the target machine. For example, I/O may not be supported on a 1750A target unless the proper drivers exist; if they do not, they must be written. In addition, the math routines must be modified to compensate for differences in the floating point representation of the target processor as well. Examples of different approaches to modifying the math routines will be provided with the test suite. The set up time for the ACEC can run from less than a day (for a VAX) to two weeks (for a less sophisticated target computer).

The amount of work required to run the test suite will also vary widely between targets. On self hosted compilation systems, it is possible to compile and run the suite within 10 hours. On remote targets, however, downloading can take a significant amount of time (as can compilation).

The analysis of the test results is straight forward. The primary analysis tool will be MEDIAN which makes the task of analysis fairly automated. Some time will be saved if the compiler is self hosted since the FORMAT tool can be then used to automate the preparation of test results as input data for MEDIAN. A remote target generally requires more time for collection and preparation of data.

16. What support will be provided to the ACEC user?

No direct support (other than the guides) will be provided to the ACEC user if problems arise. If the ACEC user runs into a problem, then he/she can file a problem report according to the procedure outlined in the ACEC User's Guide. If the problem is determined to be valid, then it will be corrected in the next version of the test suite. An ACEC user may submit change requests which include new test problems. These test problems may be considered for incorporation into the second release of ACEC.

## ACEC Organization

Purpose:

(1) To explain the scope of the ACEC effort.
(2) To provide an index of what is covered by individual tests.

Top Level Classification:

I. Compile Time Efficiency

-- A test harness has been constructed to measure compile time efficiency for Unix and VMS. Users can construct other test harnesses as needed.

II. Execution Time Efficiency

-- Multiple Categories (see below).

III. Test for Existence of Language Features

-- Not specifically tested for; to be addressed by ACVC

IV. Code Size Efficiency

-- Benchmark Code Expansion Size
-- Run Time System Size (obtained from link map)
-- Application Size

V. Usability

-- Not specifically tested for.

VI. Capacity Tests

-- Not specifically tested for.

Intermediate Level Classification:

II. Execution Time Efficiency

A. Language Feature Efficiency

1. Required
-- Referenced by LRM section.

2. Implementation Dependent
-- Referenced by LRM section.
--- attributes (LRM Appendix A)
--- record representation clauses
--- interrupts
--- language interface
--- unchecked programming

B. Pragmas

   1. Predefined

   2. Implementation Defined

C. Optimizations

   1. Classical

   2. Effect of Pragmas

   3. Static Elaboration
      -- Aggregates
      -- Tasks

   4. Language Specific
      -- Habermann-Nassi transformation for tasking
      -- delay statement optimization

D. Performance Under Load

   1. Task Loading
      -- task creation
      -- task termination
      -- task abortion
      -- Dining Philosophers Problem
      -- task starvation
      -- task delay

   2. Levels of Nesting
      -- Static
      -- Dynamic

   3. Parameter Variation

   4. Declarations

E. Trade Offs

   1. Design Issues
      -- Order of Evaluation
      -- Default .vs. Initialized Records
      -- Order of Selection (rendezvous)
      -- Scope of Usage
         --- global
         --- local
         --- shared

   2. Coding Style Variation

F. Operating System Kernel Efficiency

1. Task Scheduling
2. Exception Handling
3. File I/O
4. Memory Management/Storage Reclamation
5. Elaboration
6. Run Time Checks
7. Interrupt Handling

G. Application Profile Tests

1. Classical
   -- Whetstone
   -- Dhrystone
   -- Ackerman's
   -- Computer Family Architecture
   -- Sort Variations
   -- Math Applications
   -- Livermore Loops
   -- Knuth Loops

2. Ada in Practice (taken from)..
   -- E-3A Simulator
   -- Navigation Algorithms
   -- Radar Tracking Algorithms
   -- Communication Algorithms
   -- Electronic Warfare
   -- Avionics

3. Ideal Ada
   -- AI applications
   -- Data Encryption

IV. Code Size Efficiency

A. Expansion Code Size

B. Run Time System Size

C. Executable File Size

-- New Documents Available --

# APSE E&V REFERENCE SYSTEM

The Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Reference System is a pair of documents developed, and periodically updated, by the APSE E&V Task, sponsored by the Ada Joint Program Office and led by the US Air Force Avionics Laboratory. The documents are entitled the "E&V Reference Manual" and the "E&V Guidebook."

APSE E&V Task Purpose -- The Ada community needs the capability to assess APSEs and their components, and to determine their conformance to applicable standards. The technology required to fully satisfy this need is extensive and largely unavailable. The purpose of the APSE E&V Task is to provide a focal point for addressing this need by (1) identifying and defining specific technology requirements, (2) developing selected elements of the required technology, (3) encouraging others to develop some of these elements, (4) collecting information describing existing elements, and (5) making E&V technology information available to government agencies, industry, and academia.

E&V Reference Manual -- The manual provides a framework for understanding APSEs and their assessment, and establishes common terminology. One chapter discusses an APSE as a whole and its assessment. Other chapters are indexes to APSE component characterization and assessment, organized by life cycle activities, APSE tool category, APSE function, and attribute to be assessed. An entry in an index consists of a description, cross references to other entries in the Reference Manual, and cross references to the "E&V Guidebook." The manual is intended to help a variety of users obtain answers to their questions. As a stand-alone document it is intended to help a user find useful information about index elements and relationships among them. In conjunction with the Guidebook, it is indended to help users find criteria and metrics for assessment of APSEs and their components.

E&V Guidebook -- The Guidebook provides descriptions of specific instances of assessment technology. These include evaluation (assessment of performance and quality) and validation (assessment of conformance to a standard) techniques. For each category of item to be assessed (e.g. compilation system, test system, whole APSE, etc.), there are brief descriptions of applicable tools and aids -- such as test suites, questionnaires, checklists, and structured experiments -- and references to primary documents containing detailed descriptions. The Guidebook also contains synopses of documents of general historical importance to the entire field of Ada environments and their assessment.

## E&V Task Products and Schedule

E&V Reference Manual -- Version 1.0 (March 1988); Version 1.1 (November 1988)
E&V Guidebook -- Version 1.1 (September 1988)
Ada Compiler Evaluation Capability(ACEC) test suite -- (August 1988)
CAIS Implementation Validation Capability(CIVC) tests -- (January 1989)

## MAILING LIST FOR E&V PRODUCTS

If you would like to receive instructions for obtaining the E&V Reference System documents and other E&V products as they becomes available, attach your business card or fill in your name and address and send to Mr. Raymond Szymanski, AFWAL/AAAF, Wright Patterson AFB, OH 45433-6543. .

Name ______________________________

Address ______________________________

______________________________

# APSE E&V REFERENCE SYSTEM

AdaJUG
30 November 1988
R. Szymanski

## OVERVIEW

- APSE E&V Task Background -- Need and Purpose
  -- Process and Products
- E&V Reference Manual -- Organization
  -- Example Usage
- E&V Guidebook -- Organization
  -- Example Synopses
  -- Example Tools and Aids
- Summary

## DEFINITIONS

**APSE — Ada Programming Support Environments**

**E & V — Evaluation and Validation**

**Evaluation — Assessment of Performance and Quality**

**Validation — Assessment of Conformance to a Standard**

## NEED FOR APSE E&V TECHNOLOGY

- **IMPORTANCE OF ENVIRONMENT DECISIONS**
  - Large, Critical Ada-based Systems
  - Major Investments for Software Developers
  - Major Influence on Software Maintenance

- **DIFFICULTY OF ASSESSING APSEs AND TOOLS**
  - APSEs are Complex Systems
  - Great Diversity of Choice and Viewpoints
  - Rapid Technological Change
  - Lack of Relevant Historical Data

## E&V TASK PURPOSE

To Providé a Focal Point for Addressing Community Need for E&V Technology — Assess APSEs and Components

ACTIVITIES

1) Identify and Define Requirements
2) Develop Selected Elements
3) Encourage Others to Develop Some
4) Collect Relevant Information
5) Disseminate Information

## E&V TASK PROCESS

| | |
|---|---|
| SPONSOR | Ada Joint Program Office |
| LEADER | Air Force Avionics Laboratory<br>Mr. Raymond Szymanski, Chair |
| TECHNICAL ADVISORS | The E&V Team — Goverment, Industry and Academia Representatives |
| CONTRACTORS | TASC — Technical Support and Reference System<br>Boeing — Ada Compiler Evaluation Capability (ACEC)<br>SofTech — CAIS* Implementation Validation Capability (CIVC) |

* Common APSE Interface Set — MIL-STD-1838

## E&V TASK PRODUCTS

E&V REFERENCE SYSTEM
- E&V Reference Manual — Mar 1988
- E&V Guidebook — Aug 1988

Ada COMPILER EVALUATION CAPABILITY -- Aug 1988

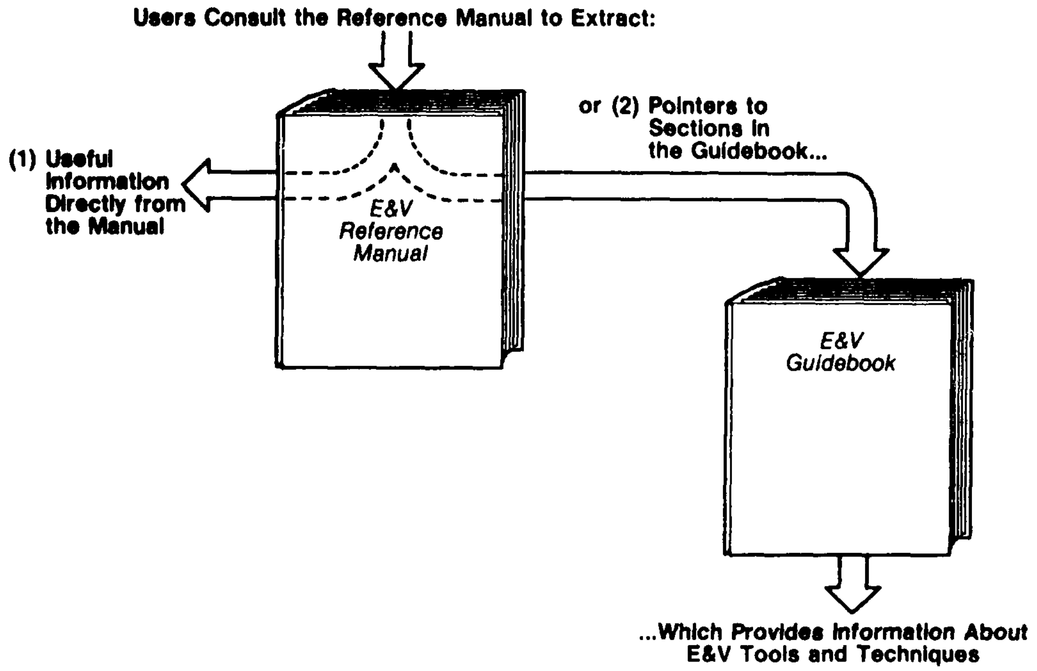
CAIS IMPLEMENTATION VALIDATION CAPABILITY — June 1989

---

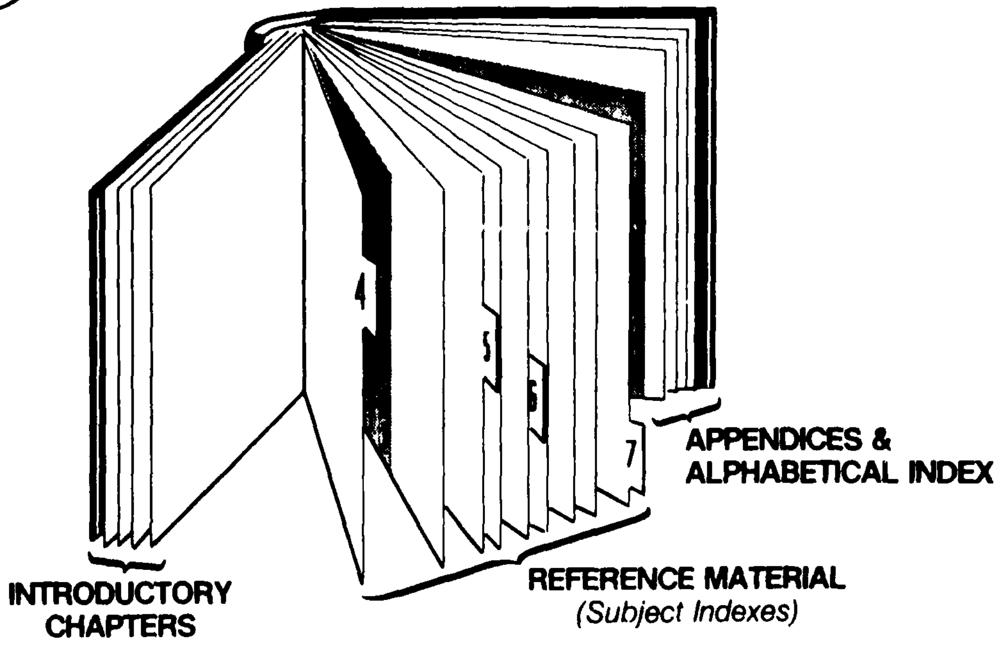
## OVERVIEW

- APSE E&V Task Background -- Need and Purpose
  -- Process and Products
- ⇨ E&V Reference Manual -- Organization
  -- Example Usage
- E&V Guidebook -- Organization
  -- Example Synopses
  -- Example Tools and Aids
- Summary

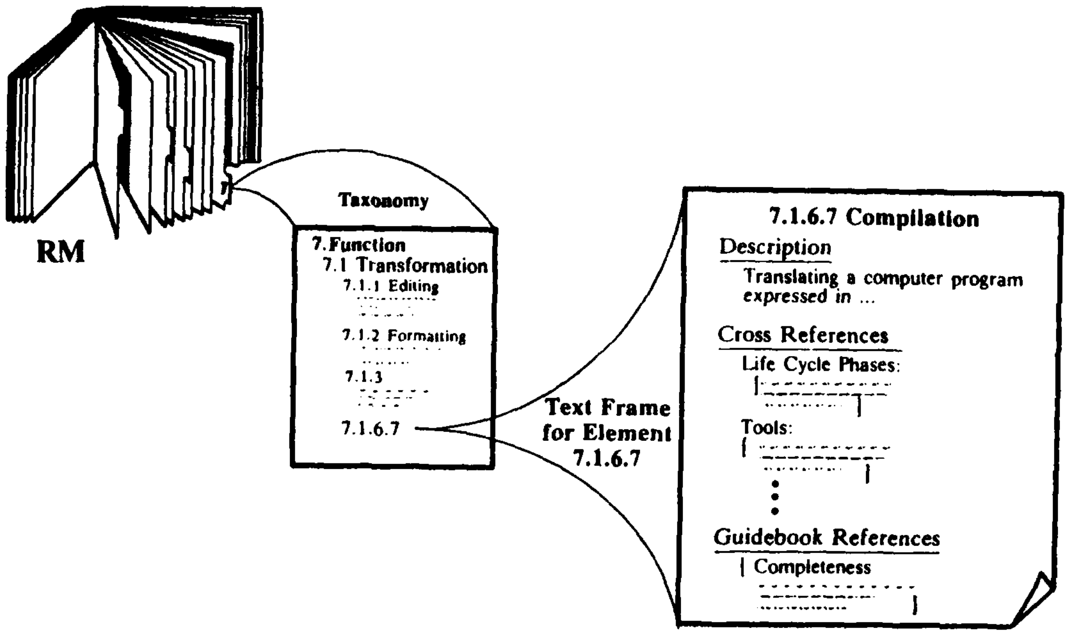
## USE OF THE REFERENCE SYSTEM


Users Consult the Reference Manual to Extract:
(1) Useful Information Directly from the Manual
or (2) Pointers to Sections in the Guidebook...
E&V Reference Manual
E&V Guidebook
...Which Provides Information About E&V Tools and Techniques

## REFERENCE MANUAL ORGANIZATION

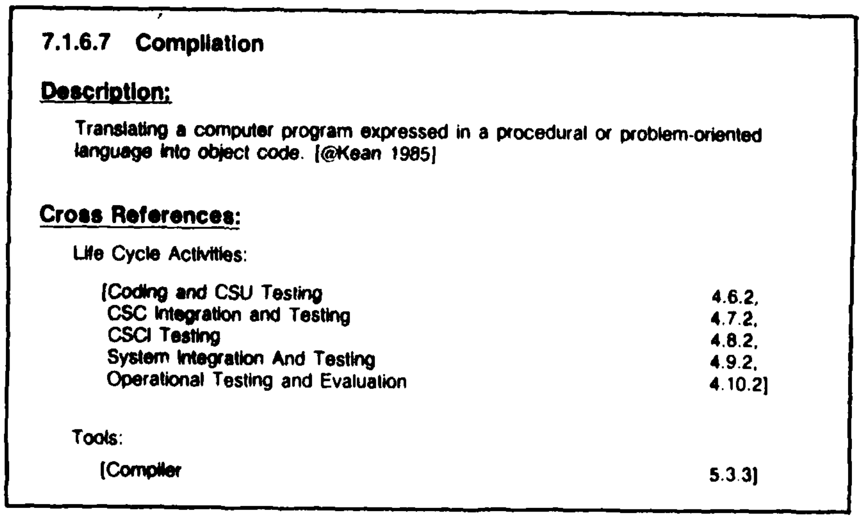

4
5
6
7
APPENDICES & ALPHABETICAL INDEX
INTRODUCTORY CHAPTERS
REFERENCE MATERIAL (Subject Indexes)

## INDEXES AND TEXT FRAMES



## TEXT FRAME EXAMPLE #3

(PAGE 1 OF 2)



**7.1.6.7 Compilation**

**Description:**

Translating a computer program expressed in a procedural or problem-oriented language into object code. [@Kean 1985]

**Cross References:**

Life Cycle Activities:

| | |
|---|---|
| [Coding and CSU Testing | 4.6.2, |
| CSC Integration and Testing | 4.7.2, |
| CSCI Testing | 4.8.2, |
| System Integration And Testing | 4.9.2, |
| Operational Testing and Evaluation | 4.10.2] |

Tools:

| | |
|---|---|
| [Compiler | 5.3.3] |

## TEXT FRAME EXAMPLE #3

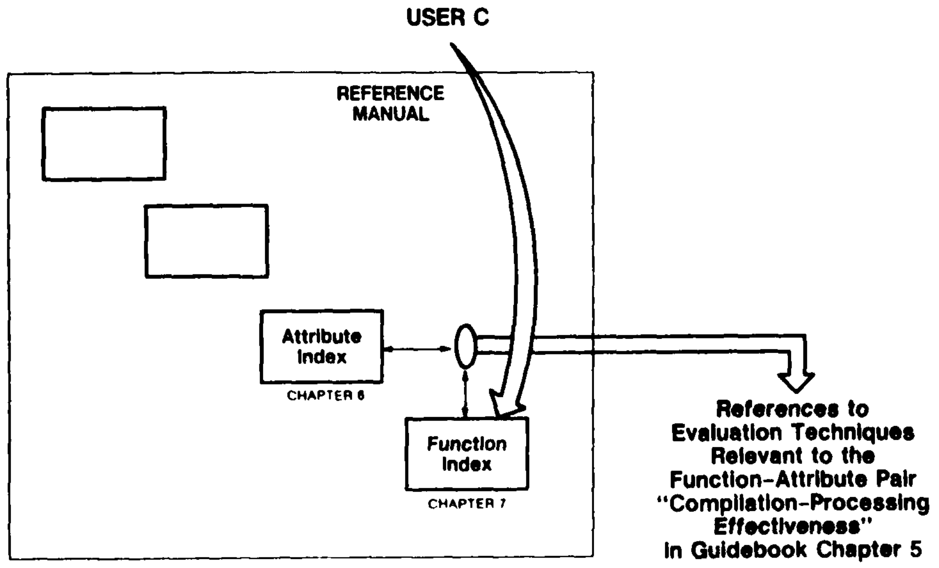
(PAGE 2 OF 2)

**Guidebook References:**

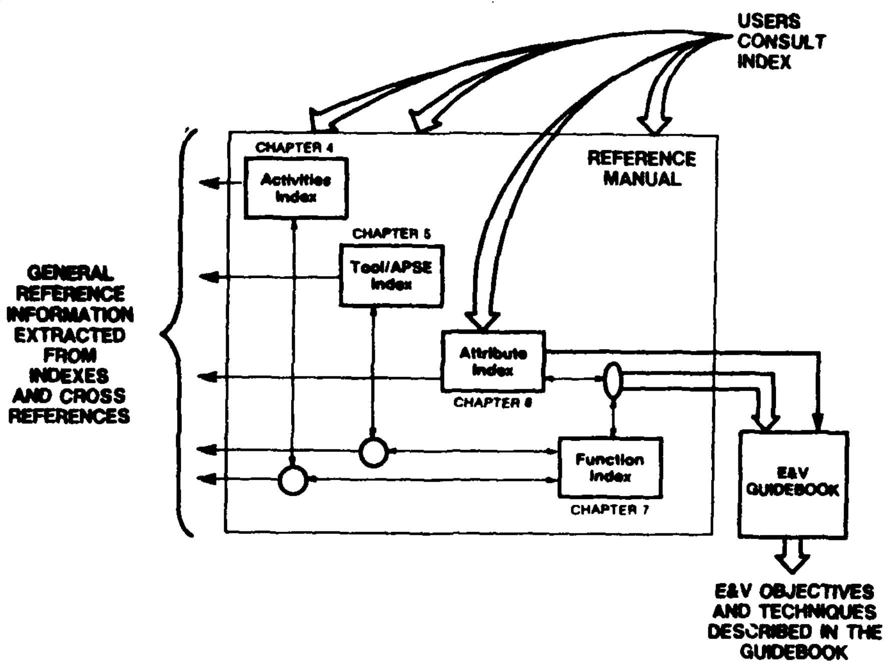
| | | | | |
|---|---|---|---|---|
| *[Anomaly Management | 6.4.2. | @GB: | ARTWEG Catalogue of Ada Runtime Implementation Dependencies | 5.10; |
| Capacity | 6.4.6. | @GB: | IDA Benchmarks | 5.2; |
| Capacity | 6.4.6. | @GB: | MITRE Benchmark Generator Tool | 5.6; |
| Completeness | 6.4.9. | @GB: | ACVC | 5.1; |
| Power | 6.4.21. | @GB: | Compilation Checklist | 5.8; |
| * Processing Effectiveness | 6.4.22. | @GB: | IDA Benchmarks | 5.2; |
| * Processing Effectiveness | 6.4.22. | @GB: | ACEC | 5.3; |
| * Processing Effectiveness | 6.4.22. | @GB: | PIWG Benchmark Tests | 5.4; |
| * Processing Effectiveness | 6.4.22. | @GB: | University of Michigan Benchmark Tests | 5.5; |
| * Processing Effectiveness | 6.4.22. | @GB: | ARTEWG Catalogue of Ada Runtime Implementation Dependencies | 5.10; |
| * Retargetability | 6.4.27. | @GB: | ARTEWG Catalogue of Ada Runtime Implementation Dependencies | 5.10; |
| * Storage Effectiveness | 6.4.31. | @GB: | IDA Benchmarks | 5.2; |
| * Storage Effectiveness | 6.4.31. | @GB: | ACEC | 5.3; |
| * Storage Effectiveness | 6.4.31. | @GB: | PIWG Benchmark Tests | 5.4; |
| * Storage Effectiveness | 6.4.31. | @GB: | ARTEWG Catalogue of Ada Runtime Implementation Dependencies | 5.10] |

*NOTE: Normally, the concern of evaluation is the quality of a tool. However, this evaluation technique focuses on the product of the tool rather than the tool itself.

## USAGE SCENARIO #3


USER C
REFERENCE MANUAL
Attribute Index
CHAPTER 6
Function Index
CHAPTER 7
References to Evaluation Techniques Relevant to the Function-Attribute Pair "Compilation-Processing Effectiveness" in Guidebook Chapter 5

## COMBINED VIEW



## OVERVIEW

- APSE E&V Task Background
  - -- Need and Purpose
  - -- Process and Products
- E&V Reference Manual
  - -- Organization
  - -- Example Usage
- ⇨ E&V Guidebook
  - -- Organization
  - -- Example Synopses
  - -- Example Tools and Aids
- Summary

## GUIDEBOOK ORGANIZATION

1. Introduction
2. Structure and Use of the Guidebook
3. Integration of E&V Technology
4. Synopses

} Early Chapters 1

5. Compilation System Assessors
6. Target Code Generation Aids and Analysis Assessors
7. Test System Assessors
8. Tool/Host Interface Accessors
9. Ada Design Support Assessors
10. Configuration Management Support Assessors
11. Distributed System Development and Runtime Support Assessors
12. Distributed APSE Assessors
13. "Whole APSE" Assessors
14. Adaption Assessors

99. Other Assessors

} Formal Chapters 1

---

## INTEGRATED APSE ASSESSMENT
*(GUIDEBOOK SECTION 3.3 – DECISION SUPPORT APPROACH)*

- **SPECIFY "ESSENTIAL FEATURES"**
  - — Subject to Yes/No Tests
- **SPECIFY "DESIRABLE FEATURES" AND CRITERIA**
  - — Attributes and Function-Attribute Pairs
- **SPECIFY WEIGHTS**
  - — Subjectively Chosen
- **PERFORM INDIVIDUAL ASSESSMENTS AND GENERATE COMBINED SCORE AND SUMMARY**

## EXAMPLE SYNOPSIS

**4.2 HOUGHTON: A TAXONOMY OF TOOL FEATURES FOR THE Ada PROGRAMMING SUPPORT ENVIRONMENT (APSE)**

Citations:

[Houghton 1983] R.C., Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," National Bureau of Standards, NBSIR-81-2625, February 1983.

Synopsis:

This paper puts forth a taxonomic classification of APSE features. The features included satisfy the criteria that they are "within current technology" and are "oriented to the Ada language." The top two levels of the classification are as follows:

- Input
  - Subject
  - Control Input
- Function
  - Transformation
  - Management
  - Static Analysis
  - Dynamic Analysis
- Output
  - User Output
  - Machine Output

For each of the second-level elements above, a third-level list is given, and some discussion is provided. The paper includes the results of a survey in which the second and third-level elements under "Function" are each rated as "Required," "Important," or "Useful."

## EXAMPLE SYNOPSIS

**4.6 TOOLS AND AIDS FOR E&V**

Citations:

[@E&V Report 1987] "Tools and Aids Document, Version 1.0," September 1987, Appendix C of "Evaluation and Validation (E&V) Team Public Report," Air Force Wright Aeronautical Laboratories, September 1987.

Synopsis:

This document was prepared by the Requirements Working Group (REQWG) of the E&V Team. It identifies the community's E&V technology needs, provides definitions of those needs, and prioritizes them in order of their relative importance. The purpose of this document is to provide pertinent information to those agencies willing and able to fund the development of E&V technology. It reflects the E&V Requirements Document (see synopsis [4.5]) and views on the subject obtained from surveys conducted among the E&V Team and appropriate ARPANET-MILNet Interest Groups.

## EXAMPLE COMPILER TEST SUITE SUMMARY

**5.4 PIWG BENCHMARK TESTS**

**Purpose:** Identification of performance characteristics of Ada compilers. The tests examine the performance of the compiler itself in terms of compilation speed, as well as the quality of the generated code for both processing and storage effectiveness. The test suite measures performance for both isolated language features and composites or mixes of language features (using the Whetstone and Dhrystone tests).
[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22; @RM: Storage Effectiveness 6.4.31]

**Primary References:**

**Host/OS:** Unrestricted

**Vendors/Agents:** [PIWG]

**Method:**
Automated test suite.
Inputs: PIWG source code, Ada compiler and runtime system, and host (and target) computer.
Process:
1. Obtain the latest PIWG tests
2. Compile and run tests according to the documentation.

Outputs: Reports on the outcome of each test run.

## EXAMPLE COMPILER TEST SUITE SUMMARY
*(PAGE 1 OF 3)*

**5.3 Ada COMPILER EVALUATION CAPABILITY (ACEC)**

Purpose: The purpose of this test suite is best stated by the following quote taken from the introduction in the ACEC Reader's Guide: "The ACEC .... consists of a portable test suite and support tools. ... It contains test problems designed to measure the execution time and size of a systematically constructed set of Ada examples. The support tools assist the ACEC user in executing the test suite and analyzing the results obtained." The scope of coverage provided by the test suite is shown by the following excerpts from the ACEC classification taxonomy:

II. Execution Time Efficiency
- A. Language Feature Efficiency
  - 1. Required (referenced by LRM section)
  - 2. Implementation Dependent (referenced by LRM section)
    - attributes (LRM Appendix A)
    - record representation clauses
    - interrupts
    - language interface
    - unchecked programming
- B. Pragmas
  - 1. Predefined
  - 2. Implementation Defined
- C. Optimizations
  - 1. Classical
  - 2. Effects of Pragmas
  - 3. Static Elaboration
    - aggregates
    - tasks
  - 4. Language Specific
    - Habermann-Nassi transformation for tasking
    - delay statement optimization
- D. Performance Under Load
  - 1. Task Loading
    - task creation
    - task termination
    - task abortion
    - Dining Philosophers Problem
    - task starvation
    - task delay
  - 2. Levels of Nesting
    - static
    - dyn

Param

## EXAMPLE COMPILER TEST CAPABILITIES CHECKLIST *(PAGE 1 OF 2)*

**5.11 ARTEWG RUNTIME ENVIRONMENT TAXONOMY**

**Purpose:** Describes the basic elements of Ada runtime environments and provides a common vocabulary. The following excerpt is taken from the introduction to the Taxonomy section. "If a runtime environment for an Ada program is composed of a set of data structures, a set of conventions for the executable code, and a collection of predefined routines, then the question arises: what are examples of these elements, and moreover, what is the complete set from which such elements are taken when a particular runtime environment is built?...It should be noted that the dividing line between the predefined runtime support library on one hand, and the conventions and data structures of a compiler on the other hand, is not always obvious. One Ada implementation may use a predefined routine to implement a particular language feature, while another implementation may realize the same feature through conventions for the executable code. ... This taxonomy concerns itself primarily with those aspects of the runtime execution architecture which are embodied as routines in the runtime library. It does not treat issues of code and data conventions, nor issues related to particular hardware functionalities, in any great depth."
[@RM: Runtime Environment 7.2.3.5, @RM: Power 6.4.21]

**Primary References:**
[ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environment Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.

**Vendors/Agents:** [ARTEWG]

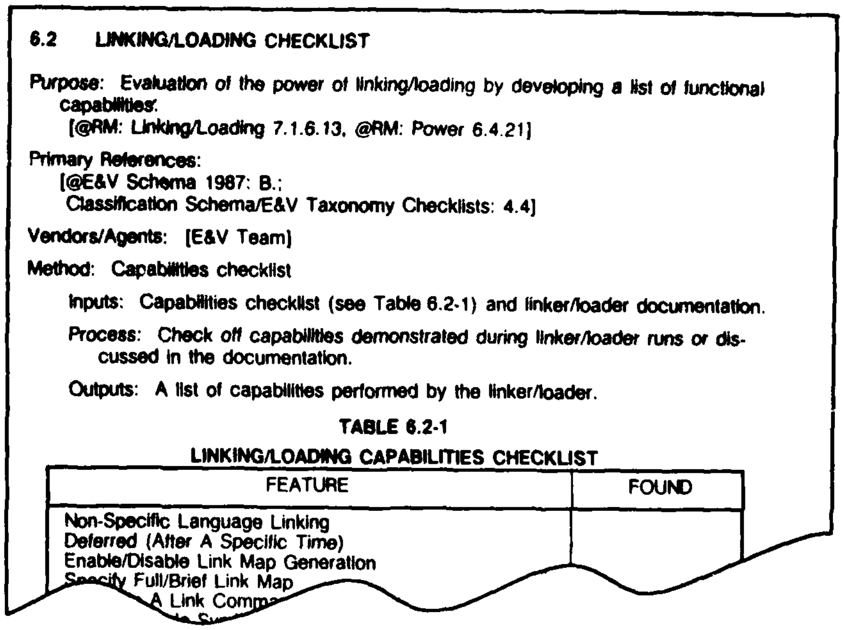
**Method:** Capabilities checklist
Inputs: Capability checklist (see Table 5.11-1)and runtime environment documentation.
Process: Check off capabilities demonstrated by the runtime environment or discussed in the documentation.
Outputs: A list of capabilities performed by the runtime environment.

## EXAMPLE COMPILER TEST CAPABILITIES CHECKLIST *(PAGE 2 OF 2)*

TABLE 5.11-1
RUNTIME ENVIRONMENT TAXONOMY

| FEATURE | FOUND |
|---|---|
| Runtime Execution Model | |
| Dynamic Memory Management | |
| Processor Management | |
| Interrupt Management | |
| Time Management | |
| Exception Management | |
| Rendezvous Management | |
| Task Activation | |
| Task Termination | |
| I/O Management | |
| Commonly Called Code Sequences | |
| Target Housekeeping Functions | |

## EXAMPLE TARGET CODE ASSESSOR



**6.2 LINKING/LOADING CHECKLIST**

**Purpose:** Evaluation of the power of linking/loading by developing a list of functional capabilities.
[@RM: Linking/Loading 7.1.6.13, @RM: Power 6.4.21]

**Primary References:**
[@E&V Schema 1987: B.;
Classification Schema/E&V Taxonomy Checklists: 4.4]

**Vendors/Agents:** [E&V Team]

**Method:** Capabilities checklist

**Inputs:** Capabilities checklist (see Table 6.2-1) and linker/loader documentation.

**Process:** *Check off capabilities demonstrated during linker/loader runs or discussed in the documentation.*

**Outputs:** A list of capabilities performed by the linker/loader.

**TABLE 6.2-1**
**LINKING/LOADING CAPABILITIES CHECKLIST**

| FEATURE | FOUND |
|---|---|
| Non-Specific Language Linking | |
| Deferred (After A Specific Time) | |
| Enable/Disable Link Map Generation | |
| Specify Full/Brief Link Map | |
| A Link Comm | |

## EXAMPLE TARGET CODE ASSESSOR
*(PAGE 1 OF 2)*

**6.5 DEBUGGING CAPABILITIES CHECKLIST**

**Purpose:** Evaluation of the power of debugging by developing a list of functional capabilities.
[@RM: Debugging 7.3.2.5, @RM: Power 6.4.21]

**Primary References:**
[@E&V Schema 1987: B.;
Classification Schema/E&V Taxonomy Checklists: 4.4]

**Vendors/Agents:** [E&V Team]

**Method:** Capabilities checklist

**Inputs:** Capabilities checklist (see Table 6.5-1) and debugger documentation.

**Process:** *Check off capabilities demonstrated by the debugger or discussed in the documentation.*

**Outputs:** A list of capabilities performed by the debugger.

## EXAMPLE TARGET CODE ASSESSOR

*(PAGE 2 OF 2)*

TABLE 6.5-1
DEBUGGING CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Instrumentation | |
| Statement | |
| Branch | |
| Unit | |
| CSC | |
| Machine Level Debugging | |
| Symbolic Debugging | |
| Tracing | |
| Breakpoint Control | |
| Data Flow Tracing | |
| Path Flow Tracing | |
| Selectable Level Of Granularity | |
| Display | |
| Program Source | |
| History | |
| Stack | |
| Tasks | |
| Breakpoints | |
| Tracepoints | |
| Memory | |
| Collections And Global Heaps | |
| [illegible] | |

## EXAMPLE TEST SYSTEM ASSESSOR

*(PAGE 1 OF 2)*

### 7.1 TESTING CAPABILITIES CHECKLIST

**Purpose:** Evaluation of the power of testing by developing a list of functional capabilities. [@RM: Testing 7.3.2.10, @RM: Power 6.4.21]

**Primary References:**
[@DeMillo 1986] R.A. DeMillo, "Functional Capabilities of a Test and Evaluation Subenvironment in an Advanced Software Engineering Environment," Georgia Institute of Technology GIT-SERC-86/07, 20 October 1986.

**Vendors/Agents:** [GIT]

**Method:** Capabilities checklist

**Inputs:** Capabilities checklist (see Table 7.1-1) and testing system documentation.

**Process:** Check off capabilities demonstrated by the testing system or discussed in the documentation.

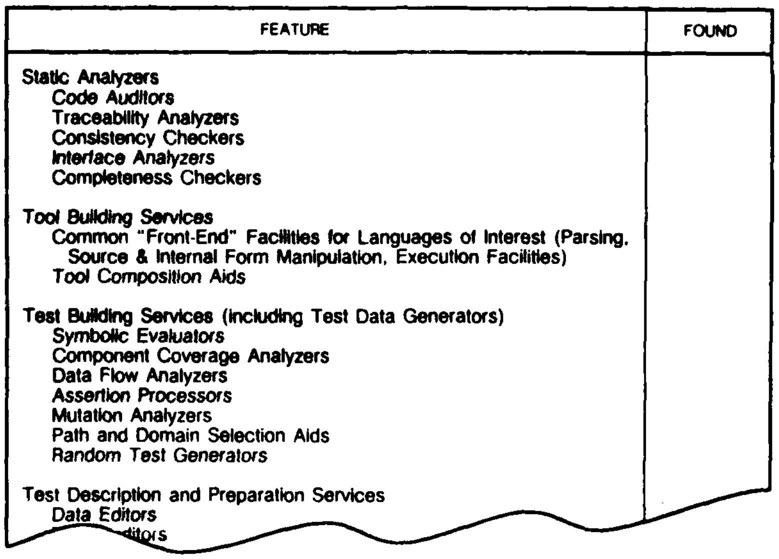
**Outputs:** A list of capabilities performed by the testing system.

## EXAMPLE TEST SYSTEM ASSESSOR

*(PAGE 2 OF 2)*

TABLE 7.1-1

TESTING CAPABILITIES CHECKLIST



| FEATURE | FOUND |
|---|---|
| Static Analyzers | |
| Code Auditors | |
| Traceability Analyzers | |
| Consistency Checkers | |
| Interface Analyzers | |
| Completeness Checkers | |
| Tool Building Services | |
| Common "Front-End" Facilities for Languages of Interest (Parsing, Source & Internal Form Manipulation, Execution Facilities) | |
| Tool Composition Aids | |
| Test Building Services (including Test Data Generators) | |
| Symbolic Evaluators | |
| Component Coverage Analyzers | |
| Data Flow Analyzers | |
| Assertion Processors | |
| Mutation Analyzers | |
| Path and Domain Selection Aids | |
| Random Test Generators | |
| Test Description and Preparation Services | |
| Data Editors | |
| ditors | |

## EXAMPLE TOOL/HOST INTERFACE ASSESSOR

### 8.1 CAIS IMPLEMENTATION VALIDATION CAPABILITY (CIVC)

Purpose: Assess the conformance of CAIS implementations to the DoD-STD-1838 and DoD-STD-1838A standards. The CIVC consists of a test suite, analysis tools, and associated documentation which enable validators and CAIS implementors to determine the completeness of CAIS implementations with respect to conformance to the standards. A suite of tests to be compiled and executed with interfaces provided for a CAIS implementation. Analysis tools are utilized for aiding the users in selecting tests and obtaining results.

[@RM: Kernel 7.2.3.3, @RM: Completeness 6.4.9]

Primary References: [CIVC 1988] "CIVC I Implementor's Guide," Air Force Wright Aeronautical Laboratory, CVC-VREL-19, Draft August 1988.

Host/OS:
Not Applicable

Vendors/Agents:
TBA

Method: Automated test suite

Inputs: The CIVC test suite, CAIS implementation, Ada compiler and runtime system, and host computer.

Process:

1. Obtain the CIVC test suite
2. Compile and run the tests
3. Collect and analyze the results.

Outputs: Report describing the conformance of various aspects of the CAIS implementation to DoD-STD-1838 or 1838A.

## EXAMPLE TOOL/HOST INTERFACE ASSESSOR

### 8.2 TOOL SUPPORT INTERFACE EVALUATION

**Purpose:** Evaluation of tool support interfaces in terms of four criteria: level, appropriateness, implementability, and performance. Five "scenarios" were designed and used to exercise a prototype CAIS implementation and a prototype PCTE implementation. The scenarios involved a configuration management system, an edit-compile-link-test cycle, a conference management system, a window manager, and a design editor.

[@RM: Kernel 7.2.3.3]

**Primary References:**
[Long 1988] F.W. Long, and M.D. Tedd, "Evaluating Tool Support Interfaces," Ada in Industry, Proceedings of the Ada-Europe Conference, Munich, 7-9 June 1988, Cambridge University Press, 1988.

Host/OS: Sun

Vendors/Agents: [College of Wales, UK]

Method: Structured experiment

Inputs: The source code for the scenarios, the tool support interface(s) (CAIS, PCTE, other), Ada compiler and runtime system, and host computer.

Process:
1. Obtain the source code for the scenarios
2. Compile and run the scenario(s)
3. Collect the results.

Outputs: Objective results and subjective conclusions concerning the impact on tool writers and the cost and behavior of the interface implementation.

## EXAMPLE WHOLE-APSE ASSESSOR
*(PAGE 1 OF 3)*

### 13.1 APSE CHARACTERIZATION

**Purpose:** The purpose of this form is to provide an overview or summary of the capabilities and features of an APSE. This form can be used as an initial information gathering device to begin the process of whole-APSE assessment. This information would then be supplemented by results of detailed evaluations or examinations of attributes that are of specific interest to the potential buyer or user of an APSE.
[@RM: Whole APSE Issues 3., @RM: Capacity 6.4.6; @RM: Cost 6.4.11;

@RM: Maturity 6.4.18; @RM: Operability 6.4.20; @RM: Power 6.4.21;

@RM: Required Configuration 6.4.26]

Primary References:

Host/OS: Not Applicable.

Vendors/Agents: [E&V Team]

Method: Questionnaire

Inputs: Blank APSE characterization form (see Fig. 13.1-1) and APSE documentation.

Process:
1. Have APSE vendor complete the APSE characterization form
2. Select APSEs for further investigation based on information gathered from step 1.

Outputs: Completed APSE characterization form.

## EXAMPLE WHOLE-APSE ASSESSOR

*(PAGE 2 OF 3)*

Name/Acronym:

Vendor:

Address:

Phone Number:

Cost ($, no charge, not available/applicable):

| | | | |
|---|---|---|---|
| Purchase | ________ | Seminars | ________ |
| Maintenance | ________ | In House Classes | ________ |
| Documentation | ________ | Educational Videos | ________ |
| On-Line Help | ________ | On-Line Tutorials | ________ |
| Hot-Line Support | ________ | | |

Problem Reporting/Resolution Procedures:

Frequency of Updates:

Usage Limitations (License Restrictions):

Host/Target(s) — Required Configurations:

Peripherals Supported:

Languages Supported & Interoperability Features:

Summary of Features:

Life Cycle Support — Capabilities/Major Activity:

Methodology Support:

Management Support:

Application-Specific Capabilities:

Documentation Support (editors, word processors, document generators, desktop publishing):

## EXAMPLE WHOLE-APSE ASSESSOR

*(PAGE 3 OF 3)*

File/Database/Program Library Management (hierarchical, relational):

Access Control — Level of Granularity:

Integration Mechanism (standard file structures, database, standard intertool interfaces):

User Interface (command language, menus, icons) — Flexibility vs. Consistency:

Extensibility:

Support for Distributed Development:

Capacity (number of users, size of project):

Typical Usage Scenarios:

Developer:

Production Process/Vehicles:

Date First Released:

Previous Use:

References (documentation, evaluation results, case histories):

## OVERVIEW

- APSE E&V Task Background
  - -- Need and Purpose
  - -- Process and Products
- E&V Reference Manual
  - -- Organization
  - -- Example Usage
- E&V Guidebook
  - -- Organization
  - -- Example Synopses
  - -- Example Tools and Aids
- Summary

## SUMMARY

THE E&V REFERENCE SYSTEM SHOULD HELP USERS TO:

- Gain Overall Understanding of APSEs and Approaches to Assessment
- Find Useful Information — Terminolgy, Definitions, Relationships
- Find Assessment Criteria/Metrics and "Pointers" to Specific Evaluation or Validation Techniques
- Find Descriptions of Evaluation or Validation Techniques
- Find Guidance in the Selection, Interpretation, and Integration of Evaluation and Validation Techniques and Results

## FOR MORE INFORMATION

CONTACT:

Raymond Szymanski
AFWAL/AAAF
Wright Patterson AFB, OH 45433-6543

(513) 255-2446/6730 or AV 785-2446/6730

Milnet Address: szymansk@ajpo.sei.cmu.edu

## Title Slide -- The APSE E&V Reference System

The subject of this presentation is a pair of documents known collectively as the

"APSE E&V Reference System."

The titles of the documents are the

"E&V Reference Manual"
and the
"E&V Guidebook"

## Overview

The presentation will proceed in the order indicated here. Since the E&V Reference Manual and the E&V Guidebook are only two of several products being produced by an activity known as the "APSE E&V Task", we'll begin with some background information on that activity -- the need for E&V technology, the purpose of the task, the process by which the task is attempting to achieve that purpose; and some key products under development.

We'll address the Reference Manual -- its organization and its use. Then we'll address the Guidebook, including a number of examples of the material it contains. Since the Reference Manual has been the subject of previous public presentations, we plan to devote more attention today to the second document -- the E&V Guidebook.

Finally, we'll summarize and tell you how to get the documents or more information about other E&V Task activities and products.

## Definitions

"APSE" refers to Ada Programming Support Environments.
"E&V" refers to Evaluation and Validation.

"Evaluation" means Assessment of Performance and Quality
--- for example, testing performance of a compiler --
which can mean either the speed of compilation or the speed of the code generated by a compiler

"Validation" means Assessment of Conformance to a Standard.
--- for example, testing a compiler implementation to see if it conforms to the language definition
(As a practical matter, of course, when we say a product passed a Validation Test, we usually mean that, "We failed to prove that it doesn't conform.")

So these documents address the assessment of Ada Programming Support Environments --
What does that mean?
How can we do it?
What technology exists to help us do it?

You would be interested in knowing about these documents (or perhaps having copies of your own) if you expect to --
Use an Ada support environment,
Assess an environment or its components,
Select or buy an environment or some components,
Manage or consult with anyone who fits these categories.

## Need for E&V Technology

The Ada community feels a need for E&V technology. The main reason for this, of course, is that people need a basis for decision -- is this particular APSE, or tool-set, or tool good enough for my application? -- or, of several that are good enough, which is "best"?

Furthermore, the decisions to be made can be very important. Consider, for example, the large, critical Ada-based systems to be developed in coming years -- ATF, LHX, Space Station, etc. The effectiveness, reliability, and cost of these systems will be strongly influenced by the environments used to develop them. From the point of view of the software developing organization, the decision to select an APSE can be a crucial investment decision with long-lasting influence on a number of projects and the organization's methods of operation, training, and competiveness. From the point of view of a software maintenance organization, the environment used will strongly influence that organization's effectiveness, as well as its operations and training.

New technology is needed because of the difficulty of assessing APSEs and their components -- tools. First, some of the difficulty exists simply because an APSE is a very complex system, with many inter-related parts. Second, there is a confusing diversity of choice with respect to tools, tool-sets, and "whole APSEs" -- and there are a number of ways of viewing APSEs. Third, the state-of-the-art of APSE architecture and some categories of tools is undergoing rapid change. Finally, there is a lack of historical data relevant to APSEs -- partly because of the general pace of technological change and partly because of Ada itself -- still a relatively new language.

## E&V Task Purpose

The purpose of the E&V Task is to provide a focal point for addressing the need felt by the general software development community -- the need to assess APSEs and components of APSEs -- and the need to have an available technology for performing such assessments.

We want to achieve this purpose by engaging in the four activities shown here:

1) by identifying and defining elements of the required E&V technology;
2) by developing selected elements of it ourselves;
3) by encouraging others to develop some elements; and
4) by collecting and disseminating information about the whole subject. The E&V Reference Manual and the E&V Guidebook are the products that serve this collection and dissemination function.

## E&V Task Process

The process is organized as shown here. Our sponsor is the Ada Joint Program Office. The task is a tri-service activity whose lead agency is the US Air Force Avionics Laboratory located at the Wright Patterson AFB in Dayton, Ohio. Ray Szymanski is the leader of both the E&V Task and a team of Government participants known as the E&V Team -- which is supported by a group of invited technical advisors from industry and academia.

Three contracts have been awarded:

TASC provides general technical support and produces certain documents known collectively as the reference system.
Boeing is developing a test suite for compiler performance evaluation known as the ACEC.
SofTech is developing a test suite for validation of CAIS implementations known as the CIVC.

## E&V Task Products

This slide lists some key products of the E&V Task, and indicates when each was, or is scheduled to be, ready to be considered for approval for general distribution. There have been other products as well -- including an annual Public Report -- but typically, these others are not of general interest to the wider community.

As indicated on the slide the E&V Reference System is made up of two documents called the Reference Manual and the Guidebook. There is a close relationship between these two, as will be shown on the next slide.

The other two products listed are the two test suites, ACEC and CIVC, being developed by Boeing and SofTech, respectfully.

## Use of the Reference System

This slide depicts the relationship between the two documents that make up the Reference System. Users would typically consult the Reference Manual first, and then follow one of two courses. Some users might simply browse through parts of the manual seeking to learn some of the terminology used to characterize APSEs, or to learn definitions of terms and key relationships between them -- in other words, these users would extract useful information directly from the manual itself -- as indicated on the left side of the diagram.

Other users would use the manual to find pointers to sections in the Guidebook -- as pictured on the right. The material found in these sections would describe specific instances of "E" -- Evaluation -- or "V" -- Validation -- technology.

Examples of both of these kinds of uses will be given shortly.

Users may also decide to consult the Guidebook directly -- so it has been designed to be easy to use as a stand-alone document as well.

## Reference Manual Organization

The document has three main parts -- as illustrated here.

The first two chapters contain introductory material, similar in content to the material of this presentation -- background, organization, how-to-use instructions. The third chapter provides a discussion of APSEs considered as a whole, rather than as the sum of many small components or functions.

The middle section, which is by far the largest, contains the reference material itself. These chapters conprise a set of subject indexes and text frames that follow a standard format.

The remaining section contains several appendixes and a composite, alphabetical index.

## Indexes and Text Frames

The standard format used throughout the main section of the manual is illustrated here. Each of these chapters is organized as a hierarchical taxonomy of elements. For example, Chapter 7 is called the "Function Index" as shown. One of its top-level subdivisions is titled "Transformations", which is further subdivided into "Editing," "Formating," etc. At a still lower level is the element numbered "7.1.6.7 Compilation." for each low-level element, there is a "Text Frame" and each Text Frame has a standard structure -- with three parts, as shown here. The three parts are labeled "Description," "Cross References," and "Guidebook References."

## Text Frame Example (page 1 of 2)

Here is a text frame copied from the Function Index. It is Frame Number 7.1.6.7 "Compilation." The first two parts of it -- "Description" and "Cross References" -- are shown on this view. Let's go right to the remaining part -- shown on the next slide.

## Text Frame Example (page 2 of 2)

Under this third part -- "Guidebook Reference" -- we find five attributes listed -- some more than once. For example, "Processing Effectiveness" is listed five times, because five different test suites pertient to the combination, "Compilation-Processing Effectiveness," are summarized in the Guidebook -- one is called "IDA Benchmarks" others are "ACEC," "PIWG," "U.Michigan," and "UK-AES." We see that all five of these are addressed in Chapter 5 of the Guidebook.

## Usage Scenario #3

Chapter 2 of the Manual makes use of three scenarios to illustrate how the document is to be used. In the first scenario "User A" simply looks up a definition of one of the attributes in the Attribute Index. In the second scenario "User B" consults the life cycle Activities Indes and uses its cross references, to find out what functions are typically used in a specific phase or activity-group. Since out audience today is made up of such intelligent people, we are skipping directly to the third and most complex scenario.

Here is the third and final example. Suppose, now, your name is "User C," and you want to evaluate an APSE in terms of how well it performs a certain function -- such as Compilation. Evaluation objectives are stated in terms of attributes or function-attribute pairs. So you begin by looking up "Compilation" in the Function Index, and find out which attributes are associated with it. You find this out by consulting the Guidebook References -- represeted by the elongated ellipse in the pictorial. Each function-attribute pair listed points to a specific section in the E&V Guidebook (such as the five references to Chapter 5 we saw in the proceeding slide) that contains information about a specific instance of an Evaluation or Validation technique.

## Combined View

Based upon the types of scenarios illustrated, we construct a combined conceptual view or framework of the entire reference system --- as pictured here. The diagonal row of boxes represent Indexes; the small circles represent Cross References; the ellipses represent Guidebook References. The framework as a whole provides paths, within and between indexes, that users can follow to extract information directly -- or to find sections in the Guidebook that describe elements of E&V technology. We can think of the indexes as analogous to the cabinets of a card catalog system in a public library -- with its Title Index, Author Index, and Subject Index. In our model we have four subject indexes -- Activities Index, Tool/APSE Index, Attribute Index, and Function Index -- as shown.

The Function Index is seen to be directly related to all of the other indexes. It is drawn in an "anchoring" position in the diagram. Attributes play a key role in the E&V process -- assessment objectives are defined either in terms of attributes (such as interoperability, cost) or in terms of function-attribute pairs (such as compilation-processing effectiveness, or editing-power).

This conceptual structure has an open-ended quality. We could add another Index by adding another chapter -- represented by another box along the diagonal row. Additional Cross References and Guidebook References can be added as well -- represented by the connections above or below the row of boxes.

## Guidebook Organization

The E&V Guidebook has three main sections. First, is the introductory and background material -- Chapters 1 through 4. Second, the main bulk of the document -- Chapters 5 and beyond -- containing descriptions of specific E&V techniques. These are called "Formal Chapters" because they follow a standard format throughout. Third, there are several appendices -- not listed here.

The material in the Formal Chapters -- 5 through 99 -- is what the Reference System is really all about. Each of these chapters contains all the procedures and techniques associated with a particular group of tools or tool-sets to be assessed -- such as

Chapter 5 -- Compilation System Assessors
Chapter 7 -- Test System Assessors
Chapter 10 -- Configuration Management Support Assessors

Our main objective is to let people know about these things -- so that they will be used and improved. When we say that a chapter contains all of the relevant procedures and techniques, we mean these are all that the E&V Team currently knows about. We expect that you will look these documents over and let us know about additional things that are out there, so we can include them in future editions of the Reference System documents.

Most of the remainder of this presentation consists of displays of excerpts from various sections of the Guidebook -- to illustrate its format and contents.

## Integrated APSE Assessment

Chapter 3, on integrated APSE assessment, contains a brief outline of an approach based on "Decision Support" techniques -- involving identification of essential features, desirable features, and weighting factors. One of our E&V Team members (Major Pat Lawlis) is now working on a PhD thesis at Arizona State U. -- aimed at developing this type of approach.

## Example Synopsis - 1

Chapter 4 provides a series of synopses of earlier documents that contain important historical or background information. Here is an example of one of those synopses. Its subject is a 1983 paper by Houghton of the National Bureau of Standards -- which put forth a taxonomic classification of APSE features. This paper had a significant impact on later efforts at RADC and within the E&V Team itself -- work that influenced parts of the classification schema contained in the E&V Reference Manual.

By reading these -- there are 19 synopes in the current version of the Guidebook -- the reader can get an idea of the historical foundations and evolution of APSE and APSE-assessment technology.

## Example Synopsis - 2

Here is a synopsis of one of the E&V Team's own reports -- called the "Tools and Aids Document." This document was produced by our Requirements Working Group. It identifies E&V technology needs, provides definitions of those needs, and prioritizes them in order of their relative importance. As a matter of fact, the Guidebook "formal chapters" -- 5 through 14, listed here a few slides back -- were named and ordered in accordance with the definitions and priorities presented in this "Tools and Aids Document."

## Example Compiler Test Suite Summary - 1

Here's an example of one of the actual instances of E&V technology -- the PIWG Test Suite. It's one of the entries in Chapter 5, which is devoted to Compilation System assessors. All of these summaries follow a standard format -- illustrated here -- namely:

Purpose
Primary References
Host and Operating System
Vendors or Agents, and
Method

## Example Compiler Test Suite Summary - 2

Here's another example of a Compilation System assessor -- the Ada Compiler Evaluation Capability (ACEC) test suite. We devote three whole pages to summarizing this one in the Guidebook -- partly because its a more complex piece of technology, and partly because it's our own E&V Team product -- of which we are quite proud. Shown here is a portion of the first of the three pages -- it contains part of the ACEC classification taxonomy -- used to portray the scope of coverage and to characterize individual tests or groups of tests.

## Example Compiler Test Capabilities Checklist - 1

The previous two Evaluation examples were based on test suites. Another category of Evaluation Technology is the use of a checklist. This slide depicts our summary of one of the ARTEWG reports -- "A Framework for Describing Ada Runtime Environments." The next slide presnts
-- go to next slide --

## Example Compiler Test Capabilities Checklist - 2

Here is a checklist that is based on the taxonomy presented in that ARTEWG report. It could be used as an aid in understanding the capabilities of a particular implementation -- by determining (and checking off) those aspects of the runtime execution architecture which are embodied as routines in the runtime library.

## Example Target Code Assessor - 1

Here is the beginning of a text frame that presents a checklist for evaluating the power of the linking/loading capabilities of a system.

## Example Target Code Assessor - 2 (page 1 of 2)

Here is a frame that introduces a checklist for evaluating debugging capabilites.

-- go to next slide --

## Example Target Code Assessor - 2 (page 2 of 2)

And here is the beginning of the checklist.

## Example Test System Assessor (page 1 of 2)

Here's one that deals with the evaluation of the Testing Capabilities of an environment (or tool set).

-- go to next slide --

## Example Test System Assessor (page 2 of 2)

And here's the corresponding checklist.

By the way -- some of these checklists represent the result of considerable effort and iteration on the part of our team members, while others represent an initial stab at the issue, and are included as "place holders" in the current version of the Guidebook. We regard these Reference System documents as belonging to all of us in the Ada community -- that is, "Your Documents." We hope that many of you will help to improve and expand them in the future -- by sending comments and suggestions.

## Example Tool/Host Interface Assessor - 1

Here is a frame summarizing the CAIS Implementation Validation Capability (CIVC) -- under development by the E&V Task. This automated test suite will be used to assess the conformance of CAIS implementations to the DOD-STD-1838 and/or -1838A standards.

## Example Tool/Host Interface Assessor - 2

This frame summarizes an effort reported at the June 1988 Ada Europe Conference. It describes a comparative evaluation of prototype PCTE and CAIS implementations -- based on four criteria and five evaluation scenarios.

## Example Whole-APSE Assessor

This and the following two slides present a two-page form designed to be useful in characterizing an APSE as a whole. It could be used, for example, as an initial information-gathering device. It addresses attributes such as capacity, cost, maturity, etc.

-- show two following slides --

## Summary

In summary, the E&V Reference System is designed to help users in several ways --

It should help them gain an overall understanding of APSEs, as well as approaches to assessment of APSEs and their components.

It should help them find useful information such as -- the standard terminology used in discussions of APSEs and their assessment, the definitions of these terms, and relationships between elements in one part of the system and elements in another -- for example, which attributes pair up with which functions for evaluation purposes.

It should help them formulate assessment criteria or metrics, and locate descriptions of specific evaluation or validation techniques used to perform those assessments.

It provides them with descriptions of specific tools and aides -- such as questionaires, checklists, test suites, and structured experiments -- which can be used to assess APSEs and APSE components.

Finally, it provides guidance in selecting, interpreting, and integrating assessment techniques and results.

## For More Information

If you would like to have your name and address placed in the E&V Task data base, send your business card or equivalent to the name and address shown here. You will then receive future notices about E&V Task products as they become available for distribution.

Version 1.0 of the Reference Manual is available from the Defense Technical Information Center -- its DTIC Number is A197388. DTIC numbers for the other documents will also be forthcoming.

1988 USAF-UES SUMMER FACULTY RESEARCH PROGRAM

Sponsored by

AIR FORCE OFFICE OF SCIENTIFIC RESEARCH

Conducted by the

Universal Energy Systems, Inc

FINAL REPORT

# Ada Compiler Evaluation Capability

| | |
|---|---|
| Prepared by: | Mike Burlakoff |
| Academic Rank: | Assistant Professor |
| Department & University: | Computer Science Department<br>Southwest Missouri State University<br>Springfield, Mo., 65804 |
| Research Location: | Air Force Avionics Laboratory,<br>AFWAL/AAAF-3, WPAFB, Oh, 45433 |
| USAF Researchers: | Lt. Robert Marmelstein, Lt. Marc Pitarys |
| Date of Report: | August 8, 1988 |
| Contract: | F49620-87-R-0004 |

## ACKNOWLEDGEMENTS

I wish to thank the Air Force Systems Command and the Air Force Office of Scientific Research (AFOSR) for sponsorship of this research. LTC Cavender of AFOSR was especially helpful in this sponsorship. Mr. Ray Szymanski of the Avionics Laboratory, System Avionics Division, Systems Evaluation Branch, Logistics Technology Group was responsible for my being assigned to this project and was my administrative point of contact during this period. Universal Enerqy Systems was always available to answer questions and provide help in the administrative aspects of the program.

The administrative staff at Southwest Missouri State University (SMSU) was also supportive of this work. Dr. Jerry Berlin, Dean of the College of Science and Mathematics endorsed support of this effort. Dr. Bruno Schmidt, Department Head, Computer Science, approved the use of the VAX computer facilities for installing and executinq the test suite. John Province of SMSU Computer Services aided in the installation of the test suite.

I could not have achieved many of the results that were achieved in this effort without the excellent technical support throughout this period. Marc Pitarys provided the help that was needed to become familiar with the technical aspects and provided initial support and direction. Bob Marmelstein provided guidance and direction for the areas to be investigated. Their help and encouragement clearly added to the results of this research project.

# Ada Compiler Evaluation Capability

by

Mike Burlakoff

## ABSTRACT

The initial phase of the Ada Compiler Evaluation Capability (ACEC) test suite and support software is presently beinq delivered to the Air Force by the Boeing Company ACEC contractor. The system has undergone formal contractor testing with additional Air Force evaluation and testing. The Air Force determined that it would be desirable to provide additional Independent Validation and Verification (IV&V) of this initial delivery. The primary purpose is to verify test results and to determine whether any usability improvements in the products could be made.

Following are the major areas which were investigated: Execution of the test suite and analysis of the results, verification of procedures for use of the test suite, review of statistical support software and review of the major documentation for users of the system.

## I. INTRODUCTION:

In 1975 the Department of Defense (DoD) High Order Language Working Group was formed with the goal of establishing a single high order language for use in DoD systems (in particular, in Embedded Computer Systems). Following establishment of technical requirements and international competition, the Ada language as currently defined in (1) was selected. One of the major goals of Ada is to reduce the rapidly increasing costs of software development and maintenance in military systems.

Early in the development process it was realized that the acceptance and benefits derived from a common language could be increased substantially by the development of an integrated system of software development and maintenance tools. The requirements for such an Ada Programming Support Environment (APSE) were stated in the STONEMAN (2) document. STONEMAN identifies the APSE as support for "the development and maintenance of Ada application software throughout its life cycle." (2)

In June 1983 the Ada Joint Program Office (AJPO) proposed the formation of the E&V Task and a tri-service APSE E&V Team, with the Air Force designated as the lead service. In October 1983 the Air Force officially accepted responsibility as the lead service for the E&V Task.

The purpose of the E&V Task is to provide a focal point for addressing the need to provide the capability to assess APSEs and their components and to determine their conformance to applicable standards, such as the Ada Language Standard (1). This will be accomplished by (1) identifying and defining specific technology requirements, (2) developing selected elements of the required technology, (3) encouraging others to develop some elements, and (4) collecting information describing existing elements. This information will be made available to DoD components, other government agencies, industry and academia (3).

The Ada Compiler Evaluation Capability (ACEC) is one of the technology initiatives of the E&V effort. The E&V team proposed the initial ACEC concept and has made valuable contributions in the guidance and direction of this technology. The Boeing Military Airplanes (BMA) Software and Languages Organization is the contractor responsible for the work.

To provide insight into the process of executing the ACEC and using the analysis tools, the following summary is provided:

Before test problems are translated for execution, ACEC source code is "included" in the test problem. The purpose of the source code is to perform initialization, compute timing, produce output, etc. Some of this code precedes the test problem and some follows it. This process permits new test problems to be easily added to the test suite with no test measurement overhead needed. Following test suite execution, some of the outputs from these "included" programs are: Memory size, minimum and mean of the execution time, counts of how many iterations/repetitions the problems was executed, standard deviation of time measurements, a code (#) which signifies that a specified timing confidence interval was not reached, etc. These outputs can be saved for later processing by ACEC support software for later analysis.

A program called FORMAT reads the above resulting raw data and generates Ada source code consisting of two initialized data array aggregates. One of the aggregates gives the "minimum" execution time, and the other the "space" measurement. (Note that a user does not need to execute FORMAT, but can generate these aggregates manually).

The source code output by FORMAT is then hand edited into a Ada package called MED_DATA.Ada. MED_DATA.Ada contains the formal type declarations, etc. In particular, it contains a two dimensional array type. One of the dimensions ranges is user supplied names of the systems to be analyzed (VAX, IBM, ...). The range of the other dimension is all of the ACEC test names. Therefore, when the user edits in the initialized values into MED_DATA.Ada, a correct Ada program exists. MED_DATA is then compiled, and serves as the input for the ACEC statistical analysis and output program MEDIAN.

## II. OBJECTIVES OF THE RESEARCH EFFORT:

The initial phase of the Ada Compiler Evaluation Capability (ACEC) test suite and support software is presently being delivered to the Air Force by the Boeing Company ACEC contractor. The system has undergone formal contractor testing with additional Air Force evaluation and testing. The Air Force determined that it would be desirable to provide additional Independent Validation and Verification (IV&V) of this initial delivery. The primary purpose is to verify test results and to determine whether any usabillty improvements in the products could be made.

My assignment as a participant in the 1988 Summer Faculty Research Program (SFRP) was to perform the above IV&V of the ACEC. Following were the major areas to be investigated:

a. Execution of the test suite and verification of results.

b. Use of contractor supplied tests procedures and scenarios.

c. Use of statistical analysis support tools to determine correctness and usefullness.

d. Review and critique of the ACEC User's (4) and Reader's Guides (5).

e. Analysis of specific test problems which demonstrated incorrect or exceptional results.

## III. ACCOMPLISHMENTS AND RESULTS:

a. The ACEC test suite was executed on the AFWAL VAX-11/780 and the Southwest Missouri State University (SMSU) VAX-11/750. The results of both systems were verified that they agreed with the results presented in the Boeing Test Report, dated 27 May 1988.

b. A separate execution of the test suite was performed on the SMSU system to purposely follow the procedures and scenarios provided in the User's Guide. The majority of the procedure worked as specified. Minor comments are listed in (6).

c. Two subsets of the ACEC were repeatedly executed on the SMSU VAX, during various times of the day, using both CPU and Elapsed (clock) time computed measurements. The purpose was to investigate the consistency of these computed times. The results were processed by the ACEC statistical analysis tools. The times were approximately equal. For this system and tests, the CPU times were the most consistent. It should be noted that this was a small subset of tests with a small number of repetitions, and no conclusions should be drawn from this experiment.

d. The statistical analysis tools were used on much of the resulting data and the results were analyzed.

e. The ACEC User's and Reader's Guides were reviewed. The specific comments are listed in (6) and (7). In general, the documents were complete, well written and provided the needed ACEC users and background information. The major recommendations were: (1) Additional summaries to present the extent and coverage of the ACEC test suite, and (2) Additional examples in portions of both manuals.

## IV. RECOMMENDATIONS:

### 1. REVIEW OF TEST FAILURES:

The ACEC Software Test Report, dated 27 May 88, gives a summary of test results. Many of the tests failed for various reasons on several of the hosts (except for the Dec VAX). A standard classification failure mnemonic is listed for the failures. However, in reviewing the test logs at the end of the document, in many cases, reasons for the failures are not given. All test failures should be investigated and problem tests should be corrected.

## 2. REVIEW OF TEST SUITE COVERAGE AND CORRECTNESS:

The entire test suite should be reviewed to verify that the LRM features listed as comments in the tests are correct. The LRM feature should reflect the major purpose of the test. The LRM features could be used as a guide to verify that the ACEC LRM test coverage is complete. Other categories that should be reviewed for completeness of coverage are other typical ACEC programming example tests (benchmarks) and other embedded computer system applications.

The ACEC is a large and rigorous set of test problems which are designed to measure and compare performance of various hosts and targets which use Ada compiled software. Because of the size and complexity of the test suite, it may be that the performance of some of the tests may not be as intended. Individual test problems should be reviewed for being correct Ada and for testing what was intended to be tested (that is, not being amenable to "unintended optimization"). Where appropriate, any problems of test portability should also be reviewed (8).

## 3. DELIVERY TAPE HELP FILE:

The User's Guide which is supplied with the ACEC is complete and helpful in guiding a user in the use and execution of the ACEC. However, as experience is gained in using the system, improved procedures and techniques may become available, and frequent updates to the User's Guide would be needed. It may be helpful if the delivery tape contained a file such as README.DOC which provided a quick summary of procedures to use the ACEC. This could simply be a summary such as presently provided on pages 15-17 of the User's Guide (updated with improvements in use, etc). One of the more valuable uses of this file could be suggestions and hints on lessons learned from executing the ACEC on specific host and targets.

## 4. VERSION DESCRIPTION DOCUMENT:

Appendix V, Feature Cross References lists test problems which use a particular Ada Language Reference Manual (LRM) feature. A reader could easily misconstrue this appendix and assume that the tests listed under each of the features were specially constructed to test that feature, where in fact, all that is meant is that the test USES the feature. An appendix which listed all LRM features specifically tested by the ACEC may be quite useful. Consideration should be given on whether the present appendix should be kept.

## 5. EXPANSION SIZE COMPUTATION:

The test problem expansion size computation is presently initiated in the INCLUDE program, and then "included" in the STOPTIME2 program. This is not a part of the INCLUDE process and should be coded elsewhere. The STOPTIME2 program would be a more appropriate place for this computation.

## 6. FORMAT RECOMMENDATIONS:

FORMAT outputs a mnemonic ("UNRELIABLE") wherever the desired timing confidence level was not reached (a code "#" in the raw data file). The User's Guide discusses how a user may change this process on page 86. However, it may be preferable for FORMAT to always output all measurements and add a suffix code ("#", etc.) to the values. These would also be input and output by MEDIAN. A user will then have all the raw measurements along with an indicator which flags unusual measurements.

Presently, when a subset of the ACEC is executed, and the results are processed by MED_DATA, the user must manually edit out all test names that are not used. An alternative is for the user to indicate missing tests by the "OTHERS => err_no_data" code. When this is done, MEDIAN outputs a full report, with much of the data as "err_no_data" codes showing null values for the tests which were not run. Recommend that FORMAT contain an option which creates the package declarations, with only the test names that are needed for MED_DATA (e.g., create the complete source file MED_DATA.Ada).

FORMAT presently outputs both time and space measurements. Since only one or the other can be used at one time for statistical analysis, a user must presently manually delete one of them from the FORMAT output. A option should exist so FORMAT would produce only one of the measurements (with default being time).

## 7. MEDIAN RECOMMENDATIONS:

As stated under FORMAT recommendations, all "raw" measurements should be output by FORMAT and MEDIAN. Unusual measurements should be followed by a suffix error code. Also, since Standard Deviation is an indicator of the degree of timing error, output this value on the Timing Measurements Table.

For a user with a non-statistical background, the MEDIAN outputs may appear to be extremely detailed and complex. It may be that the explanations are presented from a statistical viewpoint rather than a compiler evaluator's viewpoint. Following are suggestions which may aid in understanding some of the outputs:

a. Since the Histograms relate to the Problem, Systems and Residual table, the Histograms would be more understandable if they followed that table.

b. The Problem, Systems and Residual Factors table is the basis for much of the statistical analysis. Consider using an explanation page similiar to the following format:

"The purpose of this table is to provide a statistical basis for comparison of systems performance. From this data, a user may determine which systems and/or problems performed as expected or much better or worse than expected.

The data given for each problem and system represents Residual Factors. The final column gives Problem Factors for each problem.

A Residual Factor of 1.0 indicates a exact expected performance result. The Residual Factor will be large ... (Use the remainder of the paragraph on page 49 of the Reader's Guide along with the explanations of the character indicator fields. Then add the paragraph and equations presented at the beginning of the present explanation on page 49. Then continue the explanation as follows:

The Problem Factor values approach the mean of the problem times, however the values are scaled because System Factor for the first system is arbitrarily chosen to be 1.0.

The last line of the table gives the System Factor. This is computed by using ... (give an example of how computed). Note that the first system is assigned a value of 1.0 and the computations for the other systems are based on this value. The amount by which the System Factor is greater or less than one provides a measure of how much slower or faster the system is compared to the first system."

c. The Histogram explanation needs to state that the Actual Factor column relates to the Residual Factors listed on the Residual, Problem and Systems Factors table. That is, for each Slot Number, the value under Entries Per Slot refers to the number of Residual Factors for that system or entire data set.

d. It may be helpful to change the title of the "Summary Data for each System" table to something like "Summary Data of Residual Factors for each System".

e. The last summary table should be of interest to most users. It would be helpful if all eight possible outcomes (VALID, CMP_TIME,...) were output in this table. If necessary a code could be used for the titles. This information is probably already computed, so it should only be a matter of finding a way to display the data.

## REFERENCES

1. DoD. Ada Programming Language, ANSI/MIL-STD-1815A, 22 Jan 83.

2. DoD. "Stoneman". Requirements for Ada Programming Support Environments. Feb 80.

3. DoD. Evaluation & Validation (E&V) Plan. Version 4.0. 4 Jun 87.

4. Boeing. ACEC Technical Operation Report. User's Guide. 10 Jun 88.

5. Boeing. ACEC Technical Operation Report. Reader's Guide. 10 Jun 88.

6. Burlakoff, M. ACEC User's Guide Comments. 7 Jul 88.

7. Burlakoff, M. ACEC Reader's Guide Comments. 14 Jul 88.

8. Leavitt, T.C. ACEC Working Paper Number 21. 23 May 88.

# TOOLS AND AIDS DOCUMENT

Version 2.0
September 1988

Prepared by

Evaluation and Validation Team
Requirements Working Group

for the
Ada Joint Program Office

The Task for the Evaluation and Validation of Ada Programming Support Environments (APSEs) is sponsored by the Ada Joint Program Office

CONTENTS

1.0 INTRODUCTION . . . . . 1
1.1 Purpose of the Evaluation and Validation Task . . . . . 1
1.2 The Need for E&V Technology . . . . . 2
1.3 Purpose of the Tools and Aids Document . . . . . 2
1.4 Scope . . . . . 3

2.0 TYPES OF ASSESSORS . . . . . 3
2.1 Requirements and Specifications . . . . . 4
2.2 Guidelines . . . . . 4
2.3 Metrics . . . . . 4
2.4 Benchmarks. Tests, and Test Suites . . . . . 4
2.5 Questionnaires . . . . . 4
2.6 Decision Aids . . . . . 4
2.7 Monitored Experiments . . . . . 4

3.0 ASSESSOR CAPABILITIES . . . . . 5
3.1 Near-Term Assessor Acquisition Candidates . . . . . 5
3.1.1 Compilation System Evaluators . . . . . 6
3.1.1.1 Compiler Evaluators . . . . . 6
3.1.1.2 Code Generation Evaluators . . . . . 7
3.1.1.3 Program Library Evaluators . . . . . 7
3.1.1.4 Runtime Evaluators . . . . . 7
3.1.2 Target Code Generation Aids and Analysis Toolset Evaluators . . . . . 7
3.1.3 Test Systems Assessors . . . . . 7
3.1.4 CAIS (Common APSE Interface Set) Evaluation and CAIS Validation Assessors . . . . . 8
3.1.5 Requirements/Design Support Evaluators . . . . . 8
3.1.6 Configuration Management Support Evaluators . . . . . 8
3.1.7 Distributed Systems Development and Runtime Support Evaluators . . . . . 8
3.1.8 Distributed APSE Evaluators . . . . . 8
3.1.9 "Whole APSE" Assessors . . . . . 8
3.1.10 Transportability Evaluators . . . . . 8
3.1.11 Methodology Support Evaluators . . . . . 9
3.1.12 Interoperability Evaluators . . . . . 9
3.1.13 Multilingual APSE Evaluators . . . . . 9

APPENDIX A ACRONYMS . . . . . 10
APPENDIX B E&V TEAM REQUIREMENTS WORKING GROUP MEMBERSHIP . . . . . 11

## 1.0 INTRODUCTION

The Tools and Aids Document is the result of deliberations of the Requirements Working Group (REQWG) of the Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Team concerning technology required to evaluate and validate APSEs and their components. This document is a reflection of the APSE E&V Requirements Document and the state of current APSE tools. It also reflects views on the subject which were obtained from a number of surveys conducted among the APSE E&V Team and appropriate ARPANet-MILNet Interest Groups.

### 1.1 Purpose of the Evaluation and Validation Task

The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and their components and to determine their conformance to applicable standards (e.g., DOD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the Evaluation and Validation (E&V) Task is to provide a focal point for addressing the need by (1) identifying and defining specific technology requirements, (2) developing selected elements of the required technology, (3) encouraging others to develop some elements, and (4) collecting information describing existing elements. This information will be made available to DoD components, other government agencies, industry, and academia.

Validation is the process of determining conformance of an APSE or APSE component to existing standards. For example, Ada compilers are currently required to undergo validation by the Ada Validation Organization (AVO) to insure conformance to the Ada language standard (MIL-STD-1815A). In the future, validation may encompass additional standards such as the Common APSE Interface Set (CAIS) developed by the KAPSE (Kernel APSE) Interface Team/Industry and Academia (KIT/KITIA).

Evaluation is the process of assessing characteristics or attributes of an APSE or APSE component for which there may or may not be standards. Examples of such attributes include usability, efficiency, and maintainability. In the absence of standards, such attributes are free to vary across different APSE implementations. Consequently, these attributes are of interest to users when selecting between APSEs because they contribute to, or detract from, overall APSE quality and suitability for different applications or methodologies. Even in cases where standards do apply to APSE components (e.g., MIL-STD-1815A and Ada compilers), evaluations will be used to supplement information gained during validation processes.

It is anticipated that the primary benefits of E&V will be to encourage the development of quality APSEs, to promote interoperability and transportability, and to provide users and developers with a uniform and comprehensive means for assessing and selecting APSEs suitable for their specific applications and methodologies.

## 1.2 The Need for E&V Technology

Technology for the assessment of APSEs and APSE components (tools) is needed because of the difficulty in assessing APSEs and because of the importance of the decisions made based on these assessments. The importance of an APSE selection is evident when one considers the large, critical, Ada-based systems to be developed in the coming years. The effectiveness, reliability, and cost of these systems will be strongly influenced by the environments used to develop and maintain them. From the point of view of a software developing organization, the decision to select an APSE can be an important investment decision with long-lasting influence on a number of projects and the organization's method of operation, training, and competitiveness. From the point of view of a software maintenance organization, the environment used will strongly influence the organization's effectiveness as well as the cost of its operations and training.

The difficulty of assessing APSEs and tools exists for several reasons. First, an APSE represents very complex technology with many elements, which can be assessed individually or in combination. Second there is a confusing diversity of choice with respect to individual tools, tool sets, or "whole APSEs"; and there are a number of ways of viewing APSEs; see Chapter 3 of the E&V Reference Manual. Third, the state of the art of APSE architecture and of some categories of tools (e.g., graphic design tools) is undergoing rapid change. Finally, there is a lack of historical data relevant to APSEs, partly because of the general page of technological change and partly because we are dealing with Ada, a relatively new implementation language. E&V technology provides methods and techniques to overcome these difficulties and provides a basis for determining performance and other attributes of APSEs.

In addition to the need for assessment technology itself, there is a need for information about this technology. Potential buyers and users of APSEs and tools need a framework for understanding APSEs and their assessment, as well as information about specific assessment techniques. Similarly, vendors of tools and APSEs need to be aware of the deficiencies of current products, as well as the criteria to be used in the assessment of future products. Such awareness on both sides, expressed in a common terminology, should speed up the evolution of better software development environments.

## 1.3 Purpose of the Tools and Aids Document

There exists a critical need to support the Ada community, including compiler and tool builders as well as Ada users and educators, in the selection and improvement of APSEs and APSE components. The purpose of this document is to provide pertinent information to those agencies willing and able to fund the development of E&V Technology (these agencies include, but are not limited to, the AJPO, STARS, JIAWG, Major Program Offices of the services, etc.). To this end the Tools and Aids Document identifies the community's E&V technology needs, provides definitions of those technology needs, and prioritizes them in order of their relative importance.

In order to simplify the discussion, the term "assessor" is used in this document to refer to both tools and aids for use in evaluation and/or validation. APSE component assessors are defined in Section 2 of this document, and range through guidelines, checklists, benchmarks and experimental procedures. Acquisition of assessors includes incorporation of existing capabilities into the E&V assessment set, purchase of commercial products, or development of needed technologies and implementations of these technologies for APSE component assessment.

The Tools and Aids Document provides amplification from the APSE E&V team on:

- The kinds of assessors to acquire,
- The prioritized ordering of assessor acquisition,
- The rationale for the priorities.

## 1.4 Scope

The APSE E&V Requirements Document identifies APSE attributes and functionality that are perceived to require evaluation and/or validation (ie., assessment). The Tools and Aids Document identifies the kinds of assessors that need to be acquired to perform the evaluation and/or validation of the functions. The document is intended to provide the AJPO and other potential sponsors with a reference for use in the allocation of resources, RFP preparations, and source selection for Tools and Aids to support the tasks of APSE E&V.

The Tools and Aids Document is a pragmatic guide to assessor acquisition based on the APSE functions available which need evaluation and/or validation, and on the technologies and implementations of these technologies available as APSE function assessors. Through its prioritization of needs, the document emphasizes near-term acquisition of assessors. The document also provides guidance for long term assessor acquisition strategies by identifying some of the assessors that require further development.

## 2.0 TYPES OF ASSESSORS

Assessors are the mechanisms for providing information about certain characteristics of APSE components, including functionality, performance, maturity, and the suitability of documentation.

Types of assessors include, but are not limited to, the following:

- Requirements and Specifications
- Guidelines
- Metrics
- Benchmarks, Tests, and Test Suites
- Questionnaires
- Decision Aids

- Monitored Experiments

Each assessor type may be implemented in a number of ways, such as automated tools, tests and batteries of tests, and/or manual procedures.

## 2.1 Requirements and Specifications

Requirements and specifications enumerate the necessary functionality, characteristics, or performance of an APSE function or tool. These may include measures that may be made quantitatively by other assessors or by judgement alone. As standards are adopted for various APSE functions, they will be included here and used as the basis for the validation of the designated functionality.

## 2.2 Guidelines

Guidelines provide recommendations for the use or construction of an APSE function or component. Furthermore, guidelines may describe characteristics or qualities the tool should have.

## 2.3 Metrics

Metrics provide quantitative data about selected characteristics of an APSE or an APSE component.

## 2.4 Benchmarks, Tests, and Test Suites

Benchmarks are standard tests used to measure the execution performance or acceptability of an APSE function. Benchmarks may test one specific aspect of an APSE function, or may test a number of functions. Tests and Test Suites are instruments used to measure the performance, correctness, or other characteristics of APSE functions.

## 2.5 Questionnaires

Questionnaires are used to gather data not easily attainable by examination of the APSE or APSE component itself. Examples of such data might include historical information, typical usage scenarios, implementation strategies, enhancement perceptions, problems reports, etc.

## 2.6 Decision Aids

Decision aids allow a user to assess an APSE function from a particular point of view. Decision aids may combine the results of a number of assessors, each of which is weighted based on its usefulness for the view being considered.

## 2.7 Monitored Experiments

Monitored experiments, based on model projects involving an aggregation of APSE functions or tools, can be performed on APSEs or APSE components to gather data in a systematic and controlled manner. These experiments can be used for both qualitative and quantitative assessments of the functionality, usability, and performance, as well as for the more informal characteristics of APSEs.

## 3.0 ASSESSOR CAPABILITIES

A number of APSE function assessor capabilities have been identified as being important for providing an APSE E&V capability. Recommendations for near-term assessors are found in Section 3.1 below. The premise for near term attention is that E&V capabilities can be acquired by assembling existing assessors or by developing the assessors using existing, proven technology. They are ordered by acquisition priority determined by the E&V team.

Long term E&V capabilities require additional development of technology, or the development of more detailed requirements. Some long-term evaluator capabilities are listed in Section 3.2 below. The list should not be considered exhaustive, in that a number of other specific assessors will require development.

## 3.1 Near-Term Assessor Acquisition Candidates

The following prioritized list of assessment capabilities is recommended. Priorities are based on the importance to the development of mission critical software, the availability of the APSE functions to be evaluated, and the technical feasibility of developing the assessor.

1. Compilation System Evaluators
2. Target Code Generation Aids and Analysis Toolset Evaluators
3. Test Systems Evaluators
4. CAIS Evaluation and Validation Assessors
5. Ada Design Support Evaluators
6. Configuration Management Support Evaluators
7. Distributed Systems Development and Runtime Support Evaluators
8. Distributed APSE Evaluators
9. "Whole APSE" Evaluators
10. Transportability Evaluators
11. Methodology Support Evaluators

12. Interoperability Evaluators

13. Multilingual APSE Evaluators

### 3.1.1 Compilation System Evaluators

This section includes Compiler Evaluators, Code Generation Evaluators, Program Library Systems Evaluators and Runtime Systems Evaluators.

For the purposes of this document, the compilation system is defined as those APSE components which are Ada-specific and are required for validation: the compiler, the code generator, the program library management system, and the runtime support system. While each of these components have characteristics which should be assessed individually, the assessment of their combined functionality will be more critical to the successful development of mission critical software.

The immediate criticality of assessor development for these four compilation system components is made evident by the many large-scale projects with requirements for the use of Ada which are presently being procured or are planned for near-term procurement. These large scale projects include the Strategic Defense Initiative, the NASA Space Station, the STARS program, Army Tactical Command and Control System, Army WIS, and the ATF, ATA and LHX programs being evaluated for common avionics systems under the auspices of the Joint Integrated Avionics Working Group (JIAWG). The successful performance of these systems depends upon the quality/extent of code generation support and execution support found in the compilation system. APSE development teams are in the process of trying to determine which products are of sufficient quality to support the development of their complex systems. Tools to assist in these evaluations are needed now.

#### 3.1.1.1 Compiler Evaluators

Compiler evaluators provide capabilities which measure areas such as compiler performance, code and/or time optimizations, implementation of real-time embedded programming features, usability, completeness of documentation, and completeness of configuration management and control practices. The issues being probed include how "good" are the compilers, and in what ways are they good.

It is recognized that the Ada Compiler Evaluation Capability (ACEC) contract is an attempt to provide the evaluation technology required for an Ada compiler. Available funding levels have restricted the scope of that effort to something significantly less than what is actually needed, so there is an immediate need to allocate additional funds for the procurement of compiler evaluation technology which is not found in the ACEC. The current ACEC acquisition is restricted to the provision of a test suite which can measure object code execution efficiency of Ada compilation systems.

Additional urgent requirements exist for the assessment of compiler performance, real-time embedded programming features, useability, symbolic debugging support, and other aspects of compilation that cannot be directly assessed through examination of object code.

### 3.1.1.2 Code Generation Evaluators

The generation of efficient code for embedded targets such as MIL-STD-1750A, 68020, 80286, etc is of prime importance in the compilation system. Assessors should evaluate both target and native host code generators for performance, efficiency, usability, modifiability, and completeness of documentation.

### 3.1.1.3 Program Library Evaluators

Program Library Management Evaluator Systems include evaluators to verify characteristics such as the completeness of documentation, performance, efficiency, functional capabilities, and usability of APSE supplied program library management systems, as examples.

### 3.1.1.4 Runtime Evaluators

Runtime evaluators are those which measure characteristics such as the performance, efficiency, and usability of the runtime system. These would also include evaluation of the completeness of documentation and configuration management and control practices of the runtime system.

Ada Runtime evaluation is needed to evaluate the performance of target runtime support systems (RTSS), typically a runtime executive and library of runtime services. Mission critical software is particularly sensitive to timing and efficiency requirements as well as the amount of code needed for RTSS. The ability to make crucial decisions about the capability of a particular Ada RTSS to meet the demands of the application often determines the success or failure of a mission critical project. Providing sound evaluators for RTSS is essential to the success of both Ada and the mission critical systems to which it is applied. Performance measures will include the required space of the run time software. An important factor in RTSS space requirements is the ability to factor out unused services to reduce the support library size.

### 3.1.2 Target Code Generation Aids and Analysis Toolset Evaluators

These evaluators will provide tools to evaluate host-target system cross-assemblers; host-based target linkers and loaders; host-based target system instruction-level simulators/emulators; host-based target-code symbolic debuggers; and host-based target system instrumentation interfaces which provide visibility into target processes during mission critical software execution.

### 3.1.3 Test Systems Assessors

These assessors will examine the ability of the APSE or APSE component to support and facilitate the planning, development, execution, evaluation and documentation of tests of mission critical software.

## 3.1.4 CAIS (Common APSE Interface Set) Evaluation and CAIS Validation Assessors

CAIS assessors provide measurements about how "good" the CAIS is.

The CAIS evaluation assessment capability is to be developed to assure that the implementations of the CAIS will provide acceptable performance and other characteristics not covered by validation.

CAIS validation assessors will determine if the CAIS is in conformance with the DoD Standard.

## 3.1.5 Requirements/Design Support Evaluators

These evaluators will measure the suitability and effectiveness of various software definition, specification, and design tools. This will specifically include evaluators of Ada Program Design Language (PDL) implementations and/or guidelines in the use of Ada as a PDL.

## 3.1.6 Configuration Management Support Evaluators

These evaluators will examine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the contents of software systems. This will include monitoring the status, preserving the integrity of released and developing versions, and controlling the effects of changes throughout the lifetime of the software system.

## 3.1.7 Distributed Systems Development and Runtime Support Evaluators

These evaluators will assess the ability of the APSE or APSE components to support software development for distributed processing systems, and to provide runtime support for distributed processing systems.

## 3.1.8 Distributed APSE Evaluators

These evaluators will assess the ability of two or more distributed APSEs to communicate in cooperative ways in supporting the development of mission critical software at diverse geographical locations.

## 3.1.9 "Whole APSE" Assessors

Assessors which assess APSE macro characteristics, such as the overall performance, efficiency, usability, completeness of documentation, and configuration management and control practices of the entire APSE system.

### 3.1.10 Transportability Evaluators

These evaluators assess the ease with which an APSE or APSE component can be moved to other specified hosts or APSEs without change in functionality. Transportability is measured as the degree to which this relocation can be accomplished without reprogramming.

### 3.1.11 Methodology Support Evaluators

These evaluators assess the extent to which the APSE or APSE components support software development methodologies.

### 3.1.12 Interoperability Evaluators

These evaluators assess the ability of an APSE to exchange database objects and their relationships with other specified APSEs in forms usable by APSE components and user programs without conversion. Interoperability is measured as the degree to which this exchange can be accomplished without conversion.

### 3.1.13 Multilingual APSE Evaluators

These evaluators assess the extent to which the APSE or APSE components support the analysis/development of mission critical software where multiple source languages are involved. Multiple source language support includes the construction of Ada programs which interface to units written in other languages; and/or the support for the maintenance of files of programs not written in Ada (such as documentation); and/or support for programs written completely in languages other than Ada (e.g., existing programs written in FORTRAN, Pascal, C, LISP, etc.).

## APPENDIX A

## ACRONYMS

| | |
|---|---|
| ACEC | Ada Compiler Evaluation Capability |
| AJPO | Ada Joint Program Office |
| APSE | Ada Programming Support Environment |
| AVO | Ada Validation Organization |
| CAIS | Common APSE Interface Set |
| E&V | Evaluation and Validation |
| JIAWG | Joint Integrated Avionics Working Group |
| KAPSE | Kernel Ada Programming Support Environment |
| KIT | KAPSE Interface Team |
| KITIA | KAPSE Interface Team Industry/Academia |
| NASA | National Aeronautics and Space Administration |
| PDL | Program Design Language |
| REQWG | Requirements Working Group |
| RFP | Request For Proposal |
| RTSS | Runtime Support System |
| STARS | Software Technology for Adaptable, Reliable Systems |
| WIS | WWMCCS Information System |
| WWMCCS | World Wide Military Command and Control System |

## APPENDIX B

## E&V TEAM REQUIREMENTS WORKING GROUP MEMBERSHIP

| | |
|---|---|
| Becky Abraham | Wright-Patterson AFB |
| Jerry Brookshire | Texas Instruments Corporation |
| Mike Burlakoff | Southwest Missouri State University |
| Peter Clark | TASC |
| Bard Crawford | TASC |
| Dan Eilers | Irvine Compiler Company |
| Linda Elderhorst | NATC |
| Fred Francl | Sonicraft, Inc. |
| Greg Gicca | Sanders Associates |
| Marlene Hazle | MITRE Corp. |
| Alan Impicciche | Naval Avionics Center |
| Elizabeth Kean | Griffiss AFB |
| Pat Lawlis | AF Institute of Technology |
| Tom Leavitt | Boeing Military Airplanes |
| Ronnie Martin | Purdue University |
| Sandi Mulholland | General Dynamics |
| Vicki Rhoden | Wright-Patterson AFB |
| Helen Romanowsky | Rockwell International |
| Mary Ann Tompkins | Lockheed Corporation |
| Nelson Weiderman | Software Engineering Institute |

## APSE E&V Task Activities on Evaluation and Validation of CAIS and CAIS_A

06 June 1988

Presentation authors :

Gary McKee
Jeff Facemire
Dr. Bard Crawford
Dr. Tim Lindquist
Raymond Szymanski

---

## E&V Task Purpose

To provide a focal point for addressing community needs for E&V Technology -- to assess APSE's and their components

Evaluation -Assessment of performance and quality

Validation - Assessment of conformance to a standard

ACTIVITIES

1) Identify and define requirements
2) Develop selected assessment elements
3) Encourage others to develop other elements
4) Collect and disseminate information on this technology

## E&V Task Products

E&V REFERENCE SYSTEM

- E&V Reference Manual -- Mar 1988
- E&V Guidebook -- July 1988

Ada COMPILER EVALUATION CAPABILITY (ACEC)
-- Aug 1988

CAIS IMPLEMENTATION VALIDATION CAPABILITY(CIVC)
-- Feb 1989

E&V Task Public Report
-- annually

## CIVCWG

CAIS Implementation Validation Capability Working Group of the E&V Task

Charter

To provide a forum for review of the CIVC project and to provide technical input and feedback to the CIVC contractor.

Activities

- Receive presentations from the CIVC contractor (quarterly)
- Evaluate the CIVC taxonomy and recommend extensions and modifications
- Examine and report on validation issues related to the DoD-Std-1838

## What is the CIVC ?

- Collection of Ada test programs and administration tools used for determining conformance of a CAIS implementation to DoD-Std-1838.

- Used by :
  - A CAIS validation organization for issuing validation certificates to conforming implementations;
  - CAIS implementation vendors to determine completeness and readiness of their implementation.

- Should be capable of validating all legal CAIS implementations. Portability should be maximized.

## CIVC Approach

- Taxonomy
  - enumerates and organizes CAIS entities

    (Eg., nodes, relationships, attributes, lists, etc.)

- Framework
  - Automated support for managing complexity and volume
  - Use taxonomy structure to organize test objectives on particular CAIS entities
  - Provides traceability between 1838 sections and derived test objectives

- Implementor's Guide
  - Extracted from the framework information
  - Details in hardcopy form, all test objectives and the scenarios for the test objectives
  - Aids both CIVC test writers and CAIS implementors

## CIVC Status

- Presentations to the CIVCWG on a quarterly basis
- Initial build of the CIVC due in first quarter of 1989
- Coverage of the initial build includes :
  - Node Model
  - Process Management
  - Other areas

## CIVC Major Issues

- Determining completeness of the CIVC.
  - Test objectives to the 1838 specification
  - Test cases to the test objectives
- Determining effectiveness coverage of the CIVC.
- Management of the large amount of CIVC information
  - Test objectives
  - Test scenarios
  - Test code
  - Mapping to the corresponding 1838 references
- Enhancing upgradeability of the CIVC for 1838 to that needed for proposed DoD-Std-1838 revision A (1838A).

## SEVWG

**Standards Evaluation and Validation Working Group of the E&V Task**

**Charter**

**To provide a forum for the issues and approaches to evaluating and validating implementations of APSE related standards.**

**Activities**

- **Identify standards of interest**
- **Analysis of October 1986 release of the CAIS standard (now DoD-Std-1838)**
- **Analysis of the proposed standard 1838A**

## Preliminary Analysis of the CAIS

- **Approaches to generating tests**
  - **Test predicates**
  - **Abstract machine analysis**
- **Validating implementation dependencies**
  - **Pragmatic limits**
  - **Access control**
  - **Validation triplets**
- **Tough problems for validation**
  - **Asynchronous facilities**
  - **Device input and output**
  - **Minimum configuration implementations**
- **Criteria for evaluating CAIS implementations**
  - **Performance**
  - **Others : maturity, integration with host, supportability, etc.**

## Open Issues in analysis of 1838A

- **Test case selection criteria**
- **Test effectiveness**
- **Test correctness**
- **Validating CAIS rev.A**
  - **Resource control facilities**
  - **Typing**
  - **Transactions and triggering mechanisms**
- **Effort to develop a validation suite**
- **Effort to validate an implementation**

## E&V Reference System

**The Reference Manual will help users to :**

- **Gain overall understanding of APSE's and approaches to assessment**
- **Find useful information -- Terminology, Definitions and Relationships**
- **Find assessment criteria/metrics and "pointers" to specific evaluation or validation techniques**

**The Guidebook contains :**

- **Descriptions of specific instances of assessment techniques**

## Open Issues

- **How to evaluate APSE's as a whole rather than as the sum of their parts ?**

- **What European developed APSE assessors, or APSE component assessors, should be included/synopsized in the Guidebook ?**

# MINUTES

# OF THE

# EVALUATION AND VALIDATION TEAM

1-4 DEC 1987

The task for the Evaluation and Validation of Ada* Programming Support Environment (APSEs) is sponsored by the Ada Joint Program Office (AJPO).

## TABLE OF CONTENTS

SECTION PAGE

1.0 Tuesday, 1 December 1987 . . . . . . . . . . . . . . . . . . 1
1.1 Introduction . . . . . . . . . . . . . . . . . . . . . . . 1
1.2 Ada Performance Issues . . . . . . . . . . . . . . . . . . 1
1.3 An Investigation of Run-Time Performance of Data Conversion in Ada Programs . . . . . . . . . . . . . . . 4
1.4 An Introduction to the Mission of the Naval Air Test Center(NATC) . . . . . . . . . . . . . . . . . . . . 6

2.0 Wednesday, 2 December 1987 . . . . . . . . . . . . . . . . . 9
2.1 Opening Remarks . . . . . . . . . . . . . . . . . . . . . 9
2.2 Common Ada Programming Support Environments (APSE) Interface Set (CAIS) Implementation Validation Capability (CIVC) Status Report . . . . . . . . . . . . 9
2.3 Ada Compiler Evaluation Capability (ACEC) Status Report . . . . . . . . . . . . . . . . . . . . . . . . . 12
2.4 E&V Reference System Status Report . . . . . . . . . . . . 18

3.0 Friday, 4 December 1987 . . . . . . . . . . . . . . . . . . 20
3.1 General Business . . . . . . . . . . . . . . . . . . . . . 20
3.2 Introduction to the Joint Integrated Avionics Working Group (JIAWG) . . . . . . . . . . . . . . . . . 20

LIST OF ATTENDEES . . . . . . . . . . . . . . . . . . . . . . . 22

## 1.0 TUESDAY, 1 DECEMBER 1987

### 1.1 Introduction

E&V Team chairman, Raymond Szymanski, welcomed members of the E&V Team to San Diego for the December meeting. He announced that visit requests have been received by Lloyd Styles at Navy FCDSSA, where the Wednesday through Friday sessions of the December meeting will be held. A list of those whose visit requests had been processed was distributed. Ray then introduced the featured speakers for the Tuesday afternoon session: Dr. Dan Eilers, Dr. Robert Fainter, and Linda Elderhorst.

### 1.2 Ada Performance Issues

Dr. Dan Eilers
Irvine Compiler Corporation

Dr. Eilers introduced his presentation by stating that many performance issues that he would address are also addressed by the Ada Compiler Evaluation Capability (ACEC) taxonomy. His intent was to focus on some of the issues that are not being incorporated into the first iteration of the ACEC. Dan identified six areas that address performance of Ada compilers:

1. Full use of the target machine's capabilities
2. Full implementation of Chapter 13 language features
3. Capacity issues
4. Static elaboration of constant aggregates
5. Optimizations (this is particularly important on 1750As)
6. Implementation trade-offs

In discussing full use of a target machine's capabilities, Dan emphasized that the compiler must support the data types specified for its target and the complete variety of instructions available. A good compiler will also support both the integer register set and floating point register sets, and all memory and processor modes available on the machine. Dan stated that in the area of instructions, Ada is deficient in addressing logical operations and transcendental functions. He added that many Ada compilers do not support floating point operations.

Capacity is an important aspect of usability on embedded processors. How a compiler handles such operations as memory reclamation is critical. Unfortunately, the Ada Compiler Validation Capability (ACVC) does not adequately test for this. In some compilers, capacity optimizers are affected by complicated expressions, numbers of statements or declarations, data structures, and use of tasking. Dan explained that there are two basic ways that a compiler approaches memory reclamation: by using dynamic sized objects, and by tasking related dynamic memory (task activation records.)

Static elaboration of constant aggregates is a problem because Ada has no mechanism for addressing Read Only Memory (ROM). Aspects of constant aggregates that contribute to the problem involve the address attributes, unchecked conversion, components of other constant aggregates, nested constant aggregates, allocators initialized to constants, and foldable expressions. If a compiler can efficiently handle constant aggregates, a program size can be greatly reduced. An example of an application program that uses a lot of data (constant aggregates) is a cockpit display.

Dan expressed the opinion that, if a compiler handles memory reclamation and static elaboration well, optimization is less necessary. Standard optimization, Ada specific optimization, handling of redundancies, and dead code elimination are all aspects of optimization. Dan expressed the opinion that the ACEC is addressing this area adequately. Aspects of redundancy being examined include literals, image/value attributes, descriptors for unconstrained parameters, and within/between compilation units. Dead code elimination addresses compiler generated dead code, user generated dead code, and dead code in run-time systems for features not used.

For the remainder of his presentation, Dan focused on areas of the Ada Language Reference Manual (LRM) that are unclear or inefficient. General areas addressed were:

- The case statement
- Records with holes
- Discriminant records with (multiple) dynamic components
- Array descriptors
- Subtypes
- Generics
- Library units
- Exceptions
- Tasking

Dan explained that the CAIS statement is understood by programmers and usually is implemented via a jump table or, less frequently, decision tree. The jump table in Ada causes problems with space and time whereas the decision tree is efficient only for sparse tables because of capacity requirements. Validation tests in this area require the alternate implementation.

Records with holes are a problem because the compiler can not lay them out sequentially. Ada's concept is implicit assignment for all types and implicit comparison of records for equality. Records with holes have to be compared field by field. One way of handling this is to fill all holes with a zero for easy block comparison. The trade-off issue centers around whether block comparisons will be done more often than allocation of records.

Discriminant records with dynamic components force one to choose whether to store the components in the record (preferably at the end of the record) or store them on a heap. The trade-off is easy component access vs. easy copy/ compare.

Ada allows fairly advanced array types: dynamic arrays, unconstrained arrays, etc. At issue is the array descriptor that stores upper and lower bounds of the array. One can store them with the array or separately. There are disadvantages to either option which force the implementor to choose between slicing efficiency or reference efficiency.

Adding to the list of issues, Dan identified subtypes as another problem area. Ways of constraining integer types are by using a subtype that is compatible to the base type or by using a derived type that is incompatible with the base type. The trade-off issue is conversion efficiency vs. storage (packing) efficiency.

Generics forces the implementor to choose whether to share the body of the generic between instantiations or do a macro expansion of the generic. The trade-off is space efficiency in the case of shared vs. time efficiency in the case of macro expansion. Most compilers use the macro expansion (copy) method, but some applications are better supported by the shared method. Text_I/O routines, which are generic, are better supported by the shared method. Dan stated that a potential answer to this issue is use of a pragma that will tell the compiler whether you choose to share or copy the generics. This means that the compiler would have to implement both mechanisms. The ACEC would not know the name of this pragma and would therefore have difficulty in evaluating the handling of generics.

Library units are an issue because subunits are compiled separately. The optimizer, which looks at the enclosing unit of a subunit, does not know which variables of the enclosing unit are referenced by the subunit. Therefore, optimization is inhibited. A potential solution could be a compiler option that permits compilation of subunits as include files. Modularity of library units for development is good, but it creates problems.

Exceptions are an issue in that the implementor must decide whether it is better to have continual overhead, or overhead only when an exception is raised.

Tasking is a major area of concern. The implementor must decide how to use float/integer registers, and which scheduling algorithm to use: run until blocked or time slicing.

The discussion prompted comments from team members concerning the temptation present for compiler vendors to attempt to build compilers that do well on the ACEC tests, but may not support real applications. John McBride asked what the ACEC should do in terms of addressing the various implementation options being presented. Dan stated that these are all issues to which there is no easy answer. The user will have to decide which method of dealing with these issues is best suited to his application. Ideally, the ACEC will represent a broad spectrum of applications and will rate compilers on all alternative ways of addressing these problem areas. Dan's goal in this presentation was to raise consciousness concerning these issues. The Ada community will need to know what the ACEC addresses and what it does not address.

In conclusion, Dan observed that until there is an agreed upon preferred implementation strategy, or combination of strategies, or pragmas for the problem areas, it will be virtually impossible to rate one implementation higher than another in these major areas. For this reason, it is important to make the ACEC tests as unbiased as possible.

## 1.3 An Investigation of Run-Time Performance of Data Conversion in Ada Programs

Dr. Robert Fainter
Arizona State University

Dr. Fainter reported on an investigation of run-time performance of data conversions in Ada programs. This work, which was supported by a contract with McDonnell Douglas, focused on development of Ada programs for 1750A targets. The programs dealt with data conversion in real time. The purposes of the study were to recommend techniques for improving run-time performance with data conversion, and to evaluate a specific Ada compiler.

Systems used were the Micro VAX II with VAX/VMS 4.5 operating system, the VERDIX Ada Development System (VADS) version 5.4, and a RAID debugger, which was used as a run-time executive. RAID simulates execution time of 1750A instruction sets, thus permitting observation of 1750A execution times of programs run on the Micro VAX.

This project was performed as a statistical study with run-time data being collected and a statistical analysis of the data being performed. The F statistic and analysis of variance technique were used, which allowed simultaneous comparison of several means. It was necessary to find the mean execution time of several different Ada programs and to make a statistical comparison of these means.

Preliminary analysis determined the data types to be used: fixed point and floating point. Fixed point data types consist of a fixed number of fractional digits. The 1750A does not implement fixed point, so the Ada compiler had to do the implementation in software. The two integer sizes usable in the 1750A for fixed point are 16 and 32 bit. In this study, it was necessary to explain how to choose the delta to get fractional bits and how to choose the range so that one can get long fixed point or short fixed point. After deciding on the number of delta bits or the number of fractional bits wanted in the fixed point, the team chose a range, so that the compiler would select a word length to represent that range. It was necessary to specify deltas and ranges so that the compiler would select either 16 or 32 bit representations for the fixed point data.

Floating point is the native data type on 1750As, and there are 32 and 48 bit lengths for the floating point data.

The experiment focused on comparisons of converting among various fixed point data types, and among floating point data types, and between fixed and floating point data types. It was necessary to decide what the significant data conversions were. Preliminary work involved selecting specific data conversions. Converting from fixed to floating point, changing precision from 16 to 32 bits or 32 to 48 bits, and changing the delta in fixed point processing were identified as significant. With this data available, the task was to design data conversions that would be representative of these significant data conversions.

The compiler studied uses the 1750A integer data type for fixed point implementation. Because there were two integer lengths, two precisions were needed and two deltas were needed. To study conversions among fixed point types, four fixed point types were used: single precision with N delta bits, single precision with M delta bits, and double precision with both N and M delta bits.

Two floating point types studied required two precisions on the 1750A. No ranges were chosen for the floating points because it appeared that the difference in ranges was constraint checking rather than data type.

The six data types were compared against programming techniques. One concern was determining how to write programs for the compiler so as to get the most efficient processing. The three programming techniques chosen to do the conversions were:

- Comparing data conversons within arithmetic expressions (explicit conversions)

- Converting directly into temporaries, and performing the arithmetic within the temporaries

- Converting all fixed point data into floating point data and doing all arithmetic in floating point, then converting back to fixed point.

It was believed that the third method would be the least efficient. Floating point typically takes a long time, but this proved to be faster in some cases than doing the arithmetic in fixed point.

The statistical experiment consisted of comparing data conversion conditions and programming techniques. The result was in a two-way factoral design with 14 levels of factor a and three levels of factor b (programming technique). Each cell had three programs so that a mean execution time could be calculated. In all, 126 Ada programs were written to perform data conversion. In attempting to compile some conversion conditions, internal compiler errors were generated. In some instances converting the delta, but keeping the same precision, could not be compiled.

After programs were compiled, conversions were performed on nine data types. The best conversion time was for converting long floating point to short floating point. The worst conversion was long floating point to short fixed point with 1 delta bit. Second best time was for converting short fixed point to short floating point. One conclusion made was that, in many instances, it takes less time to do a lot of manipulation if you convert all fixed point into floating point and do the arithmetic in floating point, and, if necessary, convert back to fixed point. Perhaps the reason for this is that fixed point processing is implemented in the software rather than in the hardware on the 1750A. It was noted that there was not a great variance from the best conversion time to the slowest conversion time. Bob speculated that the reason for this was that all the programs used were similar in size and number of instructions.

Fred Francl asked if it might not be even more efficient to convert from fixed to floating point in the compiler and avoid all fixed point arithmetic. Bob observed that there are relative error bounds in floating point but absolute error bounds in fixed point, some applications might require use of absolute error bounds. Other applications may permit the relative error bound inherent in floating point and therefore make use of the greater speed available. Along with the speed would be added space in the run-time support library.

The reason for conducting the experiment is that the application for which McDonnel Douglas collected data requires precision of timing and synchronization, thus microseconds are important.

Bob emphasized that the experiment focused on converting data types, not on timing the actual computation in either fixed or floating point. There was great interest expressed in obtaining this information as well.

The Team expressed interest in hearing about more controlled experiments of this nature, and thanked Bob for his presentation.

## 1.4 <u>An Introduction to the Mission of the Naval Air Test Center (NATC)</u>

Linda Elderhorst
Naval Air Test Center, Patuxent River, MD

The mission of the Naval Air Test Center (NATC) is to conduct test and evaluation of vehicles, systems, and subsystems. NATC is a principal contractor full scale development (FSD) testing facility, and supports research, development and training in test techniques, instrumentation, and test facilities. NATC hosts more than forty tenant activities.

Flight testing is the primary source of data collection on the effectiveness of aircraft and weapons, but it has limitations. It is costly, dangerous, and cannot reproduce combat conditions, and is inherently a pilot event. It was noted that Patuxent River is the location of the Navy's test pilot school.

Simulation augments flight testing. Traditional simulation activities are offered to pilots before test flights. These activities conform to flight test, enhance safety, inform the pilot of the data they should observe, and offer controlled, repeatable tests. Simulation offers early detection, diagnosis, and documentation of problems.

The increasing complexity of weapons systems has expanded the role of simulation to encompass more than a single weapons system or test article. The need exists to evaluate system mission effectiveness in a mission oriented test environment that is covert and secure. This requires a facility designed for integrated weapon system testing, for evaluating interoperability, and for providing objective evaluation of human factors.

In order to meet these needs, the Navy has designed the Air Combat Environment Test and Evaluation Facility (ACETEF). This is a unique, evolving facility that is designed for testing integrated weapons systems. Six testing issues are addressed by this concept:

- Test realism (simulate a realistic test environment)
- Threat (War and Reserve Modes - WARM)
- Security
- Integration and interoperability
- Cost
- Complexity

Linda noted that this facility is a test and evaluation center whose role is to find problems, diagnose them, and document them. These activities center around the development of systems before government purchase of the systems.

A major aspect of the activities Linda is involved with is testing of Operational Flight Programs (OFPs). Testing is performed against the stated requirements for both simulation and flight testing. Pilots and other testers must write test reports. Hundreds of test points exist for each subsystem in the weapons system. Even so, the testing methodology is not adequate. Testing focuses on the specification, but does not always address aspects of performance of a weapon system.

ACETEF augments flight testing, which simulates combat environments. The ACETEF facility includes an anechoic chamber, which is a unique, secure, TEMPEST approved test chamber, and several laboratories. This facility has been evolving over the past ten years, and is the most comprehensive, largest facility of its kind located in one place. Linda noted that the Air Force has various testing facilities, but that they are scattered around the country. One laboratory connected with ACETEF is the Manned Flight Simulator Laboratory. This lab provides a system that generates the out-the-window display simulation. The simulation environment has high fidelity cockpits, imaging systems a freedom of motion base, a 40 foot dome, and other high fidelity features. The manned flight simulator will receive interoperational capability in February 1988. Programs scheduled to use the facility are the V-22, F-14D, A-6F, and F-18. Future systems that may use the facility include the ATF/ATA, T-45, and the F-14 tactical decision aide.

Linda stated that ACETEF is a large, complex system and is confronting the transition to Ada. Resources used to run simulations include multiple computer systems, many of which are networked. Linda stated that this creates a unique set of requirements for an Ada Programming Support Environment (APSE). Simulation activities require the management of information within the APSE. Multiple simulations will be run concurrently. Modules that run simulations are dynamically assigned to computing resources. Data to optimize distribution of modules should be available through the APSE. Types of information needed are a list of system services that the module needs, memory allocation, input/output (I/O) channels required for the computer cycles, maximum delay the module can tollerate, etc. The center of this APSE will be executives that assign computing resources. Engineers will use the APSE to write simulation problems.

Members of the Team debated whether such an environment could indeed be categorized as an APSE. The question was asked concerning how needs will change when the move is made from FORTRAN to Ada in terms of tools and their capabilities.

Linda next focused on the PAVE PILLAR concept of avionics architecture. Whereas in the past, testing of the OFP treated the software as a black box, this newer concept of modular, interoperable software calls for classes of generic processors with software that is dynamically reconfigurable on-the-fly within a class of processors. The OFP may be as much as 50 times larger than that of the F-15. This is complex, time critical code, and requires more than black box testing. This leads to the issue of the advent of Ada in APSEs that will make the black boxes transparent. Ada is required for the FSD phases of the new weapons systems, and FSD on the ATF is scheduled to take place in three years. The first of these OFPs will be coming to ACETEF at that time. It is hoped that with an APSE, these OFPs will be more reliable, and improved documentation will aid in writing a more definitive test plan to detect, diagnose, and document the problems.

Linda stated that the facility does some research and development in the area of testing. They are involved in trying to educate the community in the need for new and more complete testing of systems.

In concluding, Linda stated that the next generation avionics architecture and the architecture of the Manned Flight Simulator Laboratory are only two examples of an emerging class of systems which will require a type of APSE designed with these systems in mind. She stated that the help of the E&V Team is needed in defining requirements and communicating them to the Ada community.

At the conclusion of Linda's presentation, the Tuesday meeting of the E&V Team was adjourned.

## 2.0 WEDNESDAY, 2 DECEMBER 1987

### 2.1 Opening Remarks

The E&V Team was welcomed to the Navy Fleet Combat Direction System Support Activity (FCDSSA) by Lloyd Styles. FCDSSA is the life cycle facility for the Navy's ship board computer system software. A group within FCDSSA is responsible for S3 aircraft software maintenance, while another group works with satellite communications software. The bulk of the organization is involved with Navy tactical data systems software support. Lloyd's group works with support software for the CMS2 compiler and a shared environment on the UYK-43.

E&V Team chairman, Raymond Szymanski opened the floor for a discussion of concerns. Several problems were identified concerning the new NET host and potential solutions were discussed. Ray announced that the Team will continue in its present form until after the June meeting. Possible reorganization options will continue to be investigated. Guests Jack Foidl of TRW, Jon Wood and Diane Millars of MITRE were introduced.

## 2.2 Common Ada Programming Support Environments (APSE) Interface Set (CAIS) Implementation Validation Capability (CIVC) Status Report

Jeff Facemire
SofTech

Jeff Facemire opened his presentation with a summary of the areas of development that were addressed during the past three months. These areas included:

- Drafting the Software Requirements Specification. Work in this area during the past three months has included simplifying the complexity of the test administrator. A requirement for test execution parallellism has been added that allows the use of multiple processors.

- Developing a taxonomy to be used for classifying all CAIS entities and operations on these entities. This will aid in determining the overall problem of validation. (This will be ongoing.)

- Cross-referencing tests produced by Arizona State University (ASU) for the normal processing operations of the CAIS Operational Definition (OD) to the taxonomy to determine coverage of the taxonomy and of the ASU tests. (ASU tests for exceptional processing and for List_Utilities were not addressed.)

- Developing a CIVC framework. The framework concept developed out of ideas put forth by the CIVCWG, who identified a need for traceability of test objectives identified in the taxonomy to DOD-STD-1838 requirements.

- Delivering a CIVC presentation to the CAIS Editorial Board at the KIT public review of CAIS Revision A in October.

Jeff noted that the number of identified test objectives for CIVC has grown from 6,000 identified in September to approximately 6,400 which have been identified as of the end of November. These have been primarily in the areas of the static and exceptional portions of the taxonomy.

In discussing the CIVC Software Requirements Specification, Jeff stated that the requirements are being derived from tha activities involved in the validation process. The Structured Analysis and Design Technique (SADT) used by SofTech graphically illustrates these activities and the inputs, controls, outputs, and mechanisms (ICOMS) that are involved with each function or activity. Jeff explained that from a software perspective, the mechanisms can be viewed as the software programs. The steps in the CAIS validation process are:

1. Validation Requests
2. CAIS Implementation pre-validation
3. Analysis
4. CAIS Implementation validation

In response to a question from a Team member, Jeff explained that prevalidation would be a running of the CIVC test suite at the vendor's site, probably without government personnel present.

Performing the actual CAIS implementation validation involves four activities. These are:

- Selecting tests
- Setting up the test environment
- Executing interfaces
- Collecting and summarizing results.

The test administrator software supports only the first and last activities listed. Setting up the test environment involves using those CAIS interfaces outside DOD-STD-1838 which are typically host dependent and implementation dependent. This function is viewed as a manual process done by the user, and possibly supported by the vendor. The Implementor's Guide will detail what is viewed as a common global environment for all of the tests. Jeff stated that it is possible to include tests to ascertain whether the test environment is set up correctly. Gary McKee stated that in his opinion it should be a requirement for the test suite to verify that this environment has been established.

The test suite program supports the third step, executing interfaces. This program consists of several classes of tests. Within the context of CIVC, a class is an independent group of tests that can be run separately, perhaps on a separate processor. Within each class, a user can select individual tests that he wishes to run. Test lists show all tests within a class that can be run. A text editor permits the user to select tests within the class. Dependency tests are automatically included. Each class resets the test environment to its original condition. If, after running a class of tests the environment is not cleaned up, the implementation is considered to have failed that portion of validation. Work will be done in this area to permit the identification of the exact point of failure.

Tim Lindquist asked if the use of separate processes on the same machine or multiple users on the same machine might be an alternative to using separate processors. Jeff expressed the opinion that this might be possible with global preconditions for all test classes if the access rights are modified. Tim suggested that parameterizing tests so that they can be run by different users might be expedient. Tom Leavitt asked about using access synchronization and concurrency on a given system for running the various classes of tests. Jeff observed that the CAIS is designed to be a concurrent environment. It is the requirement to make tests independent of the data objects they operate on. He observed that the access control tests are all in one class. John Stanton observed that it is highly unlikely that a vendor would provide more than one processor to run the tests on, and stated that it is more practical to design the test suite to use multiple streams on a single machine. John McBride emphasized that allowing for concurrency by means of the various classes is a new concept since September, and refinement will take place. Gary McKee commended the concept of classes and requested that further discussion on this topic be deferred until the CIVCWG meeting.

Jeff focused on the remaining steps in performing CAIS implementation validation: setting up the test environment and executing interfaces. He noted that proper test execution would result in a postcondition that can be verified. The end product of performing CAIS implementation validation is a compliance report.

The next topic for discussion was the testing taxonomy, which is divided into three sections: static processing, normal processing, and exceptional processing. The static taxonomy addresses Ada compiler checks that have to be performed on a CAIS implementation, such as typing, proper value of constants, presence of exception names, etc. Normal processing is the execution of interfaces that do not generate exceptions. Exceptional processing tests are those, such as pragmatic limit testing, explicit exception raising, etc., delineated within each of the interfaces. DOD-STD-1838 describes conditions under which given exceptions can be raised. Some unknown situations will have to be addressed as they are encountered. Much detail has been added in the area of exceptions since September.

Some problem areas have been identified in the taxonomy. Although nodes fit in well, nodes and relationships that form trees have yet to be accommodated. Other items, such as attributes have required special handling. The CAIS Editorial Board identified additional areas that need attention: dangling secondary relationships, and multiple ways of raising exceptions. To address these areas, multiple scenarios have been developed to test single test objectives.

A new area of development since September is the framework. The purpose of the framework is to trace requirements from DOD-STD-1838 phrases to CIVC test objectives as organized in the taxonomy, and back. Because of the huge number of requirements in the standard and the numerous test objectives, it is necessary that this traceability function be an automated one.

In addition to building confidence in the CIVC effort, the framework will provide such benefits as allowing CIVC developers to accurately assess the impact to the CIVC test suite of changes to the standard, or changes in interpretation of the standard. It also aids in defining the scope of the problem.

The framework was originally envisioned as a tool that would provide two-way tracing of requirements as stated in sentences and phrases of the standard to test objectives. Jeff stated that not all phrases of the standard can be mapped to test objectives, as the standard contains descriptive material, definitions, etc., that can not be paired to test objectives. The framework, then, was modified to show CAIS entities and operations that can be performed on these entities, plus information as one side of the framework. On the other side are test objectives which use a particular CAIS interface related to each test objective. Connected with a test objective can be several scenarios, or test cases. Each scenario has a specific precondition, postcondition, and parameter set. The information obtained via the framework will feed into the Implementor's Guide.

After presenting the CIVC progress report, Jeff opened the floor for questions. The question was asked concerning the date that CIVC will become operational. Jeff stated that the Preliminary Design Review (PDR) is scheduled in January 1988 and Critical Design Review (CDR) will be in April 1988. The first test suite is due in October or November of 1988. This sparked discussion of CAIS implementations under development, and tools that will run on the CAIS. Questions were raised concerning the usability of DOD-STD-1838 implementations, and upward compatibility between 1838A and 1838.

## 2.3 Ada Compiler Evaluation Capability (ACEC) Status Report

Thomas Leavitt
Boeing Military Airplane Company

Mr. Leavitt reported that the ACEC Critical Design Review (CDR) had been held in mid-November 1987. A preliminary version of the ACEC test suite has been distributed to the Software Engineering Institute (SEI) at Carnegie-Mellon University, where testing is taking place. Errors are being identified via problem reports and Boeing is addressing these errors. Documentation on the ACEC is in progress. Descriptions of test problems have been prepared and are ready for distribution to ACEC Working Group members. These will be included as an appendix in the Version Description Document, and are intended for use in conjunction with other cross-reference lists and the actual test problems.

ACEC documents are in various stages of completion. Final drafts of the Software Development Plan and the Software Requirements Specification were delivered on 28 August 1987. Team members are encouraged to submit comments on these documents which will be considered for inclusion in the next iteration of these documents. The Implementor's Guide and the User's Guide, which are due in January, are undergoing revision. Team comments on ways to improve these documents are sought. Any comments received in time will be considered. Tom noted that some comments have already been received concerning inclusion of examples of step-by-step descriptions of how to perform some operations and a request for more information addressing system dependent modifications needed for calling task interrupts.

Tom reported highlights of the CDR. It was recommended that computing the time between clock ticks might be better than using predefined time specification, since some compilers are not accurate in this area. As a result, ACEC computes the values and will print an error message if the system time specification does not agree with the computed time. This error message will go to the responsible implementor with a request to fix the error. Another recommendation was that ACEC produce additional tests associated with interrupt processing, especially tests associated with raising interrupts while in the interrupt handler for another interrupt. This issue is under study. It was noted that the Ada Language Reference Manual (LRM) does not address this issue in a satisfactory way. Tom stated that interrup test problems are being written to observe whether all interrupts are being processed. If it is determined that some are being ignored in an implementation, an error message will be printed stating that the system is ignoring some interrupts and a time for these will not be reported. Other studies in progress deal with:

- An Artificial Intelligence (AI) frame system under development to process AI work. This program is written as a collection of generic packages instantiated with types that correspond to particular applications.

- Systematic tests for the effects of performance degradation caused by varying numbers of tasks in the tasking system.
- Inclusion of tests for allocation and reclamation of memory.
- Methods of increasing format and output options of some support tools.

A taxonomy has been developed which is to be incorporated into the specifications. A decision was made at the CDR to finalize it after the E&V Team meetings so that pertinent comments can be incorporated. All comments received by Boeing prior to 11 December will be considered. The draft taxonomy presented is based on work done by the ACEC Working Group in September. The draft taxonomy is presented here in its entirety.

ACEC CLASSIFICATION TAXONOMY

TOP LEVEL CLASSIFICATION

I. Compile Time Efficiency

- A test harness will be constructed to measure this; it will be done for UNIX and VMS.

II. Execution Time Efficiency

- Multiple Categories

III. Test for Existence of Language Features

- Not specifically tested for; to be addressed by ACVC.

IV. Code Size Efficiency

- Multiple Categories

V. Usability

- Multiple categories; not specifically tested for.

VI. Capacity Tests

- Multiple categories; not specifically tested for.

INTERMEDIATE LEVEL CLASSIFICATION

II. Execution Time Efficiency

A. Language Feature Efficiency

1. Required

a. Referenced by LRM section

2. Implementation Dependent

   a. Referenced by LRM section.

      - Attributes (LRM Appendix A)
      - Clauses
      - Interrupts
      - Language interface
      - Unchecked programming

B. Optimizations

1. Classical

   - Folding
   - Common subexpression elimination
   - Loop invariant motion
   - Strength reduction
   - Dead code elimination
   - Register allocation
   - Loop merging
   - Boolean expression optimization
   - Algebraic simplification
   - Order of expression evaluation
   - Jump tracing

2. Effect of Pragmas

3. Other

   - Habermann-Nassi transformation for tasking
   - Delay statement optimization

C. Degradation

1. Classical

   - Ackermann
   - Tower of Hanoi

2. Task Performance

   - Task creation
   - Task termination
   - Task abortion
   - Dining Philosophers Problem
   - Task starving

3. Other

   - Unchecked deallocation

D. Trade-offs

   1. Coding Styles

      - Order of Evaluation
      - Default vs. Initialized
      - Order of Selection (rendezvous)
      - Scope of Usage
        - o global
        - o local
        - o shared

   2. Language Feature Selection

      - Referenced by LRM section.

E. Operating System Kernel Efficiency

   1. Task Scheduling

   2. Exception Handling

   3. File I/O

   4. Memory Management/Storage Reclamation

   5. Elaboration

   6. Run Time Checks

F. Application Profile Tests

   1. Classical

      - Whetstone
      - Dhrystone

   2. Ada in Practice (taken from)

      - E-3A Simulator
      - Navigation Algorithms
      - Radar Tracking Algorithms
      - Communication Algorithms

   3. Ideal Ada

      - AI applications
      - Kernel Algorithms

IV. Coding Size Efficiency

A. Expansion Code Size

B. Run Time System Size

C. Executable File Size

Tom explained that the taxonomy addresses only areas of evaluation that are being addressed in some manner by the ACEC Version 1 Test Suite.

In referring to the first item listed in the taxonomy, Tom commented that determining compile time efficiency will be done indirectly by observing the time needed to compile various compilation units of the ACEC test suite. It was observed that specific areas, such as aspects of program libraries, can not be addressed by executing tests. Members of the E&V Team expressed the opinion that the taxonomy should state more clearly that no specific compile time tests are included, but that this data is obtainable as a by-product of compiling the tests in the ACEC suite. This data includes time required to compile and link an executable program that can be down-loaded.

The third item in the taxonomy, Tests for Existence of Language Features, is not specifcally addressed by the current version of the ACEC test suite. Tom explained that there are tests to determine execution time of features, which will reveal whether features have been omitted or included. Comments were made that the Ada Compiler Validation Capability (ACVC) does not supply sufficient information about Chapter 13 Language Feature implementation. Dan Eiler stated that an Appendix F has to be submitted by compiler vendors with their pre-validation which states which features are implemented in a given compiler. This information is to be submitted in camera ready form for inclusion in the Validation Summary Report (VSR). The most expedient way of obtaining this information on a particular implementation is to contact the Ada Validation Facility (AVF), according to John Stanton. It was noted that no specific requirement exists concerning which Chapter 13 features are to be implemented in a compiler; therefore, content differs vastly from one compiler to another.

A question was raised concerning the propriety of including the categories Usability and Capacity Tests in the taxonomy if they are not specifically tested for. It was noted that, although no specific tests for these areas have been developed, information concerning usability (which is intended to address all of the "ilities") and capacity will be gleaned as a side effect of performing some of the tests in the suite. This information led to a discussion of the aptness of the title, "taxonomy." This subject is to be addressed by the ACEC Working Group.

It was observed that items I, II, and IV in the high level taxonomy are addressed. The question was asked as to what percentage of the tests developed address each of the three areas. Tom explained that the test programs are designed to report code expansion size, execution time and information about compilation time for most tests. In addition, specific tests reveal specific information. Topics addressed by particular tests include interrupts, overhead procedure calls, various optimization techniques, etc. Portions of the documentation serve as a guide to special problem descriptions and tests that address specific issues. A question was asked as to whether effects of optimization or simply the presence of optimization features is addressed. Tom explained that the tests will reveal whether or not optimization in a system is performed. Two samples of output are available for review.

Questions were forthcoming concerning the use of pragma suppress. Tom stated that most tests in this area are run with the presence of pragma suppress.

An area of great interest to the Team dealt with tasking and rendezvous. Tom explained that tasking tests are performed which vary the priorities of tasks in order to test results of different tasks arriving at rendezvous in different orders. He noted that, in some systems, this makes for interesting results.

Tom explained that item F lists sources of tests for execution time efficiency. The second subheading, Ada in Practice, lists sources derived from actual work being conducted by Boeing, including some technology insertion efforts.

Sandi Mulholland requested that information be included in the ACEC documentation explaining that run-time systems can vary in size, depending on what is included in the particular sub-set being used. Tom noted that the Reader's Guide for the ACEC describes these variables. Mike Mills stated that an embedded run-time issue that is critical deals with predicting maximum time for rendezvous, an area not addressed by the LRM. Tom observed that there is at present no way of determining exactly the maximum time for rendezvous, but that information gained as a result of tasking tests can be used to estimate rendezvous time. He further observed that it is best to assign priorities to tasking problems. A need exists for dynamic priorities, which is not incorporated in the language at this time. Documents produced by the Ada Run Time Environments Working Group (ARTEWG) were cited as a source of good information.

In concluding the ACEC Status Report, a summary of comments received at the Ada/Jovial Users' Group (AdaJUG) was presented by E&V Team chairman, Ray Szymanski. A presentation of the ACEC structure and support tools was given at this conference, and a Birds-of-a-Feather session was conducted. Questions had been asked concerning the intended use of the ACEC, and were answered from a policy perspective by Maj. Al Kopp. It was noted that policy on use will be determined by the individual program offices. Other questions concerned possible evaluation facilities, and the probability of doing evaluation concurrently with validation. Of particular interest was Jon Squire's assessment of the ACEC. Mr. Squire is chairman of the Performance Issues Working Group (PIWG). It was revealed that he had been asked to keep abreast of ACEC development and provide necessary criticism. He praised the product and the approach and commended all parties involved. Ray complimented the Team on its contribution to the quality of the ACEC and other Team products. He acknowledged that there are some gaps, and cited funding constraints as one cause of this. He lauded the ideas contributed by new Team members and expressed hope that these will influence future developments.

## 2.4 E&V Reference System Status Report

Dr. Bard Crawford
The Analytic Sciences Corporation (TASC)

Bard Crawford presented a review of the development of the E&V Reference Manual and the E&V Guidebook. He explained that an E&V Classification Schema had been developed first. Its purpose was to create a framework that would be used in the Reference Manual. It is intended that users of the E&V Reference System will be introduced to this system via the Reference Manual. This document provides general information, such as definitions of terms, etc., and refers the user to the E&V Guidebook. The Guidebook also provides references to sources such as the ACEC or CIVC efforts.

Bard focused on the Reference Manual structure, which is by text frames. Each element in the Reference Manual provides a three-part text frame which includes general description, cross-references, and in most cases, Guidebook references. Eventual automation of the Reference Manual is being considered. Technology is developing that would facilitate this.

The original draft of the Reference Manual was produced in January 1986. Since that time, three updates to this draft have been completed, and the first official version is being submitted for government approval and will be distributed to the Team very soon. Recent modifications to this document have included the restructuring of the chapter on use of the manual, minor changes to the attribute taxonomy, and other editorial changes. The chapter on use of the manual (Chapter 2) now includes a series of diagrams showing various uses of the manual, and relationships between indexes in the manual. The manual also refers to the five E&V categories as found in chapters of the Guidebook. It is planned that the Reference Manual will be updated yearly to keep it in conformance to relative standards, to add elements currently not addressed, and to address upgrades to the Guidebook.

Whereas a majority of the work performed during the last quarter focused on the Reference Manual, future project effort will focus on the Guidebook. The first draft of the Guidebook appeared in February 1986, with the second draft following in September 1987. The first public release version is planned for April 1988. Issues to be discussed during the December meeting of the CLASSWG relating to the Guidebook include the document's organization, tracking of the ACEC and CIVC efforts, whole APSE evaluation, information on related efforts, and plans for future iterations of the document.

Bard informed the group that he has been asked to serve on a panel at the Ada Conference in Boston next week. This panel will address the topic, "What does Ada need now?" Issues to be considered are Ada support technology needs, technology insertion needs, and the role of standards. He will entertain comments and questions from team members.

John Stanton asked what plans exist for impacting the Ada community with information contained in the Reference System documents. Bard stated that the CLASSWG and REQWG will address this issue of technology transfer. He affirmed that this is indeed a concern of the Team. A perceived problem has been that AJPO has not identified technology transfer as a priority for the E&V Team.

At the conclusion of this presentation, the Wednesday session of the E&V Team was adjourned. Individual working groups met for the remainder of the Wednesday session and all day on Thursday, 3 December.

## 3.0 FRIDAY, 4 DECEMBER 1987

### 3.1 General Business

Chairman Raymond Szymanski convened the Friday, 4 December 1987 session of the E&V Team. He stated that no final decision has been made concerning the meeting place for the March meeting. An announcement will be made over the NET well in advance of the meeting so that team members can make necessary arrangements. The need for more working group time was expressed, and will be taken into consideration.

Ray thanked host Lloyd Styles for his help in arranging a successful meeting. Tricia Oberndorf, chairman of the KAPSE Interface Team (KIT) was thanked for her technical advice and expertise. Ray then presented awards to team members who had served for the past two years.

### 3.2 Introduction to the Joint Integrated Avionics Working Group (JIAWG)

Capt. Victoria Rhoden
ASD/TAEA

Capt. Rhoden presented a definition of the Joint Integrated Avionics Working Group (JIAWG), explained how the JIAWG relates to the Advanced Tactical Fighter (ATF), and highlighted ways the E&V Team can impact development of this project.

The JIAWG was created by the Department of Defense in response to Congress's requirement that the Army, Navy and Air Force work together to reduce redundancy in the avionics systems of the three major weapons systems under development: the Air Force ATF, the Navy ATA, and the Army LHX. The JIAWG is composed of representatives from these three programs. Within the JIAWG are several subgroups, including a hardware task group and a software task group. The hardware task group addresses 16 bit and 32 bit standards under development, a standard for a high speed data bus (HSDB), and a standard for a bus similar to the 1553 architecture. Each task group includes members from each of the three services. The Air Force is the lead service for the JIAWG, with Col. Mike Borky and Maj. Bob Lyons representing the ATF SPO. The three major weapons systems drive activities of the JIAWG. The involvement of the three services and their contractor teams create unique problems for the JIAWG as well as offering unique opportunities. Much information is competition sensitive in nature.

The requirement for commonality of software engineering environments (SEE) is affected by the varying architectures involved. JIAWG is tasked with defining a SEE by 1988 which is to be an APSE. This is to include the host hardware and tool set. During the demonstration/validation phase, this SEE is not a requirement, but will be common by the time the ATF reaches full scale development (FSD). Only the ATF prime contractors could impose such a requirement for their subcontractors during the current phase.

The SEE Specification has been drafted. In determining requirements, a survey was conducted of the prime contractors for the three major systems. It was learned that all developers are using VAX/VMS based tools. However, the tools vary greatly from contractor to contractor. The approach used in selecting tools will be for each contracting team to select a tool for a specific function. The primes from each team will all meet together to decide which of these selected tools is the best one for that specific function. Schedules and money are constraints on the time allowed for this selection process. All contractors will be required to procure the chosen tools. Criteria have not been established for tool selection.

Capt. Rhoden stated that the E&V Team possesses knowledge in the areas of evaluating compilers, whole APSE issues, and the CAIS. The E&V Reference System contains good information that can help the JIAWG and the various contractors. The JIAWG needs the kind of expertise available within the E&V Team for establishing criteria for tool selection and for addressing other issues.

Capt. Rhoden stated that the impact of the JIAWG is yet to be determined.

The floor was opened for discussion. It was noted that the NASA SEE allows for a wide variety of hardware and uses an entity similar to a CAIS which permits use of common tools.

The presentation reemphasized the need for transfer of E&V technology to the whole Ada community.

Following this discussion, the December meeting of the E&V Team was adjourned.

## LIST OF ATTENDEES

Abraham, Rebecca Capt.
AFWAL/FIGAda
WPAFB, Ohio 45433-6543

Adams, Karyl
C.J. Kemp Systems, Inc.
318 E. Dry Creek Road
Phoenix, AZ 85044

Brookshire, Jerry
Texas Instruments
P.O. Box 86905, MS 8476
Plano, TX 75086

Clark, Peter
TASC
55 Walkers Brook Drive
Reading, MA 01867

Crawford, Bard
TASC
55 Walkers Brook Drive
Reading, MA 01867

Eilers, Dan
Irvine Compiler Corp.
10821 Sky Park Cir., #1
Irvine, CA 92714

Elderhorst, Linda
Code SY31H, NATC
Systems Engineering Test
  Directorate
Patuxent River, MD 20670

Facemire, Jeff
SofTech
1300 Hercules Dr.
Suite 105
Huston, TX 77058

Fainter, Robert
Computer Science Dept.
Arizona State University
Tempe, AZ 85287

Foidl, Jack
TRW, Systems Division
Suite 205
9265 Sky Park Ct.
San Diego, CA 92123-4213

Francl, Fred
Sonicraft, Inc.
8859 S. Greenwood
Chicago, IL 60619

Holmes, Tracy
GTE Government Systems
1 Federal St.
Billerica, MA 01821

Impicciche, Alan L.
NAC Code 826
Naval Avionics Center
60000 E. 21st. St.
Indianapolis, IN 46219

Kean, Elizabeth
RADC/COEE
Griffiss AFB, NY 13441

Lawlis, Patricia Maj.
AFIT/ASU
3318 E. Dry Creek Rd.
Phoenix, AZ 85044

Leavitt, Tom
Boeing Military Airplane Co.
P.O. Box 7730, MSK80-13
Wichita, KS 67277-7730

Lindquist, Tim
Computer Science Dept.
Arizona State University
Tempe, AZ 85287

McBride, John
SofTech
300 Hercules Dr.
Suite 105
Houston, TX 77058

McKee, Gary
GARICAR
P.O. Box 3009
Littleton, CO 80121

Marmelstein, Robert Lt.
AFWAL/AAAF-2
WPAFB, Ohio 45433-6543

Millars, Diane
MITRE Corp.
Burlington Rd.
Bedford, MA 01730

Mulholland, Sandi
General Electric

Rhoden, Victoria Capt.
ASD/TASE

Romanowski, Helen
Rockwell International
400 Collins Rd. NE
Cedar Rapids, IA 52498

Stanton, John
AJPO, Rm 30139
(Fern ST/C107)
The Pentagon
Washington, D.C. 20301-3081

Tompkins, Mary Ann
1911 N. Ft. Myer Drive
Arlington, VA 22209

Wills, Betty
CCSO/XPTB
Tinker AFB, OK 73145

Maher, Patrick
Mail Drop T4060, Dept PB531
Motorola GEG
Tempe, AZ 85282

Martin, Ronnie
Software Eng. Research Center
Dept. of Computer Science
Purdue University
W. Lafayette, IN 47907-2004

Mills, Mike
ASD-AFALC/AXTS
Wright-Patterson AFB, OH 45433

Obendorf, Tricia
Code 423
NOSC
San Diego, CA 92152-5000

Roby, Clyde
IDA

Shirley, Jane
TASC
Dayton, OH

Styles, Lloyd
FCDSSA, U.S. Navy
200 Catalina Blvd.
San Diego, CA 92147

Weiderman, Nelson
Software Engineering Institute
Carnegie-Mellon University
Pittsburg, PA 15213

Wood, Jon
MITRE Corporation
Burlington Road
Bedford, MA 01730

# MINUTES

# OF THE

# EVALUATION AND VALIDATION TEAM

# GENERAL SESSION

2-4 MARCH 1988

The task for the Evaluation and Validation of Ada* Programming Environment (APSEs) is sponsored by the Ada Joint Program Office (AJPO).

*Ada is a registered trademark of the U.S. Government (Ada Joint Program Office).

TABLE OF CONTENTS

SECTION . . . PAGE

1.0 WEDNESDAY, 2 MARCH 1988 . . . 1
1.1 Welcome, Introductions, and General Business . . . 1
1.2 Designing a Decision Support System for APSE Evaluators . . . 1
1.3 United Kingdom Evaluation System . . . 1
1.4 CAIS Implementation Validation Capability . . . 5
1.5 The Reference System . . . 5
1.6 Ada Compiler Evaluation Capability (ACEC) . . . 6

2.0 THURSDAY, 3 MARCH 1988 . . . 6

3.0 FRIDAY, 4 MARCH 1988 . . . 6
3.1 IAW Productivity Study Findings . . . 6
3.2 Announcements . . . 6
3.3 Working Group Status Reports . . . 6

## 1.0 WEDNESDAY, 2 MARCH 1988

### 1.1 Welcome, Introductions, and General Business

Evaluation and Validation (E&V) Team Chairman, Raymond Szymanski, welcomed members of the E&V Team to Denver for the March meeting. He announced that the Reference Manual was not received on time due to a snafu; according to Barbara Fleming of the Ada Joint Program Office (AJPO) it is being submitted today to Public Affairs. The document will be approved for public release within a few weeks. The AJPO is making plans to announce at the upcoming Ada JOVIAL User's Group (AdaJUG)/Special Interest Group Ada (SIGAda) meeting that the document is available. Order forms will be available and the Team should have the document first.

Pat Lawlis will discuss plans for integrating her doctoral dissertation work with some elements of the team. Also, Kermit Terrell will be pinch-hitting for Tom Leavitt who is ill.

### 1.2 Designing a Decision Support System for APSE Evaluators

Major Patricia Lawlis
AFIT/ASU

Pat Lawlis' presentation was entitled "Designing a Decision Support System for APSE Evaluators." Its purpose is to help decision makers deal with the "information overload" which will result when one consults the many documents which will be available giving information on choosing software for an Ada Programming Support Environment (APSE). The Decision Support System will be an interactive system and will use the Reference Manual and Guidebook to get pointers to a lot of our E&V technology, resulting in evaluation information which can be turned into magnetic media and fed into the decision support system. It would be somewhere between an information system and an expert system, and would give information to lead to a decision. Opinions and suggestions from the Team on concept, content, feasibility, types of literature to be included, reviews of design products, prototype, etc. would be welcome.

Ray Szymanski announced his intention to support Pat on this in any way that he can, including Net usage for information passing. The team is to feel free to use the Net for information swapping, etc.

Ray stated that the Public Report, which covers the activities of the team for about the last year and a half, was put together in December. It was shipped to Betty Wills who has been getting a laser print copy ready. It will be submitted for approval through appropriate channels.

### 1.3 United Kingdom Evaluation System

Nelson Weiderman
Software Engineering Institute
Carnegie-Mellon University

The information came from a variety of sources: marketing literature from Ada

Evaluation System, ADS documentation, a set of slides I put together in London last week, and from a meeting on Ada Review. There are two parts to Ada Evaluation in the United Kingdom (UK). One is the Ada Evaluation System (AES), the other is the Ada Evaluation Service. The sponsor is United Kingdom Ministry of Defense (MOD). It has been developed by a company called Software Sciences, Ltd., a private company of approximately 1,200 employees. Their business is standards, quality assurance, testing, and they provide some technical help to their export industry.

The Information Technology Department is carrying out the Ada work, and has been involved with evaluation and validation for some years. They do evaluation and validation for PASCAL, and are working for MODULA 2, C, and PROLOG. They have validation suites for those languages and are working on evaluation suites. They have had quite a bit of experience in doing this kind of work. They won this contract over the National Computer Center (NCC) of the UK, and the NCC is among those doing the validation for Europe.

The AES was announced in September of 1987 and there were many articles in the British press at that time. The AES is connected with the NATO initiative. Senator Nunn decided that the Warsaw pact countries were doing a lot more cooperating than the NATO countries. He sponsored a bill called the Nunn Amendment or the NATO initiative which said that the NATO countries should get together and cooperate more on technology questions. It is a three year effort, initially signed by ten nations in December of 1986. The Kingdom of Denmark signed on after that so that there are a total of eleven nations now cooperating on the amendment. The primary focus of this agreement is enhancement of APSEs, particularly concentrated on Ada, developing and demonstrating tools, developing methods and tools for APSE evaluation and interface standards such as the CAIS and the Portable Common Technology Environment (PCTE). It is coordinated by a special working group on APSE (Virginia Caster is the Chair of that group) which is part of another larger organization. The initiative is funded by the individual participating countries. It appears to be about 10 million dollars worth over three years. The evaluation effort is close to two million dollars.

In evaluation versus validation, evaluation is the more difficult problem. It takes longer to perform, and is more subjective. There is more human involvement, and results are more than just pass/fail. There is more interpretation and looking for subtle differences.

AES has two phases. Phase I covers the Ada compilation system, linkers, loaders, and symbolic debuggers. The general size is 200 tests, 400 compilation units, and 100,000 lines of Ada code. In addition to those 100,000 lines it has an automated test harness and report generator. There are 1,000 questions grouped into several checklists and the current release is Release A Version 1.1. Phase II covers APSEs. They are going to add 70-80 tests to compiler system tests, going to develop evaluation of APSE tools and all of these APSE components are included.

A good number of Ada tools will be evaluated; they will be based on usage scenarios: you write a script of what you would like to do in an Ada environment then you try to implement in a particular Ada environment. It will include around 200 actual Ada test programs and checklists, 1,000 questions; and is due under NATO agreement, February 1989. It may be delivered to the MOD by the middle of this year, or it may be behind schedule.

Question: (Bard) The implication of that slide to me is that their APSE evaluation approach is one in which it is just a composition of individual tools and individual function evaluations rather than whole APSE evaluation (an entire phase of the life cycle). Is that your impression?

Nelson: I think that is what I would come away with as well. I am not sure. I have not seen enough detail on Phase II; I have only seen Phase I detail.

A question was asked concerning requirements.

Nelson: The only thing I have seen in terms of requirements is the NATO agreement, and that is very vague; it did not have details at all. I do not know whether the MOD gave them detail requirements or not. The MOD did have a fairly detailed report on evaluation, but did not deal with APSE tools.

Question: You said there is a slight schedule slippage but are not sure what it is.

Nelson: They are due to turn this over in July 1988 to MOD and they were acting as though they may not meet the July 1988 date, but I did not get the impression that the February 1989 date was in jeopardy.

Other questions followed and an inquiry was made about service. Service has three parts:

1) Distribution of AES itself to whoever wants it,

2) Formally evaluating Ada compilation systems, BSI is going to do that,

3) Distribution of Ada evaluation reports.

Everything that BSI evaluates will be made public in two ways: a subscription service and a one-of-a-kind copy. Each evaluation is going to take the form of a detailed report based on AES test suite. Service available to whoever wants it, vendors, contractors, etc. Cost of Service: $1,750 "do it yourself" version which has 600 questions; and a $20,000 version which has 1,000 questions and uses Assessor Guidelines Support, which provides guidance for answering the questions. The idea is to get tests out to vendors and people who are more experienced in evaluation, let them do the test and if they see that their product is going to get a good evaluation, go to BSI and get a full blown evaluation. There are several types of subscription services, annual subscriptions, individual reports. The British maintain the copyright. There is no third party distribution, only internal use, no distribution of derived reports, or opportunity for client comment.

A discussion followed over who will sign up for evaluation, etc.

There was a question over copyright issues. Ray said that the understanding of the NATO agreement is that items developed under NATO agreement are properties of all countries for defense purposes; how then can the UK copyright?

Nelson: U.S. has access to the test suite. AJPO will receive it and decide who gets it. The Europeans are very much for openness and ease of distribution. The NATO purpose is to share the information for defense purposes.

Evaluation reports come from independent tests, unbiased, greater scope than validation reports, economic benefits of one agency doing evaluation, means of identifying high quality products, and assists vendors in improving products. Estimates of the effort required are somewhat optimistic. Rehosting will probably take more than three days, executing test suites will probably take five to ten days, evaluating non-testable features will take more than two to five days, and the estimate of six manweeks total is very low. Test suites are written in Ada. The test suite contents include quality of code generation, compile time performance, quality of diagnostics, compile-time and run-time limitations, tool functionality, reliability, ease of use, and quality of documentation. Two compilers have been evaluated so far, but results have not come out yet.

AES Test Suite Groups, evaluation of non-quantifiable features, and example questions for error messages were all discussed. The point was made that one should not always rely on vendor documentation to answer questions. The test harness is the interactive user interface for this product. It has a preprocessor which allows for tailoring of the test and rehosting. The preprocessor provides elaborate text processing facilities, and an example command script template was presented. Documentation is voluminous. Completed evaluations include Alsys Ada for Sun Workstation and Systems Designers cross compiler from VAX to Motorola 68020. Others are being worked on. Software Engineering Institute's (SEI's) experience was an aborted attempt due to a bad install procedure.

The review group discussed the size of the evaluation report; the summary should be more quantitative than subjective. In comparison with the ACEC, the two suites are largely complimentary rather than overlapping. The NATO agreement is designed to promote cooperation among countries. It is being distributed widely in Europe, but people over here do not seem to know what it is about. The E&V Team should be aware. A question was asked if would not a better product result if AES and ACEC were put together? Ray Szymanski promised to research answers, but urged the team to concentrate its attention on technical rather than policy issues.

Ray thanked Nelson for an excellent presentation. Ray went on to say that certain technology, software in particular, is handled differently than other information. There are certain rules and regulations governing distribution of that technology. Occasionally the person who is the producer, in this case the AJPO, can put forward a case for the release of the technology. On the other hand, there are other occasions where certain technology is restricted and if the releasing authority decided your reasons are not acceptable, they will not allow you to release that technology. Currently the compiler evaluation technology is under the restricted regulation. I did not make the decision. We have not made a concerted effort to convince the authorities that we want this released. The AJPO has not decided to go to the trouble of doing that. As to whether or not it is in the best interests of the Ada program to do so, they know what your concerns are, that you believe it is in the best interest of the Ada program to get that released. We are bound by regulations. Questioned on who would the team lobby? Ray replied himself and that he has made his concerns to the AJPO known.

## 1.4 CAIS Implementation Validation Capability

Jeff Facemire
SofTech, Inc.

Preliminary Design Review (PDR) was held in San Diego in January, between then and now we have been working on detailed design, primarily the test suite itself. Our delivery dates are March 14 for the Implementor's Guide, Test Report Reader's Guide, and Software Product Specification. Critical Design Review (CDR) will be in Arizona on April 13. On framework, we have developed about 200 test objectives, concentrating on Chapter 4 of the CAIS. We have about 50 scenarios, but do not necessarily have scenarios for every one of the test objectives yet. We are going to develop test objectives and place these in accordance with a taxonomy that we could use some kind of a cross-referencing capability to trace the 600 page specification to know where information came from. Discussions followed over cross-referencing and use of a product called "Guide", Implementor's Guide, involvement in the NATO/Nunn amendment. MITRE is doing an Independent Verification and Validation (IV&V) effort for TRW; if they are doing the same thing we are, why not get together and come up with a more powerful system. We got together in December and informally worked up a cooperation. We are starting with Chapter 4; MITRE decided they would like to collect and upgrade public domain tests which include the Arizona State University (ASU) tests, MITRE prototype tests, etc. It was noted that we should make sure there is good coordination with MITRE and that their tests are incorporated into the framework.

## 1.5 The Reference System

Bard Crawford
The Analytic Science Corporation

After the December meeting (influenced by REQWG discussion) Ray directed us to do three things:

1) prepare a draft announcement circular and ask Tim Lindquist to be our man in Phoenix and to leave an announcement in a prominent place in the back of the room at SIGAda/AdaJUG.

2) contact Al Kopp at AJPO to find out whether they were going to have a presentation.

3) prepare a canned presentation on the Reference Manual in a form that anyone could use if deputized.

The draft announcement was prepared for distribution at the Phoenix meeting; Al Kopp was contacted and said yes, they plan to have a presentation. We offered to send him one slide and verbiage; a draft 20-minute slide presentation was prepared for future use by various team members.

Bard summarized the current status of the Reference Manual and Guidebook, and did a trial run through the draft "canned presentation." Team members offered a number of suggestions for making the presentation more effective.

## 1.6 Ada Compiler Evaluation Capability (ACEC)

Kermit Terrell
Boeing Military Airplane Company

Mr. Terrell's presentation dealt with the structure of the user documentation, Implementor's Guide, and the Reader's Guide -- are we putting in things we should not, are we leaving things out we should not? The purpose of the Implementor's Guide is to show someone how to run the ACEC; it contains information on what an ACEC is, requirements, etc. and its main subject is executing the ACEC. The guides concentrate on two systems, VAX/VMS and UNIX, since they have wide distribution. Execution of the ACEC requires preparation, initialization, and exceptional tests.

The remainder of the presentation dealt with an introduction to the structure of the user documentation.

## 2.0 THURSDAY, 3 MARCH 1988

The E&V Team meeting was reconvened on Thursday, March 3, at 8:00 a.m.

Ray discussed the timing of next meeting, maybe moving it from June to May. He hoped to have plans made before adjournment.

## 3.0 FRIDAY, 4 MARCH 1988

The E&V Team meeting reconvened on Friday, March 4, at 8:00 a.m.

## 3.1 IAW Productivity Study Findings

Lt. Robert Marmelstein
AFWAL/AAAF-3

Lt. Bob Marmelstein gave a presentation on the IAW Productivity Study.

## 3.2 Announcements

The next team meeting will be in Denver on 24-26 May 1988.

## 3.3 Working Group Status Reports

Working Group Status Reports were presented by the group chairpersons.

# MINUTES

# OF THE

# EVALUATION AND VALIDATION TEAM

# GENERAL SESSION

24-26 MAY 1988

The task for the Evaluation and Validation of Ada* Programming Support Environment (APSEs) is sponsored by the Ada Joint Program Office (AJPO).

## TABLE OF CONTENTS

SECTION PAGE

1.0 TUESDAY, 24 MAY 1988 . . . . . . . . . . . . . . . . . . . . . . 1
- 1.1 Welcome, Introductions and General Business . . . . . . . . . 1
- 1.2 The Reference System . . . . . . . . . . . . . . . . . . . . . 2
- 1.3 Environment Evaluation . . . . . . . . . . . . . . . . . . . . 4
- 1.4 CAIS Implementation Validation Capability (CIVC) . . . . . . . 7
- 1.5 Ada Compiler Evaluation Capability (ACEC) . . . . . . . . . . 9

2.0 WEDNESDAY, 25 MAY 1988 . . . . . . . . . . . . . . . . . . . . . 12

3.0 THURSDAY, 26 MAY 1988 . . . . . . . . . . . . . . . . . . . . . 12
- 3.1 General Comments and Announcements, Interim Session . . . . 12
- 3.2 Closing Session . . . . . . . . . . . . . . . . . . . . . . . 12
  - 3.2.1 Working Group Status Reports . . . . . . . . . . . . . . 12
    - 3.2.1.1 Ada Compiler Evaluation Capability Working Group (ACECWG) Status Report . . . . . . . . . . . . . . 13
    - 3.2.1.2 E&V Technology Classification Working Group (CLASSWG) Status Report . . . . . . . . . . . . . . 13
    - 3.2.1.3 Requirements Working Group (REQWG) Status Report . . 14
    - 3.2.1.4 Standards Evaluation and Validation Working Group (SEVWG) . . . . . . . . . . . . . . . . . . . . 15
    - 3.2.1.5 CAIS Implementation Validation Capability Working Group (CIVCWG) . . . . . . . . . . . . . . . . . . . 16
  - 3.2.2 Discussion with Grady Booch . . . . . . . . . . . . . . . 16

Document Announcement: APSE E&V Reference Manual . . . . . . . . . . 20

List of Attendees . . . . . . . . . . . . . . . . . . . . . . . . . 21

## 1.0 TUESDAY, 24 MAY 1988

### 1.1 Welcome, Introductions and General Business

The Evaluation and Validation (E&V) Team quarterly meeting began with opening remarks by chairperson Raymond Szymanski.

After welcoming the team, Mr. Szymanski introduced Donald Mark of RADC, who was attending for Liz Kean and Shawn Fanning, a CAIS-A designer from SofTech who will be participating in the meeting. He then made the following announcements:

- The general session and some of the working groups meetings will be attended by a convention recorder, Denise Conner, as an experiment in expediting the minutes.
- The ACEC will be released through DACS and the Federal Software Exchange. Both organizations maintain a list of eligible recipients.
- The AJPO cannot, at this time, fund a large amount of operational support for the ACEC; however, they are willing to share funding with other sources interested in that technology.
- The AJPO is also interested in co-funding new technology starts. Ray Szymanski recommended that the team choose an area of technology they would like to see developed.
- Mike Burlakoff will be performing an independent verification and validation of the ACEC under the Summer Facility Research Program at Wright-Patterson AFB.
- Nongovernment personnel must be "qualified" in order to attend future E&V meetings. Some possibilities do exist for attending, and Ray is willing to discuss individual circumstances.
- Oneida Resources, Inc. is the new support contractor. They will be mailing out the E&V Team Public Report, Volume III upon its release in June.
- Maj. Pat Lawlis will be receiving limited AJPO support for her Ph.D dissertation at Arizona State University (ASU).
- The Ada-Europe Environments Committee has a meeting scheduled before the actual Ada Europe Conference begins. Major presentations will be given by the following three groups:

    1. the E&V Team,
    2. the German Public Interface Working Group,
    3. Dr. Rovino, PCTE.

Dr. Bard Crawford announced that he has draft minutes of the March, 1988 meeting, which include the ACECWG, the CIVCWG, and the General Session. The draft minutes of the December 1987 meeting were sent to Ray Szymanski. All the minutes will need some work in preparation for the next public report.

## 1.2 <u>The Reference System</u>

Dr. Bard Crawford
The Analytic Sciences Corporation (TASC)

Dr. Crawford announced that the ASU activities and Dr. Lindquist's team support will continue through the TASC contract, and that TASC will also be supporting Gary McKee's work as an E&V Team consultant. He then began his presentation on the Reference System which is composed of the Reference Manual and the Guidebook.

The Reference Manual is the starting point for users and includes information such as the terminology used, how to set up the framework of different types of relevant elements, and how terms are defined and cross-referenced. It also provides references to the Guidebook, which offers information on specific E&V technology, for example, the ACEC.

The official Version 1.0 of the Reference Manual was approved in March and distributed in April. A Defense Technical Information Service (DTIC) number has been applied for, and information will be forthcoming on how to obtain this document.

There has been no Version 1.0 of the Guidebook because of significant organizational changes; however, Version 1.1 has an anticipated release date of early August, followed in two weeks by Version 1.1 of the Reference Manual. The main thrust of the coming year will be to add more detail and more instances of technology to the Guidebook. Version 2.0 of each document is scheduled for mid-1989.

The Reference Manual was mentioned in the AJPO presentation at Special Interest Group Ada (SIGAda) in Phoenix. Copies of the announcement flier were distributed and a mailing list will be compiled from the responses. A similar flier will be distributed at the Washington Ada Symposium (WAdaS) in June.

Dr. Crawford has prepared a slide presentation with script addressing the reference system and other E&V activities which will be available for use by any team member. Portions of this presentation will be used at the Ada-Europe joint meeting.

Ms. Hazle asked about sending the flier to the Language Control Facility Newsletter, the Ada Information Clearinghouse Newsletter, and other related newsletters. Dr. Crawford suggested that the matter be discussed further by the team before taking that action. [A copy of the document announcement can be found on page 21].

Next, an outline of the Guidebook was given. The first four chapters serve as an introduction, and the remaining formal chapters are in the same prioritized order as the REQWG's Tools and Aids Document.

The first four chapters have been rewritten. Chapter 2 of the Guidebook is much smaller than its counterpart in the Reference Manual which includes diagrams and examples of different types of scenarios. Chapter 2 of the Guidebook merely states the document's organization. Chapter 3 concerns the integration of E&V technology, which is a whole-APSE issue. Although following the original organization, Chapter 3 has been revised extensively. Chapter 4 is a synopsis of documents covering general background. They can be included for one of two reasons: (1) it is historically important, for example a half-page synopsis of the Stoneman Document; or (2) it is a single synopsis of a report which covers more than one instance in the later chapters.

Mr. McKee requested that Dr. Crawford define an assessor. Dr. Crawford responded by saying that an assessor is a method of determining value, in E&V terminology there are two categories, evaluation or validation. Evaluation is a performance or quality assessment. Validation is an assessment of conformance to a standard.

Ms. Hazle asked why the capabilities checklists were separated out, rather than included as assessors. Dr. Crawford responded that these checklists are assessors too, but they are a simpler technology. The distinction was made to give a little more insight into the limited amount of available E&V technology.

In Chapter 3, added terminology is used in an effort to clear up some of the confusion over what is meant by whole-APSE assessment.

The term "limited scope" refers to a specific bit of technology, such as in Chapter 13, where an assessor which takes a whole-APSE approach to a single phase or group of activities. The "full scope" is an integrated approach to assessing an APSE as a whole, leading to some final judgment or decision. An example of a whole-APSE assessor would be setting up an experiment to evaluate a particular APSE in a specific phase. An example of an integrated whole-APSE assessment is a decision support system combining other assessment tools or individual whole-APSE assessments, and assigning weighting factors to them to arrive at a final decision.

Mr. Burlakoff asked for an example of a maturity or reliability assessor. The example given was that of a checklist. Dr. Lindquist inquired about the object of a maturity test. Dr. Crawford stated that maturity was an attribute of an APSE or a component of the APSE.

Responding to a question from Mr. Leavitt, Dr. Crawford stated that an example of a maturity assessment would be a summary of completed projects which had used the tool in question.

Dr. Gray commented that Dynamic Research Corp. has a tool that they claim gives an objective determination of portability, reliability, and maintainability. Dr. Crawford said that one should remember that it is tools being evaluated and not the application produced by them, but if a tool is written in Ada, its source code could be evaluated using DRC's tool.

Mr. Burlakoff asked if the goal of the assessor was to apply to a complete range of tools. Dr. Crawford answered that it could apply to a whole-APSE, a tool set, a particular tool, or just a single function of a tool. The REQWG has compiled a prioritized list of the types of assessors the community would like to see. Dr. Crawford concluded by introducing the next speaker.

## 1.3 Environment Evaluation

Lewis Gray
TRW

Dr. Gray began by mentioning his work with the Software Organization Subgroup Working Group for Development Standards in Ada. This subgroup is compiling examples of implementations of the 2167A software organization. They are concerned with what CSCIs are in Ada terms. They are currently drafting a second report with examples from various vendors of what a CSCI is and how the tools are implemented. There will also be a section concerning government guidelines.

The topic of Dr. Gray's presentation was how a programming support environment was chosen for an Ada project last year. He began by defining a whole-APSE evaluation as: "an assessment of quality or performance of the APSE as a whole rather than assessments of component parts." An example would be using several different tools but evaluating the job as a whole. A limited whole-APSE assessment would be a numerical comparison of two evaluated APSEs, choosing the highest figure.

The Army World Wide Military Command and Control System (WWMCCS) Information System (AWIS) project involved recommending a foundational environment for Phase 1 of the contract. The environment chosen had to also prepare foundations for the evaluation in Phase 2 which will compare the first environment with the Software Development and Maintenance Environment (SDME).

Lt. Marmelstein asked what the AWIS project was about. Dr. Gray stated that the AWIS involved implementing the Army part of the WWMCCS Information System and replacing it incrementally with new Ada software. The Request for Proposal (RFP) stated certain environments to be evaluated, the Digital Equipment Corporation (DEC) Ada, Ada Language System (ALS), Honeywell, and the IBM compiler. TRW proposed to evaluate all of these systems except the IBM. The RFP was amended in 1986 to allow any environment as long as the code could be ported. This opened the door to the evaluation of the Rational.

The reasoning for this evaluation to be called a whole-APSE evaluation is two-fold. First, the VAX/VMS and the Rational environment were used to integrate and test the CMT WIS precursor tool. The following were recorded for each APSE and compared; the user interface, system interface, functionality, and performance. Second, the APSEs were numerically rated on the following characteristics: 1) acquirable functionality, 2) performance, 3) initial cost, 4) procurability, 5) reliability, 6) maintainability, 7) growth potential, 8) impact on facilities, 9) impact on training, and 10) impact on personnel. The ratings were totaled with the highest figure being recommended, which turned out to be the VAX/VMS.

During the course of the task, the functional and technical requirements for the foundational AWIS environment was defined. Software Engineering Institute's (SEI's) Environment Evaluation Methodology was used to evaluate the relevant characteristics of the technical requirements. Other characteristics of each environment that was relevant to the selection process was defined and evaluated resulting in the recommendation of the AWIS environment.

The SEI methodology is in six phases. The first phase involves defining the activities of interest. In the second phase, the applied criteria for judging how well the environment supports the activities is described. The third phase defines the generic experiment. The fourth phase involves the environment-specific experiment. In the fifth phase, the specific experiment is run, and the analysis occurs in the sixth phase.

The product integration experiment includes: creating a program library, stubs, and harness directories; creating test input data; selecting the components of the next build; assembling the compilation units into the build; and performing the top-down testing which involves executing the top program library unit using stubs for lower units; adding to, modifying, and deleting the unit source code as needed; creating a new executable image; and retesting the modified unit.

The evaluative questions of the product integration experiment involve functionality and performance. Functionality describes the mechanics of creating a program library, entering Ada source code into the library, and translating a compilation unit into the library. Questions in the area of performance are: 1) how long does it take the environment to create a program library or to translate a compilation unit into a program library? and 2) what is the system recompilation behavior that results from adding comments to a library unit specification, or adding a subprogram specification to a package specification?

In correcting and recompiling the units, the VAX took 1 hour, 8 minutes, 53 seconds and the Rational took 1 minute, 14 9/10 seconds. There are many specific differences between the DEC and the Rational. The VAX set can generate on-call program unit templates. The Rational generates on-call completion and formatting of program unit declarations, formal parameters, statements (e.g., loop, case, conditional, etc.), private parts of packages, and subprogram/package bodies.

When asked about the complexity of the two systems, Dr. Gray stated that the training period on the Rational was longer due to the number of tools involved and the object-oriented environment of Ada.

When adding a subprogram to a package specification, the VAX compiles the package specification, the package body, and all library units that depend upon the package specification. The Rational will compile the subprogram and the package body.

Regarding a unit test, the VAX creates a new test drive file, writes a test routine in the file using source code templates when applicable adding "with" clauses when needed, compiles the test routine, links, and then executes it. Rational attaches a command window to the unit's library, writes the test routine in the command window using the editor's source code generation functionality, and then executes the test routine. There is more visibility and no writing of "with" clauses using the Rational.

In the first year of the project, software requirements analysis had to be done and a database management system had to be used. The customer wanted a recommendation for only one system. The VAX met 100% of the functional requirements while the Rational met only 40-65%.

Mr. McKee commented that according to Dr. Gray's statements the early life cycle is not adequately supported on the Rational at present. Dr. Gray concurred that there are no tools to support requirements analysis.

Ms. Holmes stated that looking just at the Ada items, the Rational appeared the better system, but the decision was made due to the DEC's versatile performance. Dr. Gray agreed.

Dr. Gray stated that they are trying to write Ada so that it is not machine dependent; it could then be maintained on whatever system may be available in the future.

There were certain mandatory criteria for the evaluation: no schedule slips, no extraordinary effort in meeting delivery deadlines, and delivery and installation no later than the Preliminary Design Review (PDR).

Characteristics were weighted according to their importance and values were assigned along the following lines. Functional requirements: support of more than 75% received a 5, 50-75% received a 3. Performance: 5 was awarded to the fastest, and any system exceeding the minimum requirement received a 3. Cost: the VAX as the most expensive system was given a 0 and the Rational a 1. An economic approach was taken; the weights and values were then adjusted to be comparable.

The VAX/VMS was recommended as a result of the AWIS APSE evaluation study; it was found to be more useful as a foundation environment. The study report mentioned the possibility of enhancing the foundation environment before full-scale development as some important VAX/VMS-based tools were found to be inferior to their Rational counterparts.

Dr. Gray stated that the next phase is a large scale development researching the integrated environment. It will be evaluated and compared to the Software Development and Maintenance Environment.

Dr. Gray's presentation ended with a question and answer session.

## 1.4 CAIS Implementation Validation Capability (CIVC)

Jeff Facemire
SofTech, Inc.

Mr. Facemire brought the E&V Team up to date. Three meetings have been attended since March. Two were North Atlantic Treaty Organization (NATO) reviews concerning the two efforts by MITRE and SofTech. The purpose of the review was to discover any duplication of effort and to discover if there could be a cooperative effort between the two groups. It was found that the groups are doing two different tasks.

MITRE's task concerns a particular implementation on a particular host. SofTech's work is on a broader scale. Both groups are continuing to work independently.

In April a pre-Critical Design Review (CDR) was held at ASU. An evaluation was done of NATO's Ministry of Defense (MOD) test harness to see if it could be coordinated with SofTech's test harness. This evaluation effort caused the original CDR to slip resulting in redefining its purpose to cover test scenarios and objectives. The test harness and test administrator will be covered at a later date.

At the pre-CDR, issues were discussed that had been raised at the Preliminary Design Review. A presentation was given addressing the MOD test harness and its general capabilities of what it could provide or was designed to do, and what its capabilities are; examples were given of SofTech's test objectives and scenarios. The next step was to contrast its capabilities with CAIS Implementation Validation Capability (CIVC) requirements.

The idea was to start a cooperative merging between CIVC test administrator and the MOD test harness. The conclusion was that there seemed to be a reasonable overlap and that the test harness could be used to merge with SofTech's test administrator. The MOD test harness may provide the potential for automation.

The main topics of the pre-CDR were test objectives and scenarios. The organization of the Implementor's Guide was also discussed; this included the introductory material which is composed of the rationale and an explanation of terms and symbols, test objectives, and scenarios. The test objectives are certain points of interest to be tested within the CAIS. The questions being asked concern creating a process and establishing a relationship. Several scenarios may be needed to fully test these out.

There are currently 162 test objectives. The general requirements section of the CAIS is a major section of Chapter 4. If Chapter 4 could be covered in the first build, this would provide a measure of assurance in the validity of the CAIS implementation.

Access control of the CAIS is covered in the first build. The primary and secondary efforts are covered in a broad sense. Of the two major chapters, Chapter 4 deals with general information while Chapter 5 deals with the specific requirements. Process control testing is addressed in Chapter 4 along with node relationship attributes.

In the Implementor's Guide, a typical test objective describes the test objective, assigns a particular number, and gives references to where in the 1838 document the test objective was derived. In the numbering scheme, there are three parts to the taxonomy. One deals with the compilation type of testing, and are called static tests; all test objectives of this type start with an S. Normal processing covers everything but those parts that raise exceptions; these start with an N. The exceptions processing tests which include capacity tests start with an E. Numbering between dashes identifies the particular CAIS entity with which the test objective is associated. The end number is the sequential number of test objectives for that entity.

Of the terminology and diagrams used in the Implementor's Guide, the system, structural, process, and file node diagrams are from the CAIS. References to the structural nodes are similar to directories in the CAIS. A generic node was created for use in those scenarios that the testing of the relationship was the important consideration. Case sensitivity is used throughout the scenarios. Capital letters indicate an actual name while lower case letters are similar to a variable.

Handles are pointers to objects; in obtaining a handle to a file the process could have a pointer to that file. A designation was composed where a handle, name X, to a node had an intent Y, with the intent being what is intended to be done with the file. A slash notation was used to distinguish between relationship attributes and node attributes.

A sample scenario was shown illustrating precondition and post-condition when changing the value of a user-defined attribute.

There are 162 test objectives and approximately 50 scenarios for those objectives. By the end of the year it is estimated that 250 to 300 scenarios will be completed.

Mr. Crawford asked if the scenarios expected to be completed by year's end are designed to cover the 162 objectives. Mr. Facemire responded yes, and that many of the objectives may require one to three scenarios for testing which is the reason for the 300 estimate.

Dr. Lindquist inquired if the number of programs needed for the scenarios had been estimated. The response was no, because the scenarios will be combined into dependent trees before being made into Ada programs. The reason being that the postcondition of one scenario may be used for another. These scenarios can possibly be placed end-to-end in the Ada program so it is expected that a significantly lower number of Ada programs will be needed.

Mr. Szymanski asked for the total estimated number of objectives to have a theoretical 100% coverage. Mr. Facemire indicated 7,000. This concluded Mr. Facemire's presentation.

## 1.5 Ada Compiler Evaluation Capability (ACEC)

Tom Leavitt
Boeing Military Airplane Company

The subject of Mr. Leavitt's presentation was Boeing's experiences during formal qualification testing which occurred in April. The testing was done on the following five systems: 1) AIMS version V20208A, Sperry 1631 which is the 1750A; 2) Apollo DN3000, Aegis 9.6, ALSYS 3.2.1; 3) DEC VAX Station II, VMS 4.6, DEC Ada V1.5-43; 4) Harris 800-2, VOS/VUE 6.1.0, Harris Ada 3.1; and 5) TeleSoft, VMS 4.6, TeleSoft Ada 3.15, VAX Station 2000.

The number of working days that it took for each of the systems are: 10 working days for the AIMS, most of which was spent in downloading programs; the Apollo took 6 days; the DEC took 2 working days (the primary development was done essentially on the DEC Ada compiler); the Harris - 17 working days; and TeleSoft took 7 working days. From the test results, the percentages of success indicate that the TeleSoft and DEC ran over 90% of the problems. The TeleSoft system has several errors. For one, it does not do pre-emptive priority scheduling, which will be required for validation purposes not tested for in ACVC 1.9. Another set of errors was due to constraints placed on representation clauses and site clauses. In addition, there was also failure in compiling the programs.

The Apollo and the Harris did not successfully run a large percentage of tests. The major reason for test failures in Apollo was due to capacity being limited in the compiler. The failures on the AIMS were mainly representation clauses that were not supported, including non-support for pre-emptive priority scheduling and for interrupt tests.

The Apollo's failures were due mainly to compile time errors. While all the tasking tests compiled, they executed unreliably. When the tasking problems were run with the debugger invoked, they executed appropriately each time. A problem was found on the TLD system. A procedure with more than 1,000 lines would cause some tables to overflow.

There were problems getting anything to run on the Harris which is one reason for the high number of working days to get through the system. One major problem was that the math library on the Harris died and no test program that used math functions could be run. There were also constraints on the representation clauses that caused a number of programs not to work. Performance improved after it was found what parameters should be specified.

A discussion followed on various ways to interpret the tests and how to evaluate and document results. Mr. McKee stated that the ability to get good results depends on the expertise on that system. He recommended that the level of expertise of the system be given when reporting any findings. The categories would be initial user, mid-range user, and expert to avoid misleading the users of the information. Any vendor would have to claim himself an expert. Mr. Leavitt then turned the discussion to the tools used in the testing.

The INCLUDE processor is a tool which includes timing code into the test problems. It could have failed in either of two ways: compile time failure due to loop mismatch, or no measurement data output. The format tool extracts the timing measurements from the printed results file. It was used on four of the five trial systems. The AIMS system has no file system; so just the results were entered. In a number of cases, the Harris system incorrectly generated leading zeros; it was an error on the run time system.

The MEDIAN test data was run with known output, with all error codes, with a large number of systems, and with one system. It performed as expected.

In the operational software, there were several points of correctness observed. One was the timing and problem coding. The second was the correctness of the measurements, whether the numbers produced were consistent with the number of constructions executed on the machine. The testing for correctness of translation addressed built-in checks, compilation time errors, and run time errors.

In the demonstration of the system tests, four test problems failed on all of the trial systems. The ss686 was supposed to fail. It was decided that one test would be set up to fail. Task54 and Task55, which both failed, specify a length clause for a task type which is smaller than the size required. On the DEC Ada a constraint was raised at the wrong place. It should have been raised when a procedure was called or at the caller, instead it was raised in the body. In other cases, a run time system error message was produced; the exception was not generated and the program did not regain control. All the test problems that reported null time were examined. None of the test problems were unexpectedly optimized to null.

For measurement, some small quick executing test problems were selected. The machine code generated was examined for consistency. It was determined that the compilers were translating the statements correctly for six randomly selected outliers. In checking the translation, the errors detected by self-checking code are flagged as run time errors and MEDIAN output is picked up by the format program.

Mr. McBride asked what review process was used to determine if the test itself was actually testing the measure of interest. Mr. Leavitt stated this was done mainly in review by the ACEC team. In addition, there was some independent review from SEI. Mr. Burlakoff's E&V activity summary may address that question also.

Software Problem Reports (SPRs) break down the problems against this related software. There were two errors in MEDIAN. One was that when there were more than 1,500 examples, it would divide by zero which produced an error. The second problem was in running an application where the model fit the measurements with zero error. That was also divided by zero and generated 100 percent.

The test suite had 47 errors, the majority of which were related to exceptions for better error handling. These are currently being examined.

The test suite SPRs involved:

- Test problems in s0724+40 not loop invariant.
- Test problems print error messages, but no error measurement code.
- Some test problems do not contain exception handlers to print error measurement data.

There was a duplicate test name problem, an include file that left a blank, and some command files that contained a few errors.

Under system highlights, the AIMS major characteristic was the slow downloading process. On the Apollo, there were problems with program size limits and with the tasking execution. There were no major problems on the DEC Ada. The Harris environment is fragile and there were program size limits with a few run time errors. There were no major problems with TeleSoft although it does not support pre-emptive priority scheduling.

In the overall system factors, the execution time for the AIMS was 1.24; Apollo was 1.00; DEC was 0.90; Harris was 1.07; and TeleSoft was 1.00. This is the overall average result of all of the tests.

In his summary, Mr. Leavitt stated, that although there is difficulty in some cases, the execution of the ACEC on these trial systems is feasible. It was found that some items cannot be done on all systems due to system limitations. The test effectiveness is the exercising of the language features. Tools were used in the process of running FQT. FORMAT and MEDIAN were used and worked. Some limitations were found in the trial systems. Most of the unsupported features are mentioned in the document.

Mr. Leavitt then closed out his presentation by entertaining questions from the team.

## 2.0 WEDNESDAY, 25 MAY 1988

The entire day was devoted to individual working groups.

## 3.0 THURSDAY, 26 MAY 1988

### 3.1 General Comments and Announcements, Interim Session

Chairperson Raymond Szymanski reopened the general session. The following items were announced:

- The next E&V Team meeting will be 7-9 September, at WPAFB, OH.
- An offer has been made from the commander of FCDSSA at Point Loma in San Diego for the December E&V Team meeting, 6-8 December.
- In September a representative from the National Security Agency will join the Team.
- Guy Taylor (Navy) may rejoin the Team.
- An unscheduled presentation by Grady Booch will be given later on this morning.

The general session was adjoined until 10:30.

### 3.2 Closing Session

The closing session was comprised of working group reports and a presentation given by Grady Booch.

#### 3.2.1 Working Group Status Reports

## 3.2.1.1 Ada Compiler Evaluation Capability Working Group (ACECWG) Status Report

The basic topic of the ACECWG was the Phase III planning. This was comprised of the following: compiler diagnostics which includes compilation error, informational messages, linker errors, and run time errors. It was suggested that "Bad/Fair/Good" be replaced with "Pass/Fail" and other additional information. Other topics were space reclamation; additional application profiles; library system; symbolic debug; single system evaluation; and the Ada Run Time Environment Working Group (ARTEWG) proposed extensions.

Action items included: 1) a write up of the discussion of group comments on diagnostics evaluations (Lt. Marmelstein), 2) a "plan of attack" to be prepared for briefing the ACEC (Lt. Marmelstein), and 3) a revised ranking of Phase III activities (Lt. Marmelstein).

## 3.2.1.2 E&V Technology Classification Working Group (CLASSWG) Status Report

Ms. Martin reported all members present with the exception of Peter Clark. There were no deliverables due this quarter. Accomplishments include: 1) reviewed elaboration of usability-related whole APSE assessment issues; 2) reviewed mappings of tools to function in the Reference Manual, Chapter 5; 3) completed additional synopses for Chapter 4 of the Guidebook; 4) refined checklists in the Guidebook; 5) reviewed the Guidebook in preparation for summer release.

A key issue was the disappearance of team members with outstanding action items. Projected work/action items included:

1) Reference Manual - review/refine tool and mappings to functions to ensure coverage of new priority areas.

| | |
|---|---|
| - R. Martin | Test Systems |
| - G. McKee | CAIS Implementations |
| - P. Lawlis | Ada Design Support Tools |

2) Guidebook - provide missing synopses.

| | |
|---|---|
| - G. McKee | CAIS (DOD-STD-1838) |
| - G. Gicca | WIS Compiler Evaluation Guidelines |

3) Guidebook - refine checklists.

| | |
|---|---|
| - P. Lawlis | Database Management Capabilities<br>File Management Capabilities |
| - R. Martin | Scheduling Capabilities<br>Tracking/Productivity Assessment Capabilities |
| - F. Frank | Instruction Level Simulation/Emulation Capabilities |
| - G. McKee | Title and Introductory Section |

|  |  |
|---|---|
| - J. Facemire | 8.1 CIVC Description |
| - R. Martin | Provide Reference for Test Systems Capabilities Checklist |
| - P. Lawlis | Create Whole APSE Usability Assessment Checklist |

There are no deliverables due next quarter and no presentations planned.

### 3.2.1.3 Requirements Working Group (REQWG) Status Report

REQWG reported no personnel changes. There were no deliverables due this quarter. The group's accomplishments include: a proposal for developing new technology including enhancements to ACEC (from a prioritized list of requirements) and new starts: model project, symbolic debugger assessment, target code generation aids and analysis toolset evaluators; a plan for disseminating information on the team and its products; and a decision to update the Tools and Aids Document, but not the Requirements Document.

Key issues:

- New technology should be developed according to written requirements.
- It is important for E&V information to be disseminated in a coordinated fashion.
- Lessons-learned when running the ACEC should not be lost.

Projected work:

- Look for organizations interested in sharing funding for development of new technology.
- Develop guidelines for capturing lessons-learned when running ACEC.
- Refine plans for E&V information dissemination.

Presentations are planned at the following meetings by REQWG members:

- ASEET in June by Pat Lawlis.
- WAdaS in June by Ray Szymanski with AJPO.
- Ada-Europe in June by Gary McKee.
- AdaJUG in July by Becky Abraham or Marlene Hazle.
- SIGAda in August by Gary McKee or Tim Lindquist.
- TriAda in the fall.
- SAE in October.
- AdaExpo in October.
- NSIA in October by Ronnie Martin.
- ACM in October.
- AdaJUG - E&V Birds of a Feather - in November.

- IEEE Environments in December by Marlene Hazle.

Other possible vehicles for disseminating information include mailing lists such as INFO-Ada, and bulletin boards such as EV-INFO, Ada-INFO, JIAWG, and STARS. In addition, product descriptions could be placed in publications including the following:

Ada Letters
Ada IC Newsletter
LCF Newsletter
Government Computer News
Federal Computer Week
Defense Science and Electronics
Defense Science
AF Times
Avionics Week
Systems Command Newsletter

Action items carried over include expanding distributed APSEs/cross-development attributes to a checklist for use in the Guidebook; life cycle support from whole APSE view; and discovering if the Europeans are working on or will be working on whole-APSE issues.

New action items are as follows: put the status report on the Net (Pat Lawlis); update the Tools and Aids Document and make it available on the Net and to Mr. Szymanski for approval (Pat Lawlis); put proposed changes to the purpose paragraph on the Net (Becky Abraham); and put a description of the Target Code Generation Aids and Analysis Toolset Evaluations with approach for development on the Net (Ronnie Martin).

#### 3.2.1.4 <u>Standards Evaluation and Validation Working Group (SEVWG) Status Report</u>

SEVWG reported no deliverables due. The group's accomplishments include: the Guidebook entry for CAIS/CAIS-A; interchange meeting planning; and issues and strategies for CAIS Revision-A. The reported key issues were: upgradeability of tests from 1838 to 1838A; test selection criteria; test effectiveness measures; cost estimation-building and running; and the impact of new mechanisms included in the design of Revision-A.

The projected work includes: reviewing the KAPSE Interface Team/Compliance Working Group (KIT/COMPWG) validation policy for the CAIS-A; filling in the issues and strategies for CAIS-A; and reviewing and commenting on the CAIS-A. The deliverables due for the next two quarters are: in December, CAIS REV-A 1.0. There are no presentations planned.

The action items are as follows: Guidebook entries (Gary McKee, Tim Lindquist, Jeff Facemire); distribute validation policy draft-CAIS (J. Foidl); comments of the validation policy (the entire team); summarize the interchange meeting on the Net (Tim Lindquist); update the issues and strategies (Tim Lindquist); comment files for review process (J. Foidl); and comment on the "NOT_IMPLEMENTED" exception to CAIS-A (Gary McKee).

### 3.2.1.5 CAIS Implementation Validation Capability Working Group (CIVCWG) Status Report

CIVCWG reported no personnel changes. There were no deliverables due. CIVCWG accomplishments were: a review of the SofTech presentation on the CIVC contract for DOD-STD-1838; a discussion of the validation tests - criteria and priorities; and a discussion of the validation diagnostic output/summary. Key issues include: the conversion of MOD test harness for CIVC, the interface to non-CAIS functionality, and the test selection criteria and coverage (breadth vs. depth).

The projected work includes the review of KIT/COMPWG validation policy for the DOD-STD-1838 and a review of the SofTech presentation at the September meeting. There are no deliverables due next quarter. The planned presentation is the SofTech presentation on the CIVC contract. Actions items include: comments on the validation policy (the entire working group); clock comments for 1838 and 1838A (Gary McKee); and the Guidebook entries, Chapter 4 synposis of CAIS, and Chapter 8 title (G. McKee).

### 3.2.2 Discussion with Grady Booch

Bard Crawford introduced Grady Booch, who accepted an invitation to speak on what he has been doing in the Ada world. With Mr. Booch was Dudley McBride, account representative for the Front Range area, and Robert Sjodin, a technical representative for Rational.

Mr. Booch has been working for Rational on object related design issues and gathering material for a new book. Mr. Booch's research into categorization led him to a book by George Lecoff titled Women, Fire and Dangerous Things. Mr. Booch cited one specific example from the book. An aboriginal language called Dydridl is being corrupted by western influence. Due to their method of categorization there is no room for terms such as microwave oven. This problem he then applied to his own recent experiences.

Mr. Booch attended the International Conference on Software Engineering in Singapore. He was responsible for the Tools Fair which involved both the tools exhibition and the tools presentations. The exhibition had approximately 30 presenters. There were 16 tool presentations from all over the world. The categorization problem is revealed here through the variety of tools that do not fit any of the established categories. An example cited was a tool set called Pygmalion from SRA in Japan. Mr. Booch found this set difficult to classify because it is unlike any he has seen in the U.S.

This problem of categorization is also occurring with the R1000. It is hard to classify, because it does not easily fit any known categories. The validation tests reflect this problem. There are aspects to this model that the ACVC can not test. One of these is incremental compilation.

A decision was made in 1982 to use DIANA as the integrating mechanism. There are a number of implications to this. First, DIANA is semantically richer and these semantics become a part of the information stored. This means that not only are Ada units represented as DIANA trees but everything is represented as such. Every object in the environment is in some way represented in some form of DIANA. Another intermediate language used is Buyer Design Facility which is very like DIANA.

This is the sixth version of DIANA, and it is being optimized in a number of ways. On the Ada level, five representations were gone through until one was found reasonable in its efficiency. Another way of optimization was microcoding some of the common DIANA operations, such as retrieving and setting attributes. With DIANA, there is the potential for any system to have incremental compilation. With this, changes can be made to generic bodies without forcing recompilation of any points of instantiation, and also, changes can be made to package specifications without forcing massive amounts of recompilation.

The subsystem mechanism arose when it was discovered that Ada was too small. People were managing collections of packages with no tool support defining sets of packages together. This is really an environment issue more than one of language.

Rational is working with Phillips in Sweden. They are building a fire control system for the navies of Sweden, Denmark, and Finland. Multiple R1000s with the subsystem mechanism is being used to structure the three systems simultaneously. A single system has been built with the common requirements of the three. It is composed of 60 subsystems and there are separate bodies or representations for each subsystem. So a new system can be produced by picking up the bodies of the subsystems that meet the specific requirements of a particular implementation. They found that incremental compilation saved hours of recompilation time.

Dr. Eilers questioned if the multiple representation was on a unit level basis. Mr. Booch replied that they were on a subsystem unit basis. A new subsystem would be spawned to change the body of a single package.

Mr. Booch in replying to Dr. Eilers question concerning additional compilation stated that the problem is the desire to have multiple representations of things. The solution lies in having new representations of subsystems rather than doing a single statement basis. When trying to keep two sets of documentation in synch, a mechanism can be applied that creates a path between the two so that when one changes the other changes automatically. There are two reasons additional compilation was not used. First, it does not work very well in this model. Second, the problem can be solved in another way. Another approach is to use generics to achieve the effect of additional compilation.

The view Mr. Booch is taking in categorizing tools is looking at what piece of the life cycle it tries to address and what processes it tries to automate; its characteristics. The problem is a lack of knowledge of what characteristics the tools have.

Mr. Crawford asked the subject of Mr. Booch's book. Mr. Booch stated that it was on object-oriented design. He said that 80% of the customers were trying to apply these techniques in a variety of ways. There are people outside the Ada community who are very interested in object-oriented design; Nicholas Barrett is working on a new object-oriented language called Oberon. Mr. Booch expects to have most of his research finished by the end of the summer.

Mr. Booch attended an Ada presentation held in Washington, D.C., sponsored by Ralph Crafts. The audience of 250 people was composed of press and congressional staff. Brian Phlug from Boeing spoke on Boeing's specialization with Ada. There was a presentation from NASA and SEI. Mr. Booch gave a presentation on the international market for Ada. The result will be a white paper to be circulated among some key congressmen.

In response to an inquiry from Mr. Szymanski, Mr. Booch suggested that the E&V Team look at the international community with regard to tools, to study the trends and what people are actually working on. He mentioned a joint environment project with Dr. Kashida of SRA and a consortium of universities. One of the people to contact in the U.S. is Dr. Lloyd Williams of the Rocky Mountain Institute for Software Engineering. The project is not focused on a particular language, but addresses the problems of software development. A paper was published at the last conference which was held in Singapore.

Mr. Booch announced that the International Software Conference is held yearly; it is sponsored by 20 to 30 different computer societies. The next one is in Pittsburg in October. The program chair is Larry Druffle and Mr. Booch is the tools chair. Papers are to be in by September. This conference is a good place to hear ideas for tools that may become products in the future.

Mr. Leavitt asked if Mr. Booch had heard of the ACEC and what he expected from it. Mr. Booch replied that he knew of its existence but does not have any predefined expectations. He stated that some customers are developing their own tests which are domain specific.

Mr. Szymanski asked why the vendors have not released any statistics on individual compilers. Mr. Booch answered that what statistics they have are in the abstract and are therefore meaningless. We should decide upon characteristic tasks that need to be performed in software engineering and use that basis to evaluate different environments based on the performance of those tasks.

In his closing remarks, Mr. Booch stated that we are just beginning to see the productivity factors of Ada and that there is a need for stability in the next few years. Mr. Booch thanked the E&V Team for the opportunity to speak with them.

After thanking Mr. Booch for his presentation, Mr. Szymanski adjourned the 24-26 May 1988 quarterly meeting of the E&V Team.

## LIST OF ATTENDEES

| | |
|---|---|
| Abraham, Rebecca Capt.<br>AFWAL/FIGAda<br>WPAFB, Ohio 45433-6543 | Burlakoff, Mike<br>S.W. Missouri State Univ.<br>Computer Science Dept.<br>Springfield, MO 65804 |
| Camp, John<br>AFWAL/AAAF-2<br>WPAFB, Ohio 45433 | Crawford, Bard<br>TASC<br>55 Walkers Brook Dr.<br>Reading, MA 01867 |
| Eilers, Dan<br>Irvine Compiler Corp.<br>10821 Sky Park Cir., #1<br>Irvine, CA 92714 | Facemire, Jeff<br>SofTech<br>1300 Hercules Dr.<br>Suite 105<br>Houston, TX 77058 |
| Fanning, Shawn<br>SofTech<br>16875 W. Bernardo Dr.<br>San Diego, CA 92127 | Foidl, Jack<br>TRW, Systems Division<br>Suite 205<br>9265 Sky Park Ct.<br>San Diego, CA 92123-4213 |
| Francl, Fred<br>Sonicraft, Inc.<br>8859 S. Greenwood<br>Chicago, IL 60619 | Gicca, Greg<br>GTE Government Systems<br>1 Federal St.<br>Billerica, MA 01821 |
| Gray, Lewis<br>TRW Federal Systems Group<br>2900 Fair Lakes Parkway<br>Fairfox, VA 22033 | Hazle, Marlene<br>MITRE, Corporation<br>Burlington Rd.<br>Bedford, MA 01730 |
| Holmes, Tracy<br>GTE Government Systems<br>1 Federal St.<br>Billerica, MA 01821 | Impicciche, Alan L.<br>NAC Code 826<br>Naval Avionics Center<br>60000E 21st. St.<br>Indianapolis, IN 46219 |
| Lawlis, Patricia Maj.<br>AFIT/ASU<br>3318 E. Dry Creek Rd.<br>Phoenix, AZ 85044 | Leavitt, Thomas<br>Boeing Military Airplane Co.<br>P.O. Box 7730, MS K80-13<br>Wichita, KS 67277-7730 |
| Lindquist, Tim<br>Computer Science Dept.<br>Arizona State Univ.<br>Tempe, AZ 85287 | McBride, John<br>SofTech<br>300 Hercules Dr., Suite 105<br>Houston, TX 77058 |
| McKee, Gary<br>GARICAR<br>P.O. Box 3009<br>Littleton, CO 80121 | Mark, Donald<br>RADC/COEE<br>Griffiss AFB, NY 13441 |

Marmelstein, Robert Lt.
AFWAL/AAAF-2
WPAFB, Ohio 45433-6543

Martin, Ronnie
Software Eng. Research Center
Dept. of Computer Science
Purdue University
W. Lafayette, IN 47907-2004

Rhoads, Barbara
Oneida Resources, Inc.
3578 Kettering Blvd.
Suite 300
Dayton, Ohio 45439

Szymanski, Raymond
AFWAL/AAAF-2
WPAFB, Ohio 45433

# MINUTES

# OF THE

# EVALUATION AND VALIDATION TEAM

# GENERAL SESSION

7-9 SEPT 1988

The task for the Evaluation and Validation of Ada Programming Support Environments (APSEs) is sponsored by the Ada Joint Program Office (AJPO).

## TABLE OF CONTENTS

SECTION PAGE

1.0 WEDNESDAY, 7 September 1988 . . . . . . . . . . . . . . . . . . . 1
 1.1 Welcome, Introductions, and General Business . . . . . . . . 1
 1.2 Independent Validation and Verification (IV&V) of Ada Compiler Evaluation Capability (ACEC) . . . . . . . . . . 2
 1.3 Evaluating Ada Compilers . . . . . . . . . . . . . . . . . . 5
 1.4 Joint Integrated Avionics Working group (JIAWG) . . . . . . 8
 1.5 Ada Compiler Evaluation Capability (ACEC) . . . . . . . . . 14

2.0 THURSDAY, 8 September 1988 . . . . . . . . . . . . . . . . . . . 18
 2.1 Ada Based Integrated Control System (ABICS) . . . . . . . . . 18

3.0 FRIDAY, 9 September 1988 . . . . . . . . . . . . . . . . . . . . 26
 3.1 Closing Session . . . . . . . . . . . . . . . . . . . . . . . 26
  3.1.1 General Comments and Announcements . . . . . . . . . . . 26
  3.1.2 Working Group Status Reports . . . . . . . . . . . . . . 27
   3.1.2.1 CAIS Implementation Validation Capabilities Working Group (CIVCWG) Status Report . . . . . . 27
   3.1.2.2 Requirements Working Group (REQWG) Status Report . 29
   3.1.2.3 Ada Compiler Evaluation Capability Working Group (ACECWG) Status Report . . . . . . . . . . 30
   3.1.2.4 E&V Technology Classification Working Group (CLASSWG) Status Report . . . . . . . . . . . . . 31
   3.1.2.5 Standards Evaluation and Validation Working Group (SEVWG) Status Report . . . . . . . . . . . 33
  3.1.3 Closing Remarks . . . . . . . . . . . . . . . . . . . . . 34

LIST OF ATTENDEES . . . . . . . . . . . . . . . . . . . . . . . . . 35

## 1.0 WEDNESDAY, 7 SEPTEMBER 1988

### 1.1 Welcome, Introductions, and General Business

Chairman Ray Szymanski opened the September meeting of the Evaluation and Validation (E&V) Team by welcoming the team back to Wright-Patterson Air Force Base. Mr. Szymanski brought the team up to date on Don Jennings' illness and announced that donations for Don were being accepted. Mr. Szymanski, then, introduced his Branch Chief, Donna Morris, to the team.

Mr. Szymanski informed the team that the latest revision of the Ada Compiler Evaluation Capability (ACEC) Questions and Answers was sent out the week prior to the meeting. He indicated that the Ada Joint Program Office (AJPO) has been directed by the House Appropriations Committee to address file efficiency. The Ada Validation Facility along with Lt. Marmelstein and Mr. Szymanski are drafting a plan to address the House Appropriations Committee's direction. The draft plan will be presented by Mr. Szymanski to the AJPO on September 20. The plan will be refined over the next few months. A report is due back from the House Appropriations Committee in February. The involvement of E&V Team members at this point has not been addressed by the AJPO.

Mr. Szymanski stated that handouts on the ACEC are available for review. Any comments will be taken by Dale Landing throughout the week. These comments must be for minor adjustments only. The ACEC Questions and Answers will be formatted as a special supplement to the Language Control Facility Newsletter.

Items of interest concerning the move of AJPO to Air Force Systems Command include:

- Virginia Castor will be remaining at her current position at the Department of Defense (DoD).

- No changes are expected in E&V Team activities.

- The team has a new sponsor, Lt. Col. Rick Gross.

Mr. Szymanski commented that distinguished reviewer contractors who are not directly funded by AJPO and do not have a direct contract with the E&V Team are not present at this session. Draft copies of the request for a legal opinion on this matter and the necessary background information has been distributed to local management and to the contracting office for comment. The letter will be presented next week to the contracting office and then forwarded to the Judge Advocate General (JAG). The JAG has proposed a six-week turnaround from the time the letter is received; therefore, it will be late October or early November before a decision is made.

Mr. Szymanski stated that there are two upcoming presentations at the Ada JOVIAL User's Group (AdaJUG), one on the ACEC and one on the Reference System. There will be a Birds of a Feather session on Wednesday of the AdaJUG. The Birds of a Feather session on the Reference System is still uncertain. The ACEC will be presented at the Society of Automotive Engineers (SAE) meeting in October by Tom Leavitt; it will also be presented at the Ada Expo.

The next E&V meeting will be held December 5-7 in San Diego, California. This will accommodate those attending the AdaJUG the prior week.

There is to be a public review of the Ada Language System/Navy (ALS/N) on September 21. The target machines will be covered - the UYK44, UYK43, AYK14, and AdaVAX which is the VAX/VMS target. The military target machines and the AdaVAX portion have all been passed by the ACEC. At this point, the throughput is not equivalent to the Digital Equipment Corporation (DEC) Ada; it is about four times as slow. In trying to improve performance, they have reduced the Kernel Ada Programming Support Environment (KAPSE) interface which cuts down the portability.

Mr. Szymanski announced that Liz Kean has been retired from the E&V Team. Don Marks will be her replacement. Mr. Szymanski eulogized Liz saying, "She was a productive member; an original member of the E&V Teamers and one of the most consistent people on the team. She was well-known for her work on taxonomies with respect to Ada Programming Support Environments and her work in RADC contracting, especially those who tried to develop compilers in the Ada World for 1750A target."

Gary McKee asked Mr. Szymanski to explain the earlier comments on the House Appropriations Committee activities. Mr. Szymanski answered that on June 10 the House Appropriations Committee issued a paragraph with respect to the AJPO, giving them direction in five areas. One of those areas stated that the Ada Joint Program Office will do compiler efficiency as part of validation procedures. This pushes the Ada Compiler Evaluation Capability right in the forefront as it is currently the tool being reviewed to perform this additional function. Question number three of the ACEC Questions and Answers gives a brief synopsis of this matter.

Mr. Szymanski showed a brief video to the assembled group before turning to the day's speakers.

## 1.2 Independent Validation and Verification (IV&V) of Ada Compiler Evaluation Capability (ACEC)

Mike Burlakoff
Southwest Missouri State University (SMSU)

This presentation covered a ten-week summer faculty program sponsored by the Air Force Office of Scientific Research (AFOSR) on the Ada Compiler Evaluation Capability (ACEC).

Mr. Burlakoff displayed a slide summarizing the ACEC system. The following is the information he presented:

- The ACEC system consists of approximately 1,074 systematically developed individual tests which measure and permit performance comparisons of Ada systems.

- Example test categories:

  - Efficiency of various language features.

    - Implementation dependent options.

    - Optimizations.

    - Performance under load such as task loading.

    - Tradeoffs such as design issues and coding style.

    - Operating System Kernel efficiency testing such as Task Scheduling, Exception handling, I/O and Memory Management.

- Applications, both classical and Environmental Control System (ECS) specific.

    - Primary ACEC outputs are time and space performance.

    - Statistical tools permit automated analysis of results and comparison of performance between systems.

The primary purpose of the Independent Verification and Validation (IV&V) was to test the usability of the ACEC. The major areas that were investigated include:

- The execution of the Test Suite and verification of the results.

- The use of contractor supplied Test Procedures/Scenarios.

- The use of Statistical Analysis Tools to determine correctness and usefulness.

- The reviewing/critiquing of both the User's and Reader's Guidebooks.

The testing was done on two hosts, the AFWAL VAX 11/785 and the SMSU VAX 11/750. The times on the 750 were almost double that on the 785; this is substantially slower.

The User's Guide was used on the scenarios and procedures which were verified. The comments which were minor were given in the final report and also in the User's Guide.

Two subsets were chosen to be repeatedly submitted as batch jobs through the 24-hour period on the VAX 11/750. For this experiment, the time calculations were consistent. Mr. Burlakoff noted two things: (1) the times were given in microseconds, and (2) there was some divergence of time in the multi-programming system.

The analysis tools were used and analyzed. Also, the User's and Reader's Guides were reviewed. All comments with recommendations were submitted in a final report.

For the User's and Reader's Guide, an overview of test suite coverage early in the document was recommended. A major recommendation for the User's Guide was additional examples on the setup/use of the analysis tools. For the Reader's Guide, the recommendation was for additional explanation and examples of statistical computations and output. It was felt that the statistical tools in particular needed additional examples.

The miscellaneous recommendations were the inclusion of a delivery tape help file; for the FORMAT program, suggestions to generate a complete Ada package of raw data; and for the MEDIAN program, better explanations and suggested output changes. Another recommendation was the continued evaluation of the test suite.

Mr. McKee questioned what was meant by the FORMAT program suggestion. Mr. Burlakoff responded by saying that in running the subset of the ACEC some manual work is involved in generating the data needed for MEDIAN. The recommendation was for FORMAT to generate the complete MEDIAN data saving the users some manual editing.

Mr. Burlakoff observed that the ACEC is a quality product based upon the following factors:

- The wide range of coverage.

- The extensive set of tests.

- The difficult Ada issues measured (Tasking, I/O, Exceptions, Memory Management, Interrupts, etc.).

- The completeness and usability of the User's and Reader's Guides.

Questions were then entertained from the team.

Dr. Crawford asked why the 750 results were almost twice as slow as the 785 results. This was due to using different hardware. Mr. Leavitt stated that the 750 is not a chip base processor; it is, in general, just a slower system.

Mr. Leavitt was then asked if the results corresponded to his expectations. He stated that there are several options that can be bought on the hardware depending which one is factored in. Mr. Burlakoff agreed stating that it alsodepended on the I/O.

Mr. Szymanski asked Mr. Burlakoff for his recommendation of issues to be further addressed. Mr. Burlakoff said he would like to see a continual evaluation of the ACEC as the program develops.

As a follow-up, Lt. Marmelstein commented that there is a need to look at other areas: test failures and the need to run MEDIAN; the test flag outlier, running extremely slow or extremely fast; and the code generated as part of the evaluation. These will be pursued in Phase III. There is also a need to have a single system report.

Mr. Burlakoff was asked to what degree attending the E&V meetings helped him. He said he was aided through his involvement in decisions on the ACEC, and by seeing the early versions of the User's Guide and the Reader's Guide.

Mr. Szymanski announced that a follow-on to this activity with Mike Burlakoff is being pursued. If this idea is realized the team will be notified.

Mr. Burlakoff was asked about the level of difficulty encountered by someone unfamiliar with the system. He stated that it would be time consuming but a review of the User's Guide would be an advantage.

Lt. Marmelstein said that a copy of the test suite was given with a guide to the Ada Validation Facility (AVF) for preliminary evaluation. They were able to get the test suite up and running within a week, but they are familiar with running the ACEC which would be of help.

Dr. Crawford asked, if he had a major program in electronic warfare, would there be a test in the ACEC for this. Mr. Leavitt stated that there are a limited number of tests run directly from applications. The question is the use of the language. Any application has to be written to use the features of the language. The design was to include enough of a sample of the language features and instructions so that if anything in a system is noticeably strange it will show up in one place or another.

This concluded Mr. Burlakoff's presentation.

## 1.3 Evaluating Ada Compilers

Lt. Marc Pitarys
AFWAL/AAAF-3

Before the second presentation, Mr. Szymanski announced that depending upon the outcome of the JAG reading concerning the legality of distinguished reviewers, one of two things will occur. If the response is positive, an effort will be made to have all distinguished reviewers attend the December meeting. If the ruling is negative, they will still be invited for one last meeting, the December meeting. This will give them the opportunity to close out any unfinished business with the team. They will also be given a proper send-off for their productivity and accomplishments on the E&V Team over the last three years.

Mr. Szymanski then introduced Lt. Marc Pitarys of AFWAL/AAAF-3, the Software Concept Group. The presentation covered the work that has been going on in evaluating Ada compilers in-house at the Avoinics Laboratory. Also discussed were the experiences and observations using the ACEC, and the preliminary results from using the ACEC on several compilers with several target processors. A few brief conclusions were also offered.

Lt. Pitarys offered three reasons for the evaluation effort. First, there is little quantitative data on Ada's performance for embedded computers. Second, data is needed for compiler selection, avionics design, and cost decisions. Third, the Avionics Laboratory has compilers and processors.

In the Ada compiler evaluation, some of the toolsets were targeted towards both the 32-bit and the 16-bit processors. Benchmarks included not only the ACEC but the Performance Issues Working Group (PIWG) and the Common Ada

Missile Packages (CAMP) also. The results from the PIWG were distributed to other DoD organizations and to industry that includes some of the other compiler vendors.

At the present time, accomplishments include an evaluation of four Ada 1750A compilers targeted to several 1750A processors including the V1750A. This used the PIWG benchmarks. Also evaluated were the DEC's Ada compiler on three VAX targets using the ACEC and the ALSYS Ada compiler on the Z-248 which is an IBM compatible. The TLD Ada/1750A compiler was evaluated on a Sperry 1750A using the ACEC.

Several factors became apparent when testing the cross compilers with the ACEC. When setting up, math packages used in the test suite and the COM files have to be modified. This takes time, sometimes a full day. The COM files need to be modified since they target the DEC Ada compiler. The proper commands must be inserted to execute the cross compilers on the VAX. The text I/O package must be modified, and the account quotas must be adjusted.

When testing a cross compiler with the ACEC, it is done in two phases. The first time, compile, link and execute as many tests as possible. After the first run, the failed tests can be identified and removed. As is known from the ACEC, half of the files provided contain multiple tests and if one test fails in the file it has to be removed and the file recompiled to run the remaining tests.

There is a possibility that the multi-test files will have to be reduced. This may apply in particular for a 1750A compiler when the storage capacity of the processor is exceeded.

Areas that consume a lot of time include downloading and entering the data in the MED_Data package.

The estimated testing time using the ACEC is from one man-week to one man-month. Many factors contribute to this:

- The amount of time it takes to change the math package and for the compiler to work.
- Downloading.
- Separating the files.
- Narrowing down to the failures.

This is just running the data; to analyze it would take longer.

Lt. Marmelstein interjected that concerning time consumption the link phase takes three times as long as the compilation phase and setup time depending on the target.

Lt. Pitarys reiterated some general observations on the ACEC stating that it stretches the VAX accounts. The quotas that the system sets up can be adjusted several times. Concerning memory, a lot of disk space can be consumed by running the ACEC with several compilers which could be a serious problem. Also, Central Processing Unit (CPU) time is noticeable on the VAX. When compiling, it is probable that every available CPU second will be consumed.

It was noted that the ACEC shakes and breaks the compilers. The following can be identified after running some tests.

- The limits of how many variables the programs can have.
- The size of the programs should be in the upper limit of the programs' size that the compilers can accept.
- Stack overflows.
- There are internal compiler errors for a system access violation, VAX error, or illegal VAX OP code being executed.

Lt. Pitarys suggested the following areas for the team to pursue.

- Classify the errors and failed tests in order to determine what tests are critical using the compiler selection for some type of weighing system for errors.
- Assistance for interpreting single compiler test results.
- Comments by the evaluator on the MEDIAN data.

Although MEDIAN gives a lot of output, some things can be missed without the specific comments of the evaluator. MEDIAN may say there is a compiler failure or a run type failure but not disclose what the error is. It could be a storage size problem or a numeric error exception.

In the preliminary results, the system factors are a combination of compiler and system performance. The DEC Ada compiler was tested running the MicroVAX 3, VAX 11/785, and MicroVAX 2; also evaluated were the ALSYS Ada Compiler for the Z-248 and the TLD compiler for the 1750A.

It was asked how the system factor is calculated. Mr. Leavitt explained that the system factor is essentially a result of fitting the model and computing the raw data in two dimensional ranges for each system and each problem. Fitting one factor for each problem and one for each system, they are sent through a data system to compute those factors. They then represent the average performance of the system. The performance of that problem is averaged over all the systems and is scaled as a standard. Everything else is relative to the standard with a large number meaning that it ran slower than the first one.

Mr. McBride asked if these numbers represent execution time only or a combination of compilation time and execution time. When running the analysis program, it computes system factors depending on the data chosen to be put through. Execution times, compilation times, code space, and expansion space can all be run.

For the system factors, the same execution codes used on the DEC Ada compiler were run on the three different VAX targets. The VAX 11/785 ran 2.23 times slower than the MicroVAX 3. The MicroVAX 3 operated faster and had a higher disk access rate which is important for the length of time it takes to redirect the memory. On the host machine, the system factors should be looked at along with the percentage of tests executed.

Mr. Facemire asked why one version has ten times, with another having four, and some having no unreliable times. Lt. Pitarys stated that many factors figure into the loading on the host. These are different machines and the times measured are within the range of error. It is a combination of hardware and system loading.

Lt. Pitarys concluded his presentation with the following statements.

- The AFWAL/AAAF is evaluating compilers for embedded targets.
- The ACEC is the preferred test suite for obvious reasons.
- The results are made available to other DoD users when possible.
- The testing will continue until other organizations step in to fill the data void.

Mr. Weiderman asked what were the most important aspects of the ACEC to Lt. Pitarys, and also what additional information he gathered outside of PIWG.

Lt. Pitarys stated that one of the most important distinctions between the PIWG and the ACEC were the tools provided to analyze the data. In this area, the ACEC provides the mechanism for analyzing the data. Also, the ACEC covers many features of the language. PIWG does not address all of the issues in the language. So the conclusions are that the ACEC is more comprehensive and more scientific, but the PIWG is faster.

Mr. Weiderman followed-up by asking if Lt. Pitarys' opinion on any of the compilers had changed as a result of this effort. Lt. Pitarys stated that the ACEC gives the user more confidence with the problems in the compiler. There is always the problem of whether the compiler will be able to handle a large amount of code. This is caught sooner with the ACEC which provides application benchmarks.

Lt. Pitarys in answering a question from Mr. Szymanski stated that there are no comparisons between the ACEC and PIWG in documentation. He then concluded his presentation.

## 1.4 <u>Joint Integrated Avionics Working Group (JIAWG) Software Task Group</u>

Michael T. Mills
ASD-AFALC/AXTS

Mr. Szymanski made the following announcements concerning the agenda.

- The meeting time of the Working Groups would be switched.

- A briefing on the Ada Based Integrated Control System (ABICS) effort would be given at 4:00 Thursday.

- Lt. Marmelstein would give an Ada Programming Experiment (APEX) demonstration at 1:00 Wednesday.

Mr. Szymanski proceeded to introduce the third speaker of the morning, Mike Mills.

The Joint Integrated Avionics Working Group (JIAWG) was formed a year and a half ago. It is composed of several test groups and subtest groups in both hardware and software. The JIAWG is headed by the deputy program managers of the three programs along with a contractor board consisting of program managers from the contractor end as a steering group. The steering group is composed of an architecture and a software section. The architecture section covers the common 16-bit processor, 32-bit processor, several different hardware type items, and sensors. The software test group is comprised of the Ada Issue Subtest Group, the Reuse Group, and the Common Software Engineering Environment Group. Each of these groups are headed by the three principals from the three services.

The presentation concentrated mainly on the Ada group. The Ada Subtask Group started receiving contractor influence around July 1987. A meeting was held in Washington with all contractors present. Each team presented different issues, part of which were the Ada issues concerning items to be changed in the language. In February 1988, a JIAWG position paper was widely circulated; after receiving contractor input a second paper was published. Later a presentation was given at the AdaJUG which had decided to establish a JIAWG corner where JIAWG-type presentations can be given. The first presentation was on Ada issues.

A meeting held on August 9 was a milestone which brought all the contractors together. At that time, there were 26 different issues which have been narrowed down to four language issues that everyone agrees on. On August 23, these were put on the bulletin board, and a presentation was given at the Ada Language Issues Working Group which provided some feedback.

Work is starting with the Software Engineering Group. The chair is John Goodenough; this group maintains the language in the International Standards Organization (ISO). He is also at the Software Engineering Institute (SEI) and in charge of the Ada 9X project. This is the next revision of the Ada language for the 1990s. He will work with the JIAWG bringing issues before the ISO and into the language.

The next JIAWG takes place the week after the E&V meeting at the Software Engineering Institute. It will cover four categories: Ada Language Issues, Implementation Problems, Runtime Environment Problems or Standard Runtime Environment Interfaces, and Ada Architecture Incompatibilities.

In a summary of comments, the items were labeled contractor 1A, 1B, 2, and 3. The stated problem was a lack of production quality compilers. The first group had some minor concerns but the second had many issues. Alternate testing packages are being used. Preemptive priority scheduling was wanted which according to ISO is interpreted as having an unsynchronized rendezvous feature working on blind send type items and not having a full rendezvous when all that is going to be provided is a packet of data. Other issues were the pointers to static objects, and improvement of compiler implementations.

A narrative followed on how these items are being evolved. These are all prime contractor influenced. A contractor uses an alternate low level tasking package knowledge similar to what Ada Run Time Environment Working Group (ARTEWG) has proposed. They do not want to change the language. Ada tasking is the weak point in the standard, ill-suited for embedded systems. The decision was made to ignore tasking and achieve concurrence with multiple programs running under an operating system. Tasking in its current form is bad for real-time.

A rundown of the 26 issues was provided. The first three are interpreted as part of the language: one is the mandatory implementation of requirement specifications; second is preemptive priority scheduling; and the third is making delay expiration force a scheduling activity. This is a problem in compiler implementation; at this time, when performing a delay statement, the compiler sometimes does not get back to the user. An added issue is making pragma elaborate mandatory. The language issues include: dynamic priorities, and unsynchronized rendezvous where data is passed without synchronization. Some ARTEWG type items do this.

Mr. McKee asked for more information on the ARTEWG interface. A group of interfaces provide some extensions to the Ada language which are interfaces to the compiler. One of these interfaces is unsynchronized rendezvous which gives another alternative to rendezvous.

Mr. McKee asked how it is implemented. Mr. Mills said it is implemented in packages, but to the package which encompasses several procedures. This is different from the Software Engineering Environment (SEE) distributed Ada runtime which is an Ada package and mainly compiler independent.

Mr. McBride inquired if the interfaces require modifications to the compiler through the runtime system; Mr. Mills agreed. Mr. McBride stated that the interfaces are not within the definition of Ada and that they are not really accessible to application writers; whereas, the buffer task approach is something an application writer can do. Mr. Mills agreed saying the buffer task is an Ada workaround while with ARTEWG the compiler interests have to provide it in the compiler.

Another issue is the exception problem. When running, aborting the program when in flight mode or tracking down a target will cause problems. The main workaround will make sure all of the blocks have an exception handler but there would still be concern. It is at present undecided whether this is a language issue.

One that is still considered a language issue is access types to static objects. It is being debated whether or not this should be allowed.

In runtime, the concerns are about alternate tasking packages including the ARTEWG and alternate tasking package, interrupt management, and pragma interrupt task. Among the pros and cons, the main concern is the extent of functionality of the data for real-time.

Mr. Burlakoff asked if a compiler user would have to have two tasking packages with the compiler. Mr. Mills stated that this alternate is in the manner of an escape mechanism providing another type; it is an ARTEWG type item, a standard interface, but it could be a runtime type item also.

One item the contractors do not want is priority inheritance. This means that they do not want priorities to be automatically inherited. Another item, Hardware Interrupts, was considered a bad idea as far as using it in place of priority tasks.

Former issues include: exception types, ragged arrays, static pointers, and extend capability of discriminates for variant records. Items that were considered non-essential/unneeded include mailbox capability, which is really not non-essential but is part of another issue that is encapsulated in it. Another is making pragma elaborate mandatory which is apparently already interrupted in the language. Machine code insertions are to be made mandatory; some consider it already mandatory. Usage type operations should be registered to help optimizers do the machine cut insertions more efficiently declaring static data efficiently. Declaring static data allowing subroutines to remember between invocation was thought to be bad software engineering and it was agreed to throw it out. It was also decided that extending image attributes to real types was not needed. Two items that might be revisited are allowing procedures to be passed and adding a subprogram data type.

Other non-essentials include: allowing array slice; negative integers in for loops; allowing access as part of a floating point; and limiting scopes of withs and uses. Some want a capability within exception handling to identify raised exceptions to be located during a flight and analyzed later after the plane has landed.

The long term purpose of these issues is to take the awkwardness out of Ada. Things should be done in a more straightforward manner in the long term life cycle support of weapon systems.

The four issues from the August 9th meeting are: dynamic priorities, access types to static objects, converting between Ada attribute and access value, Ada task scheduling allowing priority accept and select queues and having an alternate mechanism within the language.

The reason for going to dynamic priorities was that at this time the Ada language provides only static priorities unless a rendezvous is used. However, rendezvous is mainly for task communication and changing priorities is awkward. Mode changes is one of the main requirements for this. An example would be going from a navigation function which is more critical at Terrain Following/Terrain Avoidance (TF/TA) than when flying at forty thousand feet. A way is needed to dynamically change the priorities. Another example is tolerance aspect and system use, the minimum threshold of the system; priorities might need to be changed. For this and for all features, there is a need to increase the portability and reuse. Many contractors think that using a rendezvous for a workaround is cumbersome and inefficient. Some features on dynamic priorities can be misused requiring some discipline type programming in order to avoid unwanted independencies of the system.

The second issue is allowing pointers to static objects. Some workarounds for initialization processing are being looked at. There is a lot of disagreement over this in the Special Interest Group Ada (SIGAda) and others; so it is not established if this one will make it or not.

Justification which is needed to restrict the use of pointers to dynamically declared objects will not meet the requirements of applications where dynamically declared objects are prohibited.

In response to a query for clarification from Mr. McKee, Mr. Mills said that for certain applications capability is lessened with using just dynamic objects. For example, large maps of the world and read-only memory would have to be copied into heap space which would access a lot of memory.

Mr. McKee felt that this could probably be more efficiently handled by a runtime environment control of heap space without changing the language. Mr. Mills stated, that unless the language forces the compiler to allow it, it might be a non-portability feature.

Mr. Mills stated that there is a lot of controversy over this. One suggestion is to pre-elaborate a certain point to get around the problem. But with pre-elaboration can come maintenance troubles. It could result in the source code and object code not correlating as to placement of items in the memory.

The third issue is conversion between access attribute and access value. These are requirements to be able to go back and forth between address attributes of an object and access values or pointers which point to the object. Having unchecked conversion as the only mechanism for this function is unsatisfactory as there is no guarantee that a compiler will implement access types in a way that support it. The goal is to go from attribute to pointer value converting between the two and yet have the same type of checking as when going through unchecked conversion. It was brought out at JIAWG that the two might not be the same. The access pointer could be pointing to the middle of a word which would not be the same as an attribute. However with most compilers, the two are the same. What is needed is a functionality converting one to the other when they are the same. If not, an indication can be received which can be triggered. There are a lot of different issues in which nonchecked conversion is a workaround and this type of mechanism might help in real-time embedded applications.

The fourth issue is concerned with Ada task scheduling. If items could be specified in a rendezvous, the accept queues and select alternatives should be priority ordered as opposed to current or accept queues which are first in first out (FIFO) and select being arbitrary depending on the way the compiler implemented it.

Without justification, there is no guarantee that the highest priority tasks are serviced before other tasks. There is a need for better repeatability. In some instances FIFO would be the better way but in other instances, priority order would be better. Also, for capability reasons it might be the best way to implement it. This feature would be through a pragma. So there might be two alternatives for task entry. Whether priority ordered or FIFO, doing priority order scheduling can cause task starvation for lower priority tasks. It is felt that having this feature in the language rather than an escape through implementation in general type features would help the embedded system people.

A scheduling out algorithm called rate monotonic scheduling is being looked at by the SEI and the Ada Preparatory Group. This type of scheduling depends on locations or periodicity of the interrupts or requests for the task service. Based on periodicity it will schedule high priority versus low priority for servicing tasks in that order. One disadvantage is that if a distributed run-time system arises in the processor, it would not work in the current form. This priority order type mechanism is apparently needed and might have a good chance of getting into the language.

In the Reusability Task Group working group meeting, it was raised that there is a need for a configurability mechanism other than generic risk. One possible mechanism to do this might be the discriminate for generic risk. This needs a lot of study and is something new that was brought up.

During the August 9th meeting, some issues were put in the revisit category. These were ragged arrays and discriminates of variant records; allowing procedures to be passed; and a subprogram data type.

These are out of the original 26 issues. Some items are labeled implementation issues such as allowing the user to specify what registers are already being used which is when machine cut insertions are done. This could be a help for the optimizer. Another is the identification of the raised or logging exceptions; also, critical sections as in some way for Ada to specify critical sections of code which is normally done with SIM-4 type items.

Mr. Mills displayed a flowchart of the work being done in the Ada effort. It originated with input from contractors, then was put in a position paper, circulated, and presented to the Ada Subtask Group. There were four different types of commissions. There were additions to the Ada issues; one is implementation issues and another is runtime.

Runtime is looking at a standard interface and standard kernel, something which all programs could use and provide more commonality between their software programs. With these Ada issues, JIAWG is working with SEI and the Ada Language Issue Working Group. The feedback achieved is helpful.

With the Implementation Issues, JIAWG is in touch with PIWG, the E&V Team, and runtime interests. ARTEWG is interested; SEI is involved and briefings are given at AdaJUG. SEI is working on putting the issues into the four different categories.

Language issues include runtime, implementation, architecture incompatibilities, and Ada language issues. These will go through SEI and Ada Repertoire Group which was formerly called the Ada Maintenance Committee; this is the group which decides what goes in the language and is out of the International Standards Organization. Ada is an international standard now.

Ada is also monitored by the Ada Board and the AJPO. The SAE are also kept briefed. The Ada Uniformity Repertoire Group headed by Robert Brewer is interested in creating uniformity in compiler implementations.

Mr. Szymanski asked if there were others connected to JIAWG besides the contractors and government representatives. Mr. Mills replied that the compiler implementers are only through the three primes. They are being funded out of the various primes or subs to the primes for different functions.

Dr. Crawford inquired if John Goodenough or any others sat in on the meetings of the JIAWG Issues Group. Mr. Mills said not at the present time but someone would be at the next meeting which is at the SEI.

Ms. Hazle asked that when developing guidelines or recommendations for language use for Ada 9X if the contractors would be solving similar problems in similar ways. Mr. Mills stated that these are two parallel items; the Ada 9X for the long term, and in the short term, a management type of standard in the JIAWG or an extension to the distributed runtime system in SEI, or something from the contractors. All efforts should be following the same path which should help in the overall maintenance and portability during upgrading of different processors.

A newly added task is a study of the ACEC to see where JIAWG could interface with the E&V Team. JIAWG is interested in any available information from the team.

Ms. Hazle asked if anybody in the JIAWG could decide to use the ACEC tests on all the compilers being used on the Advanced Tactical Fighter (ATF) and other programs. Mr. Mills said yes. The procedure is that a group within the JIAWG introduces the idea and it is brought up the chain to be okayed by the Steering Group Committee; then it will be mandated through JIAWG.

Replying to a question from Mr. Leavitt, Mr. Mills stated that the approach to ACEC is undecided, but it would be more than just implementing it. It would involve interfacing with the team and doing anything possible to help.

Mr. Szymanski proposed the following scenario: The JIAWG tests the ACEC and finding areas to be enhanced places their findings before the E&V Team. The team would entertain the suggestion pending on the availability of funds. Mr. Leavitt agreed with this scenario as did Mr. Mills who went on to state that he was unsure of the funding. This concluded Mr. Mills' presentation.

## 1.5 <u>Ada Compiler Evaluation Capability (ACEC)</u>

Lt. Robert E. Marmelstein
AFWAL/AAAF-3

Before beginning his presentation, Lt. Marmelstein made the following announcements.

- Phase 1 of the ACEC was finished in August culminating in the delivery of Version 1.

- The ACEC was shipped out to the Data Analysis Center.

- A kickoff meeting was held when Version 1 was delivered. This will be discussed further during the Ada Compiler Evaluation Capability Working Group (ACECWG). The meeting had centered on the additional execution type performance tests which are going to be in the suite. Additional areas also to be incorporated are: diagnostic message evaluation single system reporting, labor, robustness, and evaluation and symbolic debugger.

- Another deliverable included the following action items: (1) a revised ranking of the additional areas. A single system report for Version 1 will be done; and (2) a summary was completed of the discussion on diagnostic messages at the last ACECWG meeting.

- An ACECWG agenda is available to the team.

The presentation was a preview of a briefing for the System Program Offices (SPOs) and possibly the JIAWG. Validation is not enough. If there is only one Ada compiler available for a particular target, some people are mandated to use it. Sometimes when primes propose to use compilers, the government assumes that the primes have already evaluated the compiler which is not always the case. This assumption can lead to cost overruns, productivity problems, and scheduling delays later on. Lt. Marmelstein suggested that compiler evaluation be mandated by JIAWG. He will speak to other SPOs advocating this now that a product has been released.

Major points in the ACEC objectives include: comparing the performance of several Ada compiler systems; isolating the strong and weak points of a specific system; determining what significant changes were made between releases of a specific compiler; and predicting the performance of differing Ada design approaches.

A summary of the ACEC approach include: looking at compile time, execution time, and code size. Runtime size should not be explicitly measured; data can be extracted from it and run through MEDIAN as long as it is consistent and quantifiable. This includes benchmark results from other test suites if need be.

Lt. Marmelstein briefly referred to the slide on execution test classification. The slide displayed the following information:

1. Individual Language Features

   - Required
   - Implementation Dependent

2. Optimizations

3. Performance Under Load

4. Design Trade Off's

5. Operating System Efficiency

6. Application Profiles

   - Classical
   - Ada in Practice
   - Ideal Ada

In the briefing, Lt. Marmelstein intends to emphasize that it is in the best interest of the DoD system program offices to use the ACEC if and when a government evaluation service is set up. Then DoD offices would not need to have their primes run the ACEC, but could obtain data from an evaluation service before making a decision.

There are quite a few application profile tests in Version 1 of the ACEC. For the next version, a neural network example and data base example will be added. Any suggestions from the user community will also be considered.

Dr. Crawford inquired as to the sizes of the applications and to the amount of code involved. Lt. Marmelstein stated that the Ada data base is over ten thousand lines. Many applications are on the order of 500 to 2000 lines before inserting the timing. Some of them will increase when timing is inserted.

Mr. McKee asked how bit manipulation critical facility is tested since it is not provided by the language definition. Lt. Marmelstein replied that pack array can be used.

In summarizing the capability of the MEDIAN tool, Lt. Marmelstein referred to Lt. Pitarys' earlier remark about the system factors and group factors. These are problem factors indicating a relative hardness of problem for each system rather than for the systems tested.

The system performance is composed of system factors and residual values that are used. This tells the relative optimization that a compiler performed on each test problem.

Lt. Marmelstein showed a sample of the MEDIAN data; the problem factors on the column and systems; factors on the column below and residuals are what is inbetween in the matrix. Runtime and compiler time indicates the type of errors occurring which prevent the execution.

A sample histogram of the residuals using MEDIAN was displayed. One test ran with a residual .08. This will show if the test has been optimized away, which happens with some tests. There are some optimization tests which can be prepared where one test handout optimized the version and another one leads itself throughout the optimization; you can then compare the two.

The summary of the residual data shows the percentage of tests which ran. This is important because a system or compiler may have a very good system factor but the percentages of tests might be low. Basically, it is possible to look very good with a system factor when, in fact, few tests have run. This chart will be included as part of the report.

The ACEC was sent to the Data Analysis Center for Software. If interested in getting a copy, the number is listed in the ACEC Questions and Answers (Q&A) along with the address.

Mr. Szymanski asked how much time would be allotted in the briefing to defining the charts. Lt. Marmelstein indicated that everyone is familiar with the information, but he would like input as to its comprehensiveness. Mr. Szymanski stated that although some details have been seen several times, unless they are seen several times more they do not immediately attract attention. Lt. Marmelstein stated that during the briefing there will be more detail.

Mr. Szymanski stated there are still details on the charts that have not been addressed; they should be detailed or have some verbiage attached. Not knowing the level of expertise of the audience, as much detail as possible should be used. But Lt. Marmelstein indicated that if some information were given in detail the discussion would become mechanical.

Mr. Szymanski then suggested having copies of ACEC Questions and Answers. He also stated that in the short term funding has been assured for making the improvements.

Mr. Leavitt stated that the following questions arose at a briefing of the ACEC at Princeton:

- How long does it take to run.

- How long does it take to compile a half million lines of code.

- What kind of documents are available.

Mr. Weiderman asked if when distributing the ACEC results from five different systems should those systems be identified or called A, B, C, D, and E. Lt. Marmelstein chose not to identify them.

Mr. Leavitt found that a complete set of data from one system and its results were not in the distribution package. Mr. Szymanski replied that the software test report has been deleted from the set of documents originally listed from the Q&A and additional documentation made available through DACS.

Lt. Marmelstein said that it is a political issue as to whether or not the government can legally distribute the results as far as actually naming the compilers showing the tests. Mr. Szymanski interjected that the issue is being addressed.

Mr. Weiderman inquired whether this was not giving partial data for each of the five systems. Mr. Leavitt said that was correct. A couple sets of sample data was available to the team: (1) the results of the average of all the compiler systems on the test problems; and (2) the average results of VAX targeted systems. But the sample does not contain a data set that reflects, for some unnamed implementation x, its results.

Mr. Szymanski stated he saw it as a point of reference for further understanding of the ACEC. Lt. Marmelstein said when having one system and using MEDIAN it is not to be used as a pure data point in determining whether or not the compiler not named is useful or not.

Mr. Mills wondered if any tests or plans for tests to measure performance of compilers with distributed runtime systems are available, and what was its capability of working with distributed type targets. Lt. Marmelstein said there were no tests planned. The reason being that right now there are not any commercial compilers being sold that (1) will allow that, and (2) the way that test problems have to be set up would be extremely dependent on the implementation and not addressed in the Language Reference Manual (LRM). Even if they did exist, it was not known if a portable version could be done. The problem is two-fold: (1) lack of compilers for implementation; and (2) how to write that as portable.

Mr. Leavitt stated that one compiler claims to have distributed run times, parallel processors, shared memory, and if there is a multiple tasking system, assign task processors. Mr. Leavitt said it had been out either two or three months and that by running the existing tasking test on those there will be some benefit. Lt. Marmelstein said it may be able to take existing tasking tests and modify them to run on the line compiler.

Dr. Crawford suggested that adding titles to slides would aid the audience. Lt. Marmelstein agreed and brought his presentation to a close.

## 2.0 THURSDAY, 8 SEPTEMBER 1988

### 2.1 <u>Ada Based Integrated Control System (ABICS)</u>

John Perdzock
Maurice Pierce, McDonnell Douglas
David Cobb, McDonnell Douglas

Captain Abraham introduced John Perdzock, the program manager of the Ada Based Integrated Control System (ABICS) which completed its flight testing earlier this summer. He then introduced Maurice Pierce and David Cobb of McDonnell Douglas who would speak on the actual software.

Mr. Perdzock stated now that ABICS has just completed a flight test a video would be made of the airplane in flight. The purpose would be to make a tape for release that would cover the whole program. One of the key features of the video would be seeing the F-15 in flight, a Flight Critical Software Demonstration. This was one of the main objectives of the program.

ABICS was a joint program with many participants and multiple sponsors. The Flight Dynamics Laboratory was involved with the weapon fire control, a flight control software type of application. The Naval Air Systems Command entered the program from the aspect of inertial reference systems. The Air Force Flight Test Center was the responsible test organization. McDonnell Douglas was the prime contractor on the effort with subcontracting provided by Litton for the inertial system and Lear-Siegler for some of the Ada computers.

Program objectives included:

- Demonstrate that Ada is suitable for real time applications as validated through flight test.

- Demonstrate the viability of multifunction inertial reference system (in which the former inertial systems on board the F-15 were removed and replaced by one set of the ring laser gyros).

In January 1983, the baseline flight test program with fire control equations programmed in Assembly language moved to a Digital Flight Control System, also programmed in Assembly language. In 1984, accomplishments included an Ada Digital Flight Control System, and, in 1986, Fire Control Laws programmed in Ada. At the present time, ABICS 3 has added the navigation and redundancy management system programmed in Ada.

The schedule for ABICS 3 started in 1986 to develop all the hardware or software and test it in the laboratory. It went to the airplane in December 1987 and was put through ground tests and flightworthiness tests. It started flying in March 1988. Twenty pilots tested the system. It is now in the final phase of aircraft flight for the video. Next is demodification which takes the aircraft back to its original configuration.

Concerning the hardware installed in the airplane, the INAs were the inertial navigation assemblies. There were five processors in those boxes. Only one had Ada processing on the navigation processor. The Rolm Hawk was the main Ada processor in this program.

The digital electronic flight control system had four identical channels of computation. Each channel had the following software programmed into it in Ada:

- Flight control laws for the F-15.

- Modified F-15 flight and fire control laws for the coupled fire control functions.

- Offline built-in test functions.

The Rolm Hawk, which is a 32-bit machine, had the majority of the Ada software in it; fire control laws; flight coupler laws (which tied the weapon delivery to the flight control system); and on-board simulation which was a mechanization that tested the fire control without having to put up a real target. If the software failed the airplane would lose a major portion of its function, its weapon delivery performance. The inertial navigation was also mission critical giving the position, latitude, and longitude of the airplane. It provided the velocity and altitude of the airplane to the fire control solution and fire control equation without which the airplane could not perform the primary mission weapon delivery.

Mr. Perdzock offered the following explanation of bringing the airplane back to the original configuration. When the inertial navigation unit was installed, the standard F-15 inertial measurement unit was removed. In its place, one of the inertial navigation assemblies was installed which did not perform the identical additional functions. Because of that the airplane was limited in what it could interface with. The airplane was changed so that on a day-by-day basis a switch could occur between the standard F-15 inertial measurement units and the new ABICS inertial navigation assembly.

All the Ada software will not be deployed in the F-15 fleet. This is a research and development airplane. All of the systems installed on the airplane are unique, one of a kind systems. In the standard F-15, weapon delivery software resides in the central computer. This computer is not functionally capable of supporting the amount of software it took to do the integrated flight fire control. The standard F-15 flight control system is an analog flight control system although the new F-15E does go to a digital flight control. Mr. Perdzock then turned the floor over to Maurice Pierce.

Mr. Pierce stated that his involvement with ABICS was primarily with the integrated flight and fire control software. His part of the presentation would briefly cover the redundancy management and flight control aspects of the program.

Integrated Fire Flight Control (IFFC), a research project from 1981 to 1983, involved the concept of coupled flight control. It had a computer guiding the airplane for weapon delivery to reduce pilot workloads and increase accuracy of weapon delivery. There were two fire control functions: one was an air to air gunnery scenario in which the aircraft by locking on to another aircraft with radar could track and fire a gunshot on the target aircraft. Another was a bombing scenario. The advantage with this scenario was that an operator could attack with the aircraft in a three to five G turn changing altitude the entire time. It approached the target and released in a turn making it very hard to intercept the attacking aircraft.

ABICS was a follow-on to the integrated flight and fire control program. The normal F-15 had an analog flight control system. Special flight control laws were required for both gunnery and bombing coupler mode. It was necessary to change the analog computer to accommodate the coupled flight control. Therefore, the idea of using digital flight control became a very important aspect of the follow-on to the IFFC.

With a digital flight control system, new flight control laws could be used or the existing flight control laws could be changed with software. Hardware changes were not required; therefore, this was more constructive, efficient, and cheaper.

The ABICS program was an evolutionary process following the integrated flight and fire control program. The digital flight control system for the F-15 was developed and tested in ABICS-I. The original digital flight control system was programmed in the Assembly language. These control laws were programmed in Ada from the same flowcharts and flight tested in 1984.

ABICS-II was to integrate the digital flight control system with the integrated flight and fire control software. This had been previously done in the Assembly language. ABICS-II was to take the flight and fire control laws which were programmed in Assembly for IFFC, program them in Ada from the flowcharts, and then flight test them. ABICS-III added in the IISA boxes.

These provided an integrated inertial sensor assembly that would provide both navigation and flight control sensor output thus eliminating the standard flight control sensors on the F-15. The redundancy management function took the output from the two ring laser gyro boxes to determine which was the best input to the flight control system. It had 15 combinations of outputs to compare and therefore was very redundant and safe.

Redundancy management programmed in Ada involved many matrix transformations. Due to different coordinate systems, it was floating point intensive. The software was programmed in Ada in a floating point format; the original algorithm was programmed in Assembly in a fixed point format. The Ada program was too slow for the 80 hertz frame required for digital flight control. There was not time to reduce the execution time of the software. Litton was involved in using Ada fixed point feature in navigation. However, with the schedule of this program there was no time to do the same type of work with redundancy management which was a more complicated type of software than navigation.

In the ABICS configuration, three units in the plane involved Ada. The IISA units integrated inertial sensor assembly, had two assemblies with five processors each. One processor was devoted to navigation which was programmed in Ada. The digital flight control computer had the standard F-15 flight control laws. The equations which translated pilot input into control surface position were programmed in Ada. There were also two sets of special flight control laws. These were designed to interface with the integrated flight and fire control coupler and were programmed in Ada. The built-in test functions were also programmed in Ada.

The Rolm Hawk/32 computer contained all of the integrated flight and fire control software that involved taking sensor input from the airplane, radar input from target, and inertial sensors position input for ground targets. It took that input and calculated the proper orientation of the plane in order to deliver a weapon on target. If the pilot chose, the fire control computer would be coupled to the flight control computer and the computer would actually steer the airplane to deliver the weapon. An on-board simulation function was available to create a simulated target for the pilot to practice switchology and technique without real targets. All the software required for these functions was programmed in Ada.

For overall control, there was a scheduler written in Ada in the computer. Also, there was a scheduler which controlled the order and time at which different modules, i.e., navigation fire control equations and flight control output, were executed. This was all written in Ada in the Rolm Hawk. The compiler itself provided floating point math and I/O functions, etc.

Concerning the software approach for ABICS, the algorithms were coded in Ada from flowcharts of the algorithms. The productivity of the Assembly language program was evaluated versus the Ada programming. An important feature was the flight control function which is traditionally done in fixed point Assembly. This is a time consuming process. Ada programming was done in floating point. This produced a large savings in program productivity.

It was also decided to evaluate software maintenance of Ada high level programming. Also, evaluation was done on the computers to be used for real-time Ada applications and compilers.

The writing of real-time Ada software was investigated; whether it was portable or if there was some requirement for customizing it to the hardware or system that made it nonportable.

Part of the program involved 16-bit versus 32-bit processors. The issue is adequate throughput and adequate memory for real-time applications. This was a function of the compiler. It was decided to evaluate various compilers and software developing environments. Software environments involve how real-time Ada software is written.

Ada software commenced with writing Ada algorithms, and compiling and testing Ada modules individually. Software already developed, possibly on a main frame system, was used in the actual flight hardware. The first step involved primarily checking for syntax errors and the overall structure, the algorithm design, if variables were placed correctly, and bench testing.

Testing was done using data generated from testing the Assembly or another language version. A lot of testing was initially done on the Ada algorithm by comparing the open loop output in real-time with the previous Assembly versions. Although not a requirement for developing Ada software, this was an economical way of verifying the real-time, checking if the software was performing correctly, and looking for possible bugs in the compiler where a calculation may be wrong even if programmed correctly. This verified in real-time that the algorithm was giving the same output for the same input as previously tested in the Assembly language program.

We also tested the algorithms in real-time by giving certain inputs, observing the output, verifying that the algorithm is performing correctly, and performing acceptance tests. The flight simulation was checked that the hardware and software could be installed in the airplane and would run in real-time.

Pilots flew in the simulator verifying that the algorithms performed correctly. The simulator also provided real-time dynamic input for changing target positions from radar and the changing position of the airplane. This was critical for testing dynamic real-time algorithms for airplanes.

Pre-flight tests involved installing the equipment on the airplane. Functional testing of the equipment on the airplane determined if it was following its programming. Environmental testing involved all the subtle problems with putting the electronic equipment on the airplane. The findings would self-check the algorithms and hardware to see if they were working properly. The flight test was actually to have a pilot fly and verify that the system was performing as planned.

The IISA unit or inertial navigation assembly (INA) is a ring laser gyro based system and provided attitude and velocity data. This unit was accurate when used not only for navigation but for flight control sensors. The other unit is the Control Display Unit (CDU). It interfaced with the hardware. This would be how a pilot entered the latitude and longitude to align a system, displayed the latitude, longitude, and velocities while flying. There was also a lot of capability with the system to inspect memory and debugging of the system. The INA, in addition to being a very accurate sensor, was very redundant. This was accomplished with the redundancy management software.

There was an identical INA in the airplane, oriented differently, which combined the individual accelerometer and gyro outputs from the two units. The digital flight control computer had the ability to chose an input set to use. If the input failed, the computer would use the redundancy management software to choose another input set.

Responding to a question on the frequency at which the accelerometer and gyro data entered the software system, Mr. Pierce stated that there was a special high speed data bus where the data entered the digital flight control computer. It was 1,000 hertz; although the digital flight control computer executed at 80 hertz. The idea was to reduce the transport lag by extracting the data at 1,000 hertz. This was a matter of design intent. It is generally considered that 40 hertz is the very minimum frame used for flight control and 80 hertz is considered optimal. 80 hertz meant a 12.5 millisecond frame. The standard F-15 flight control equation took up about 1/4 of the frame in Ada. For the digital flight control system, a pragma was used to suppress runtime checking.

All the exceptions were handled in the IFFC in the same manner. One of the problems was in the bombing equations. If the pilot went into the bombing mode without a target being designated in the central computer, the radar might send either a zero range or a large scale range for a target. This would possibly either cause a divide by zero or a floating point overflow. The bombing algorithm was constantly updating the bombing equations relative to the target's position.

If a target is not designated it could cause an error that could crash the program. An example of how that was handled was that in a module the numeric errors were intercepted and sent to the executive with an error flag. This gave the pilot a message, and there was then a procedure to be followed. Switchology had to be checked to determine what action to take.

Additionally, several safety precautions were taken inside the computer. If coupled to the flight control system (which it should not be) the flight control system was caused to uncouple so the pilot had full manual control of the airplane. This helped in not giving an erroneous input to the flight control system. Variables were reinitialized for bombing. A wait state was entered while the pilot corrected the toggle switch and then restarted.

With the integrated flight and fire control software, one of the things stressed was making an algorithm machine independent, portable, and safe. Advantage was taken of the built-in features of Ada. In doing things certain ways, a major item was the hardware and compilers available. When starting the original digital flight control system, the compiler was the Irvine compiler. It was not a validated version of the compiler and there were many problems. In addition, the floating point algorithm had to be written. Litton did the software in fixed point working closely with the compiler manufacturer to improve that. There was a major problem during runtime checking with the compiler and the system on which it runs.

The IISA units for the navigation function had both Ada and Assembly language. The digital flight control computer, a microprocessor based computer, had a Z-8002 microprocessor with four channels. Two were devoted to the pitch and two were devoted to the roll and yaw control of the airplane.

ABICS-II was to take the digital flight control system that already existed in Ada and add special flight control laws. This was for the integrated flight and fire control which had also been programmed in Ada with the fire control algorithm and flight control coupler algorithm. This caused a problem in the ABICS-II. The same nonvalidated version of the Irvine compiler was used and the Avionics Integration Computer (AIC) also had Z-8002 microprocessor.

Also, one of the problems was the 16-bit architecture of the AIC. The Z-8002 did not have the throughput required for the algorithm. Implemented in memory, the Ada modules had to be broken out on different pages of memory in order to get them working with the microprocessor. This meant a lot of hand tailoring of the compiler output. It had a software floating point function as opposed to a hardware floating point function. This was another factor that reduced the throughput of that computer. As stated previously, the compiler was not validated. Some features were not available in Ada and others were disqualified because of the throughput.

A major problem was the Ada compiler did not produce an executable object module. It produced an Assembly language program which had to be assembled and linked with the rest of the software. Many things had to be done by hand, putting in origin statements, hooks to part of memory, and hooks to subroutine and other algorithms. This was a big factor in reducing productivity in ABICS-II. There was also human error and some problems with the reliability of the hardware that was used for the integrated flight and fire control.

In flight testing the integrated flight and fire control software, there was concern over the throughput of the computer. An obvious step was to go to a 32-bit architecture which eliminated the problems with memory. The addition of a 32-bit architecture added theoretically up to four megabytes of memory. The computer actually had two megabytes which was more than adequate for the software used. Giving a big savings as far as productivity of the program, the Rolm Hawk computer had a high level operating system and a validated compiler which was designed to work with it. Therefore, the compiler produced runnable object modules that execute under the Rolm Hawk operating system. Another point was that ABICS-II had unreliable hardware and the Rolm Hawk computer was a very reliable computer.

One of the ways ABICS-III evaluated choosing a compiler and hardware was through a benchmark which indicated the speed of a computer in Ada. No acceptable benchmarks were found. Instead three algorithm modules were taken from the integrated flight and fire control software already written in Ada which represented different types of applications. There was a stability axis transformation algorithm module involving a lot of assignments and also receiving data from memory and storing it back into memory. The display module involved checking what mode was in, what orientation the airplane was in, etc., and many statement type applications. The Kalman filter for the gunnery equations involved many floating point operations. The Rolm Hawk was tested with the sample algorithm running in a simulated frame just as on the AIC. There was a significant reduction in the time required to execute those modules.

Another step was to take the AIC computer used in ABICS-II and without changing any other hardware in the box replace a CPU card with a CPU card from the Z-8000 microprocessor, 32-bit microbyte microprocessor, which ran at 15 megabytes. The same software and compiler could be used on the Z-80000 as on the Z-8002. It was faster than the 16-bit architecture, but not as fast as the Rolm Hawk. The Rolm Hawk had a hardware float point mechanization, but with the Z-80000 the previous software float point mechanization was still used.

It was decided to go with the Rolm Hawk computer, but others were evaluated. This decision was based on the fact that the Rolm Hawk used a Data General Eclipse computer as a work station. The design concept behind this was that the Rolm Hawk had the same hardware architecture as the Eclipse; therefore, programs could be developed on this main frame type computer and the same software executed on this computer could also be executed in the Rolm Hawk.

The Rolm Hawk computer was a 32-bit MIL-SPEC computer. The statistics verified a throughput of 3.5 million instructions per second (MIPS). But MIPS were found not to be a very informative measure of a computer's abilities. With the test algorithm in the Rolm Hawk, it was determined that it would be more than adequate for the application. In the IFFC software, any of the modules that were under .2% were lumped in with the scheduler which called those additional modules. The important point was that the initial tests showed that the Rolm Hawk had lots of throughput and should be more than adequate for the application.

There was a problem with I/O found when running the software in real-time in the laboratory within an integrated environment. Although not a fundamental problem, the manner in which the compiler and the hardware set up to do I/O on a MIL-STD 1553 bus created a problem. Although I/O modules we. not the largest modules in the IFFC application, much time was spent doing them in the Rolm Hawk. As an example, it was all done primarily in the Assembly language in ABICS-II, and executes in approximately five milliseconds. In ABICS-III doing the I/O would take about 50% of the time which translated into around 18 milliseconds. The rest of the software executed in less time than the I/O portion. The Ada IFFC software on the ABICS-III in the Rolm Hawk executed twice as fast as the Assembly language version of the IFFC on the DGE 7180 flight control computer which was the original IFFC computer. This was a 16-bit computer.

The results of the ABICS program include:

- The standard flight control sensors were replaced by IISA sensors.

- Navigation software in Ada was coded and tested.

- Integrated flight and fire control software in Ada was coded and tested.

- One of the lessons learned especially from ABICS-III was the type of environment needed for development and implementation of real-time Ada software. For this particular program Rolm Hawk seemed to be the answer.

Concerning productivity, the programming in Ada was much more productive than programming in Assembly language. Another major factor was that taking advantage of the floating point with Ada as opposed to the fixed point Assembly language program increased productivity. Another increase was the development of a completely portable integrated flight and fire control algorithm in Ada.

Of the applications for future programs, one of the issues to be addressed in the final report is the 16 versus 32 bit processor question, the problems and the advantages. Mr. Pierce feels that the 32-bit processors are the direction to go.

The In-House Research and Development (IR&D) program investigation of other types of processor architecture and transputers would be continued. It would compare them with the same algorithm and with the Rolm Hawk and other 32-bit processors.

The flight test has shown that Ada was suitable for real-time applications with the right environment. There are now integrated flight and fire control navigation and flight control programs developed and available.

It was identified that the 32-bit processors were indicated for real-time Ada applications. Hardware floating point mechanization was essential because with Ada the floating point would be used and a bottleneck created if it was not fast.

Mr. Pierce emphasized that Ada does improve productivity, and that it does work in real-time with the right compiler and processor. Guidelines offered were to understand the applications, test the compiler and hardware together, and to do an example problem understanding its relationship to the Operational Flight Program (OFP), and, also, to always overdesign.

Mr. Pierce stated there is a need for real-time Ada users to work with hardware and software vendors in defining the proper environment. There is a need for hardware that is more than adequate. To determine this, there must be some criteria for judging the hardware.

Mr. Pierce concluded the presentation with a question and answer session.

## 3.0 FRIDAY, 9 SEPTEMBER 1988

### 3.1 Closing Session

#### 3.1.1 General Comments and Announcements

Mr. Szymanski opened Friday's session thanking the various presenters and those team members who had labored in obtaining them. Mr. Szymanski commented favorably on the ABICS briefing. He asked if there was continued interest in issuing another invitation for the December meeting; interest was expressed in viewing their final report.

Mr. Szymanski stated that he will keep the team posted on the AJPO move from DoD to Systems Command and on any legal activity. The team will be notified of any progress concerning the formal request sent to the contracting office.

Mr. Szymanski expressed concern that all events taking place concerning E&V activities are not being reported. A list will be created of public activities; meetings that were attended, briefings that were given, etc.

Mr. Szymanski then gave a brief update on the activities of John Stanton since leaving the team. Mr. Stanton is now with GEMMA Corporation and has requested a return to the team as their representative.

Gary McKee gave an impromptu briefing on the Ada Europe trip which occurred on June 6-9 in Munich, Germany. The purpose was to meet with the Ada Europe Environments Working Group. A presentation was given covering E&V activities and was considered to be very productive. Topics covered were the ACEC, CIVC, the Reference System, CAIS, and CAIS-A. The ACEC was a major interest of the Europeans.

The presenters from Germany gave a briefing on the Portable Common Tool Environment (PCTE) initiative which they call the German Portable Common Tool Environment Initiative (GPI). In connection with this effort, it was noted that standardization is approached differently in Europe. They start with the product and then write the standard describing it. Consequently, as soon as the standard is approved, they have a working version.

Initially the PCTE supported the C language. Now there is an on-going effort to support an Ada language interface. Also, it was reported that there is a movement towards a PCTE Plus that is similar in power and capabilities to the CAIS-A.

The remainder of the conference covered Ada in industry. A presentation was given regarding the use of Ada in American industry. Companies mentioned included General Electric and Boeing.

Other highlights of the conference included a presentation from NASA on their research of support software development, Software Support Environment (SSE), for the space station. They are using Ada in this effort. The requirements are interesting as they concern an environment that is to last 20 to 30 years supporting ground and flight software and must allow for the inevitable obsolescence in tools and methodologies. Another unique item concerning this environment is its heterogeneity. The mainstay of the space station activity is a standardized communications bus, a coordinated endeavor allowing any kind of hardware and software to fit.

Fred Long presented a paper at the conference covering an experiment done in Wales which implemented both the CAIS and the PCTE, comparing them by constructing several tools. The results were that the CAIS was harder to implement and was given all the handicaps possible, such as in being built in an active living environment, and still came out well.

The Europeans are using Ada in a variety of ways: financial accounting, simulations, air traffic control, and military systems. At this time, the Ada Europe effort is undergoing a financial and political reorganization.

Other presentations included a seminar on Ada Reusability with a panel of eight speakers.

Livia Ravine, who has been active in the KIT effort, gave a presentation covering an implementation of an Ada binding to the PCTE. An interesting point was that there is a validated Ada compiler running on top of this implementation. Due to its funding, in terms of its state of maturity the PCTE effort is advanced over the CAIS activity.

Mr. Szymanski stated that in consideration of this if the CAIS implementation should fail with PCTE and others CAIS would have a very hard time being sold not only in Europe but in the U.S. as well. The effort right now is to develop a validation suite for CAIS implementations.

One of the major weaknesses of the PCTE is that there is no serious development of a validation mechanism. Also, there is not a good set of access control mechanisms within PCTE. There are also some problems with the input and output; so, there are several pros and cons when looking at the two different systems.

Mr. Szymanski stated that he talked with the new chief of the Ada Europe Environments Group. He found that there is an interest in reestablishing the ties made in Edinburough two years ago. Mr. Szymanski was informed that a letter would be sent to the head of AJPO requesting the inauguration of formal ties due to the fact that all the products are scheduled for release and would be made available to them anyway. He stated that he would keep members of the E&V Team informed via the Net.

Further comments on the Ada Europe conference were given by Dr. Tim Lindquist, Peter Clark, Nelson Weiderman, Dr. Bard Crawford, and Captain Becky Abraham.

The team then turned their attention to the working group status reports.

### 3.1.2 Working Group Status Reports

#### 3.1.2.1 CAIS Implementation Validation Capabilities Working Group Status Report

The CIVCWG members attending included: Gary McKee (chair), John Camp, Lloyd Stiles, Dr. Tim Lindquist, Jeff Facemire, and John McBride.

There were no deliverables due.

Ray reviewed the CIVCWG contract status in a roundtable discussion.

Accomplishments

Items discussed during this session were:

- The Ministry of Defense (MOD) test harness which is one of the candidate test finder vehicles for controlling all the tests that SofTech is producing.

- The independent verification and validation (IV&V) of the CIVC, what is involved and how to go about doing it. The decision rests with Ray.

- The use of Hypertext as a vehicle for exploring and examining documents in the taxonomy on the framework that Jeff Facemire is developing, and delivery vehicles possibly including optical disks. This will be discussed further in the December meeting.

Key Issues

- The decision to go with the test administrator design and capabilities. During the next three months, Jeff will be putting the capabilities out on the Net, and the CIVCWG will be reviewing and commenting on them.

- The status of TRW with CAIS. This is the CAIS implementation of speed delivered to NATO as part of the NATO agreement. There are two CAIS ODs, one at Arizona State University and TRW's. Jeff is actually examining both of these as a first sanity check on the tests that are being developed.

Project Work

- There is to be a CIVC technical interchange meeting in November in Houston, TX.

- There will be a major document review.

- Some work will be done on the test administrator design.

- There are no deliverables planned for next quarter.

Action Items

- Group. Review of the Net activity from Jeff on the test administrator design and capabilities.

- Tim, Gary. CIVC document review, technical managemeni.

- SofTech. Examine the use of an optical disk for document delivery.

## 3.1.2.2 Requirements Working Group (REQWG) Status Report.

Those attending this working group session included: Major Pat Lawlis (chair), Captain Becky Abraham, Mike Burlakoff, Peter Clark, Bard Crawford, Marlene Hazle, Alan Impicciche, Tom Leavitt, Ronnie Martin, Nelson Weiderman, and Barbara Rhoads (recorder).

There are no deliverables this quarter.

### Accomplishments

- Presentations given on the Reference System at conferences.
- A new section on E&V needs has been added to the Reference Manual and the Guidebook. This section will also be added to the Tools and Aids document.
- Discussion on updating the PR activities. In specific, looking at updating the slide presentation that TASC has been working on to reflect the delivery of three products, as opposed to just one.

### Key Issues

- The dissemination of E&V information is of particular concern now that products have been delivered.
- There is a continuing concern that new technology being developed by the team under contract is developed according to requirements.
- The future direction for the REQWG.

Ray asked if the ACEC paid attention to REQWG influence. Pat said yes and no. The requirements that were established for the ACEC held within those established in the REQWG, but they were generally all encompassing.

### Projected Work

- The continuing refinement of materials for E&V information dissemination.
- The complete Tools and Aids Document Version 2.0 will be put on the Net for team comment and then will be given to Ray.
- Discussion on capturing lessons-learned information from the initial release of the ACEC.

### Presentations

Presentations planned at the beginning of next year include:

| | | |
|---|---|---|
| SIGAda | | |
| NSIA | Feb/Mar | Ronnie |
| CECOM | Mar | |
| NAECON | May | |

Action Items

Action items carried over from the last session include:

- Jerry, Sandi. Expanding the distributed APSEs/cross-development attributes to a checklist for use in the Guidebook.
- Sandi. The life cycle support from the whole APSE view.
- Nelson. Finding out if the Europeans are working on or will be working on whole APSE issues.

New action items include:

- Pat. Make changes to the Tools and Aids Document.
- Bard. Put the recommended information for the bulletin board distribution on the Net.
- Mike. Put a list of the possible target conferences on the Net.
- Becky. Research Ada news groups.
- ACECWG. Capture lessons-learned information from the initial ACEC release.
- Marlene. Put a summary of the coverage of the proposed symbolic debugging evaluation on the Net.
- Peter. Try to get information for the Target Code Generation Analysis checklist.
- Bard. Put the new model project notes on the Net.
- Pat. Put the September REQWG's status report on the Net.

## 3.1.2.3 Ada Compiler Evaluation Capability Working Group (ACECWG) Status Report

Those members attending this session of the ACECWG included: Nelson Weiderman (chair), Mike Burlakoff, Marlene Hazle, Tom Leavitt, Lt. Robert Marmelstein, and Ray Szymanski.

There were no deliverables due this quarter.

Accomplishments

- Reviewed the kickoff meeting of Phase III.
- Discussed four major areas:
    - Diagnostic messages
    - Single systems report
    - Library system robustness
    - Symbolic debugger evaluation.

- Reviewed ACEC Questions and Answers.

- Created a list of major issues for the evaluation service as part of the validation.

Key Issues

- Phase III needs more structure and test objectives prior to the Preliminary Design Review (PDR).

- The role of the team and the public for evaluation service needs to be defined.

Projected Work/Action Items

- Nelson. Put the evaluation service issues on the Net.

- Bob. Get the team's input on test objectives before the PDR.

- Ray. Keep the Team as informed as possible on the evaluation service developments.

- There are no deliverables due next quarter.

Planned Presentations

- A more complete presentation of Phase III test plans.

- A presentation on feedback from the ACEC user community.

There was no other significant information.

### 3.1.2.4 E&V Technology Classification Working Group (CLASSWG) Status Report

The members attending this session of CLASSWG included: Ronnie Martin (chair), Capt. Rebecca Abraham, Dr. Bard Crawford, Maj. Patricia Lawlis, and Peter Clark. Honorary members attending were: Gary McKee, Greg Gicca, and Jeff Facemire.

There were no deliverables this quarter.

Accomplishments

- Reviewed mappings of the tools to functions in Chapter 5 of the Reference Manual:

    - Test Systems
    - Ada Design Support Tools

- Completed additional synopses for Chapter 4 of the Guidebook:

    - CAIS (DOD-STD-1838)
    - WIS Compiler Evaluation Guidelines

- Refined checklists in the Guidebooks for the following:
  - Data Base Management Capabilities.
  - File Management Capabilities.
  - Program Management Capabilities. There will be a slight restructuring of the Reference Manual as opposed to providing a checklist.
  - Whole APSE Usability Assessment. An independent verification of the previous work. Due to a few additions, there will be an update to the APSE characterization questionnaire, but no new checklist.
  - Completed Chapter 8 of the Guidebook:
    - Title and Introductory Section
    - CIVC Description
  - Provided references for Test Systems Capabilities checklist.
- Reviewed the Reference Manual in preparation for near term delivery. This was done for consistency with the Guidebook. Other changes to the Reference Manual and the Guidebook was to bring them up to date with DOD-STD-2167A. In Chapter 4 of the Reference Manual, Life Cycle Phases was changed to Life Cycle Activities.
- Created a guidebook subject assessor matrix.

Key Issues

- The disappearance of team members who have outstanding action items.

Projected Work/Action Items

- Ronnie. Review of Chapter 4 of the Reference Manual (Life Cycle Activities) for treatment of the Testing-Related Products and Functions.
- Gary. Create/refine the tools and mappings to functions in Chapter 5 of the Reference Manual for treatment of the CAIS implementations.
- CLASSWG. Create/refine tools and mappings to function in Chapter 5 of the Reference Manual for treatment of the Requirements/Design Support System.
- Bard. Mail design slides from a seminar he attended to CLASSWG.
- Bard. Review the SEI framework for applicability when available.

- Ronnie. Review categorization used in SERC projects addressing design translation.

- Peter. Review the whole APSE Assessment Issues work to date for possible further enhancements.

- Ronnie. Locate the question list developed by Rick Fleming and review for enhancement and inclusion in the Guidebook.

- CLASSWG. Review released versions 1.1 of the Reference Manual and Guidebook for enhancements for the next release.

There are no deliverables due next quarter and no presentations planned.

There was no other significant information.

### 3.1.2.5 Standards Evaluation and Validation Working Group (SEVWG) Status Report

Those members in attendance included: Dr. Tim Lindquist (chair), Gary McKee, John Camp, John McBride, Jeff Facemire, and Lloyd Stiles.

Accomplishments

- Guidebook entry.

- Interchange meeting. Ada Europe Environments Working Group attended.

- Progress towards the Issues and Strategies for Review A.

Key Issues

The issues raised in May are still valid. Progress has been made toward resolving the following:

- Upgrading the validation tests to 1838A. This issue was resolved by being combined with the second issue.

- Test Selection Criteria. In May, five or six different test selection criteria were presented. These criteria are based on the following:

  - Upgradability of the test to 1838A.

  - Interfaces in the CAIS most frequently used by tools.

  - Interfaces having the largest impact on transportability.

  The majority of time was spent on the process that SofTech is currently going through in terms of selecting tests, and trying to establish how SEVWG could affect that mechanism so to incorporate some of the other selection criteria.

- Test Effectiveness Measures, Cost Estimation, and the Impact of New Mechanism in "A". It has been decided to include in the Issues and Strategies document an equation in which to plug the appropriate numbers to come up with an estimation of costs for creating a validation mechanism for Revision A.

Projected Work

- Review and comment on the KIT Validation policy statement.

Deliverables

- The Issues and Strategies document for Revision A in December for team review.

Action Items

- Jack. Draft Validation Policy.
- Tim. Send a hard copy of the current Issues and Strategies document to the SEVWG.
- Jack. Comment files for review process.
- Jeff. Provide a list of factors used in generating test scenarios for 1838.

### 3.1.3 Closing Remarks

Ray stated that the March meeting is tentatively set for Phoenix, Arizona. It will be held the first week of March.

Lloyd Stiles reminded everyone that he needs visit requests for the December San Diego meeting.

Ronnie Martin thanked Captain Abraham on behalf of the team for her hospitality.

Ray adjourned the September meeting of the E&V Team.

LIST OF ATTENDEES

Abraham, Capt. Rebecca
AFWAL/FIG
WPAFB, OH 45433-6543

Brown, Freda
Oneida Resources, Inc.
3578 Kettering Blvd.
Suite 300
Dayton, OH 45439

Burlakoff, Mike
Southwest Missouri State University
Computer Science Department
Springfield, MO 65804

Camp, John
AFWAL/AAAF-3
WPAFB, OH 45433-6543

Clark, Peter
TASC
55 Walkers Brook Drive
Reading, MA 01867

Crawford, Dr. Bard
TASC
55 Walkers Brook Drive
Reading, MA 01867

Facemire, Jeff
SofTech, Inc.
1300 Hercules Drive
Suite 105
Houston, TX 77058

Hazle, Marlene
MITRE Corporation
Burlington Road
Bedford, MA 01730

Impicciche, Alan L.
Naval Avionics Center
NAC Code 826
6000 E. 21st Street
Indianapolis, IN 46219

Kirkbride, Kathy
Oneida Resources, Inc.
3578 Kettering Blvd.
Suite 300
Dayton, OH 45439

Lawlis, Maj. Patricia
AFIT/ASU
3318 E. Dry Creek Road
Phoenix, AZ 85044

Leavitt, Tom
Boeing Military Airplanes
P.O. Box 7730, MSK80-13
Wichita, KS 67277-7730

Lindquist, Dr. Tim
Computer Science Department
Arizona State University
Tempe, AZ 85287-5406

McBride, John
SofTech, Inc.
1300 Hercules Drive
Suite 105
Houston, TX 77058

McKee, Gary
GARICAR
P.O. Box 3009
Littleton, CO 80121

Marmelstein, Lt. Robert
AFWAL/AAAF-3
WPAFB, OH 45433-6543

LIST OF ATTENDEES cont.

Martin, Ronnie
Software Engineering Research
Center
Department of Computer Science
Purdue University
West Lafayette, IN 47907-2004

Mills, Mike
ASD-AFALC/AXTS
WPAFB, OH 45433

Rhoads, Barbara
Oneida Resources, Inc.
3578 Kettering Blvd.
Suite 300
Dayton, OH 45439

Stiles, Lloyd
FCDSSA, San Diego
200 Catalina Blvd.
San Diego, CA 92147

Szymanski, Raymond
AFWAL/AAAF-3
WPAFB, OH 45433

Weiderman, Nelson
Software Engineering Institute
Carnegie-Mellon University
Pittsburg, PA 15213

Wills, Betty
CCSO/XPTB
Tinker AFB, OK 73145