UNCLASSIFIED

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

## REPORT DOCUMENTATION PAGE

READ INSTRUCTIONS
BEFORE COMPLETEING FORM

1. REPORT NUMBER

2. GOVT ACCESSION NO.

3. RECIPIENT'S CATALOG NUMBER

4. TITLE *(and Subtitle)*
E & V Guidebook, Version 1.1, 1988

5. TYPE OF REPORT & PERIOD COVERE
1988

6. PERFORMING ORG. REPORT NUMBER

AD-A202 809

8. CONTRACT OR GRANT NUMBER*(s)*

9. PERFORMING ORGANIZATION AND ADDRESS
Ada Joint Program Office
3D139 (1211 S. Fern, C-107)
The Pentagon, Washington DC 20301-3081

10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS

11. CONTROLLING OFFICE NAME AND ADDRESS
Ada Joint Program Office
United States Department of Defense
Washington, DC 20301-3081

12. REPORT DATE
1988

13. NUMBER OF PAGE
75

14. MONITORING AGENCY NAME & ADDRESS*(If different from Controlling Office)*
AJPO

15. SECURITY CLASS *(of this report)*
UNCLASSIFIED

15a. DECLASSIFICATION/DOWNGRADING SCHEDULE
N/A

16. DISTRIBUTION STATEMENT *(of this Report)*
Approved for public release; distribution unlimited.

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

UNCLASSIFIED

18. SUPPLEMENTARY NOTES

19. KEYWORDS *(Continue on reverse side if necessary and identify by block number)*
Ada Programming Language; APSE Evaluation & Validation Task, APSEs

Evaluation and Validation

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

The purpose of the E&V Guidebook is to provide information that will help users to assess APSEs and APSE components by: assisting in the selection E&V procedures, the interpretation of results, an integration of analyses and results; describing E&V procedures and techniques developed by the E& V task, and assisting in the location E & V procedures and techniques developed outside the E&V task. [illegible] Ada [illegible] (Ada Programming Support Environment) [illegible] data processing [illegible]

UNCLASSIFIED

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

# E&V GUIDEBOOK

VERSION 1.1
15 August 1988

Approved for public release; distribution is unlimited

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

TASC No. TR-5234-4

# EXECUTIVE SUMMARY

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 this technology,

(3) Encouraging others to develop additional elements, and

(4) Collecting information describing elements which already exist.

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

The purpose of the E&V Guidebook (this document) is to provide information that will help users to assess APSEs and APSE components by:

(1) Assisting in the selection of E&V procedures, the interpretation of results, and integration of analyses and results,

(2) Describing E&V procedures and techniques developed by the E&V Task, and

(3) Assisting in the location of E&V procedures and techniques developed outside the E&V Task.

All E&V procedures and techniques found in the Guidebook are referenced by the indexes contained in the companion document called the E&V Reference Manual.

Chapters 1 through 4 provide a general introduction to the document and other background material. Chapter 5 and later chapters are "formal chapters" built around a standard format and formal grammar. Each of the formal chapters contains all the assessment procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. The assessment procedures are described and in some instances can be applied directly from the information given in the Guidebook. In other cases, the user is directed to a primary reference for more information.

Yearly updates and extensions to this document are planned. Therefore, comments and suggestions are welcome. Please send comments electronically (preferred) to szymansk@ajpo.sei.cmu.edu, or by regular mail to Mr. Raymond Szymanski, AFWAL/AAAF, Wright Patterson AFB, OH 45433-6543.

**Note:**

"E&V Guidebook, Version 1.1" (this document) is best used in conjunction with "E&V Reference Manual, Version 1.1." The table on the following page is included for the benefit of readers who have the older manual, "E&V Reference Manual, Version 1.0," The organization of the Guidebook was significantly altered after issuance of Version 1.0 of the Reference Manual. (There is no Guidebook, Version 1.0.)

## CONVERSION TABLE

| Reference Manual, Version 1.0<br>Guidebook Reference (RM Section Number) | Actual Guidebook, Version 1.1<br>Section Number |
|---|---|
| 6.1 (3.4.1, 6.4.6, 6.4.22, 6.4.31, 7.1.6.7) | 5.2 |
| 6.2 (3.4.1, 6.4.22, 6.4.31, 7.1.6.7) | 5.3 |
| 4.12 (3.4.2) | 4.9 |
| 3. (3.4.4) | 3. |
| 8.1 (6.4.9, 7.1.6.7) | 5.1 |
| 8.2 (6.4.9, 7.2.3.3) | 8.1 |
| 5.2.3 (6.4.20) | deleted |
| 5.3.1 (6.4.20) | deleted |
| 5.1.1 (6.4.21, 7.1.1.1) | 99.1 |
| 5.1.2 (6.4.21, 7.1.6.6) | 6.1 |
| 5.1.3 (6.4.21, 7.1.6.7) | 5.8 |
| 5.1.4 (6.4.21, 7.1.6.13) | 6.2 |
| 5.1.5 (6.4.21, 7.2.1.1) | 99.2 |
| 5.1.6 (6.4.21, 7.2.1.3) | deleted |
| 5.1.7 (6.4.21, 7.2.1.4) | 99.3 |
| 5.1.8 (6.4.21, 7.2.1.7) | 5.9 |
| 5.1.9 (6.4.21, 7.2.1.10) | 99.5 |
| 5.1.10 (6.4.21, 7.2.2.3) | deleted |
| 5.1.11 (6.4.21, 7.2.2.6) | deleted |
| 5.1.12 (6.4.21, 7.2.2.7) | 10.1 |
| 5.1.13 (6.4.21, 7.2.3.2) | deleted |
| *5.1.14(6.4.21, 7.2.3.5) | 6.3, 5.11 |
| *5.1.15(6.4.21, 7.2.3.6) | 99.4, 6.3 |
| *5.1.16(6.4.21, 7.3.2.2) | 99.6, 99.4 |
| *5.1.17(6.4.21, 7.3.2.3) | 6.5, 99.6 |
| *5.1.18(6.4.21, 7.3.2.5) | deleted, 6.5 |
| *5.1.19(6.4.21, 7.3.2.6) | deleted, deleted |
| *5.1.20(6.4.21, 7.3.2.9) | 7.1, deleted |
| *5.1.21 (6.4.21, 7.3.2.10) | 6.4, 7.1 |
| *5.1.22(6.4.21, 7.3.2.13) | 6.6, 6.4 |
| *5.1.23(6.4.21, 7.3.2.14) | 6.7, 6.6 |
| *5.1.24(6.4.21, 7.3.2.15) | 6.8, 6.7 |
| 5.2.1 (6.4.25, 6.4.27) | 14.1 |
| 5.2.2 (6.4.25, 6.4.27) | 14.2 |
| 5.1.25 (7.3.2.17) | 6.8 |
| NONE | 5.4, 5.5, 5.6, 5.7, 5.10, 7.2, 8.2, 9.1, 10.2, 13.1 |

Note: There is an error in some Guidebook references in Version 1.0 of the Reference Manual. For the marked Guidebook references ("*"), there are two Guidebook section numbers listed in the right hand column. Use the section number in the right hand column corresponding to the Reference Manual section number (in parentheses in the left hand column) in which the Guidebook reference is made.

# TABLE OF CONTENTS

## TABLE OF CONTENTS (Continued)

## TABLE OF CONTENTS (Continued)

# LIST OF FIGURES

**Figure** **Page**

# LIST OF TABLES

# 1. INTRODUCTION

## 1.1 PURPOSE OF GUIDEBOOK

This document is a product of the Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Task sponsored by the Ada Joint Program Office. It is one of a pair of companion documents known as the E&V Reference System, consisting of:

- E&V Reference Manual
- E&V Guidebook.

The subject of both documents is the assessment of APSEs and their components. Specific assessment techniques typically fall into one of two categories: evaluation (assessment of performance and quality) and validation (assessment of conformance to a standard).

The purpose of the Guidebook is to provide a collection of information to support a variety of E&V users in the following ways. It should help them:

- Gain an overall understanding of APSE assessment, in particular, the selection of appropriate E&V procedures, the interpretation of test results, and the integration of analyses and results.
- Apply the various E&V procedures and techniques developed under E&V Task sponsorship.
- Find the primary sources for those E&V procedures and techniques not developed by the E&V Task or not fully explained within the Guidebook (due to space or other constraints).

The Reference Manual includes many "pointers" to sections in the Guidebook and other documents which describe E&V techniques in much the same way that a card catalog does in a library. Figure 1.1-1 illustrates the relationship between the documents.

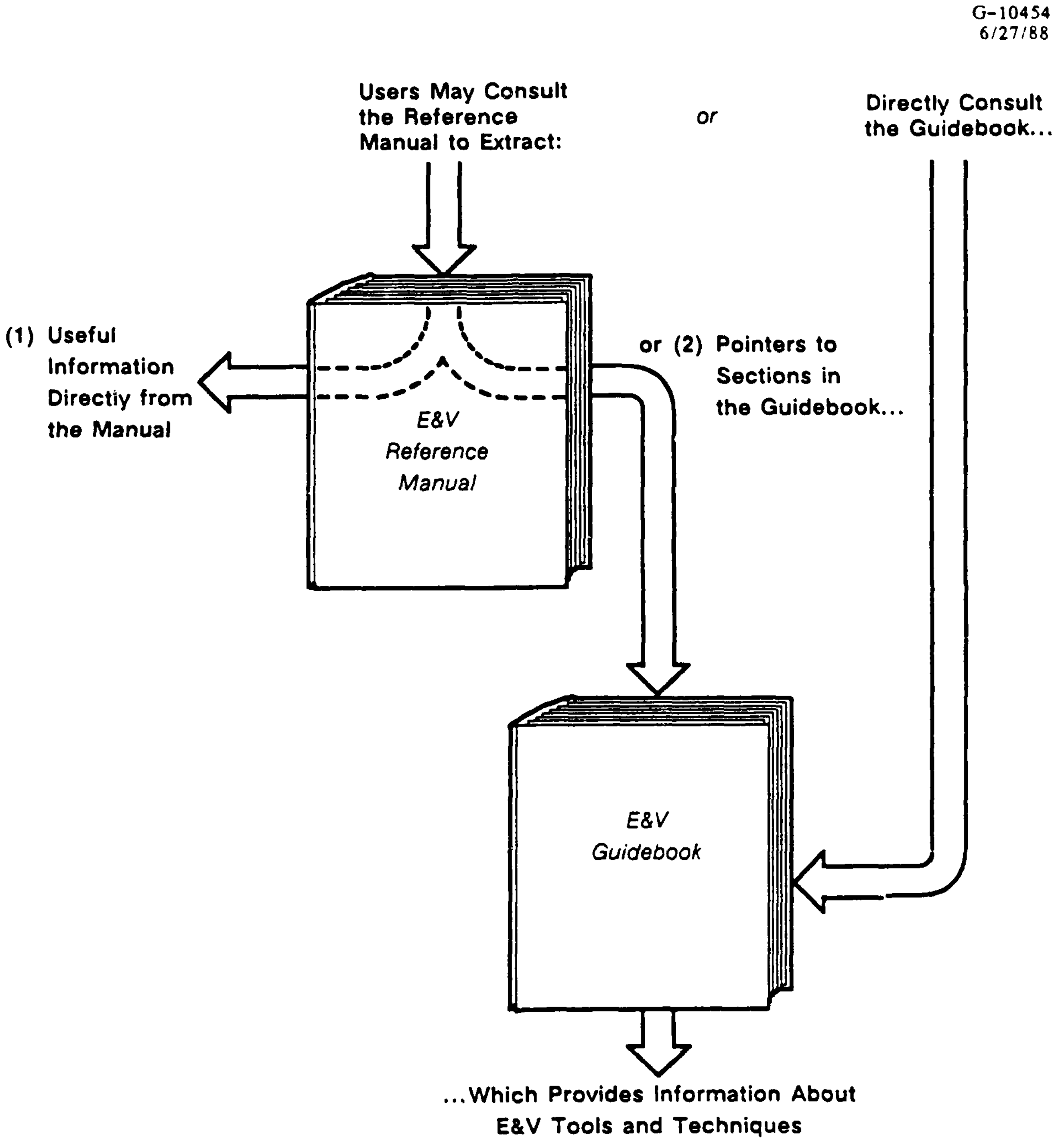

G-10454
6/27/88
Users May Consult the Reference Manual to Extract:
or
Directly Consult the Guidebook...
(1) Useful Information Directiy from the Manual
or (2) Pointers to Sections in the Guidebook...
E&V Reference Manual
E&V Guidebook
...Which Provides Information About E&V Tools and Techniques

Figure 1.1-1 Relationship Between Reference Manual and Guidebook

## 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 methods 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 [@RM 3].* 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 pace 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.

*The format used for references is associated with the "formal grammar" used beginning with Chapter 5. See further explanations in Appendix C.

## 1.3 BACKGROUND

In June 1983 the Ada Joint Program Office (AJPO) proposed the formation of the E&V Task and a tri-service E&V Team, with the Air Force designated as lead service. In October 1983 the Air Force officially accepted responsibility as lead service and designated the Air Force Wright Aeronautical Laboratories (AFWAL) at Wright Patterson Air Force Base as lead organization. In April 1984 an E&V Workshop was held at Airlie, Virginia. The purpose of the workshop was to solicit participation of industry representatives in the E&V Task. Many of the participants in the workshop have chosen to remain involved as Distinguished Reviewers, and additional industry participants have subsequently become involved in E&V Team activities.

The E&V Task publishes an annual public report. The following paragraph is quoted from the 1987 version [@E&V Report 1987] of the report:

> "The Ada community, including government, industry, and academic personnel, needs the capability to assess APSEs (Ada Programming Support Environments) and 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 existing elements. This information will be made available to DoD components, other government agencies, industry, and academia."

The team public reports contain much additional information for the interested reader. See for example, the "DoD APSE Analysis Report" [@E&V Report 1984], the "Requirements for the Evaluation and Validation of Ada Programming Support Environments, Version 2.0" [@E&V Report 1987], and the "Tools and Aids Document, Version 1.0" [@E&V Report 1987], which are synopsized in Chapter 4 [4.3, 4.5, 4.6].

Three competitive contracts have been awarded under the E&V Task. These are:

- Technical Support contract - awarded June 1985
- Ada Compiler Evaluation Capability (ACEC) contract - awarded February 1987
- CAIS Implementation Validation Capability (CIVC) contract - awarded May 1987.

The major purpose of the first of these contracts is to create and update elements of the E&V Reference System, including this document. The purpose of the second and third contracts is to create two specific elements (ACEC and CIVC) of the needed E&V technology.

## 1.4 ORGANIZATION OF THE GUIDEBOOK

*Chapter 2* provides a general description of the structure and use of the Guidebook.

*Chapter 3* provides high-level guidance to users who may need assistance in selecting instances of the technology and integrating the results of its application.

*Chapter 4* provides synopses of other documents or activities that are either too broad in scope to fit within one of the later chapters or are of historical importance to E&V activities.

*Chapter 5 and subsequent chapters* are "formal chapters" that describe or refer to specific instances of E&V technology. Each of the formal chapters contains all of the procedures and techniques associated with a particular group of tools or toolsets to be assessed, such as Compilation System Assessors or Test System Assessors. A standard format based on a "formal grammar" is used in presenting this material. See further explanations in Appendix C.

*Appendices A, B, and C* contain a list of citations, a list of acronyms and abbreviations, and a definition of the formal grammar used in the formal chapters, respectively. *Appendix D* contains the list of vendors and agents of assessment tools who are the primary sources of E&V technology.

# 2. STRUCTURE AND USE OF THE GUIDEBOOK

This chapter provides a brief explanation of the structure and uses of the E&V Guidebook. It is expected that many users have first consulted the E&V Reference Manual (see Fig. 1.1-1) and come to the Guidebook with a specific chapter and section number in hand, prepared to read about a specific instance of E&V technology. A user following this path does not particularly care about the overall structure of the document. Other users, however, may come to the document with a less narrowly-defined objective. An attempt has been made, with such users in mind, to make the Guidebook easy to use as a stand-alone document.

## 2.1 STRUCTURE

The Guidebook structure may be considered as having four major subdivisions, as follows:

- Introductory Material (Chapters 1 and 2)
- General Background Material (Chapters 3 and 4)
- Specific E&V Technology Descriptions (Chapters 5 and beyond)
- Appendices.

The introductory material is used to introduce the document and its structure. The general background material is used to introduce the general subject of APSE assessment. Chapter 3 is an "essay" designed to help users who are faced with the question of how to evaluate an APSE as a whole, or how to compare several APSEs with the objective of selecting one. (Chapter 3 of the E&V Reference Manual, dealing with whole APSE assessment issues, is a "companion essay" that provides complementary background material.) Chapter 4 provides a different kind of background material. It may be considered a "guide to the literature" of APSE assessment. It contains synopses of documents that fall into one of two categories. One category is that of documents that contain *no* specific instances of E&V technology, but contain generally useful background material. The other category is that of documents that contain or discuss multiple instances of E&V technology, which are individually covered in multiple parts of the later, formal chapters. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. Each "synopsis text frame" in Chapter 4 has the following parts:

- Citation: (the primary reference)
- Synopsis: (brief description)
- Methods: (references to specific instances of E&V technology, if any).

The formal chapters (Chapters 5 and beyond), which comprise the main bulk of the Guidebook, describe or summarize specific instances of E&V technology. The chapter subjects and titles were chosen to be meaningful and intuitive to users of the Guidebook. Thus, they focus on the *subject* of assessment (e.g., Compilation System, Test System, Ada Design Support System, etc) rather than the *method* of assessment (e.g., formal validation, subjective evaluation, etc). Within each chapter there are, in general, multiple instances of assessment technology. Some may be examples of evaluation techniques, others may be examples of validations, others may be mixtures of the two. Readers should not infer approval of the E&V Task, because a tool or technique is included in the collection, or disapproval, because a tool or technique is not included. Readers who know of instances of E&V technology not reported here are urged to contact the E&V Task chairman, in the manner described in the Executive Summary. The separate instances within a chapter are simply placed there in *chronological order*, indicating the relative timing of the material's first appearance in the Guidebook. Readers should not infer any judgment as to relative importance based on order. Each chapter thus provides a dynamically growing section of the Guidebook. Old sections will not be thrown awav or replaced by new sections describing newer techniques. Old sections may, however, be updated if a particular vendor or agent has updated material describing a technique, or has improved the technique itself without fundamentally changing the approach. Each "technique text frame" in the formal chapters has the following parts:

- Purpose:
- Primary References:
- Host/OS: (if applicable)
- Vendors/Agents: (if applicable)
- Method:

  Inputs:

  Process:

  Outputs:

The final major subdivision (the appendices) require little explanation here. The formal grammar described in Appendix C need not concern most users. It was employed because of the possibility of a future on-line, electronic version of the Reference System, supported by advanced updating and information retrieval techniques.

## 2.2 EXAMPLE USES

Instances of E&V technology may be found in two ways. A user may consult the Guidebook directly, or may first consult the E&V Reference Manual, as pictured in Figure 1.1-1. A user who comes directly to the Guidebook would typically first look at the Table of Contents. For example, a user interested in evaluating compiler performance would naturally look under Chapter 5 "Compilation System Assessors." The titles of Sections 5.2, "IDA Benchmarks," and 5.3, "Ada Compiler Evaluation Capability (ACEC)," would probably suggest themselves as relevant to this user's needs — as indeed they are.

Alternatively, the user may consult the E&V Reference Manual, which is designed to help find E&V techniques in the same way that the card catalog helps people find books in the library. For example, the Reference Manual contains both a Function Index and an Attribute Index, each of which contains cross references to elements in the other. One element of the Function Index is the function "Compilation," which is cross-referenced to a number of relevant attributes. Under the particular function-attribute pair "Compilation-Processing Effectiveness" are listed a number of Guidebook references.

Among these are the same two Sections, 5.2 and 5.3, of the Guidebook, mentioned in the previous paragraph. The user following this procedure could pick up the Guidebook and go directly to these two sections or "text frames" and find summary information concerning the IDA Benchmarks test suite and the ACEC test suite, respectively.

# 3. INTEGRATION OF APSE ASSESSMENTS

The purpose of this chapter is to provide high-level guidance for the user of the E&V Reference System (Reference Manual and Guidebook) who is interested in evaluating an APSE as a whole, or in comparing several APSEs with the objective of selecting one. While the "formal chapters" (beginning with Chapter 4 of the Reference Manual and Chapter 5 of the Guidebook) provide assistance in locating, defining, and assessing many individual aspects of APSEs, they do not provide an overall approach to weighting and combining the results of such assessments. Section 3.1 briefly discusses some relevant general background material. Section 3.2 discusses some earlier, partial efforts aimed at an integrated approach. Section 3.3 provides some additional guidance leading to a comprehensive, integrated approach.

It is necessary, first, to distinguish the subject of this chapter -- integrated whole-APSE assessment — from the subject of Chapter 13 — specific "Whole-APSE Assessors." The integrated form of whole-APSE assessment (Section 3.3) involves a combining or mixing together of the results of individual assessment steps to arrive at a decision. These individual steps may be oriented toward specific functions or tools, or may be oriented toward a "whole APSE," in relation to specific attributes or the APSE's performance in a specific life-cycle phase or activity. Thus, a whole APSE assessor (Chapter 13) might be used to evaluate the APSE's capability to support a project team during one major activity, such as preliminary design. The results of such an assessment would become one of the weighted factors of an integrating process leading to a major decision.

## 3.1 GENERAL BACKGROUND

Chapter 4 of this Guidebook contains synopses of books, articles, and documents. Some of these have historical value and are also indirectly relevant to the topic of an integrated approach to APSE evaluation because they provide definitions of an APSE or highlight issues that may be important during APSE evaluations. The Stoneman document

[@DoD 1980] defines an APSE as a layered system and includes some discussion of evaluation criteria. The Common APSE Interface Set (CAIS) and CAIS-A definition documents [@DoD 1986, @DoD 1988] describe proposed interface requirements for interfaces that exist between layers of an APSE. The motivation for these interface requirements is to support the transportability of tools and project data bases from one APSE to another. The book "Life Cycle Support in the Ada Environment" by McDermid and Ripken [@McDermid 1984] takes a top-down approach to defining a "coherent APSE," starting with requirements for a coherent life-cycle methodology; see synopsis [4.10]. Several papers in an IEEE Tutorial [@Wasserman 1981] provide relevant observations on desirable characteristics and major issues for Ada support environments; see synopses [4.11, 4.12, 4.13]. A more recent survey paper "Characteristics and Functions of Software Engineering Environments: An Overview" [@Houghton and Wallace 1987] provides a broad discussion of environments and the state of the art; see synopsis [4.15]. Chapter 3 of the E&V Reference Manual [@RM: Whole APSE Issues 3.] presents various ways of viewing an APSE and key whole-APSE attributes.

## 3.2 EARLY EFFORTS AT INTEGRATED APSE ASSESSMENT

The following quotation is from a paper by Henderson and Notkin [@Henderson 1987]:

> "Perhaps the biggest failing of environments research and development to date is the general lack of scientific evaluation of existing environments. Evaluation approaches and actual evaluations are beginning to appear, but relatively little effort has been given to this undeniably fundamental subject."

Some early efforts are mentioned briefly below.

The Software Engineering Institute (SEI) has developed a methodology [@Weiderman 1986] to evaluate certain aspects of APSEs. The methodology centers around the execution of several experiments in the environment(s) to be evaluated. The experiments are designed in a generic fashion and must be tailored or "instantiated" for each specific environment; see synopsis [4.13].

The book "*Selecting an Ada Environment*" [@Lyons 1986], written by the Ada Europe Environment Working Group, provides background discussion about a broad range of topics. In each chapter and section it provides a list of questions to be asked about the environment under consideration; see synopsis [4.9].

It is apparent that industry has devoted resources internally to comparative assessment of commercial APSEs. However, little has yet been published in the open literature describing the techniques employed.

## 3.3 TOWARDS A COMPREHENSIVE APPROACH

The published literature on assessment of software engineering environments does not include descriptions of "decision support" oriented approaches. A decision support system is one that leads a user through a structured framework that includes weighting factors and decision criteria, and supports a final decision process. As applied to APSE assessment this kind of approach would support a final decision, such as, whether a single APSE under consideration is "good enough," or which of several APSEs under consideration is "best."

The following characteristics appear to be appropriate for a decision support system designed for integrated APSE assessment:

- The system should allow the specification of a list of "essential features" that are absolutely required for the contemplated application or family of applications. Ideally, each of these essential features would be subject to a question or test that yields an unambiguous "yes/no" result.
- The system should allow the specification of a second list of attributes and function-attribute pairs that represent desirable features or criteria, which should be involved in an integrated assessment.
- The system should allow for specification of "weights" to be applied to each attribute and function-attribute pair in the second list. The weights will typically be chosen subjectively by the assessment participants.

- The system should include a mechanism to document/identify the method of assessment used for every test/metric to be employed in addressing every essential and desirable feature.
- The system should include a well-defined method of combination, leading to an overall set of pre-decision results. For example, the results may be summarized in two lines as in:

  1) satisfies all essential requirements (listed in Table A)

  2) scores 72 out of possible 100 (based on weights in Table B).

The characteristics outlined above represent a general framework that can be applied very differently by different users. At one extreme is a decision maker with little time or resources, who focuses on a short list of essential features only, and accepts answers supplied by vendors or vendor documentation. At the other extreme is a team of APSE assessors who conduct a comprehensive, detailed set of tests and "model project" experiments and expend multiple person-years of effort in a comparative, hands-on assessment of competing APSEs.

It is also possible that two assessment teams applying equal resources might differ greatly in the manner of their assessments. One might view the APSE as a support system for a particular life-cycle methodology adopted by its organization. Another might view the APSE as a project data base management system. These two teams would be likely to use very different tests, or very different weights where the same tests are used. Neither is necessarily right or wrong. In the final analysis, it is the software developer's responsibility to understand his own application area and the most critical attributes of his development support environment.

# 4. SYNOPSES

The purpose of this chapter is to provide a single place in the Guidebook for synopses of documents (or other resources), which have too broad a scope to fit within one of the subsequent Chapters. In some cases the subject document appears only in this Chapter because it does *not* contain specific instances of E&V technology. For example, the Stoneman document [@DoD 1980] does not deal with evaluation or validation of APSEs, but it has general historical importance to the entire field of Ada environments and has been selected as the first document to be synopsized. In other cases a particular document may contain multiple instances of E&V technology, which are themselves summarized or referenced in multiple parts of the Guidebook. These multiple instances can be thought of as children of a common parent. In order to avoid the redundancy of summarizing the parent document many times, the Chapter 4 synopsis is provided as a common point to which all the children may refer. The formal grammar used to structure the entries in subsequent chapters includes, therefore, a mechanism for referring back to the synopsis contained in Chapter 4. Similarly, after each synopsis there is a provision for forward references to specific techniques (if any) described in later chapters.

Most of the documents synopsized in this chapter are readily available through public sources. A few of them may be difficult or impossible to obtain for some readers; these were included because the synopsis itself was judged to be helpful in filling in a piece of the historical background.

## 4.1 STONEMAN

Citations:

[@DoD 1980] "Requirements for Ada Programming Support Environments — STONEMAN," 1980.

Synopsis:

The Stoneman document defines the APSE as a layered system. The innermost layer is referred to as the Kernel APSE, or KAPSE. The KAPSE is machine-dependent and includes the database functions and other general operating system support functions. The next layer, the Minimal APSE, or MAPSE, consists of the minimal set of tools which can support the development of software. The outermost layer, the APSE, consists of tools and functions that are project dependent. In addition to providing guidance for APSE designers, the Stoneman document provides some evaluation criteria for APSEs.

## 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."

## 4.3 E&V REPORT: DoD APSE ANALYSIS

Citations:

[E&V Report 1984] "DoD APSE Analysis Report, Draft Version 1.0," 31 August 1984, Appendix C of "Evaluation and Validation (E&V) Team Public Report", Air Force Wright Aeronautical Laboratories, November 1984.

Synopsis:

The DoD Ada Programming Support Environment (APSE) Analysis Document was prepared by the APSE Working Group (APSEWG) of the E&V Team. It contains a description and analysis of the Ada programming support environments developed by each of the armed services. The three environments analyzed were the Air Force's Ada Integrated Environment (AIE), the Army's Ada Language System (ALS), and the Navy's Ada Language System/Navy (ALS/N). The design documentation was used to determine the functionality contained in each programming environment. The functions were described in a taxonomy in order to determine the commonality and differences of each system. The taxonomy developed for this purpose was an expanded version of the function part of the taxonomy developed earlier by Houghton [@Houghton 1983]; see synopsis [4.2].

## 4.4 CLASSIFICATION SCHEMA/E&V TAXONOMY CHECKLISTS

Citations:

[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

Synopsis:

The purpose of this document was to set forth a schema, or a framework, to be used in subsequent E&V documents, especially the E&V Reference Manual [RM]. The Function Index of the schema was strongly influenced by earlier documents, such as Houghton's taxonomy [4.2], the DoD APSE Analysis Report [4.3], and the SEE tool features taxonomy [@Kean 1985]. The upper levels of the Function Index of the schema became the initial version of the Function Index of the Reference Manual. The lower levels were found to incorporate a large number of tool functions which could be evaluative in nature. These tool function features have been carried over into the Guidebook as capability assessment checklists. As a group, they are considered the Classification Schema Checklists.

Methods:

| | |
|---|---|
| [Compilation Checklist | 5.8; |
| Program Library Management Checklist | 5.9; |
| Linking/Loading Checklist | 6.2; |
| Import/Export Capabilities Checklist | 6.3; |
| Emulation Capabilities Checklist | 6.4; |
| Debugging Capabilities Checklist | 6.5; |
| Tuning Analysis Capabilities Checklist | 6.7; |
| Real-Time Analysis Capabilities Checklist | 6.8; |
| Configuration Management Capabilities Checklist | 10.1; |
| Text Editing Capabilities Checklist | 99.1; |
| Electronic Mail Capabilities Checklist | 99.3] |

## 4.5 REQUIREMENTS FOR E&V

Citations:

[@E&V Report 1987] "Requirements for Evaluation and Validation of Ada Programming Support Environments, Version 2.0," 4 December 1986, Appendix D 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. Its purpose is to set forth requirements on the E&V Task. It is intended for use by the E&V Team and by the E&V Task contractors in developing technology for the evaluation and validation of APSEs. However, its use in other E&V efforts is encouraged. The document contains three categories of requirements: (1) those on the E&V Team itself, (2) those on the E&V methods and procedures, and (3) those specifying what is to be evaluated or validated. See also the Tools and Aids Document, synopsis [4.6].

## 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.

## 4.7 STARS-SEE OPERATIONAL CONCEPT DOCUMENT

Citations:

[STARS-SEE 1985] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

Synopsis:

The Software Technology for Adaptable, Reliable Systems — Software Engineering Environment (STARS-SEE) Operational Concept Document (OCD) presents requirements from the perspective of the STARS-SEE users. It represents a consensus among the Government agencies responsible for SEE development and support, STARS-SEE implementors, and potential users. Major sections of the document describe the STARS-SEE mission, operational and support environments, and system components and functions. The primary mission centers on the development, support, reuse, management, and control of mission critical software. The STARS-SEE system is defined to consist of the people, computers, software, and procedures needed to perform the mission. Major topics discussed include (1) the types of users and associated software activities, (2) the function of the Integration and Compatibility Framework, (3) the capabilities required by the Information Storage and Retrieval System, (4) the functional capabilities of the SEE, (5) the SEE-user interaction, and (6) the hardware and software characteristics of the computer system. The functional capabilities address project planning and control, requirements specification and analysis, design specification and analysis, software prototyping and modeling, reusability, program generation and unit testing, integration testing, quality assurance, verification and validation, configuration management, software/hardware integration, post deployment software support, project communications, generation of documents, data collection, performance and productivity measurement, help and training for STARS-SEE users, the transition to and tailoring of the STARS-SEE, and knowledge engineering.

## 4.8 GRUND, ET AL.: KEY CHARACTERISTICS OF APSES

Citations:

[Grund 1985] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD — TR-85-144, MTR-9590, July 1985.

Synopsis:

This document is intended to provide basic information about Ada Programming Support Environments for people concerned with the specification or selection of an APSE. Section 1 summarizes the STONEMAN APSE requirements. Section 2 describes desirable characteristics of APSEs in five areas: compilers, run-time environments, databases, configuration management tools, and editors. A short list of questions to ask in each area is included. Section 3 describes four Ada programming support products available or under development in early 1985 in terms of their capabilities in the same five areas.

## 4.9 Ada-EUROPE: SELECTING AN Ada ENVIRONMENT

Citations:

[Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D. Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

Synopsis:

The Ada-Europe Environment Working Group, under the chairmanship of John Nissen, produced a guide which adopts the "point of view of a potential user wishing to select an environment, and provides lists of questions to be asked about the environment under consideration." It generally follows the structure proposed in Stoneman [@DoD 1980]; it "starts from the inside of the onion structure and works outwards." Each of its 19 Chapters follows a standard format. Topics are introduced and discussed, typically using one or two pages of text, and then a list of appropriate questions is provided.

## 4.10 MCDERMID AND RIPKEN: LIFE CYCLE SUPPORT IN THE Ada ENVIRONMENT

Citations:

[McDermid 1984] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

Synopsis:

This book contrasts its own approach to APSE development with that of the Stoneman report [@DoD 1980]. Stoneman takes a bottom-up approach, starting with a kernel and minimal APSE (KAPSE and MAPSE), as a foundation for extensions to more powerful and better integrated environments. McDermid and Ripken follow a top-down approach by defining requirements for a coherent life-cycle methodology. They then describe a particular instance of a coherent methodology, as a combination of existing methods used in various life-cycle phases. This description becomes the basis for a definition of a "coherent APSE" that supports the entire life cycle.

The authors use a seven-phase life cycle and state requirements for each phase in terms of (1) a system *representation* form, (2) a *transformation* method and (3) a *verification* activity. Table 4.8-1 lists the names of the seven phases (each named for its principal output) and the methods selected for each.

TABLE 4.8-1

EXAMPLE COHERENT METHODOLOGY

| PHASE (OUTPUT) | SELECTED METHOD |
|---|---|
| Requirements Expression | CORE |
| System Specification | A-7 Techniques |
| Abstract Functional Specification | A-7 Techniques |
| Module Specification | Ada and ANNA |
| Module Design | Ada and ANNA |
| Module Code | Ada and ANNA |
| Executable System | -- |

The authors are not completely satisfied with all of the methods chosen, and point out shortcomings in each case. They suggest the book be used as "a reference point for further work on APSE design and development." They stress that the coherence of the methods and ease of transition from one phase to the next is an important attribute. They also outline a phased development plan in which a larger scale APSE might be developed in the following three steps: (1) a "Clerical Support APSE," (2) a "V&V and Management Support APSE," and (3) a "Transformation Support APSE."

## 4.11 NOTKIN AND HABERMANN: SOFTWARE DEVELOPMENT ENVIRONMENT ISSUES AS RELATED TO Ada

Citations:

[Notkin 1981] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 107-133.

Synopsis:

This paper addresses software development problems that arise in three areas: programming, system composition, and management. In each area traditional methods and tools are contrasted with a more integrated approach exemplified by an experimental environment named Gandalf.

"Programming issues are those that arise when a single programmer takes a program all the way from its specifications to a working program."

"System composition issues are those that arise when a system (or a version of a system) is built by integrating many programs into one." "The two basic problems in system composition are interface control and version control." Traditional methods use isolated tools "coordinated by memory...or scraps of paper."

"Management issues are those that arise when a group of more than one person develops and maintains a system over a period of time." Three problem categories are addressed: misunderstanding, lack of information, and conflict of interest. Traditionally, these problems have been handled by non-technical means. The problem with the management approach to a management environment is that the solution to human interaction difficulties is treated by the introduction of more human interaction.

Methods:

Although this paper was not written as an example of E&V technology, the following list of environment software requirements (paraphrased from the paper) may be used as a high-level checklist:

- Concurrent multiple users must be supported
- An efficient implementation of Ada must be possible
- Efficient support for data base manipulations is needed
- A good file system is essential
- An extensible command language is needed.

It is also pointed out that the most important hardware requirement is that the software requirements listed above must be supported.

## 4.12 STENNING, ET AL.: THE Ada ENVIRONMENT: A PERSPECTIVE

Citations:

[Stenning 1981] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: a Perspective," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 36-46.

Synopsis:

This paper discusses the objectives and the design of the Ada Programming Support Environment. It is strongly influenced by the United Kingdom Ministry of Defense Ada Support System Study, which was initiated by the MoD in January 1979. According to the paper, the DoD KAPSE/MAPSE/APSE approach is strongly recommended to achieve portability. The APSE should be designed to support a project throughout its life cycle. Furthermore, it should be an open-ended environment. This would allow for the user to extend or modify existing tools. A basic configuration control manager, a complete user interface, and a complete basic tool set are necessary to develop an Ada Environment which will improve program reliability, life-cycle program costs, and promote portability.

## 4.13 WEIDERMAN: EVALUATION OF Ada ENVIRONMENTS

Citations:

[Weiderman 1986] N. Weiderman, "Evaluation of Ada Environments," Software Engineering Institute, SEI-86-MR-10, September 1986.

Synopsis:

In response to the lack of available research about the selection of APSEs, the Software Engineering Institute (SEI) has developed a methodology to evaluate these environments. The methodology centers around the execution of several experiments in the environment to be evaluated. Several experiments have been developed in the following areas: System Management; Configuration Management/Version Control; Design and Code Development; Unit Testing and Debugging. The experiments are evaluated in terms of functionality, performance, user interfaces, and system interfaces. The need for an evaluator to tailor an evaluation technique to a specific environment is addressed by the SEI study. The experiments that have been designed are generic experiments. The evaluator derives, or "instantiates," the environment-specific technique from the generic experiment. In the final phase of the evaluation, the results are analyzed. An advantage of the application of this methodology is that results can be compared from one environment to another. See also a paper describing an application of the SEI's method [@Gray 1987].

Methods:

| | |
|---|---|
| [SEI Unit Testing and Debugging Experiment | 7.2; |
| SEI Design Support Experiment | 9.1; |
| SEI Configuration Management Experiment | 10.2] |

## 4.14 BARSTOW AND SHROBE: OBSERVATIONS ON INTERACTIVE PROGRAMMING ENVIRONMENTS

Citations:

[Barstow 1981] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," in "Tutorial: Software Development Environments," ed. A.I. Wasserman, IEEE, 1981, pp. 285-301.

Synopsis:

This paper reviews key features of LISP-based environments and comments upon lessons learned from these environments and future directions. These environments encourage a "progressive enrichment" style of development rather than developments broken into distinct phases such as specification, implementation, and maintenance. The following set of lessons (described more fully in the paper) are concerned with the programmer's perception of the environment:

- It is important to be able to run an incomplete program.
- The user should be able to view the program from many different natural viewpoints, most of which are "structured" in nature.
- Intercommunication among tools is extremely important.
- The programmer should not be required to know the details of the particular language definition used in the current implementation.
- The environment's interface must be highly tuned to be as natural as possible for the human programmer.

Environment characteristics created with these lessons in mind "lead to the ultimate goal of a programming environment (which is to increase the ability of the programmer to communicate with the computer) by taking advantage of as many naturally occurring structures as possible."

## 4.15 HOUGHTON AND WALLACE: CHARACTERISTICS AND FUNCTIONS OF SOFTWARE ENGINEERING ENVIRONMENTS: AN OVERVIEW

Citations:

[Houghton and Wallace 1987] R.C., Houghton, Jr. and D.R., Wallace, "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Engineering Notes, Vol. 12 Number 1, January l987.

Synopsis:

This paper provides a comprehensive discussion of software engineering environments in general, with no focus on Ada or any specific language. Some major topics discussed are:

- Environment Types and Life Cycle Relationships
- Integration
- Human Factors
- Analysis and Software Quality
- Support for Different Types of Users
- Support for Application
- Hardware Support
- Levels of Support.

In its concluding section, the paper stresses that software engineering environments should be viewed as systems that support broad categories of users and tasks throughout the full life cycle.

## 4.16 CAIS

Citations:

[DoD 1986] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986.

Synopsis:

DoD-STD-1838, hereafter called CAIS, was developed by the KAPSE Interface Team (KIT) and the KAPSE Interface Team for Industry and Academia (KITIA) during the period from 1981 to 1986 as a first evolutionary step towards a full state-of-the-art common APSE interface standard.

The CAIS is designed to promote source-level portability of Ada programs, especially Ada software development tools. The goal of the CAIS is to promote interoperability (of database objects) and transportability (of APSE tools) of Ada software across Department of Defense (DoD) APSEs. The scope of the CAIS includes interfaces to those services, traditionally provided by an operating system, that affect tool transportability. The CAIS contains definitions for a set of Ada package specifications that will provide a standard and transportable set of interfaces to a collection of such underlying services. See also [@DoD 1988] and synopsis [4.17], and the overview paper [@Oberndorf 1988].

## 4.17 CAIS-A

Citations:

[DoD 1988] "Common APSE Interface Set, Revision A," Proposed DoD-STD-1838A, January 1988.

Synopsis:

CAIS-A is a set of Ada package interfaces designed to enhance the transportability of Ada Support Environment Tools. The scope of the CAIS-A includes the functionality affecting transportability that is needed by tools, but not provided by the language. The proposed CAIS-A contains definitions for an entity management system for software engineering tools. The primitive entities defined allow for the manipulation of devices, files and processes. CAIS-A is based on an entity-relationship approach and it allows the user to define entities, in a limited way, by means of a typing mechanism. CAIS-A also includes functionality to support tools requiring transaction processing, a rudimentary triggering mechanism, and explicit control over APSE distribution.

The CAIS-A was developed by SofTech under contract to Naval Ocean Systems Center. CAIS-A is a design enhancement to the existing CAIS (DoD-STD-1838) [@DoD 1986]; see synopsis [4.14], which was developed by the KIT and KITIA (see Appendix B) as a first evolutionary step towards a full, state-of-the-art interface standard. Designers view CAIS-A as the next step in that evolutionary process. See also [@DoD 1986] and synopsis [4.16].

## 4.18 HOGAN AND PRUD'HOMME: DEFINITION OF A PRODUCTION QUALITY COMPILER

Citations:

[Hogan 1985] M.O. Hogan, and S.M. Prud'homme, "Definition of a Production Quality Compiler," Aerospace Corporation, Technical Report, July 1985.

Synopsis:

The study that led to this report was sponsored by the Space Division of the Air Force Systems Command. The report "is organized as a set of prototype requirements, along with guidance on how to tailor the requirement for specific application areas. In this form it can be used either as a tool to help determine whether a particular compiler is of production quality or as a guide for preparing requirements for compilers to be used in the development and maintenance of software for mission critical computer resources."

Chapters 2 through 6 address interface requirements: user interfaces, machine interfaces, runtime system interfaces, compiler related components interfaces, and Ada language interfaces, respectively. Chapter 7 addresses capacity and performance requirements, Chapter 8 addresses reliability requirements, Chapter 9 addresses documentation requirements. Each of the above chapters follows a standard format in which a requirement is stated in the form: "The compiler shall . . . ," and then a "Guidance" section is provided giving background information and help in subsetting or tailoring the requirement for specific application domains.

## 4.19 NISSEN, ET AL: GUIDELINES FOR Ada COMPILER SPECIFICATION AND SELECTION

Citations:

[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al., "Guidelines for Ada compiler specification and selection," in Ada: *Language, compilers and bibliography*, ed. M.W.Rogers, Cambridge University Press, 1984.

Synopsis:

Members of Ada-Europe produced this set of guidelines based upon a taxonomy of compiler features. Their caveat is clear: "The relative value of information about different features of the compiler is a matter of judgment and circumstance. ... It is the reader's responsibility to weigh each factor according to his requirements. No liability of whatever kind shall be carried by the authors."

The taxonomy is represented by the table of contents of the guide, reproduced in part below.

2. Host and target
3. Language-related issues
4. User-interfacing and facilities
    4.1 Compiler invocation and listing management
    4.2 Compilation options
    4.3 Other features
    4.4 Errors and warnings
    4.5 Other software supplied
    4.6 Compilation management
5. Performance and capacity
    5.1 Host performance and capacity
    5.2 Target code performance
6. Compiler and run-time interfacing
    6.1 Compiler issues
    6.2 Run-time system issues
7. Retargetting and rehosting
    7.1 Introduction and definitions
    7.2 Retargetting
    7.3 Rehosting
8. Contractual matters
9. Validation

Chapter 2, Host and Target, briefly treats compiler configuration issues, and provides a checklist [Host and target checklist 14.1]

Chapter 3, Language-related issues, extracts from the Ada language reference manual [@DoD 1983] those features explicitly allowed to vary based upon machine specific characteristics [Machine-specific characteristics checklist 14.2]

Methods:

| | |
|---|---|
| [Host and target checklist | 14.1; |
| Machine-specific characteristics checklist | 14.2] |

## 4.20 WIS COMPILER EVALUATION GUIDELINES

Citations:

[WIS CEG 1985] "WIS Compiler Evaluation Guidelines," GTE Labs, Technical Report, 1985.

Synopsis:

This document presents guidelines that provide an information base on which specific compiler evaluation methodology and criteria can be built. Three types of guidelines have been identified: essential characteristics, highly recommended characteristics, and recommended characteristics. Also, certain questions that compiler vendors should be asked regarding their compilers measurable characteristics are listed. The guidelines take the view that the development of Ada compilers is an ongoing process. To address this fact, the document discusses, where appropriate, general aspects of compilers, and specific aspects of Ada compilers.

The document is broken down into four main sections. Part 1 is an introductory section. Part 2 provides background information on Ada compilers. Part 3 discusses compiler architecture issues. Part 4 then provides the main Ada compiler guidelines.

## 4.21 WIS TOOL EVALUATION CRITERIA

Citations:

[WIS PCEG 1985] G. Gicca and C. Stacey, "Proposed Component Evaluation Guidelines," GTE Government Systems, Technical Report, 16 August 1985.

Synopsis:

This document outlines a process to be used in evaluating currently available software tools for inclusion in an Ada software development environment. It defines a four phase evaluation process where each successive phase takes a more detailed view of the particular development tool. All phases have the same basic set of evaluation categories, with the definition of each being defined at a more refined level in the following phase. There are seven such categories. These are: "Functional Applicability," "Understandability," "Testability," "Evolvability," "Efficiency," "Portability," and "Human Engineering." Their definitions at a high level are:

Functional Applicability — the extent to which the tool or component fulfills a current need within a software development support environment

Understandability — the extent to which the tool or component is understandable from a systems viewpoint

Testability — the ease with which a program can be tested to verify that it performs its intended functions

Evolvability — a category that evaluates the combination of both modifiability and expandability

Efficiency — the amount of time and space required by a program to perform a function

Portability — the ease of transferring a program from one hardware configuration or software environment to another

Human Engineering — the ease of learning, operating, preparing input, and interpreting output of a program.

Each of the four phases of the component evaluation has its own rating scheme. A checklist is created for each category and for each sub-category within the basic evaluation categories. The rating scheme itself defines a set of numbers from 1 to 5. The reviewer then rates a particular tool or component by assigning a value for each sub-category. A weighting factor is then used to prioritize sub-categories and main categories. In the end a final set of numbers is produced that allows for overall comparisons between tools that offer similar capabilities.

# 5. COMPILATION SYSTEM ASSESSORS

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 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. These large-scale projects include the Strategic Defense Initiative (SDI), the NASA Space Station, the Software Technology for Adaptable, Reliable Systems (STARS) program, Army Tactical Command and Control System, Army WWMCCS Information System (WIS), and the Advanced Tactical Fighter (ATF), Advanced Tactical Aircraft (ATA), and Light Helicopter Experimental (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.

## 5.1 Ada COMPILER VALIDATION CAPABILITY (ACVC)

Purpose: Validation of the completeness of the Ada compiler by means of a compiler test suite. The ACVC consists of a test suite, analysis tools, and accompanying documentation, to enable the determination of conformance of Ada compiler implementations to the ANSI/MIL-STD-1815A. Note: The AJPO requires that Ada compilers pass the ACVC and the vendor allow the distribution of the resulting Validation Summary Report (VSR) in order for the compiler to be advertised as a commercially available Ada compiler.
[@RM: Compilation 7.1.6.7; @RM: Completeness 6.4.9]

Primary References:
[@ACVC 1987]

Host/OS: Unrestricted.

Vendors/Agents: [National Technical Information Service]

Method: Automated test suite.

Inputs:
ACVC source code, Ada compiler and runtime system, and host (and target) computer.

Process:

1. Obtain latest ACVC test suite
2. Following documentation, compile and run tests
3. Use analysis tools on test outputs.

Outputs:
Validation results;
Validation Summary Report (VSR).

## 5.2 IDA BENCHMARKS

Purpose: Evaluation of the capacity and performance of the Ada compiler by means of a compiler test suite.
[@RM: Compilation 7.1.6.7, @RM: Capacity 6.4.6;
@RM: Processing Effectiveness 6.4.22; @RM: Storage Effectiveness 6.4.31]

Primary References:
[@IDA 1985]

Host/OS: VAX/VMS or any system that can read ANSI standard tapes.

Vendors/Agents: [SofTech, Inc.]

Method: Test suite.

Inputs: IDA source code, Ada compiler and runtime system, and host (and target) computers.

Process:

1. Obtain test suite from agent
2. Compile and run Ada programs.

Outputs: Timing and storage measurements for individual language features.

## 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)
      - 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. Effects 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. Tradeoffs
  1. Coding Styles
    - order of evaluation
    - default vs initialized
    - order of selection (rendezvous)
    - scope of usage (global, local, 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
    - E-3A simulator
    - navigation algorithms
    - radar tracking algorithms
    - communication algorithms
  3. Ideal Ada
    - AI applications
    - kernel algorithms

III. Code Size Efficiency
  A. Expansion Code Size
  B. Run Time System Size
  C. Executable File Size

The first version of the ACEC is expected in August 1988.
[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22;
@RM: Storage Efficiency 6.4.31]

Primary References:
[ACEC 1988] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Aeronautical Laboratory, Document Number D500-11790-2, Draft 31 January 1988.

Host/OS: Portable

Vendors/Agents: [DACS]

Method:

Automated test suite

Inputs: ACEC source code, Ada compiler and runtime system, host and target computer.

Process:

1. Obtain the ACEC
2. Compile and run the tests according to the documentation
3. Use the analysis tools on the test outputs.

Outputs: Reports containing the execution time and/or code size for the selected tests.

## 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.

## 5.5 UNIVERSITY OF MICHIGAN BENCHMARK TESTS

Purpose: Identification of the execution efficiency of the code generated by Ada compilers. The tests measure only the performance of isolated language features as they relate to real-time performance.
[@RM: Compilation 7.1.6.7, @RM: Processing Effectiveness 6.4.22]

Primary References:
[Mich 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Electrical Engineering and Computer Science Dept., Univ. of Michigan, RSD-TR-6-86, January 1986, pp. 1-25.

Host/OS: Unrestricted

Vendors/Agents: [UMich]

Method:
Test suite.

Inputs: UMichigan source code, Ada compiler and runtime system, and host (and target) computer.

*Process:*

1. Obtain the UMichigan tests
2. Compile and run according to the documentation.

Outputs: Reports on the outcome of each test run.

## 5.6 MITRE BENCHMARK GENERATOR TOOL (BGT)

Purpose: Evaluation of the ability of an Ada compilation system to support development of very large systems in Ada. Under sponsorship of the Federal Aviation Administration, MITRE developed the Benchmark Generator Tool (BGT). The benchmark tests address capacity issues arising with large system developments. The initial version (1988) includes two types of tests: Library Capacity Tests and Dependency Maintenance Tests.

Primary References:
[MITRE BGT 1986] S.R. Rainer, and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corporation, MTR-87W00192-01, January 1988.

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [MITRE, McLean, VA]

Method:
Automated tool.

Inputs:
BGT source code, Ada compiler, and host computer.

Process:

1. Obtain the BGT
2. Compile according to the documentation.

Outputs:
Results of the above analysis, including capacity limits, link time, compilation time, etc.

## 5.7 UK Ada EVALUATION SYSTEM (AES)

Purpose: Evaluation of Ada compilers and associated linkers/loaders, program library systems, debuggers, and run-time libraries. A test suite and a methodology (AES) were developed by Software Sciences Ltd., under sponsorship of the UK Ministry of Defense (MoD). The British Standards Institute (BSI) has been sponsored by the MoD to provide an Ada Evaluation Service, using the AES. Interested parties, such as compiler vendors or potential compiler purchasers, may pay BSI to conduct an evaluation or to supply a copy of an existing evaluation report.

Primary References:
[UK AES 1986]

Host/OS: Any for which an Ada compiler exists.

Vendors/Agents: [BSI, Milton-Keynes, UK]

Method:
Automated test suite and questionnaire.

Inputs:
AES source code and questionnaire, Ada compiler and runtime system, and host (and target) computer.

Process:
Pay BSI to do an evaluation or purchase an existing evaluation report.

Outputs:
An Evaluation Report organized in a standard format.

## 5.8 COMPILATION CHECKLIST

Purpose: Evaluation of the power of compilation by developing a list of functional capabilities.
[@RM: Compilation 7.1.6.7, @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: Capability checklist (see Table 5.8-1) and compiler documentation.

Process: Check off capabilities demonstrated during compiler runs or discussed in the documentation.

Outputs: A list of capabilities performed by the compiler.

## TABLE 5.8-1
## COMPILATION CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Conditional Compilation | |
| Debug Information Generation | |
| Enable/Disable Listing | |
| Errors Only Listing | |
| Error Identification | |
| Set Default Directory For Source | |
| Set Listing Width And Height | |
| Specify Different Program Library | |
| Specify Main Program | |
| Disable Use Of SYSTEM Library | |
| Suppress All Run-Time Checks | |
| Compile Multiple Files | |
| Language Sensitive Editor Support | |
| Specify Error Limit | |
| Enable/Disable An Error Category | |
| Specify Optimization Parameters | |
| Syntax Only Checking | |
| Symbol Table | |
| Variable Set/Use Indications (Cross Reference) | |
| Object Code Listing | |
| Object Attribute Map | |
| Code Statistics | |
| Unidentified Compiler Options(Pragmas) | |
| Controlled Dynamic Storage | |
| Elaboration Control | |
| Inline Expansion Of Subprograms | |
| Interface With Other Languages | |
| Specify Memory Size | |
| Pack Data Representations In Memory | |
| Priority Control Of Concurrent Tasks | |
| Shared Variables | |
| Specify Storage Unit | |
| Specify Alternative System Characteristics | |
| Machine Code Mapping | |
| Machine Code Insertions | |
| Cross Compilation | |
| Error Reporting | |
| Exceptions List | |
| Identify Target Dependencies | |

## 5.9 PROGRAM LIBRARY MANAGEMENT CHECKLIST

Purpose: Evaluation of the power of program library management by developing a list of functional capabilities.
[@RM: Program Library Management 7.2.1.7, @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 5.9-1) and program library manager documentation.

Process: Check off capabilities demonstrated by the program library manager or discussed in the documentation.

Outputs: A list of capabilities performed by the program library manager.

TABLE 5.9-1

PROGRAM LIBRARY MANAGEMENT CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Listing Information | |
| List Of Logical Unit Names | |
| Associated File Names For Unit | |
| Units Using Specified Unit | |
| Units Used By Specified Unit | |
| Size Information | |
| Time-Stamp Information | |
| Kind Of Compilation Unit | |
| Units Used To Construct Executable | |
| Completeness And Currency Check | |
| Automatic Recompilation | |
| Spawn CLI Subprocess | |
| Create Structures | |
| Move Elements Between Libraries | |
| Move Elements Between Directories | |
| Remove Compilation Unit | |
| Library Access Control | |
| Read Only (Shared) | |
| Exclusive | |

## 5.10 ARTEWG CATALOGUE OF Ada RUNTIME IMPLEMENTATION DEPENDENCIES

Purpose: The purpose of this document is best stated by the following quotation taken from the rationale section in the catalogue: "The main goal of this catalogue is to be the one place where all the areas of the Ada Reference Manual ... which permit implementation flexibilities can be found." These implementation dependencies affect the performance and adaptation characteristics of the generated code. The text describes each known dependency by a number (which identifies the relevant section and paragraph in the Ada Reference Manual), a topic or title, a question which poses the implementation issue, a dependency type (either explicit or implicit), a rationale explaining why the dependency exists, and an Ada example to further clarify the dependency. (These descriptions could be used as the basis for an automated test suite.)
[@RM: Compilation 7.1.6.7, @RM: Anomaly Management 6.4.2,
@RM: Processing Effectiveness 6.4.22, @RM: Retargetability 6.4.27,
@RM: Storage Effectiveness 6.4.31]

Primary References:
[ARTEWG 1987] Catalogue of Ada Runtime Implementation Dependencies,"
Association for Computing Machinery, Special Interest Group on Ada,
Ada Runtime Environment Working Group, 1 December 1987.

Host/OS: Not applicable.

Vendors/Agents: [ARTEWG]

Method:
Questionnaire.

Inputs: Descriptions of implementation dependent features.

Process:

1. Select critical dependencies
2. Build and run tests for each dependency or ask vendor how dependencies are implemented
3. Select compiler and/or make coding standards based on results of step 2.

Outputs: Evidence showing how features are implemented.

## 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."

Primary References:

[ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Pro posed 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.

TABLE 5.11-1

RUNTIME ENVIRONMENT TAXONOMENT

| 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 | |

# 6. TARGET CODE GENERATION AIDS AND ANALYSIS TOOLSET ASSESSORS

These tools are used to assess 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 program execution. These assessments are also used in the case where the host computer is also the target computer.

## 6.1 ASSEMBLING CHECKLIST

Purpose: Evaluation of the power of assembling by developing a list of functional capabilities.
[@RM: Assembling 7.1.6.6, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.1-1) and assembler documentation.

Process: Check off capabilities demonstrated during assembler runs or discussed in the documentation.

Outputs: A list of capabilities performed by the assembler.

TABLE 6.1-1

ASSEMBLING CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Code Generation | |
| Macro Preprocessing | |
| Conditional Assembly | |
| Debug Information Generation | |
| Enable/Disable Listing | |
| Errors Only Listing | |
| Set Listing Width and Height | |
| Suppress All Run-Time Checks | |
| Assemble Multiple Files | |
| Specify Error Limit | |
| Enable/Disable An Error Category | |
| Syntax Only Checking | |
| Symbol Table | |
| Code Statistics | |
| Cross Assembly | |

## 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 | |
| Generate A Link Command File | |
| Enable/Disable Symbol Cross-Reference | |
| Generate Debug Information | |
| Enable/Disable Execution File Creation | |
| Specify Batch/Nobatch Operation | |
| Specify Map File Name | |
| Specify Object File Name | |
| Specify Diagnostic Output File | |
| Enable/Disable System Library Search | |
| Enable/Disable Traceback Information | |
| Library Search Capabilities | |
| Extended Options Capabilities | |
| Sharable Image Support | |
| Specify Maximum Memory | |
| Specify Optimization Parameters | |
| Force Load | |
| Enable/Disable Library Trace | |
| Specify Main Program | |
| Non-Specific Language Main Program | |
| Overlays | |

## 6.3 IMPORT/EXPORT CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of import/export by developing a list of functional capabilities.
[@RM: Import/Export 7.2.3.6, @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 6.3-1) and import/export documentation.

Process: Check off capabilities demonstrated by the import/export system or discussed in the documentation.

Outputs: A list of capabilities performed by the import/export system.

TABLE 6.3-1
IMPORT/EXPORT CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Host to Target Object Downloading<br>Target to Host Data Uploading | |

Note: This table will be expanded in a future version of the Guidebook.

## 6.4 EMULATION CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of emulation by developing a list of functional capabilities.
[@RM: Emulation 7.3.2.13, @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 6.4-1) and emulation system documentation.

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

Outputs: A list of capabilities performed by the emulation system.

TABLE 6.4-1
EMULATION CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Hardware<br>System Emulation/Simulation | |

Note: This table will be expanded in a future version of the Guidebook.

## 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.

## 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 | |
|  Name Of Current Exception | |
| Evaluate Objects | |
| Step | |
|  Single | |
|  By Discrete Amounts | |
|  Into Subprograms | |
|  Over Subprograms | |
|  To Next Scheduling Event | |
|  To Next Exception | |
|  To End of Program Unit | |
| Miscellaneous | |
|  Symbol Abbreviation | |
|  Set Context For Control | |
|  Input Debugger Command Files | |
|  Modify Variable Values | |
|  Modify Object Code | |
|  Modify Control Flow | |
|  Console Interrupt | |
|  Full Screen Mode | |
|  Keypad For Entering Commands | |
|  Virtual Clock | |
|  Special Compilation Mode | |
|  Multi-Language Support | |
|  Dynamic Interrupt | |
|  Optimization Support | |
|  Units Comprising Executable | |

## 6.6 TIMING ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of timing analysis by developing a list of functional capabilities.
[@RM: Timing 7.3.2.14, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 6.6-1) and timing analysis system documentation.

Process: Check off capabilities demonstrated by the timing analysis system or discussed in the documentation.

Outputs: A list of capabilities performed by the timing analysis system.

TABLE 6.6-1

TIMING ANALYSIS CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Timing Instrumentation | |
| Obtrusive | |
| Non Obtrusive | |
| Fraction By Section Of Code | |
| Tasking Monitor | |
| Fraction Executing | |
| Fraction Runnable | |
| Fraction Runnable And Not Executing | |
| Time Between Runnable And Executing | |
| Time Between Events | |
| System Idle Time | |

## 6.7 TUNING ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of tuning analysis by developing a list of functional capabilities.
[@RM: Tuning 7.3.2.15, @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 6.7-1) and tuning analysis system documentation.

Process: Check off capabilities demonstrated by the tuning analysis system or discussed in the documentation.

Outputs: A list of capabilities performed by the tuning analysis system.

TABLE 6.7-1

TUNING ANALYSIS CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Task Monitoring Instrumentation | |

Note: This table will be expanded in a future version of the Guidebook.

## 6.8 REAL-TIME ANALYSIS CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of real-time analysis by developing a list of functional capabilities.
[@RM: Real-Time Analysis 7.3.2.17, @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 6.8-1) and real-time analysis system documentation.

Process: Check off capabilities demonstrated by the real-time analysis system or discussed in the documentation.

Outputs: A !ist of capabilities performed by the real-time analysis system.

**TABLE 6.8-1**

**REAL-TIME ANALYSIS CAPABILITIES CHECKLIST**

| FEATURE | FOUND |
|---|---|
| Hardware-In-The-Loop<br>Non-Intrusive Instrumentation<br>Performance Analysis<br>Symbolic Trace | |

# 7. TEST SYSTEMS ASSESSORS

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

## 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.

## 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 | |
| Data Auditors | |
| Body/Stub Generators | |
| File Comparators | |
| Data/File Services | |
| Software and System Test Communications Facilities | |
| Test Execution Services | |
| Test Harness Generator | |
| Data and Error Logging Services | |
| Quality Measurement Tools | |
| Test Analysis Services | |
| Correctness Analyzers (Oracles) | |
| Instrumentation Aids | |
| Status Display Tools | |
| Data Reduction and Analysis Tools | |
| Decision Support Services | |
| Documentation Services | |
| Information Repositories | |
| Problem Report Processing and Analysis Tools | |
| Change Request Processing and Analysis Tools | |

## 7.2 SEI UNIT TESTING AND DEBUGGING EXPERIMENT

Purpose: Evaluation of an environment's capabilities, from the point of view of the unit tester. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Design Support Experiment [9.1].

Primary References:
[@Weiderman 1986, Chapter 6;
Weiderman: Evaluation of Ada Environments, 4.11]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment

Inputs: The "generic" experiment description, an APSE, and host (and target) computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in table of functional elements present and missing, elapsed time and cpu time values, and subjective judgments based on the experience.

# 8. TOOL/HOST INTERFACE ASSESSORS

These assessors evaluate or validate implementations of Ada language specifications for tool/host interface sets. Interface sets that may be assessed could include a CAIS or a CAIS-A implementation, Portable Common Tool Environment (PCTE) implementations and Ada language interfaces to the UNIX operating system and its variants (e.g., Berkeley UNIX, System V, A/UX, POSIX).

## 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.

Primary References:
TBA

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.

## 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 conierence management system, a window manager, and a design editor.

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.

# 9. Ada DESIGN SUPPORT ASSESSORS

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

## 9.1 SEI DESIGN SUPPORT EXPERIMENT

Purpose: Evaluation of the design and code development capabilities of an environment, as represented in a small project. An experiment was designed to simulate the activities normally associated with small projects, namely the design, creation, modification, and testing of a single unit or module. See also the SEI Unit Testing and Debugging Experiment [7.2].

Primary References: [@Weiderman 1986, Chapter 5;
Weiderman: Evaluation of Ada Environments, 4.11]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist of functional elements present and missing, tables of time and space data, and subjective judgments based on the experience.

# 10. CONFIGURATION MANAGEMENT SUPPORT ASSESSORS

These assessors exar.ine the performance, usability, and completeness of the APSE or APSE component functionality related to controlling the contents of software systems. This includes monitoring the status, preserving the integrity of released and developing versions, and controlling the effects of changes throughout the lifetime of the software sys:em.

## 10.1 CONFIGURATION MANAGEMENT CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of configuration management by developing a list of functional capabilities.
[@RM: Configuration Management 7.2.2.7, @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 10.1-1) and configuration management documentation.

Process: Check off capabilities demonstrated by the configuration manager or discussed in the documentation.

Outputs: A list of capabilities performed by the configuration manager.

TABLE 10.1-1

CONFIGURATION MANAGEMENT CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Version Management | |
| Archive | |
| Protect | |
| Revision Management | |
| Audit Support | |
| Configuration Library | |
| Create | |
| Delete | |
| Verify | |
| Library Elements | |
| Create | |
| Delete | |
| Fetch | |
| Reserve | |
| Unreserve | |
| Replace | |
| Differences | |
| Element Classes | |
| Create | |
| Delete | |
| Insert Element | |
| Remove Element | |
| Listings | |
| Elements | |
| Reservation | |
| History | |
| Annotation | |
| Completeness | |
| Level Control | |
| Usage Administration | |
| Test Control | |
| Procedures | |
| Data | |
| Results | |
| Failure Reporting | |

## 10.2 SEI CONFIGURATION MANAGEMENT EXPERIMENT

Purpose: Evaluation of the configuration management and version control capabilities of an environment. An experiment was designed to simulate the system integration and testing phase of the life cycle by having three separate development lines of descent from a single baseline.

Primary References: [@Weiderman 1986, Chapter 3;
Weiderman: Evaluation of Ada Environments, 4.11]

Host/OS: VAX/VMS and VAX/UNIX

Vendors/Agents: [SEI]

Method: Structured experiment.

Inputs: The "generic" experiment description, an APSE, and host computer.

Process: "Instantiate" the experiment for a specific Host/OS/APSE combination and carry it out.

Outputs: A filled-in checklist showing functional elements present and missing, a table of elapsed-time values for certain specific operations, and subjective judgments based on the experience.

# 11. DISTRIBUTED SYSTEMS DEVELOPMENT AND RUNTIME SUPPORT ASSESSORS

These assessors examine 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.

# 12. DISTRIBUTED APSE ASSESSORS

These assessors examine 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.

# 13. "WHOLE APSE" ASSESSORS

These assessors examine or measure the overall quality or performance of an APSE considered as a whole rather than as a collection of individual parts individually assessed. A specific whole-APSE assessor may be designed to achieve a limited objective, such as: evaluate the quality of an APSE in supporting a team of software developers performing a specific life-cycle phase or activity such as preliminary design or integration testing. The results of such an eva!uation could then become one ingredient of an integrated whole-APSE assessment, as described in Section 3.3.

## 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.]

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.

Name/Acronym:

Vendor:

Address:

Phone Number:

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

Purchase ____________

Maintenance ____________

Documentation ____________

On-Line Help ____________

Hot-Line Support ____________

Seminars ____________

In House Classes ____________

Educational Videos ____________

On-Line Tutorials ____________

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/Phase:

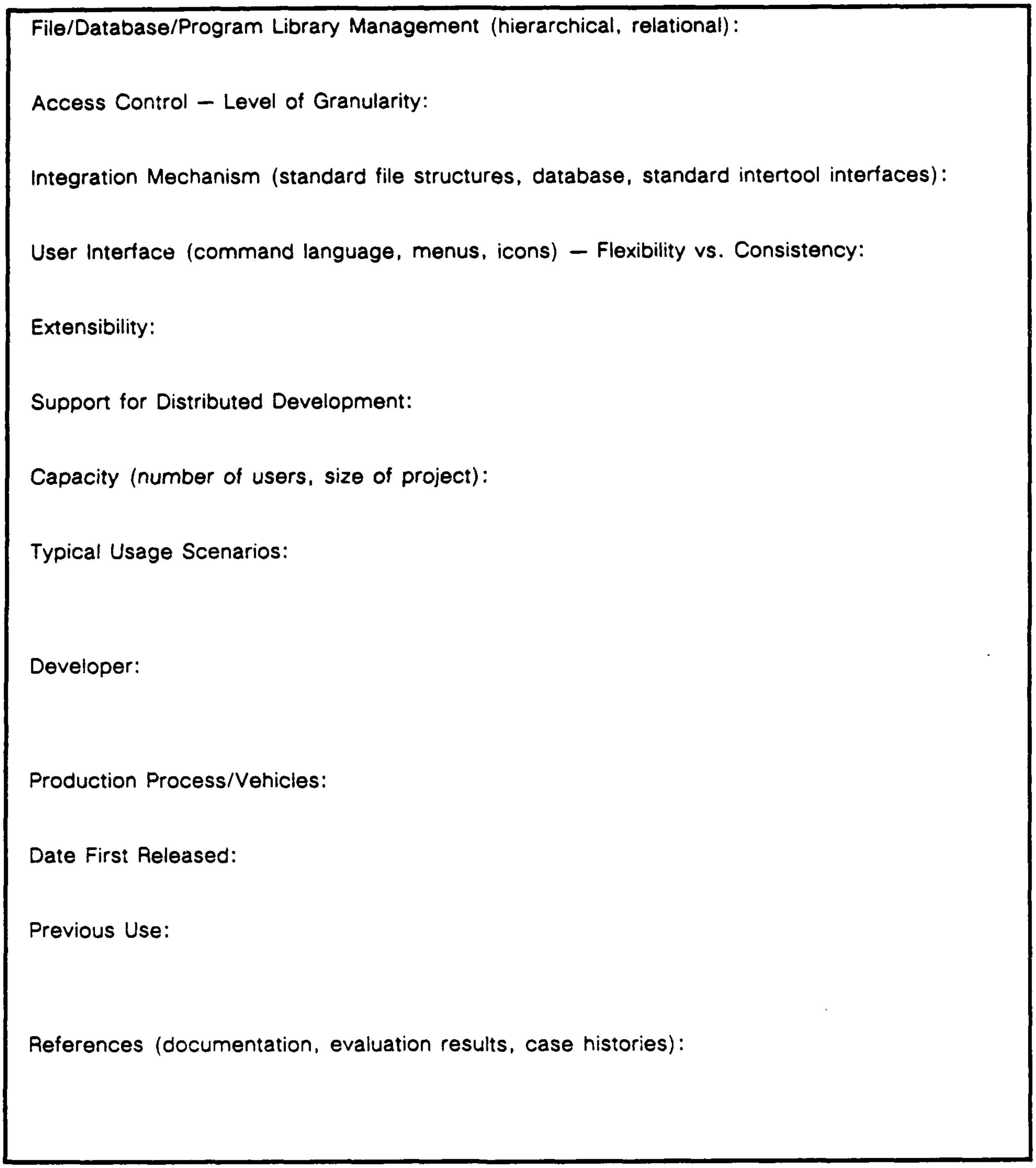
Methodology Support:

Management Support:

Application-Specific Capabilities:

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

Figure 13.1-1 APSE Characterization Form (Page 1)



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):

Figure 13.1-1 APSE Characterization Form (Page 2)

# 14. ADAPTION ASSESSORS

These assessors examine the ease with which an APSE or APSE component can be used beyond its original requirements, such as extending or expanding capabilities and adapting for use in another application or environment. This is measured as the degree to which this adaptation can be accomplished without reprogramming.

## 14.1 HOST AND TARGET CHECKLIST

Purpose: Evaluation of tools relative to host and target configurations.
[@RM: Rehostability 6.4.25;
@RM: Retargetability 6.4.27;
@RM: Transportability 6.3.4]

Primary References:
[@Nissen 1984;
Nissen, et al.: Guidelines For Ada Compiler Specification And Selection: 4.17]

Vendors/Agents: [Cambridge University Press]

Method: Assessment checklist

Inputs: Checklist and tool documentation.

Process: Fill in the appropriate answers in the following checklist.

a) Host configuration(s) required

b) Host operating system(s) required

c) Target configuration(s) supported

d) Target operating system(s) supported

e) APSE(s) supported, if applicable

f) Host-target communication supported

i) program loading

ii) program execution and debugging.

Outputs: A completed list which characterizes the tool relative to host-target issues.

## 14.2 MACHINE-SPECIFIC CHARACTERISTICS CHECKLIST

Purpose: Evaluation of tools relative to machine-specific characteristics.
[@RM: Rehostability 6.4.25;
@RM: Retargetability 6.4.27]

Primary References:
[@Nissen 1984;
Nissen, et al.: Guidelines For Ada Compiler Specification And Selection: 4.17]

Vendors/Agents: [Cambridge University Press]

Method: Assessment checklist

Inputs: Checklist and tool documentation.

Process: Fill in the appropriate answers in the following checklist.

[@DoD 1983: Lexical Elements 2.]

- [@DoD 1983: 2.1] Character set of the host and target
- [@DoD 1983: 2.2] Maximum number of characters on a line of the host and target
- [@DoD 1983: 2.3, 2.4] Is the maximum character length of an identifier or numerical literal restricted other than by line length
- [@DoD 1983: 2.8, F.] The form, allowed place, and effect of every implementation-defined pragma

[@DoD 1983: Declarations and Types 3.]

- [@DoD 1983: 3.2.1] The effect of using uninitialized variables — does the compiler flag or reject program that depends upon such variables
- [@DoD 1983: 3.5.1] The maximum number of elements in an enumeration type
- [@DoD 1983: 3.5.4] The values cf:

-- INTEGER'FIRST

-- SHORT_INTEGER'FIRST

-- LONG_INTEGER'FIRST

-- INTEGER'LAST

-- SHORT_INTEGER'LAST

-- LONG_INTEGER'LAST

- [@DoD 1983: 3.5.8] The values of:

-- FLOAT'DIGITS

-- SHORT_FLOAT'DIGITS

-- LONG_FLOAT'DIGITS

[@DoD 1983: Names and Expressions 4.]

- [@DoD 1983: 4.10] Is there a limit on the range of universal values which exceeds the capacity of the compiler
- [@DoD 1983: 4.10] Is there a limit on the accuracy real universal expressions

[@DoD 1983: Tasks 9.]

- [@DoD 1983: 9.6] The values of:

-- DURATION'DELTA

-- DURATION'SMALL

-- DURATION'FIRST

-- DURATION'LAST

- [@DoD 1983: 9.8] The values of:

-- PRIORITY'FIRST

-- PRIORITY'LAST

- [@DoD 1983: 9.11] The restrictions on shared variables

[@DoD 1983: Program Structure and Compilation Issues 10.]

- [@DoD 1983: 10.1] Initiation, communication with, and restrictions on the main program
- [@DoD 1983: 10.5] When tasks initiated in imported library units will terminate

[@DoD 1983: Exceptions 11.]

- [@DoD 1983: 11.1] Conditions under which these exceptions are raised:

-- NUMER!C_ERROR

-- PROGRAM_ERROR

-- STORAGE_ERROR

[@DoD 1983: Representation Clauses and Implementation-Dependent Features 13.]

- [@DoD 1983: 13.4, F.] The list of all restrictions on representation clauses
- [@DoD 1983: 13.1, F.] The conventions used for any system generated name denoting system dependent components
- [@DoD 1983: 13.5, F.] The interpretation of expressions that appear in address clauses, including those for interrupts
- [@DoD 1983: 13.7] The specification of package SYSTEM; which includes the values of:
  - -- MIN_INT
  - -- MAX_INT
  - -- MAX_DIGITS
  - -- MAX_MANTISSA
  - -- FINE_DELTA
  - -- TICK

[@DoD 1983: 13.7.3] For a pre-defined floating point type F, the value of:

- -- F'MACHINE_ROUNDS
- -- F'MACHINE_RADIX
- -- F'MACHINE_MANTISSA
- -- F'MACHINE_EMAX
- -- F'MACHINE_EMIN
- -- F'MACHINE_OVERFLOWS

- [@DoD 1983: 13.7.3] The values outside the range of safe numbers for real types
- [@DoD 1983: 13.10.1] Any restriction on UNCHECKED_DEALLOCATION
- [@DoD 1983: 13.10.2, F.] Any restriction on UNCHECKED_CONVERSION

[@DoD 1983: Input-Output 14]

- [@DoD 1983: 14., F.] Any implementation-dependent characteristics of the input-output packages

- [@DoD 1983: Implementation-Dependent Features F.]
    - [@DoD 1983: F.] The name and type of every implementation-dependent attribute

Outputs: A completed list which characterizes the tool relative to machine dependencies.

# 99. OTHER ASSESSORS

This chapter contains instances of E&V technology that do not conveniently fit one of the earlier chapters. It is likely that in future versions of the Guidebook some of these "miscellaneous" instances will be grouped together in new chapters, and therefore moved out of Chapter 99.

## 99.1 TEXT EDITING CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of text editing by developing a list of functional capabilities.
[@RM: Text 7.1.1.1, @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 99.1-1) and text editor documentation.

Process: Check off capabilities demonstrated during editing sessions or discussed in the documentation.

Outputs: A list of capabilities performed by the text editor.

## TABLE 99.1-1
## TEXT EDITING CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Locator Movement | |
|  Left, Right, Up, Down | |
|  Top, Bottom | |
|  Beginning/End of Line | |
|  Next/Previous Word | |
| Search/Replace | |
|  Search Forward | |
|  Search Backward | |
|  Regular Expression Search | |
|  Regular Expression Replace | |
|  Multiple Replace | |
|  Scroll Up, Down, Left, Right | |
|  Page Up, Down | |
| Buffers | |
|  Copy Text To | |
|  Copy Text From | |
|  Edit Multiple Files | |
|  Split Screen | |
| Regions | |
|  Set Mark | |
|  Delete Region | |
|  Copy Region | |
|  Move Region | |
| File Manipulation | |
|  Copy From File | |
|  Append To File | |
| Macros | |
|  Keyboard Macros | |
|  Macro Language | |
|  Language Sensitive | |
|    Identifiers/Keywords Search | |
|    Syntax Templates/Completion | |
|    Reformat (Pretty Print) | |
|    Display High-Level Structure | |
|    Display Unclosed Structures | |
|    Display Matching Structures | |
|    Display Permitted Constructs | |
|    "Comment Out" Code Block | |
|    Invoke Syntax Check/Translation | |
|  LRM Automated Access | |
| Miscellaneous | |
|  *Terminal Independent* | |
|  On-Line Help Facility | |
|  Minimal Redisplay Algorithm | |
|  Key Redefinition | |
|  Undo Capability | |
|  Spawn CLI | |
|  Command Recall and Integration | |
|  Command Type-Ahead | |

## 99.2 DATA BASE MANAGEMENT CHECKLIST

Purpose: Evaluation of the power of data base management by developing a list of functional capabilities.
[@RM: Data Base (Object) Management 7.2.1.1, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.2-1) and data base manager documentation.

Process: Check off capabilities demonstrated by the data base manager or discussed in the documentation.

Outputs: A list of capabilities performed by the data base manager.

TABLE 99.2-1

DATA BASE MANAGEMENT CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Administration | |
| Access Control | |
| Backup And Recovery | |
| Data Dictionary Management | |
| Design Aids | |
| Performance Prediction | |
| Report Generation | |
| Query | |
| Data Set | |
| User Defined | |
| Tools | |
| Subschema (View) Facility | |

## 99.3 ELECTRONIC MAIL CHECKLIST

Purpose: Evaluation of the power of electronic mail by developing a list of functional capabilities.
[@RM: Electronic Mail 7.2.1.4, @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 99.3-1) and mail system documentation.

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

Outputs: A list of capabilities performed by the mail system.

TABLE 99.3-1
ELECTRONIC MAIL CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Send | |
| Receive | |
| Immediate Forwarding | |
| Immediate Reply | |
| Archive | |
| Print | |
| Search For String | |
| Edit Message To Be Sent | |
| Read Next Message | |
| Read Previous Message | |
| Read First Message | |
| Read Last Message | |
| Position To Start Of Message | |
| Keypad Support | |
| On-Line Help Facility | |
| Send To Distribution Lists | |
| Send Across Network | |
| Mail Filing | |

## 99.4 REQUIREMENTS PROTOTYPING CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of requirements prototyping by developing a list of functional capabilities.
[@RM: Requirements Prototyping 7.3.2.2, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.4-1) and requirements prototyping documentation.

Process: Check off capabilities demonstrated by the requirements prototyping system or discussed in the documentation.

Outputs: A list of capabilities performed by the requirements prototyping system.

TABLE 99.4-1
REQUIREMENTS PROTOTYPING CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Standards Requirements Libraries | |
| Executable Specifications | |
| Fourth Generation Languages or Very High Level Languages | |
| Reusable Building Blocks and Associated Tools | |
| Man-Machine Interface Prototyping Capabilities | |
| Applications Generators | |
| Previous Software Version Import Capabilities | |

## 99.5 PERFORMANCE MONITORING CHECKLIST

Purpose: Evaluation of the power of performance monitoring by developing a list of functional capabilities.
[@RM: Performance Monitoring 7.2.1.10, @RM: Power 6.4.21]

Primary References:

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.5-1) and performance monitor documentation.

Process: Check off capabilities demonstrated by the performance monitor or discussed in the documentation.

Outputs: A list of capabilities performed by the performance monitor.

TABLE 99.5-1

PERFORMANCE MONITOR CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Hardware | |
| CPU Time (Real And Virtual) | |
| Memory Usage | |
| I/O Channel Traffic | |
| Terminal Response | |
| Terminal Connect Time | |
| Terminal Availability | |
| Disk Usage | |
| Disk Space Availability | |
| Tape Mounts | |
| Tape Drive Availability | |
| Printout Quantity | |
| Software | |
| Tool Usage | |
| Program Library Monitoring | |
| Wall Clock Time | |

## 99.6 SIMULATION AND MODELING CAPABILITIES CHECKLIST

Purpose: Evaluation of the power of simulation and modeling by developing a list of functional capabilities.
[@RM: Simulation and Modeling 7.3.2.3, @RM: Power 6.4.21]

Primary References:
[ISTAR 1987] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

Vendors/Agents: [E&V Team]

Method: Capabilities checklist

Inputs: Capabilities checklist (see Table 99.6-1) and simulation and modeling documentation.

Process: Check off capabilities demonstrated by the simulation and modeling system or discussed in the documentation.

Outputs: A list of capabilities performed by the simulation and modeling system.

TABLE 99.6-1

SIMULATION AND MODELING CAPABILITIES CHECKLIST

| FEATURE | FOUND |
|---|---|
| Conceptual Modeling Support | |
| Domain-Specific Knowledge Base | |
| Inferencing Systems | |
| Operational Environment Modeling Support | |
| User Modeling Support | |
| Model Browsers | |
| Game and Risk Models Database | |
| Functional Allocation Methodologies Database | |
| Scaling Rules Database | |
| Constraint Evaluation Tools | |
| Resource Utilization Models | |
| Precision Estimators | |

# APPENDIX A
# CITATIONS

[ACEC 1986] "Ada Compiler Evaluation Capability (ACEC) Technical Operating Report (TOR) Reader's Guide," Air Force Wright Aeronautical Laboratory, Document Number D500-11790-2, Draft 31 January 1988.

[ACVC 1987] ACVC Procedures and Guidelines, Version 1.1, AJPO, 1 January 1987.

[ALS 1984] SofTech, "Ada Language System (ALS) Specification, "CR-CP-0059-A00, August 1984.

[ARTEWG 1987] "Catalogue of Ada Runtime Implementation Dependencies," Association for Computing Machinery, Special Interest Group on Ada, Ada Runtime Environment Working Group, 1 December 1987.

[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.

[Barnes 1985] *Proceedings of the International Ada Conference*, Paris, eds. J.G.P. Barnes and G.A. Fisher, Jr, Cambridge University Press, 1985.

[Barstow 1981] D.R. Barstow and H.E. Shrobe, "Observations on Interactive Programming Environments," [@Wassermann 1981], 286-301, 1981.

[Buxton 1980] [@DoD 1980].

[CAIS] [@DoD 1986].

[CAIS-A] [@DoD 1988].

[CIVC 1985] TBA

[DACS 1979] The DACS Glossary, A Bibliography of Software Engineering Terms, October 1979.

[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.

[DoD APSE Analysis 1984] [@E&V Report: DoD APSE Analysis Report C.]

[DoD-STD-2167] Proposed DoD-STD-2167, Defense System Software Development, U.S. Department of Defense, 30 January 1985.

[DoD 1977] DoD, "Requirements for High Order Computer Languages (IRONMAN), U.S. Department of Defense, 1977.

[DoD 1980] J.N. Buxton, "Requirements for Ada Programming Support Environments — STONEMAN," U.S. Department of Defense, February 1980.

[DoD 1982] "Software Development Methodologies and Ada (METHODMAN)," U.S. Department of Defense, 1982.

[DoD 1983] ANSI/MIL-STD-1815A-1983, Reference Manual for the Ada Programming Language, U.S. Department of Defense, 17 February 1983.

[DoD 1986] DoD-STD-1838, Common APSE Interface Set (CAIS), U.S. Department of Defense, 9 October 1986.

[DoD 1988] "Common APSE Interface Set, Revision A,"proposed DoD-STD-1838A, U.S. Department of Defense, January 1988.

[DoD Trusted Computer Reoort 1983] "Trusted Computer System Evaluation Criteria," CSC-STD-001-83, U.S. Department of Defense Computer Security Center, 15 August 1983.

[E&V Plan] [@E&V Report 1984: E&V Plan A]. [@E&V Report 1987: E&V Plan A].

[E&V Reference Manual] [@RM].

[E&V Report 1984] Evaluation and Validation (E&V) Team Public Report, Volume I, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, 30 November 1984.

[E&V Report 1985] "Evaluation and Validation (E&V) Team Public Report," Volume II, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, November 1985.

[E&V Report 1987] "Evaluation and Validation (E&V) Team Public Report," Volume III, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, September 1987.

[E&V Requirements 1987] [@E&V Report 1987: E&V Requirements D].

[E&V Schema 1987] "E&V Classification Schema Report," TASC, TR-5234-2, Version 1.0, 15 June 1987.

[E&V Tools and Aids 1987] [@E&V Report 1987: Tools and Aids C].

[Gray 1987] L. Gray, "Using the SEI's Methodology for Evaluating Ada Environments: A Comparison of VAX/VMS to Rational," Proceedings of the AIAA Computers in Aerospace VI Conference, 7-9 October 1987.

[Grund 1985] E.C. Grund, L.A. Hilliard, and K.A. Younger, "Key Characteristics of Ada Programming Support Environments," MITRE Corporation, ESD-TR-85-144, 9590, July 1985.

[Henderson 1987] P.B. Henderson and D. Notkin, "Integrated Design and Programming Environment," Computer, IEEE, November 1987.

[Hogan 1985] M.O. Hogan and S.M. Prud'homme, "Definition of a Production Quality Compiler," Aerospace Corporation, Technical Report, July 1985.

[Houghton 1983] R.C. Houghton, Jr., "A Taxonomy of Tool Features for the Ada Programming Support Environment (APSE)," U.S. Department of Commerce, National Bureau of Standards, NBSIR-81-2625, December 1982, Issued February 1983.

[Houghton and Wallace 1987] R.C. Houghton, Jr. and D.R. Wallace, "Characteristics and Functions of Software Engineering Environments: An Overview," ACM Software Notes, Vol. 12, No. 1, January 1987.

[Howden 1982] W.E. Howden, "Contemporary Software Development Environments," Communications of the ACM 25(5), 318-329, 1982.

[IDA 1985] A.A. Hook, G.A. Riccardi, M. Vilot, and S. Welke, "User's Manual for the Prototype Ada Compiler Evaluation Capability (ACEC)," Version 1, Institute for Defense Analysis, IDA Paper P-1879, October 1985.

[ISTAR 1987] Workshop on Future Development Environments, Information Science and Technology Assessment for Research, Conference on Information Mission Area (IMA) Productivity, Department of Army Director of Information Systems for Command, Control, Communications and Computers, 13-15 April 1987, pp 28.

[Jackson 1985] A.R. Jackson, "Abstract Data Types and the IPSE Database," [@McDermid 1985], 135-145, 1985.

[Kean 1985] E.S. Kean and F.S. Lamonica, "A Taxonomy Of Tool Features For A Life Cycle Software Engineering Environment," Rome Air Development Center, Griffiss AFB, June 1985.

[Lehman 1981] M.M. Lehman, "The Environment of Program Development, Maintenance Programming, and Program Support," [@Wasserman 1981], 3-14, 1981.

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

[Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons and J.C.D Nissen, Ada-Europe Working Group, Cambridge University Press, 1986.

[McDermid 1984] J. McDermid and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.

[METHODMAN] [@DoD 1982].

[Mich 1986] R.M. Clapp, L. Duchesneau, R.A. Volz, T.N. Mudge, and T. Schultze, "Toward Real-Time Performance Benchmarks for Ada," Electrical Engineering and Computer Science Dept., Univ. of Michigan, RSD-TR-6-86, January 1986, pp. 1-25.

[MITRE BGT 1986] S.R. Rainer and T.P. Reagan, "User's Manual for the Ada Compilation Benchmark Generator Tool," MITRE Corp. MTR-87W00192-01, January 1988.

[Morton 1985] R.P. Morton and J.C. Wileden, "Information Interface Related Sources," Institute for Defense Analyses, SEE-INFO-003-001.0, IDA Paper p-1821, April 1985 (Appendix 2, L. Stucki, "Some Thoughts on a Taxonomy for Software Engineering Objects").

[NBS Taxonomy] [@Houghton 1983].

[Nissen 1984] J.C.D. Nissen, B.A. Wichman, et al.,"Guidelines for Ada Compiler Specification and Selection," in Ada: *Language, Compilers And Bibliography*, ed. M.W. Rogers, Cambridge University Press, 1984.

[Notkin 1981] D.S. Notkin and A.N. Habermann, "Software Development Environment Issues as Related to Ada," [@Wasserman 1981], 107-133, 1981.

[Oberndorf 1988] P.A. Oberndorf, "The Common Ada Programming Support Environment (APSE) Interface Set (CAIS)," IEEE Transactions on Software Engineering, Vol. 14, No. 6, June 1988.

[RM] "Evaluation and Validation (E&V) Reference Manual," Version 1.0, Air Force Wright Aeronautical Laboratories, AFWAL TR-88-1060, Wright Patterson AFB, (DTIC Accession Number Pending), March 1988.

[SEE Taxonomy] [@Kean 1985].

[STARS-SEE 1985] "Proposed Version 001.0," STARS Joint Service Team for Software Engineering Environments, Stars Joint Program Office, October 1985.

[Stenning 1981] V. Stenning, T. Froggart, R. Gilbert, and E. Thomas, "The Ada Environment: A Perspective," [@Wasserman 1981], 36-46, 1981.

[STONEMAN] [@DoD 1980].

[Texas Instruments 1985] The APSE Interactive Monitor, Texas Instruments, Slide Presentation to the E&V Team, 5 September 1985.

[UK AES 1986] R.H. Pierce, I. Marshall, and S.D. Blude, "An Introduction to the MoD Ada Evaluation System," Software Sciences Ltd., Report Number 5485, June 1986.

[Wasserman 1981] A.I. Wasserman, *Tutorial: Software Engineering Environments*, IEEE, 1981.

[Weiderman 1986] N. Weiderman, "Evaluation of Ada Environments," Software Engineering Institute, SEI-86-MR-10, September 1986.

[WIS CEG 1985] "WIS Compiler Evaluation Guidelines," GTE Labs, Technical Report, 1985.

[WIS PCEG 1985] G. Gicca and C. Stacey, "Proposed Component Evaluation Guidelines," GTE Government Systems, Technical Report, 16 August 1985.

# APPENDIX B
# ACRONYMS AND ABBREVIATIONS

| | |
|---|---|
| ACEC | Ada Compiler Evaluation Capability |
| ACM | Association for Computing Machinery |
| ACVC | Ada Compiler Validation Capability |
| AES | Ada Evaluation System |
| AFB | Air Force Base |
| AFWAL | Air Force Wright Aeronautical Laboratories |
| AIE | Ada Integrated Environment |
| AJPO | Ada Joint Program Office |
| ALS | Ada Language System |
| ALS/N | Ada Language System/Navy |
| ANNA | Annotation Language for Ada |
| ANSI | American National Standards Institute |
| APSE | Ada Programming Support Environment |
| ARTEWG | Ada RunTime Environment Working Group (SIGAda) |
| ATA | Advanced Tactical Aircraft |
| ATF | Advanced Tactical Fighter |
| AVF | Ada Validation Facility |
| AVO | Ada Validation Office |
| BGT | Benchmark Generator Tool |
| BSI | British Standards Institute (UK) |
| CAIS | Common APSE Interface Set |
| CIVC | CAIS Implementation Validation Capability |
| CLI | Command Language Instruction |
| CPU | Central Processing Unit |
| CSC | Computer Software Component |
| DACS | Data and Analysis Center for Software |
| DoD | Department of Defense |
| DTIC | Defense Technical Information Center |
| ESD | Electronic Systems Division (Air Force) |
| E&V | Evaluation and Validation |
| GB | Guidebook |
| GIT | Georgia Institute of Technology |
| IBM | International Business Machines Corporation |
| IDA | Institute for Defense Analysis |
| IEEE | Institute of Electrical and Electronics Engineers, Inc. |
| IPSE | Integrated Project Support Environment |
| ISTAR | Information Science and Technology Assessment for Research |
| I/O | Input/Output |

| | |
|---|---|
| JIAWG | Joint Integrated Avionics Working Group |
| KAPSE | Kernel Ada Programming Support Environment |
| KIT | KAPSE Interface Team |
| KITIA | KAPSE Interface Team for Industry and Academia |
| LHX | Light Helicopter Experimental |
| LRM | Language Reference Manual [@DoD 1983] |
| MAPSE | Minimal Ada Programming Support Environment |
| MCCS | Mission-Critical Computer System |
| MMI | Man-Machine Interface |
| MoD | Ministry of Defense (UK) |
| NBS | National Bureau of Standards |
| NTIS | National Technical Information Service |
| OCD | Operational Concept Document |
| OS | Operating System |
| PCTE | Portable Common Tool Environment |
| PDL | Program Design Language |
| PIWG | Performance Issues Working Group (SIGAda) |
| RM | Reference Manual |
| SDI | Strategic Defense Initiative |
| SEE | Software Engineering Environment |
| SEI | Software Engineering Institute |
| SIGAda | Special Interest Group for Ada of the Association for Computing Machinery (ACM) |
| STARS | Software Technology for Adaptable, Reliable Systems |
| TASC | The Analytic Sciences Corporation |
| UK | United Kingdom |
| VAX | Virtual Address Extension |
| VMS | Virtual Memory System |
| VSR | Validation Summary Report |
| V&V | Verification and Validation |
| WIS | WWMCCS Information System |
| WWMCCS | WorldWide Military Command and Control System |

# APPENDIX C
# FORMAL GRAMMAR

This appendix specifies sections of the Reference Manual and Guidebook (Reference System) as a formal grammar. The sections include chapters four through seven of the Reference Manual (RM), chapters four through 99 of the Guidebook (GB), all explicit references, the tables of contents, the indices, and the citations. The specification is presented as a partitioned grammar for convenience.

(The grammar is presented in a modified Backus-Naur form. Brackets represent optionality when alone, and may be marked by an asterisk '*' to denote 0-N instances of the production, or by a sharp '#' to denote 1-N instances. Angle brackets denote comments in place of productions which are too elaborate to express here. All terminals of the grammar are expressed as quoted literals, or composite literals based on characters and character strings.)

## C.1 FORMAL REFERENCES

Throughout the Reference System, whenever formal references are made, a single consistent set of grammar rules are used. This includes reference from one volume to the other, reference from one section in a volume to another section in the same document, and reference to documents outside the Reference System.

```
reference_list  ::=  “[” references    [“;”   references    ]* “]”

references      ::=  reference [ “,”  reference ]*

reference       ::=  “@” phrase [ “:” [ phrase ]
                     [ designator_list ]]
                     [ phrase ] designator_list

phrase_list     ::=  phrase [ “,” phrase ]*
```

```
phrase          ::= <text lacking special characters>

designator_list ::= designator [ "," designator ]*

designator      ::= lead "." ] lead [ "." digits ]*

lead            ::= digits ] caps

digits          ::= one_to_nine [zero_to_nine]*

one_to_nine     ::= ('1'—'9')

zero_to_nine    ::= ('0'—'9')

caps            ::= ('A'—'Z')
```

## C.2 FORMAL CHAPTERS

Those chapters of the Reference System which are derived from the classification schema are formally defined here.

### C.2.1 Chapter Components

The following rules define the components which are used to compose formal chapter entries.

```
header          ::= designator phrase

prolog          ::= header purpose primary host
                    [vendors_agents]

purpose         ::= "Purpose:" text

host            ::= "Host/OS:" text

primary         ::= "Primary References:" reference_list

vendors_agents::= "Vendors/Agents:" reference_list
```

```
method           ::= meth_description inputs
                     process outputs

meth_description ::= "Method:" text

inputs           ::= "Inputs:" text

process          ::= "Process:" text

outputs          ::= "Outputs:" text

citations        ::= "Citations:" [citations]#

synopsis_text    ::= "Synopsis:" text

methods          ::= "Methods:" reference_list

text             ::= < prose text >
```

### C.2.2 Chapter Entries

Each numbered section of the formal chapters follows a specific grammar rule. The following rules define the format of each chapter entry.

```
synopsi          ::= header citations synopsis_text [methods]

E&V_technology   ::= prolog method
```

### C.2.3 Formal Chapter Ordering

The formal portion of the GB is found in chapters four through ninety nine.

```
formal_chapters  ::= [ synopsis ]*

                     [ E&V_technology ]*

                     .
                     .
                     .

                     [ E&V_technology ]*
```

## C.3 TABLE OF CONTENTS

The table of contents shares some features with the rest of the formal aspects of the GB.

```
table_of_contents   ::=  [ chapter ]* index
chapter             ::=  designator phrase digits
```

## C.4 CITATIONS

The citations are found in Appendix A, and have a formal structure as defined in the following grammar. The (semantic) form of citation text is taken from the standard for IEEE Software Magazine.

```
citations           ::=  [ citation ]*
citation            ::=  key body "."
key                 ::=  "[" phrase_list "]"
body                ::=  reference_list ] phrase_list
```

# APPENDIX D
# VENDORS AND AGENTS

[ARTEWG]
Mike Kamrad (612) 782-7321
Honeywell Systems and Research Center
M/S MN17-2351
3660 Marshall St., NE
Minneapolis, MN 55418

[BSI, Milton-Keynes, UK] 0908-220908 x2313
J.B. Souter
BSI Quality Assurance
P.O. Box 375
Milton Keynes MK14 6LL
UK

[Cambridge University Press]
Cambridge University Press
32 East 57th Street
New York, NY 10022

[Defense Technical Information Center]
Defense Technical Information Center (703) 274-7633
Cameron Station
Alexandria, VA 22314

[DACS]
Data & Analysis Center for Software (315) 336-0937
RADC/COED
Griffiss AFB, NY 13441

[E&V Team]

Mr. Raymond Szymanski (513) 255-2446
AFWAL/AAAF -6730
Wright-Patterson AFB AV 785-2446
OH 45433-6543 -6730
MILNET: SZYMANSK@AJPO.SEI.CMU.EDU

[GIT]

Georgia Institute of Technology (404) 894-3180
Software Engineering Research Center
Atlanta, GA 30332-0280

[MITRE]

MITRE Corporation (703) 883-6000
Civil Systems Division
7525 Colshire Drive
McLean, VA 22102-3481

[National Technical Information Service]

National Technical Information Service (703) 487-4650
U.S. Department of Commerce
5285 Port Royal Road
Springfield, VA 22161

[PIWG]

Dan M. Roy (301) 464-6800
Ford Aerospace
7401-D Forbes Blvd.
Seabrook, MD 20706

[SofTech, Inc.]

Teresa L. Banks (513) 429-3241
SofTech, Inc.
3100 Presidential Drive
Fairborn, OH 45324-2039
ARPANET: HILLM@WPAFB-JALCF

[SEI]

Software Engineering Institute (412) 268-7700
Canegie Mellon University
Pittsburg, PA 15213

[UMich]

Russel M. Clapp, Louis Duchesneau, Richard A. Volz, (313) 764-1817
Trevor N. Mudge, and Timothy Schultze
The Robotics Research Laboratory
The University of Michigan
Ann Arbor, MI 48109