

OTM FILE COPY

WRDC-TR-89-1135

AD-A214 167

E&V REFERENCE MANUAL, VERSION 2.0



Bard S. Crawford
Peter G. Clark

The Analytic Sciences Corp.
55 Walker's Brook Drive
Reading, MA 01856

October 1989

Interim Report for Period Oct 88 to Sept 89

Approved for public release; distribution unlimited.

DTIC
ELECTE
NOV 07 1989
S E D
BT

AVIONICS LABORATORY
WRIGHT RESEARCH DEVELOPMENT CENTER
AIR FORCE SYSTEMS COMMAND
WRIGHT-PATTERSON AIR FORCE BASE, OHIO 45433-6543

NOTICE

When Government drawings, specifications, or other data are used for any purpose other than in connection with a definitely related Government procurement operation, the United States Government thereby incurs no responsibility nor any obligation whatsoever; and the fact that the government may have formulated, furnished, or in any way supplied the said drawings, specifications, or other data, is not to be regarded by implication or otherwise as in any manner licensing the holder or any other person or corporation, or conveying any rights or permission to manufacture, use, or sell any patented invention that may in any way be related thereto.

This report has been reviewed by the office of Public Affairs, (ASD/PA) and is releasable to the National Technical Information Service (NTIS). At NTIS, it will be available to the general public including foreign nations.

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

Raymond Szymanski

Raymond Szymanski
Program Manager

3 October 1989
Date

FOR THE COMMANDER:

Charles H. Krueger Jr

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

9 8 OCT 1989
Date

If your address has changed, if you wish to be removed from our mailing list, or if the addressee is no longer employed by your organization, please notify WRDC/AAAF-3 WPAFB, OH 45433 to help us maintain a current mailing list.

Copies of this report should not be returned unless required by security considerations, contractual obligation, or notice on a specific document.

89 11 07 081

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE

REPORT DOCUMENTATION PAGE				Form Approved OMB No. 0704-0188	
1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED			1b. RESTRICTIVE MARKINGS		
2a. SECURITY CLASSIFICATION AUTHORITY			3. DISTRIBUTION/AVAILABILITY OF REPORT		
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE			Approved for Public Release Distribution Unlimited		
4. PERFORMING ORGANIZATION REPORT NUMBER(S) TASC No. TR-5234-3			5. MONITORING ORGANIZATION REPORT NUMBER(S) WRDC-TR-89-1135		
6a. NAME OF PERFORMING ORGANIZATION The Analytic Sciences Corporation		6b. OFFICE SYMBOL (if applicable)	7a. NAME OF MONITORING ORGANIZATION Avionics Laboratory (WRDC/AAAF) Wright Research Development Center		
6c. ADDRESS (City, State, and ZIP Code) 55 Walker's Brook Drive Reading, Mass. 01867			7b. ADDRESS (City, State, and ZIP Code) Wright-Patterson AFB, OH 45433-6543		
8a. NAME OF FUNDING/SPONSORING ORGANIZATION Ada Joint Program Office		8b. OFFICE SYMBOL (if applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER F33615-85-C-1812		
8c. ADDRESS (City, State, and ZIP Code) Room 3E114 (1211 S. Fern St) The Pentagon Washington, D.C. 20301-3080			10. SOURCE OF FUNDING NUMBERS		
			PROGRAM ELEMENT NO. 63756D 63226	PROJECT NO. 2853	TASK NO. 01
					WORK UNIT ACCESSION NO. 01
11. TITLE (Include Security Classification) E&V Reference Manual, Version 2.0					
12. PERSONAL AUTHOR(S) Crawford, Bard S., Clark, Peter G.					
13a. TYPE OF REPORT Interim Technical		13b. TIME COVERED FROM 21OCT88 TO 30SEP89		14. DATE OF REPORT (Year, Month, Day) October 1989	
				15. PAGE COUNT 357	
16. SUPPLEMENTARY NOTATION A companion document titled "E&V Guidebook" is being released concurrently.					
17. COSATI CODES			18. SUBJECT TERMS (Continue on reverse if necessary and identify by block number)		
FIELD	GROUP	SUB-GROUP	EVALUATION		
09	02		VALIDATION		
			Ada PROGRAMMING SUPPORT ENVIRONMENTS (APSEs)		
19. ABSTRACT (Continue on reverse if necessary and identify by block number)					
The Ada community, including government, industry, and academic personnel, needs the capability to assess Ada Programming Support Environments (APSEs) and their components and to determine their conformance to applicable standards (e.g., DoD-STD-1838, the CAIS standard). The technology required to fully satisfy this need is extensive and largely unavailable; it cannot be acquired by a single government-sponsored, professional society-sponsored, or private effort. The purpose of the APSE Evaluation and Validation (E&V) Task is to provide a focal point for addressing the need by: (1) Identifying and defining specific technology requirements, (2) Developing selected elements of the required technology, (3) Encouraging others to develop some elements, and (4) Collecting information describing elements which already exist. (See Reverse)					
20. DISTRIBUTION/AVAILABILITY OF ABSTRACT			21. ABSTRACT SECURITY CLASSIFICATION		
☒ UNCLASSIFIED/UNLIMITED ☒ SAME AS RPT. ☐ DTIC USERS			Unclassified		
22a. NAME OF RESPONSIBLE INDIVIDUAL RAYMOND SZYMANSKI			22b. TELEPHONE (Include Area Code) (513) 255-3947		22c. OFFICE SYMBOL WRDC/AAAF-3

19. Continued

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

The purpose of the E&V Reference Manual (this document) is to provide information that will help users to:

- (1) Gain an overall understanding of APSEs and approaches to their assessment,
- (2) Find useful reference information (e.g., definitions) about specific elements and relationships between elements, and
- (3) Find criteria and metrics for assessing tools and APSEs, and techniques for performing such assessment.

The latter are to be found (or referenced) in a companion document called the E&V Guidebook.

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



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 Reference Manual (this document) is to provide information that will help users to:

- (1) Gain an overall understanding of APSEs and approaches to their assessment;
- (2) Find useful reference information (e.g., definitions) about specific elements and relationships between elements; and
- (3) Find criteria and metrics for assessing tools and APSEs, and techniques for performing such assessments.

The last are to be found (or referenced) in a companion document called the E&V Guidebook.

Chapters 1 through 3 provide a general introduction to the document and to the issue of assessing APSEs as a whole.

E&V Reference Manual, Version 2.0

Chapter 4 and later chapters are "formal chapters" built around a standard format and formal grammar. Each of the formal chapters corresponds to one index of an overall E&V Classification Schema. The schema adopts a relational model of the subject and process of E&V. This model will allow the user to arrive at E&V techniques through many different paths, and provides a means to extract useful information along the way.

Yearly updates and extensions to this manual 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, WRDC/AAAF, Wright Patterson AFB, OH 45433-6543.

TABLE OF CONTENTS

	Page
EXECUTIVE SUMMARY	ES-1
List of Figures	xiii
List of Tables	xiv
1. INTRODUCTION	1-1
1.1 Purpose of the Manual	1-1
1.2 Users of the Manual	1-4
1.3 The Need for E&V Technology	1-5
1.4 Background	1-6
1.5 Organization of the Manual	1-7
2. USE OF THE REFERENCE SYSTEM	2-1
2.1 System Organization	2-1
2.2 Description: Direct Reference	2-5
2.3 Cross Reference	2-6
2.4 Guidebook Reference	2-7
2.5 Reference Framework	2-10
3. WHOLE APSE ASSESSMENT ISSUES	3-1
3.1 APSE Definitions and Alternative Names	3-1
3.2 Views of an APSE	3-3
3.2.1 APSE Viewed as a Collection of Tools	3-4
3.2.2 APSE Viewed as a Methodology-Support System	3-4
3.2.3 APSE Viewed as an Information Management System	3-4
3.2.4 APSE Viewed as a User-Oriented, Interactive System	3-6
3.2.5 APSE Viewed as a Knowledge-Based Expert System	3-6
3.2.6 APSE Viewed as a Stable Framework	3-6
3.3 Key Attributes of Whole APSEs	3-8
3.3.1 Performance Attributes	3-8
3.3.2 Design Attributes	3-10
3.3.3 Adaptation Attributes	3-10
3.4 Approaches to Whole-APSE E&V	3-11
3.4.1 Benchmarks and Test Suites	3-11
3.4.2 Checklists and Questionnaires	3-12
3.4.3 Structured Experiments	3-12
3.4.4 Decision Aids	3-12
4. LIFE CYCLE ACTIVITIES	4-1
4.1 System Concepts	4-4
4.1.1 System Development Management	4-4
4.1.2 System Engineering	4-4
4.1.3 Formal Qualification Testing	4-4

TABLE OF CONTENTS (Continued)

	Page
4.1.4 System Product Evaluations	4-5
4.1.5 Configuration Management	4-5
4.1.6 Transitioning to System Support	4-5
4.2 System Requirements Analysis/Design	4-6
4.2.1 System Development Management	4-6
4.2.2 System Engineering	4-6
4.2.3 Formal Qualification Testing	4-7
4.2.4 System Product Evaluations	4-7
4.2.5 Configuration Management	4-7
4.2.6 Transitioning to System Support	4-7
4.3 Software Requirements Analysis	4-8
4.3.1 System Development Management	4-8
4.3.2 Software Engineering	4-8
4.3.3 Formal Qualification Testing	4-9
4.3.4 Software Product Evaluations	4-9
4.3.5 Configuration Management	4-9
4.3.6 Transitioning to Software Support	4-9
4.4 Preliminary Design	4-10
4.4.1 Software Development Management	4-10
4.4.2 Software Engineering	4-10
4.4.3 Formal Qualification Testing	4-11
4.4.4 Software Product Evaluations	4-11
4.4.5 Configuration Management	4-11
4.4.6 Transitioning to Software Support	4-12
4.5 Detailed Design	4-13
4.5.1 Software Development Management	4-13
4.5.2 Software Engineering	4-14
4.5.3 Formal Qualification Testing	4-15
4.5.4 Software Product Evaluations	4-15
4.5.5 Configuration Management	4-15
4.5.6 Transitioning to Software Support	4-16
4.6 Coding and CSU Testing	4-17
4.6.1 Software Development Management	4-17
4.6.2 Software Engineering	4-17
4.6.3 Formal Qualification Testing	4-19
4.6.4 Software Product Evaluations	4-19
4.6.5 Configuration Management	4-19
4.6.6 Transitioning to Software Support	4-20
4.7 CSC Integration and Testing	4-21
4.7.1 Software Development Management	4-21
4.7.2 Software Engineering	4-21
4.7.3 Formal Qualification Testing	4-23
4.7.4 Software Product Evaluations	4-24
4.7.5 Configuration Management	4-24
4.7.6 Transitioning to Software Support	4-24

TABLE OF CONTENTS (Continued)

	Page
4.8 CSCI Testing	4-25
4.8.1 Software Development Management	4-25
4.8.2 Software Engineering	4-25
4.8.3 Formal Qualification Testing	4-27
4.8.4 Software Product Evaluations	4-27
4.8.5 Configuration Management	4-28
4.8.6 Transitioning to Software Support	4-28
4.9 System Integration and Testing	4-29
4.9.1 Software Development Management	4-29
4.9.2 Software Engineering	4-29
4.9.3 Formal Qualification Testing	4-31
4.9.4 Software Product Evaluations	4-31
4.9.5 Configuration Management	4-32
4.9.6 Transitioning to System Support	4-32
4.10 Operational Testing and Evaluation	4-33
4.10.1 Software Development Management	4-33
4.10.2 Software Engineering	4-33
4.10.3 Formal Qualification Testing	4-35
4.10.4 Software Product Evaluations	4-35
4.10.5 Configuration Management	4-36
4.10.6 Transitioning to System Support	4-36
4.11 Change Requirements	4-37
4.11.1 System Development Management	4-37
4.11.2 System Engineering	4-37
4.11.3 Formal Qualification Testing	4-37
4.11.4 System Product Evaluations	4-38
4.11.5 Configuration Management	4-38
4.11.6 Transitioning to System Support	4-38
4.12 Global	4-39
4.12.1 System Development Management	4-39
4.12.2 System Engineering	4-39
4.12.3 Formal Qualification Testing	4-40
4.12.4 System Product Evaluations	4-40
4.12.5 Configuration Management	4-40
4.12.6 Transitioning to System Support	4-40
5. APSE TOOL CATEGORIES	5-1
5.1 Computer Management System	5-3
5.1.1 Command Language Interface	5-3
5.1.2 Archive, Backup, and Retrieval System	5-3
5.1.3 Security System	5-3
5.1.4 Job Scheduler	5-4
5.1.5 Resource Controller	5-4
5.1.6 Import/Export System	5-4
5.1.7 On-Line Assistance	5-5

TABLE OF CONTENTS (Continued)

	Page
5.1.8 Input and Output Services	5-5
5.1.9 Performance Monitor	5-5
5.2 Information Management System	5-6
5.2.1 File Manager	5-6
5.2.2 Database Manager	5-6
5.3 Tool Support Components	5-7
5.3.1 Virtual Operating System	5-7
5.3.2 Window Manager	5-7
5.3.3 Language Bindings to Standard Interface Specification Implementations	5-7
5.3.4 I/O Pipes	5-8
5.3.5 RAM Cache	5-8
5.4 Distributed APSE Support	5-9
5.5 Project Management System	5-10
5.5.1 Cost Estimator	5-10
5.5.2 Size Estimator	5-10
5.5.3 Scheduler	5-10
5.5.4 Work Breakdown Structure Editor	5-11
5.5.5 Resource Estimator	5-11
5.5.6 Tracking	5-11
5.5.7 Problem Report Analyzer	5-11
5.5.8 Change Impact Analyzer	5-12
5.5.9 Analysis and Reporting	5-12
5.6 Desktop System	5-13
5.6.1 Spreadsheet	5-13
5.6.2 Calculator	5-13
5.6.3 Address/Phone Book	5-13
5.6.4 Electronic Mail	5-14
5.6.5 Electronic Conferencing	5-14
5.6.6 Calendar	5-14
5.6.7 Dictionary/Thesaurus	5-14
5.7 Configuration Management System	5-15
5.7.1 Configuration Identification	5-15
5.7.2 Configuration Control	5-15
5.7.3 Configuration Status Accounting	5-15
5.7.4 Version Control	5-16
5.7.5 History	5-16
5.8 Document Generation System	5-17
5.8.1 Document Manager	5-17
5.8.2 Word Processor	5-17
5.8.3 Spelling Checker	5-17
5.8.4 Graphics Generator	5-18
5.8.5 Formatter	5-18
5.9 Requirements/Design Support	5-19
5.9.1 Definition Language Processor	5-19
5.9.2 Specification Language Processor	5-19

TABLE OF CONTENTS (Continued)

	Page
5.9.3 Enterprise Model	5-19
5.9.4 Data Model/Dictionary	5-20
5.9.5 Process Model	5-20
5.9.6 Simulators	5-21
5.9.7 Prototyping Tools	5-21
5.9.8 Design Library Manager	5-21
5.10 Distributed Systems Development and Runtime Support	5-22
5.11 Implementation Support	5-23
5.11.1 Applications (Code) Generator	5-23
5.11.2 Database Schema Generator	5-23
5.11.3 Report Generator	5-23
5.11.4 Screen Generator	5-24
5.11.5 Syntax-Directed (Language-Sensitive) Editor	5-24
5.11.6 Data Definition Language Processor	5-24
5.11.7 Data Manipulation Language Processor	5-24
5.11.8 Reusable Components Library	5-25
5.12 Compilation System	5-26
5.12.1 Translating Editor	5-26
5.12.2 Compiler (Front End)	5-26
5.12.3 Compiler (Code Generator — Back End)	5-26
5.12.4 Interpreter	5-27
5.12.5 Program Library Manager	5-27
5.12.6 Runtime System	5-27
5.13 Target Code Generation Aids and Analysis Toolset	5-28
5.13.1 Host-Target System Cross-Assembler	5-28
5.13.2 Host-Based Target Linker	5-28
5.13.3 Host-Based Target Loader	5-28
5.13.4 Host-Based Target System Instruction-Level Simulator	5-29
5.13.5 Host-Based Target System Instruction-Level Emulator	5-29
5.13.6 Host-Based Target Code Symbolic Debugger	5-29
5.13.7 Host-to-Target Downloader	5-29
5.13.8 Target-to-Host Uploader	5-29
5.14 Test System	5-30
5.14.1 Static Analyzers	5-30
5.14.2 Tool Building Services	5-31
5.14.3 Test Building Services	5-31
5.14.4 Test Description and Preparation Services	5-31
5.14.5 Test Execution Services	5-32
5.14.6 Test Analysis Services	5-32
5.14.7 Decision Support Services	5-32
6. ATTRIBUTES	6-1
6.1 Performance	6-9
6.1.1 Efficiency	6-10
6.1.2 Integrity	6-11
6.1.3 Reliability	6-12

TABLE OF CONTENTS (Continued)

		Page
	6.1.4 Survivability	6-13
	6.1.5 Usability	6-14
6.2	Design	6-15
	6.2.1 Correctness	6-16
	6.2.2 Maintainability	6-17
	6.2.3 Verifiability, Testability	6-18
6.3	Adaptation	6-19
	6.3.1 Expandability, Flexibility	6-20
	6.3.2 Interoperability	6-21
	6.3.3 Reusability	6-22
	6.3.4 Transportability	6-23
6.4	Software-Oriented Criteria	6-24
	6.4.1 Accuracy	6-24
	6.4.2 Anomaly Management, Fault or Error Tolerance, Robustness	6-25
	6.4.3 Application Independence	6-26
	6.4.4 Augmentability	6-27
	6.4.5 Autonomy	6-28
	6.4.6 Capacity	6-29
	6.4.7 Commonality (Data and Communication)	6-30
	6.4.8 Communication Effectiveness	6-31
	6.4.9 Completeness	6-32
	6.4.10 Consistency	6-35
	6.4.11 Cost	6-36
	6.4.12 Distributedness	6-37
	6.4.13 Document Accessibility	6-38
	6.4.14 Functional Overlap	6-39
	6.4.15 Functional Scope	6-40
	6.4.16 Generality	6-41
	6.4.17 Granularity	6-42
	6.4.18 Maturity	6-43
	6.4.19 Modularity	6-44
	6.4.20 Operability, Communicativeness	6-45
	6.4.21 Power	6-47
	6.4.22 Processing (Execution) Effectiveness	6-49
	6.4.23 Proprietary Rights	6-51
	6.4.24 Reconfigurability	6-52
	6.4.25 Rehostability	6-53
	6.4.26 Required Configuration	6-54
	6.4.27 Retargetability	6-55
	6.4.28 Self-Descriptiveness	6-56
	6.4.29 Simplicity	6-57
	6.4.30 Software Production Vehicle(s)	6-58
	6.4.31 Storage Effectiveness	6-59
	6.4.32 System Accessibility	6-61
	6.4.33 System Clarity	6-62
	6.4.34 System Compatibility	6-63

TABLE OF CONTENTS (Continued)

	Page
6.4.35 Traceability	6-64
6.4.36 Training	6-65
6.4.37 Virtuality	6-66
6.4.38 Visibility, Test Availability	6-67
7. FUNCTIONS	7-1
7.1 Transformation	7-3
7.1.1 Editing	7-3
7.1.1.1 Text Editing	7-4
7.1.1.2 Data Editing	7-5
7.1.1.3 Graphics Editing	7-6
7.1.2 Formatting	7-7
7.1.2.1 MIL-STD Format	7-7
7.1.2.2 Table of Contents	7-8
7.1.2.3 Predefined and User-Defined Forms	7-9
7.1.3 On-Line Assistance Processing	7-10
7.1.4 Sort/Merge	7-10
7.1.5 Graphics Generation	7-11
7.1.6 Translation	7-11
7.1.6.1 Systems Requirements	7-12
7.1.6.2 Software Requirements	7-13
7.1.6.3 Requirements to Natural Language	7-14
7.1.6.4 Preliminary Design	7-15
7.1.6.5 Detailed Design	7-16
7.1.6.6 Assembling	7-17
7.1.6.7 Compilation	7-18
7.1.6.8 Conversion	7-21
7.1.6.9 Macro Expansion	7-22
7.1.6.10 Structure Preprocessing	7-23
7.1.6.11 Body Stub Generation	7-24
7.1.6.12 Preamble Generation	7-25
7.1.6.13 Linking/Loading	7-26
7.1.6.14 Interpretation	7-27
7.1.6.15 Software and System Test Communications	7-28
7.1.6.16 Compression	7-29
7.1.6.17 Expansion	7-30
7.1.6.18 Encryption	7-31
7.1.6.19 Decryption	7-32
7.1.7 Synthesis	7-33
7.1.7.1 Design Generation	7-33
7.1.7.2 Requirements Reconstruction	7-34
7.1.7.3 Program Generation	7-35
7.1.7.4 Source Reconstruction	7-36
7.1.7.5 Decompilation	7-37
7.1.7.6 Disassembling	7-38
7.1.7.7 Test Harness Generation	7-39

TABLE OF CONTENTS (Continued)

		Page
7.2	Management	7-40
7.2.1	Information Management	7-40
7.2.1.1	Database (Object) Management	7-41
7.2.1.2	Documentation Management	7-42
7.2.1.3	File Management	7-43
7.2.1.4	Electronic Mail	7-44
7.2.1.5	Electronic Conferencing	7-45
7.2.1.6	Specification Management	7-46
7.2.1.7	Program Library Management	7-47
7.2.1.8	Test Data Management	7-48
7.2.1.9	Evaluation Results Management	7-49
7.2.1.10	Performance Monitoring	7-50
7.2.1.11	Data and Error Logging	7-51
7.2.1.12	Status Display	7-52
7.2.2	Project Management	7-53
7.2.2.1	Cost Estimation	7-53
7.2.2.2	Quality Specification	7-54
7.2.2.3	Scheduling	7-55
7.2.2.4	Work Breakdown Structure	7-56
7.2.2.5	Resource Estimation	7-57
7.2.2.6	Tracking	7-58
7.2.2.7	Configuration Management	7-59
7.2.2.8	Quality Assessment	7-61
7.2.2.9	Risk Analysis	7-62
7.2.3	Computer System Management	7-63
7.2.3.1	Command Language Processing	7-64
7.2.3.2	Input/Output Support	7-65
7.2.3.3	Kernel	7-66
7.2.3.4	Math/Statistics	7-67
7.2.3.5	Runtime Environment	7-68
7.2.3.6	Import/Export	7-70
7.2.3.7	Access Control	7-71
7.2.3.8	Job Scheduling	7-72
7.2.3.9	Resource Management	7-73
7.3	Analysis	7-74
7.3.1	Static Analysis	7-74
7.3.1.1	Comparison	7-74
7.3.1.2	Spelling Checking	7-75
7.3.1.3	Data Flow Analysis	7-76
7.3.1.4	Functional Analysis	7-77
7.3.1.5	Interface Analysis	7-78
7.3.1.6	Traceability Analysis	7-79
7.3.1.7	Testability Analysis	7-80
7.3.1.8	Test Condition Analysis	7-81
7.3.1.9	Quality Measurement	7-82
7.3.1.10	Complexity Measurement	7-83

TABLE OF CONTENTS (Continued)

		Page
	7.3.1.11 Correctness Checking	7-84
	7.3.1.12 Completeness Checking	7-85
	7.3.1.13 Consistency Checking	7-86
	7.3.1.14 Reusability Analysis	7-87
	7.3.1.15 Syntax and Semantics Checking	7-88
	7.3.1.16 Reachability Analysis	7-89
	7.3.1.17 Cross Reference	7-90
	7.3.1.18 Maintainability Analysis	7-91
	7.3.1.19 Invocation Analysis	7-92
	7.3.1.20 Scanning	7-93
	7.3.1.21 Structured Walkthrough	7-94
	7.3.1.22 Auditing	7-95
	7.3.1.23 Error Checking	7-96
	7.3.1.24 Statistical Analysis	7-97
	7.3.1.25 Statistical Profiling	7-98
	7.3.1.26 Structure Checking	7-99
	7.3.1.27 Type Analysis	7-100
	7.3.1.28 Units Analysis	7-101
	7.3.1.29 I/O Specification Analysis	7-102
	7.3.1.30 Sizing Analysis	7-103
	7.3.1.31 Data Reduction and Analysis	7-104
	7.3.1.32 Random Test Generation	7-105
7.3.2	Dynamic Analysis	7-106
	7.3.2.1 Requirements Simulation	7-106
	7.3.2.2 Requirements Prototyping	7-107
	7.3.2.3 Simulation and Modeling	7-108
	7.3.2.4 Design Prototyping	7-109
	7.3.2.5 Debugging	7-110
	7.3.2.6 Executable Assertion Checking	7-111
	7.3.2.7 Constraint Evaluation (Contention)	7-112
	7.3.2.8 Coverage/Frequency Analysis	7-113
	7.3.2.9 Mutation Analysis	7-114
	7.3.2.10 Symbolic Execution	7-115
	7.3.2.11 Regression Testing	7-116
	7.3.2.12 Resource Utilization	7-117
	7.3.2.13 Emulation	7-118
	7.3.2.14 Timing Analysis	7-119
	7.3.2.15 Tuning	7-120
	7.3.2.16 Reliability Analysis	7-121
	7.3.2.17 Real-Time Analysis	7-122
	7.3.2.18 Path and Domain Selection	7-123
7.3.3	Formal Verification	7-124
7.3.4	Problem Report Analysis	7-125
7.3.5	Change Request Analysis	7-126
	7.3.5.1 Change Impact Analysis	7-126

TABLE OF CONTENTS (Continued)

	Page
APPENDIX A CITATIONS	A-1
APPENDIX B GLOSSARY	B-1
B.1 Acronyms and Abbreviations	B-1
APPENDIX C FORMAL GRAMMAR	C-1
C.1 Formal References	C-1
C.2 Indexes	C-2
C.3 Formal Chapters	C-2
C.3.1 Chapter Components	C-2
C.3.2 Chapter Entries	C-3
C.3.3 Formal Chapter Ordering	C-4
C.4 Table of Contents	C-5
C.5 Citations	C-5
INDEX	index-1

LIST OF FIGURES

Figure	Page
1.1-1 Uses of the Reference System	1-2
1.1-2 Examples of Reference Manual Information	1-3
1.1-3 Examples of Guidebook Information	1-3
2.1-1 Reference Manual Organization	2-3
2.1-2 Reference Material Indexes	2-3
2.1-3 Text Frames	2-4
2.2-1 Sample Attributes Index Frame	2-5
2.2-2 Example of Direct Reference: User A	2-6
2.3-1 Sample Life Cycle Activity Text Frame	2-7
2.3-2 Example of Cross Reference: User B	2-8
2.4-1 Sample Functions Index Frame	2-9
2.4-2 Example of Guidebook Reference: User C	2-10
2.5-1 The Schema as Framework for the Reference Manual	2-11
2.5-2 Organization of the Guidebook	2-13
3.2-1 Stoneman APSE Architecture	3-5
3.2-2 STARS Environment Architecture	3-7
4-1 Life Cycle Activity Relationships	4-3
5-1 Tool Relationships	5-1
6-1 Attribute Relationships	6-8
7-1 Function Relationships	7-2

LIST OF TABLES

Table	Page
1.2-1 Reference Manual Users	1-4
4-1 Life Cycle Activities Mapping	4-3
6-1 Top Level Attribute Hierarchy	6-2
6-2 Complete Attribute Hierarchy	6-3
6-3 Complementary Software Quality Factors	6-5
6-4 Beneficial and Adverse Effects of Criteria on Software Quality Factors	6-7

1. INTRODUCTION

1.1 PURPOSE OF THE MANUAL

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 purpose of the Reference Manual is to provide a collection of information to support a variety of users. The collection is organized in accordance with a Classification Schema described in Chapter 2. It should help users to:

- Gain an overall understanding of APSEs and approaches to the assessment of APSE performance, quality and conformance to applicable standards.
- Find useful reference information, such as definitions of specific elements of the Classification Schema, and relationships between elements.
- Find criteria and metrics for assessing specific components, combinations of components and "whole APSEs," and locate relevant E&V techniques.

The Reference Manual includes many "pointers" to sections in the Guidebook and other documents which describe E&V techniques. Figure 1.1-1 illustrates the relationship between the Reference Manual and the Guidebook. Figures 1.1-2 and 1.1-3 illustrate the types of information to be extracted from each document. Chapter 2 provides a more detailed description of the structure and uses of the Reference System.

E&V Reference Manual, Version 2.0

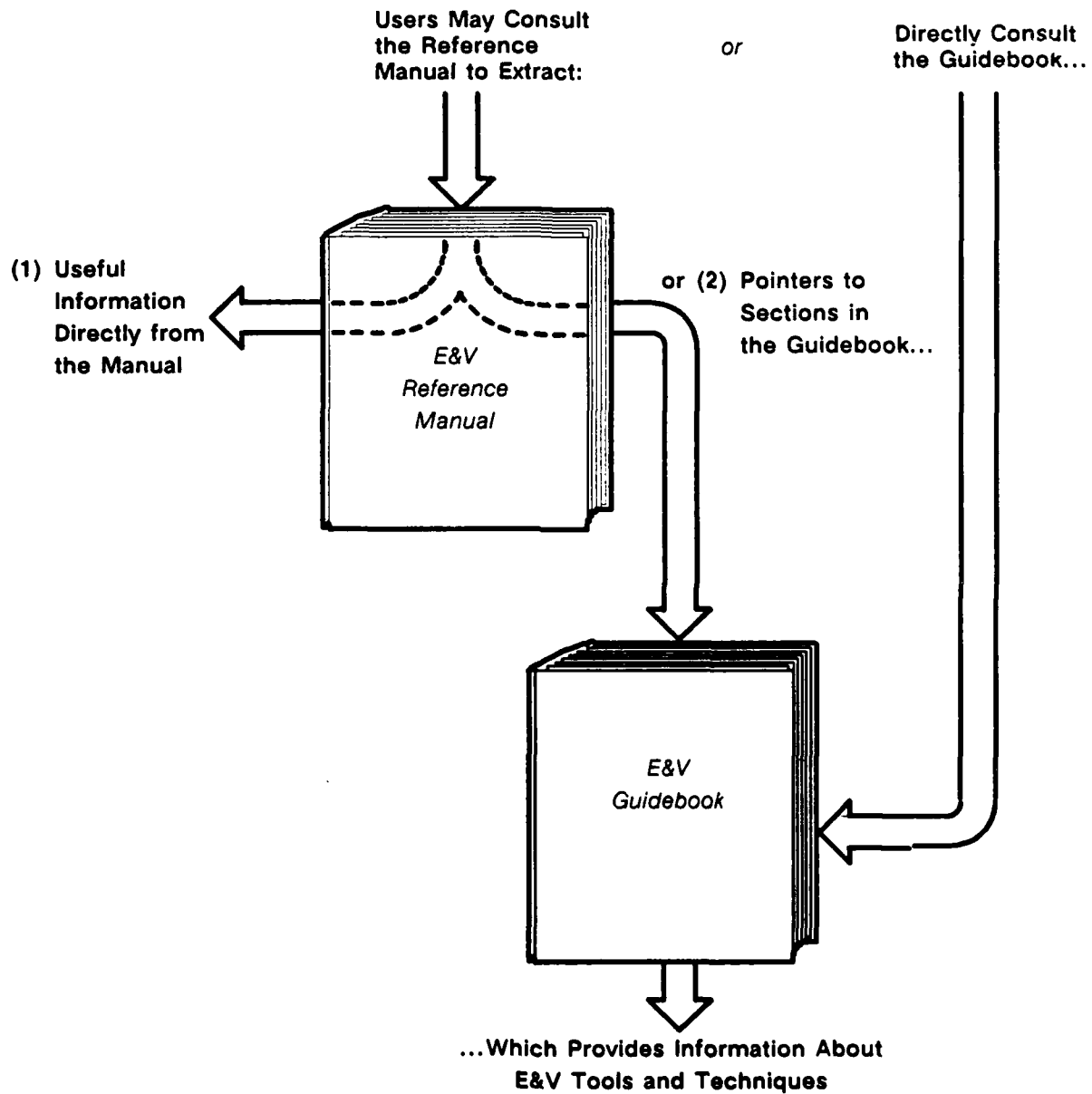
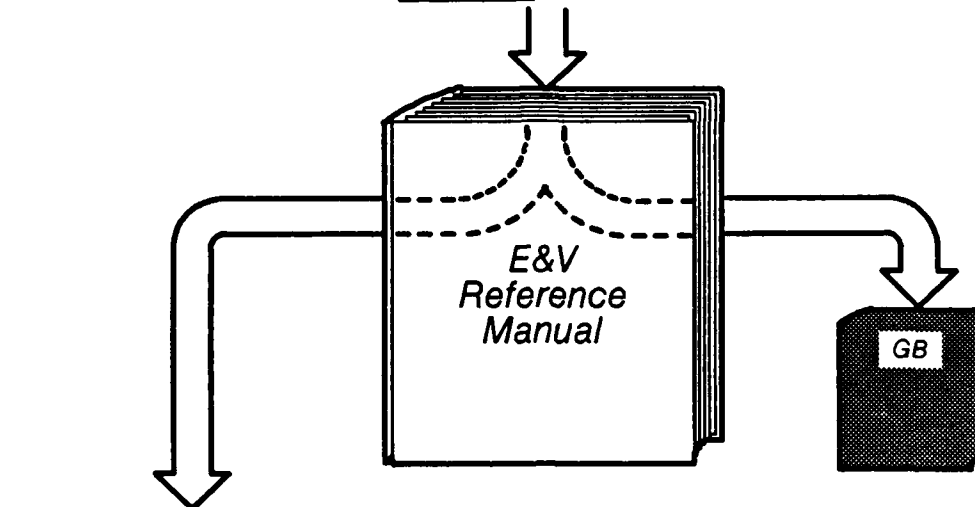


Figure 1.1-1 Uses of the Reference System

Users consult Indexes in the Reference Manual

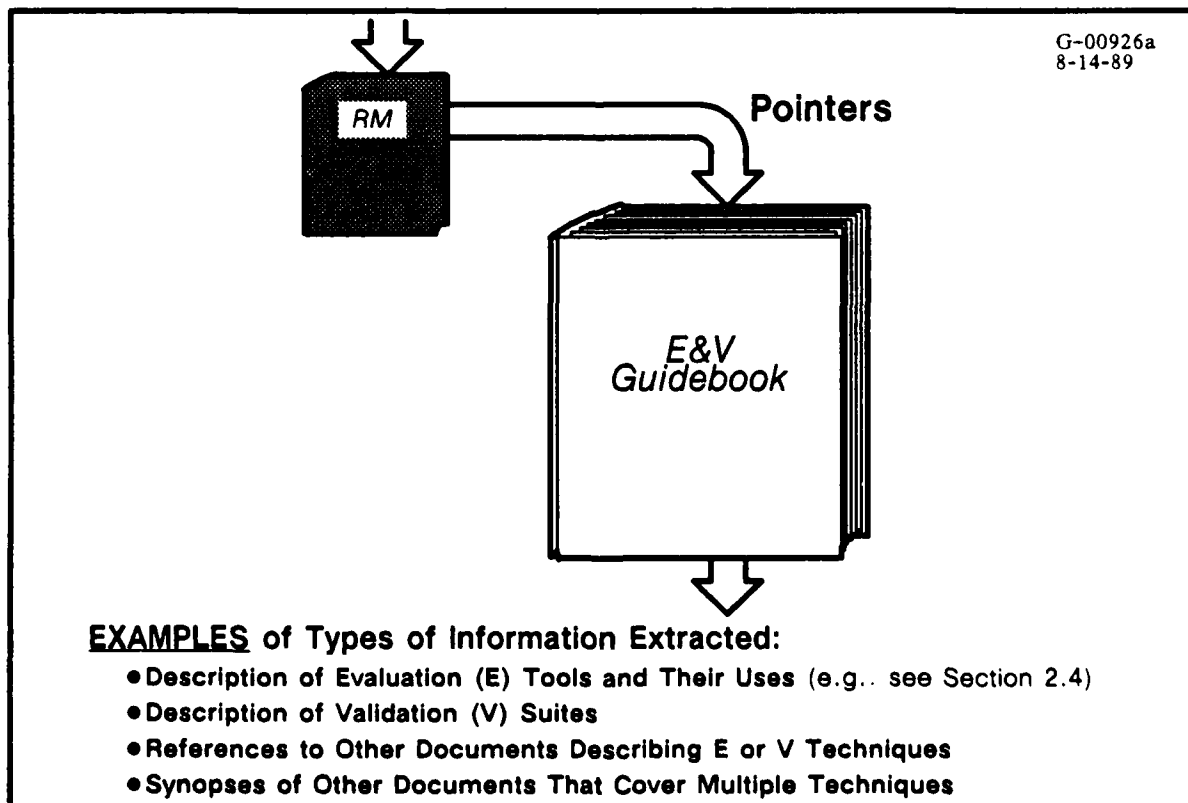
G-00925b
8-14-89



EXAMPLES of Types of Information Extracted:

- **Organization of The Indexes**
Hierarchical Structure and a Numbering System
 (e.g., The Function Index . . . see Fig. 2.1-2)
- **Names and Descriptions of Specific Elements within the Indexes**
 (e.g., The Compilation Function . . . see Figs. 2.1-2 and 2.1-3)
- **Cross References Between Elements of the Indexes**
 (e.g., The Function "Compilation" is Cross Referenced to
 Activities and Tools . . . see Fig. 2.4-1)

Figure 1.1-2 Examples of Reference Manual Information



G-00926a
8-14-89

EXAMPLES of Types of Information Extracted:

- **Description of Evaluation (E) Tools and Their Uses** (e.g., see Section 2.4)
- **Description of Validation (V) Suites**
- **References to Other Documents Describing E or V Techniques**
- **Synopses of Other Documents That Cover Multiple Techniques**

Figure 1.1-3 Examples of Guidebook Information

1.2 USERS OF THE MANUAL

Classes of people who are expected to be users of this manual are listed in Table 1.2-1. They are described in terms of their relationships to deliverable software, tools, APSEs, and APSE E&V technology. They may be associated with Government, industry, or academia. The table was adapted from material in the report of the E&V Workshop [at E&V Workshop 1984].*

While the manual is designed to be of use to people of all the classes listed, the primary users are expected to be the APSE/tool users and the E&V technology users. These are people with highly technical backgrounds and technical/managerial interests in evaluating and selecting tools and APSEs. This expectation has strongly influenced the structure and style of the manual. The primary users are likely to consult both the Reference Manual and the Guidebook. Many of the other classes of users listed in Table 1.2-1 are likely to consult the Reference Manual only.

Table 1.2-1 Reference Manual Users

CLASS	DESCRIPTION
Software Acquisition Personnel	Government program officers and commercial program managers who let contracts for software development
APSE/Tool Users	Project managers, librarians, system engineers, software engineers
APSE/Tool Builders	Environment/tool designers, implementers, managers, marketing personnel
E&V Technology Users	Government, commercial and university personnel applying E&V technology
E&V Technology Builders	Anyone developing E&V assessment techniques (E&V Task contractors, etc)
Investors	Anyone funding development or use of E&V technology (Congress, AJPO, corporations, etc.)

*The format used for references is associated with the "formal grammar" used beginning with Chapter 4. See further explanations in Appendix C.

1.3 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. 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 the Ada language, a relatively new implementation language. E&V technology provides methods and techniques to overcome these difficulties and provides a basis for determining performance and other attributes of APSEs.

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

1.4 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 Team 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. 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 manual. The purpose of the second and third contracts is to create two specific elements (ACEC and CIVC) of the needed E&V technology.

1.5 ORGANIZATION OF THE MANUAL

Chapter 2 discusses the structure of the E&V Reference System (Reference Manual plus Guidebook) and the Classification Schema upon which that structure is based. Specific directions as to how to use the manual are also included.

Chapter 3 provides a general discussion of "whole APSE" issues, in which an APSE is viewed as more than the sum of its parts. Key whole-APSE attributes are discussed along with general approaches to whole-APSE assessment.

Chapter 4 and subsequent chapters are "formal chapters" built around a standard format and formal grammar. Each of the formal chapters corresponds to one index of the Classification Schema. The structure and use of these chapters are the focus of the material found in Chapter 2.

The appendices include a description of the formal grammar, a glossary of acronyms and abbreviations, a document citation list, and a composite index.

2. USE OF THE REFERENCE SYSTEM

This chapter provides a step-by-step explanation of how to use the E&V Reference System and the Classification Schema upon which the system is based. Section 2.1 describes the organization of the material. Sections 2.2, 2.3, and 2.4 contain illustrations of uses of the system, presented in increasing levels of sophistication. Section 2.5 provides a global, conceptual view of the system framework. User A (Section 2.2) consults an index of the Reference Manual to find the description of a term that is an element of that index. User B (Section 2.3) consults an index to find an element and several cross references to related elements in another index. User C (Section 2.4) consults a combination of indexes to find references to sections in the Guidebook, which contain explanations of relevant Evaluation or Validation techniques. Brief definitions of several key words and expressions follow:

E&V	— Evaluation and Validation
Evaluation	— Assessing performance and quality
Validation	— Assessing conformance to a standard
E&V Reference System	— Two documents: the E&V Reference Manual and the E&V Guidebook
E&V Classification Schema	— A set of indexes that provide a framework for the E&V Reference Manual.

The schema was initially described in an earlier "E&V Classification Schema Report" [E&V Classification]. Subsequent changes in the schema will be updated in future versions of this manual; the schema report will not be updated.

2.1 SYSTEM ORGANIZATION

The entire reference system can be viewed as a structure of multiple indexes. For example, there is a function index and a life cycle activity index, among others. The structure is

analogous to the card catalog system in a public library, with its author index, title index, and subject index. To use the card catalog in the library, you must first locate the card that corresponds to the author, title, or subject in which you are interested. Similarly, to use the Reference Manual, you first find the element(s) in which you are interested. The way to do this is to begin by looking at the Table of Contents or the Index at the back of the manual to locate the element(s) in the "formal chapters." The names of the indexes that are formal chapters within the Reference Manual are:

- Life Cycle Activities (Chapter 4)
- APSE Tool Categories (Chapter 5)
- Attributes (Chapter 6)
- Functions (Chapter 7).

Figures 2.1-1, 2.1-2, and 2.1-3 provide a pictorial view of the organization of the Reference Manual, particularly the structure of the reference material contained in the "middle section." The chapters of this middle section are organized in a consistent, formal manner, using a formal grammar (described in Appendix C). A typical chapter corresponds to one index of the schema. A typical index is organized as a hierarchical structure of elements. For every element there is a "text frame" that has (in general) three parts as shown in Fig. 2.1-3. The text frames are built using the formal grammar. (Details of the formal grammar need not concern the user. It was employed because of the possibility of a future on-line, electronic version of the system, supported by advanced updating and information retrieval techniques.)

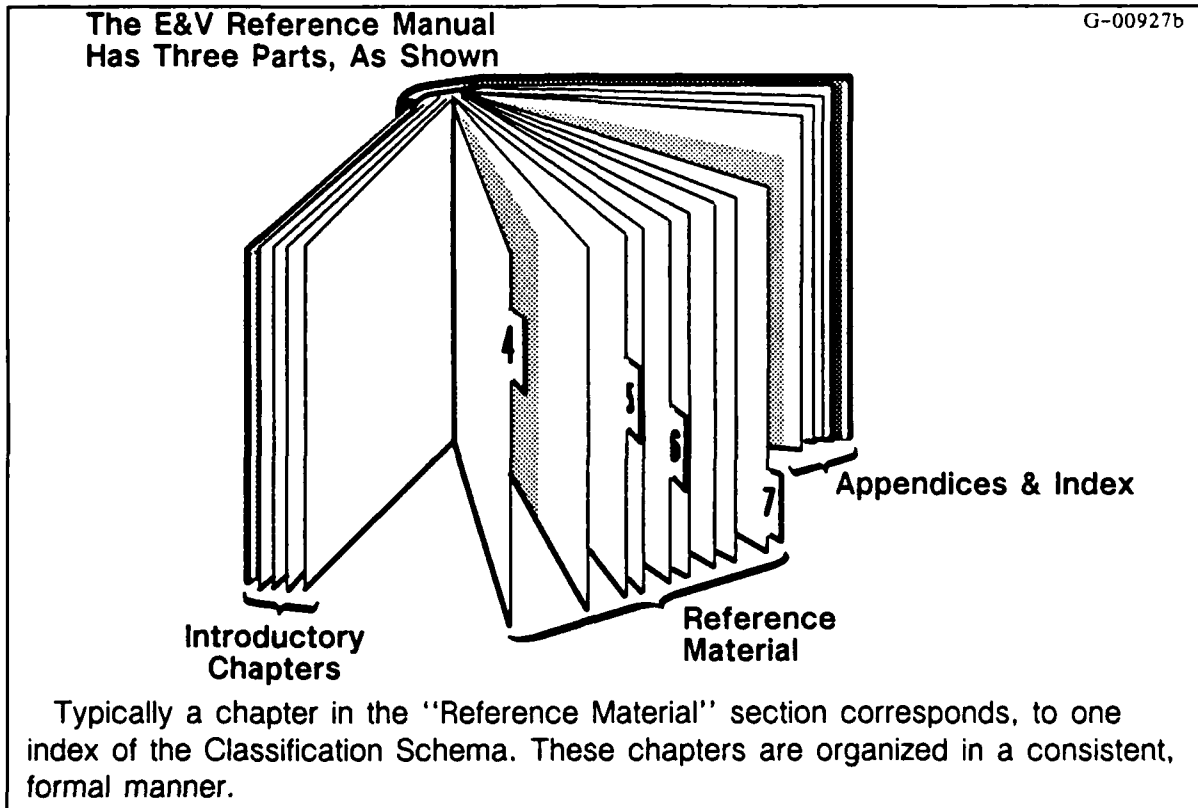


Figure 2.1-1 Reference Manual Organization

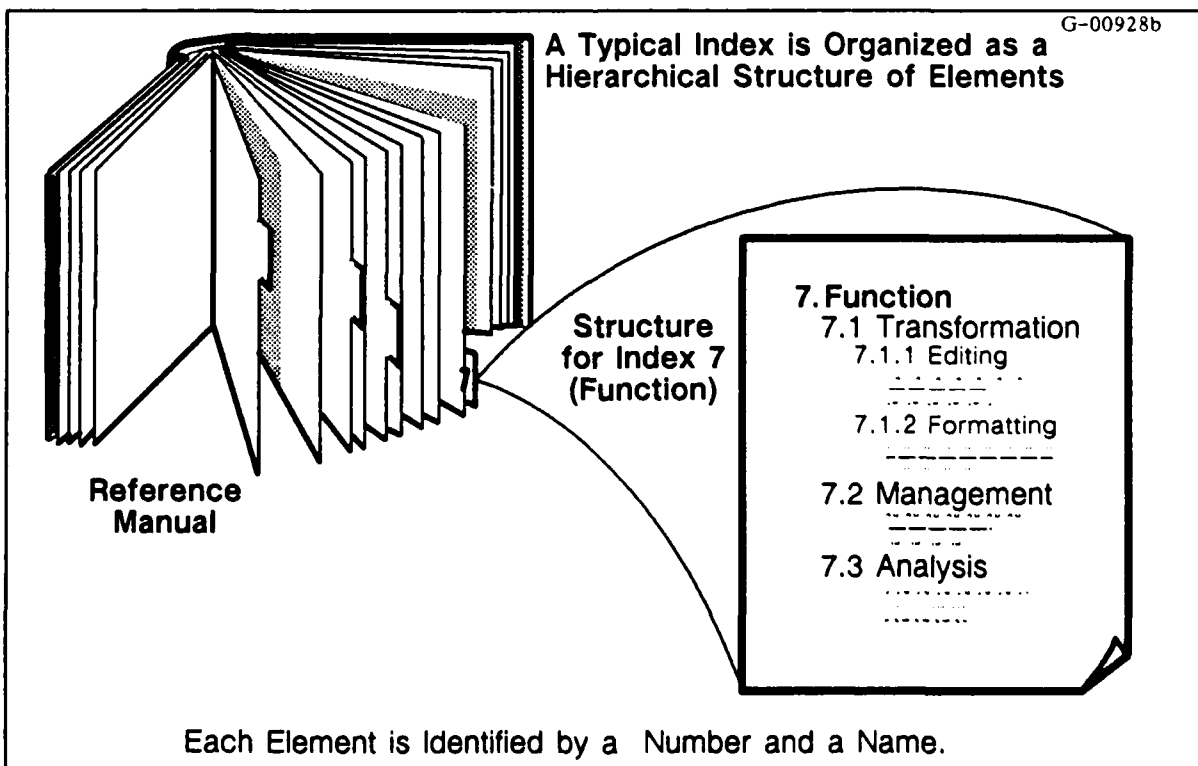


Figure 2.1-2 Reference Material Indexes

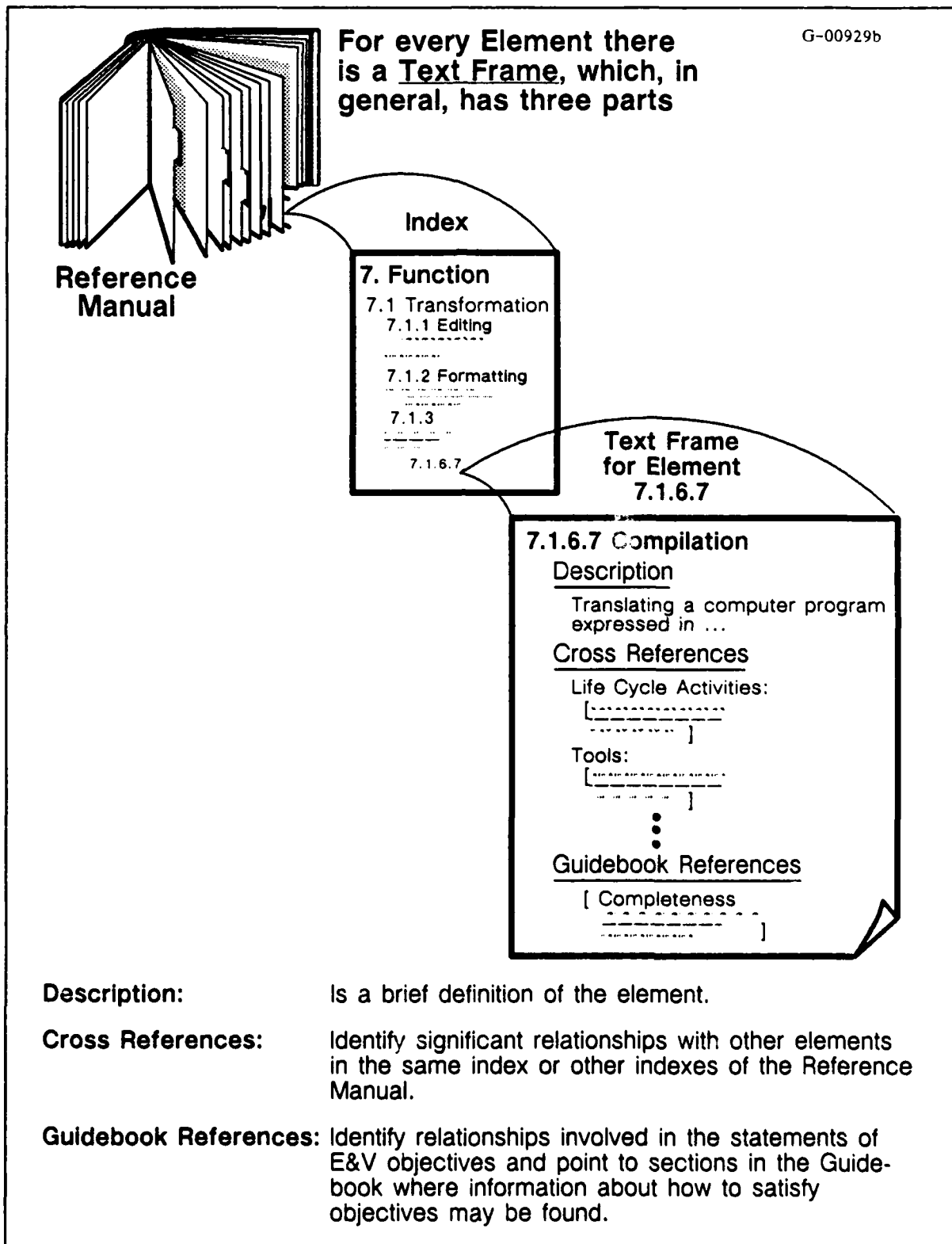


Figure 2.1-3 Text Frames

2.2 DESCRIPTION: DIRECT REFERENCE

Our first example of the use of the Reference Manual is "User A" who consults the Attributes Index to find the description of the term "Storage Effectiveness." Figure 2.2-1 is a (truncated) copy of Section (or Text Frame) 6.4.31. User A may find this frame, for example, by browsing through the Table of Contents or by looking up the term "Storage Effectiveness" in the main Index at the back of the manual. Note that the text frame contains Cross References and Guidebook References as well, but this need not concern User A, who simply seeks a description. The User A scenario is pictorially represented in Fig. 2.2-2, where the boxes are analogous to cabinets in a library card catalog system.

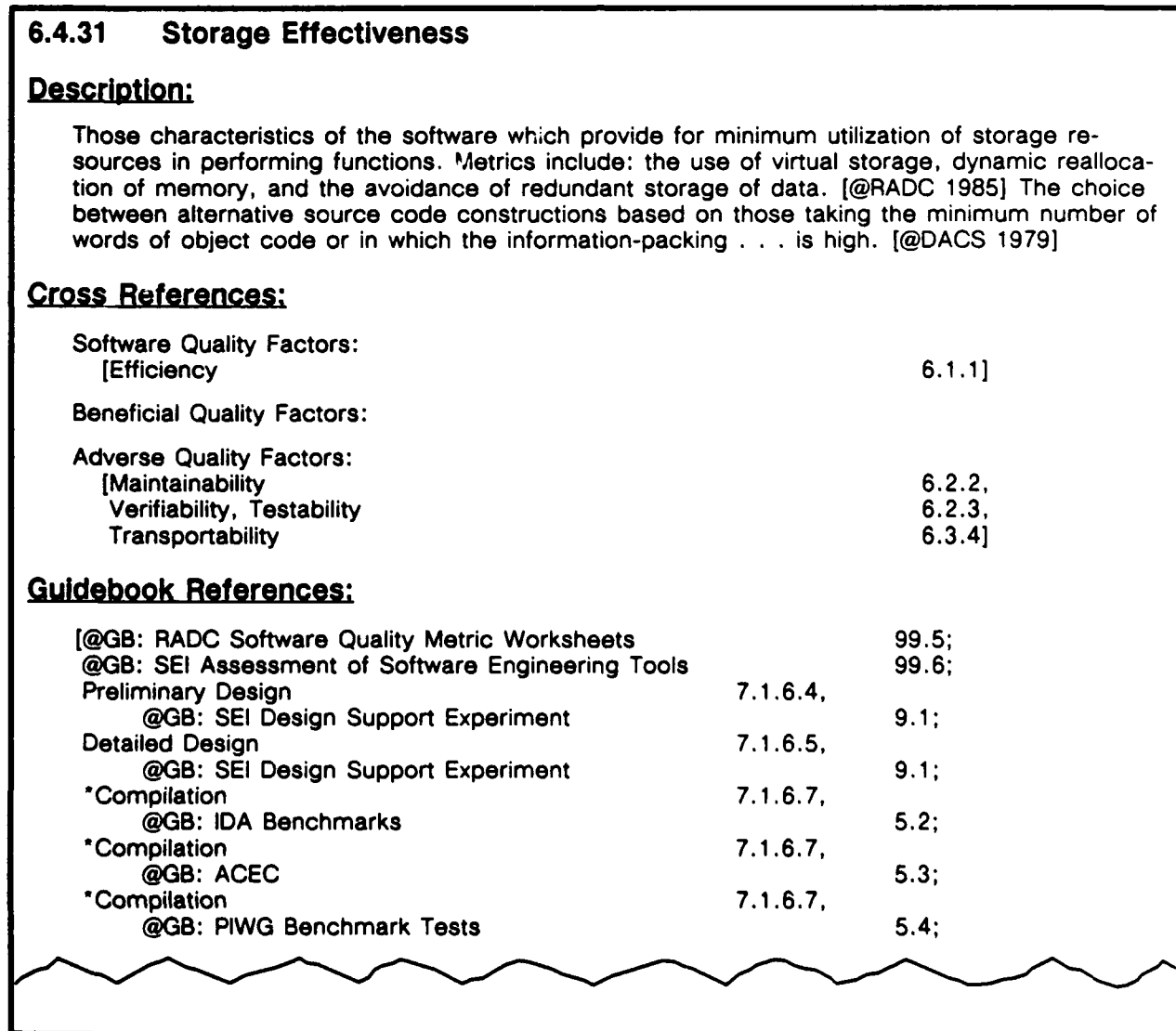


Figure 2.2-1 Sample Attributes Index Frame

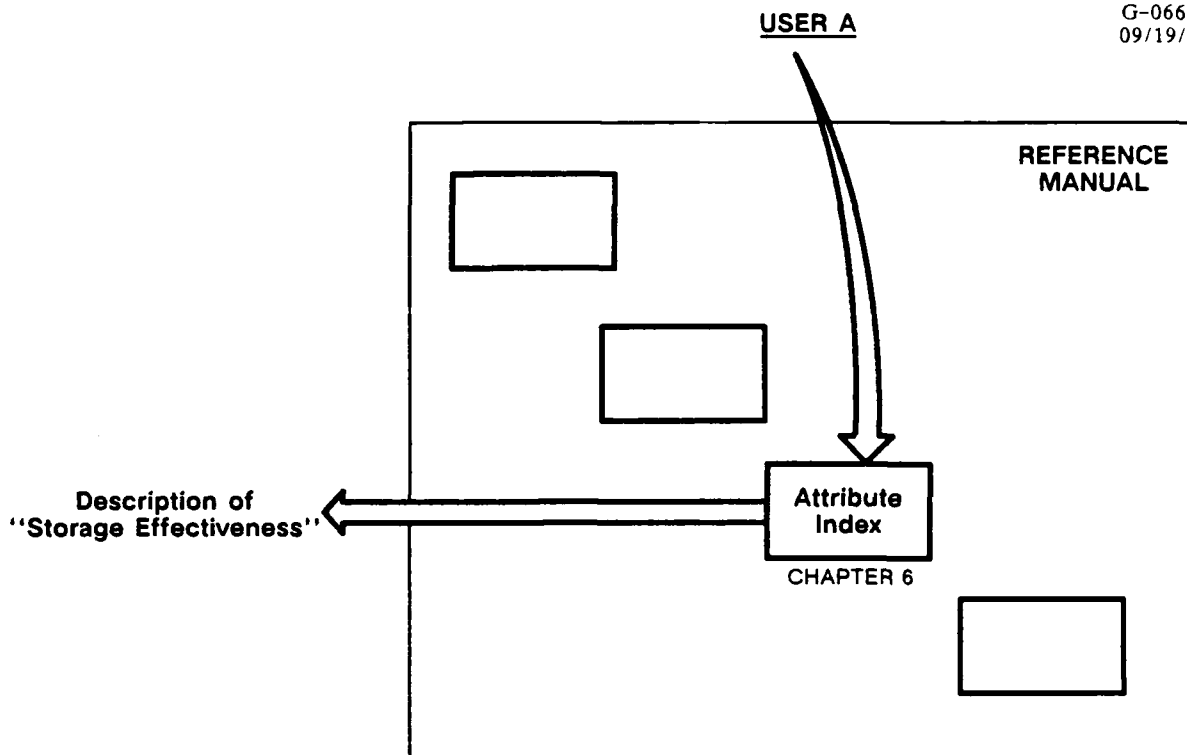


Figure 2.2-2 Example of Direct Reference: User A

2.3 CROSS REFERENCE

Our second example of use of the Reference Manual is "User B" who would like to know the names and descriptions of functions associated with a particular group of life cycle activities. User B first consults the Life Cycle Activities Index, Chapter 4, which divides the life cycle in accordance with the DoD-STD-2167A model [DoD-STD-2167A] for project development. Chapter 4 relates the life cycle activity groups to the functions typically found within an APSE and the products that are produced in each portion of the life cycle. A sample Life Cycle Activities Index Frame is shown in Fig. 2.3-1. This frame represents the sixth activity group [4.6 Coding and CSU Testing] and the second class of activities [4.6.2 Software Engineering] covered in each activity group. The Software Development File is the product for this group of activities. User B finds 61 functions that might be performed during this activity listed in the text frame. In each case another text frame is cross referenced in Chapter 7, the Functions Index, where more information about this function is found. The User B scenario is pictorially represented in Fig. 2.3-2.

4.6.2 Software Engineering**Cross References:****Products:**

[Software Development File (SDF)]

Functions:

[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Assembling	7.1.6.6,
Compilation	7.1.6.7,
Conversion	7.1.6.8,
Macro Expansion	7.1.6.9,
Structure Preprocessing	7.1.6.10,
Body Stub Generation	7.1.6.11,
Preamble Generation	7.1.6.12,
Linking/Loading	7.1.6.13,

Mutation Analysis	7.3.2.9,
Symbolic Execution	7.3.2.10,
Regression Testing	7.3.2.11,
Emulation	7.3.2.13,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Path and Domain Selection	7.3.2.18,
Formal Verification	7.3.3]

Figure 2.3-1 Sample Life Cycle Activity Text Frame**2.4 GUIDEBOOK REFERENCE**

Our third example of use of the Reference Manual is "User C" who would like to look up a function, learn what attributes are associated with it, and find evaluation techniques relevant to particular function-attribute pairs. User C first consults the Function Index, Chapter 7. A sample Function Index Frame [7.1.6.7 Compilation] is shown in Fig. 2.4-1. Functions are related to life cycle activities and tools. In the example, the function, Compilation, is related to five life cycle activity groups: Coding and CSU Testing, CSC Integration and Testing, CSCI Testing, System Integration and Testing, and Operational Testing and Evaluation. The user of the manual can refer to Chapter 4 to find more information about these activity groups. The Compiler is the tool that performs the function Compilation. Information about compilers can be found in Chapter 5. User C is interested in evaluation of the compilation function with respect to various

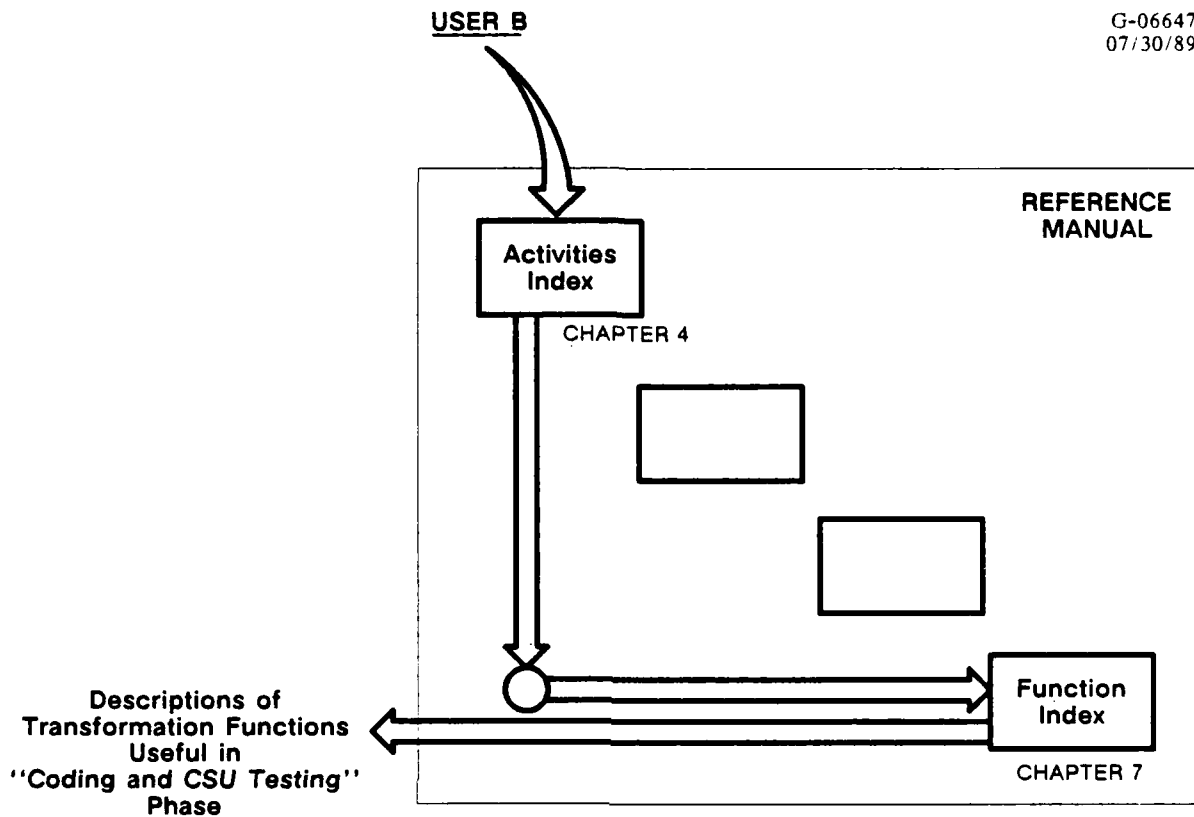


Figure 2.3-2 Example of Cross Reference: User B

attributes. For example, the function-attribute pair Compilation-Processing Effectiveness is represented by the items under Guidebook References in this text frame. The first line of the Guidebook References gives the attribute that is paired with Compilation to point to a section in the Guidebook describing a specific assessment technique. Here, the attribute Processing Effectiveness (described in Section 6.4.22 of the Reference Manual) is paired with Compilation. This pair points to sections in the Guidebook called IDA Benchmarks and ACEC, among others, which provide additional information on these two E&V techniques. A user of the Reference Manual can find references to these two techniques in the Attribute Index or the Function Index. The User C scenario is pictorially represented in Fig. 2.4-2.

7.1.6.7 Compilation

Description:

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

Cross References:

Life Cycle Activities:		
[Coding and CSU Testing	4.6.2,	
CSC Integration and Testing	4.7.2,	
CSCI Testing	4.8.2,	
System Integration and Testing	4.9.2,	
Operational Testing and Evaluation	4.10.2]	
Tools:		
[Translating Editor	5.12.1,	
Compiler (Front End)	5.12.2,	
Compiler (Code Generator — Back End)	5.12.3]	

Guidebook References:

* [Anomaly Management	6.4.2,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Anomaly Management	6.4.2,	
@GB: Compiler Assessment Questionnaire		5.12;
Augmentability	6.4.4,	
@GB: Compiler Assessment Questionnaire		5.12;
Capacity	6.4.6,	
@GB: IDA Benchmarks		5.2;
Capacity	6.4.6,	
@GB: MITRE Benchmark Generator Tool		5.6;
Power	6.4.21,	
@GB: Compilation Checklist		5.8;
Power	6.4.21,	
@GB: Compiler Assessment Questionnaire		5.12;
*Processing Effectiveness	6.4.22,	
@GB: IDA Benchmarks		5.2;
*Processing Effectiveness	6.4.22,	
@GB: ACEC		5.3;
*Processing Effectiveness	6.4.22,	
@GB: PIWG Benchmark Tests		5.4;
*Processing Effectiveness	6.4.22,	
@GB: University of Michigan Benchmark Tests		5.5;
*Processing Effectiveness	6.4.22,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Processing Effectiveness	6.4.22,	
@GB: Compiler Assessment Questionnaire		5.12;
Proprietary Rights	6.4.23,	
@GB: Compiler Assessment Questionnaire		5.12;

Figure 2.4-1 Sample Functions Index Frame

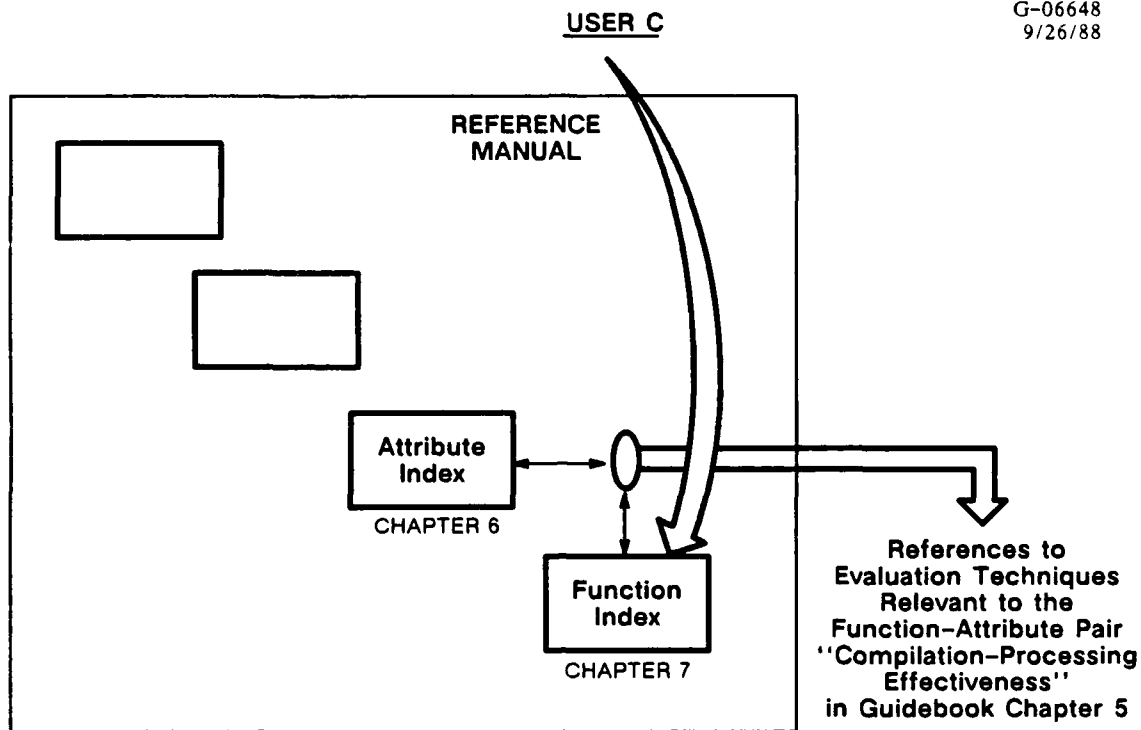


Figure 2.4-2 Example of Guidebook Reference: User C

2.5 REFERENCE FRAMEWORK

Figure 2.5-1 depicts the Classification Schema as an internal framework for the Reference Manual. The schema provides paths, within and between indexes, that users can follow to extract information directly or to find sections in the Guidebook that describe elements of E&V technology. The figure also indicates the direct relationships, that is, types of cross references and Guidebook references featured in the Reference Manual.

The Function Index is seen to be directly related to all of the other indexes. Thus, it is drawn in an "anchoring" position in the diagram. Attributes, also, play a key role in the overall E&V process, in two ways. First, many assessment objectives are defined in terms of function-attribute pairs, such as compilation-efficiency and editing-power. Second, other attributes (those not "pairing-up" with functions) represent factors or criteria by which APSEs or APSE components are assessed, independent of the functions performed. Examples of the latter are: maintainability, interoperability, maturity, and cost.

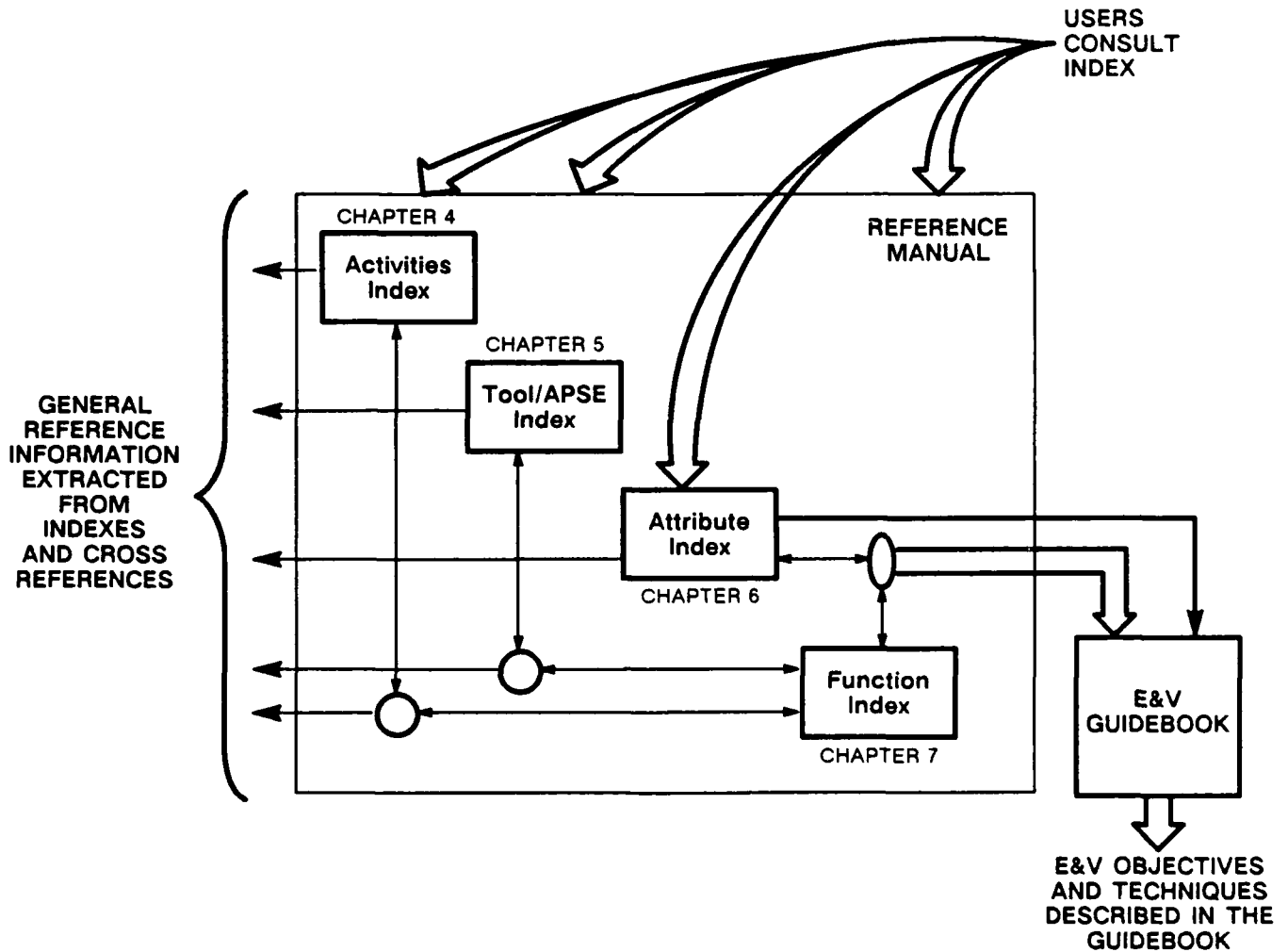


Figure 2.5-1 The Schema as Framework for the Reference Manual

Besides the direct relationships indicated in Fig. 2.5-1, indirect relationships can also be useful and are constructed by combining two or more direct relationships. For example, since activities and functions are related and functions and tools are related, it is possible to determine the relationships between activities and tools. It should be noted that not all such constructions will be useful. Such a useless relationship might be life cycle activities and attributes related via function.

The conceptual structure pictured in Fig. 2.5-1 has an open-ended quality in several ways. First, there is no fixed number of indexes; more can be added as desired. As each new

index is added a new section of the Reference Manual can be added, providing appropriate definitions and cross-references to other indexes. The new index would be represented by an additional block along the "main diagonal." Second, each individual index may have elements added to it as new understanding of the various aspects of APSEs and the E&V process is gained. The process of modifying the structure of an individual index in this way would take place within the boxes. Finally, the off-diagonal space, above and below the main diagonal, represents the notion that any two-way relationship may be included in the system. Thus, any section may be referenced by other sections, and any section may reference other sections. Also, any two indexes may be involved in a relationship of the type that defines an E&V objective or points to an E&V technique in the Guidebook. Although many of the potential combinations are not expected to be useful or relevant to E&V purposes, the structure permits the consideration of all possibilities.

The procedures described in the Guidebook are organized into chapters, as indicated in Fig. 2.5-2. Chapters 4 and beyond are "formal chapters" built on a standard format (see Guidebook). Most of the pointers from the Reference Manual to the Guidebook refer to specific instances of evaluation (E) or validation (V) techniques described in these formal chapters. There are also some "backward references" from these chapters to Chapter 4, "Synopses" [GB 4] or to the Reference Manual.

Evaluation techniques can be characterized as based on subjective judgments or objective procedures, designed to assess performance and quality. Validation techniques can be characterized as based on informal or formal procedures, designed to assess conformance to a standard. The types of E&V techniques described in the Guidebook fall into the following categories:

- Questionnaire
- Capabilities Checklist
- Assessment Checklist
- Test Suite
- Automated Test Suite
- Structured Experiment.

Examples of all of the above are spread throughout the Guidebook. Chapter 3 of the Guidebook [GB 3] provides a discussion of how to combine results from a variety of techniques to perform an integrated whole-APSE assessment.

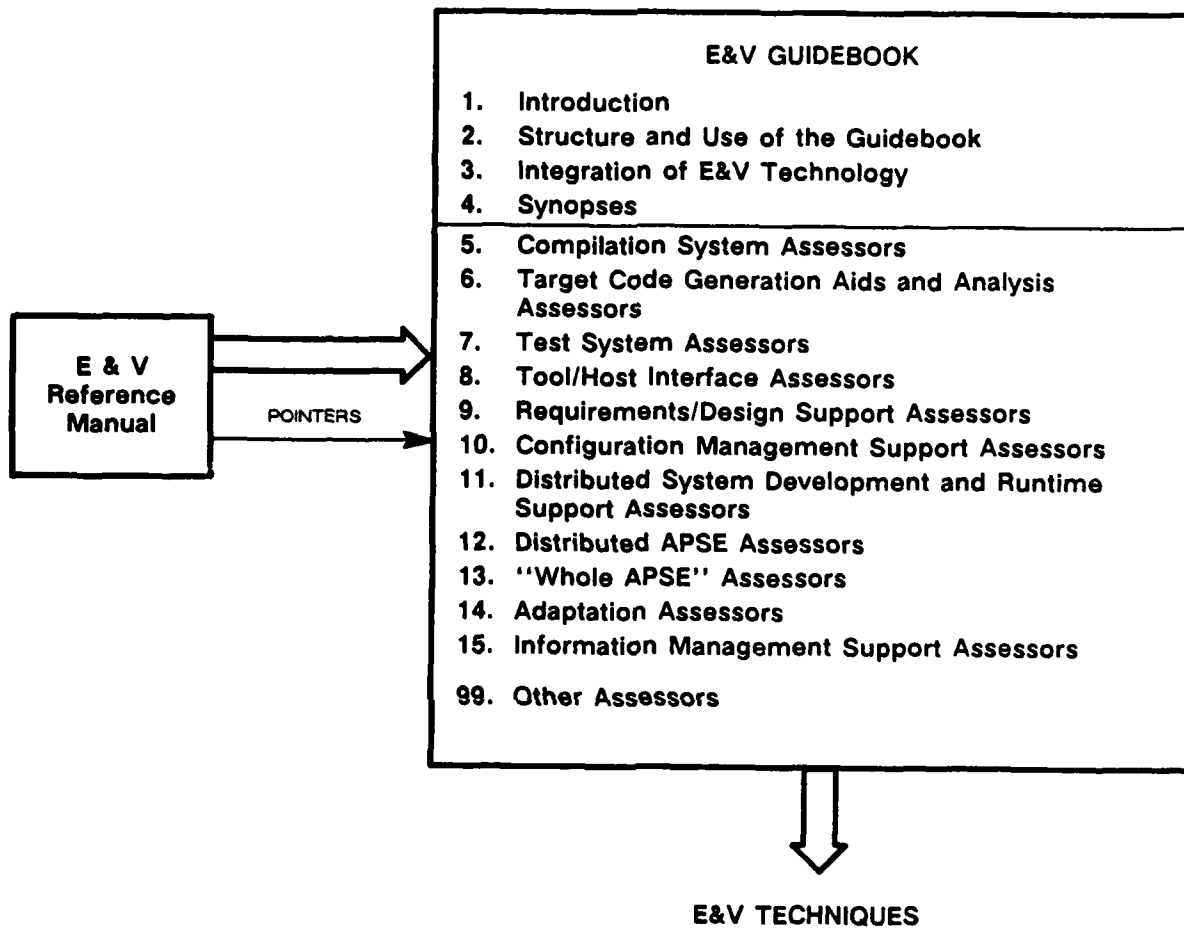


Figure 2.5-2 Organization of the Guidebook

3. **WHOLE APSE ASSESSMENT ISSUES**

This chapter is intended as a starting place for those interested in selecting or assessing an APSE considered as a "whole entity" that is "more than the sum of its parts." The chapter, therefore, cannot be organized in the index/text frame style of other chapters in this manual, because that style fits the view of an APSE as a collection of components or functions that can be evaluated one at a time and "added up." Here we take the opposite view — that the APSE is a total system which serves a project team across an entire software development life cycle — and that it should be evaluated in terms of overall project goals and team productivity, rather than in terms of individual atomic functions.

Although the technology of "whole APSE assessment" is (in the late 1980s) very immature, there are many helpful materials in the open literature. This chapter may be considered a guide to that literature. It is organized to address the following types of questions:

- What is an APSE?
- How can whole APSEs be viewed?
- What are the key whole APSE attributes?
- How can whole APSEs be assessed?
- Where can relevant information be found?

The first four questions are addressed in Sections 3.1 through 3.4, respectively. Various references to other sections of the E&V Reference Manual (RM), the E&V Guidebook (GB), and open-literature sources are distributed throughout the chapter.

3.1 APSE DEFINITIONS AND ALTERNATIVE NAMES

The acronym APSE stands for "Ada Programming Support Environment." The term "Programming Support Environment" is defined in the "IEEE Standard Glossary of Software Engineering Terminology" [IEEE 1983] as:

"An integrated collection of tools accessed via a single command language to provide programming support capabilities throughout the software life-cycle. The environment typically includes tools for design, editing, compiling, loading, testing, configuration management, and project management."

Thus, one useful definition of an APSE is the above quotation along with the stipulation that there be at least one Ada compiler among the tools provided. A similar definition for the term IPSE, which stands for "Integrated Project Support Environment," is given [Lehman 1987] as:

"An embodiment of software technology in a collection of tools for capture, representation, control, refinement, transformation, and other manipulation of project related information."

The second name and definition are broader than the first because they refer to total "project" support and information, rather than "programming" support and programs. The distinction between the two is highlighted not only in the Lehman and Turski paper, but in many others, including a survey paper by Houghton and Wallace [Houghton 1987]. The latter uses the following terms to characterize most existing environments:

- Framing Environments
- Programming Environments
- General Environments.

Framing environments concentrate on the early stages of the life cycle and tend to be methodology-specific. Programming environments concentrate on the latter part of the life cycle and are oriented toward programming, debugging, and testing. General environments contain basic tools that support all phases of the life cycle and tend to be methodology-free.

Both of the above definitions and discussion are limited in that they are tied to one, traditional view of an APSE. In this view an APSE (or IPSE) is seen as a collection of tools. The following section presents additional views, any of which may become increasingly important to those who wish to select or evaluate APSEs of the future.

Another taxonomic classification of environments is presented in a paper by Dart, Ellison, Feiler, and Habermann [Dart 1987]. The categories are:

- Language-centered environments
- Structure-oriented environments
- Toolkit environments
- Method-based environments.

The significance of the taxonomy, according to the authors, "is in its clarification of trends rather than in categorization of particular environments. A particular environment may fit a number of categories," (see discussion of views of an APSE in the following section).

Additional acronyms found in the literature on this same general topic are listed below, among others mentioned previously.

APSE	—	Ada Programming Support Environment
IPSE	—	Integrated Project Support Environment
PSE	—	Programming Support Environment
SDE	—	Software Development Environment
SEE	—	Software Engineering Environment

The term SDE, for example, has been used in the title of several ACM-sponsored conferences, and is the subject of an IEEE tutorial [IEEE 1981]. All of the above are possible key words, under which may be found useful material on whole APSEs or whole APSE E&V, the subject of this chapter.

A definition of an APSE, based on the concept of a collection of tools, is limited in the sense that it describes only a portion of the total environment for software development. As discussed in the paper "The Ecology of Software Environments" [Wasserman 1981a], the total environment or "surroundings" includes such things as the computer itself, peripheral storage and display devices, project management practices, organizational characteristics, and external constraints such as governmental directives and physical workspace factors. All of the above will influence the development and application of whole APSE evaluation techniques.

3.2 VIEWS OF AN APSE

Various ways of viewing an APSE are summarized briefly below. The views differ in terms of both how an APSE is seen — that is, what sort of conceptual model does the viewer hold

— and by whom is the APSE viewed — for example, the APSE user would typically have a different view than the APSE builder. The views presented are not necessarily mutually exclusive; an individual's view of a specific APSE may combine aspects of several of the following notions.

3.2.1 APSE Viewed as a Collection of Tools

This is the traditional view of programming support environments, and is consistent with the standard definition quoted in the previous section. An advantage of taking this view is that it permits the viewer to characterize an APSE in terms of elements of a functional taxonomy, such as that provided in Chapter 7 [Functions 7]. The characterization can be stated in terms of yes/no answers to a long list of clearly defined questions — is function x.y.z provided or is it not? A disadvantage of this view is that it may cause the viewer to neglect the crucial whole-APSE issues that are the subject of this chapter, and which are impossible to express in terms of a composition of individual low-level functions.

One version of this view is given in the Stoneman Report [DoD 1980], which pictures a multi-layered, extensible collection built around an inner kernel (KAPSE) and a surrounding minimal (MAPSE) layer of essential tools, as shown in Fig. 3.2-1. This version may have more relevance for APSE builders than for APSE users, since users need not necessarily be aware of the layer in which a particular tool resides.

3.2.2 APSE Viewed as a Methodology-Support System

In this view an APSE is seen as a system that supports a particular development methodology or a particular model of the software development process. It might be strongly tied to a standard set of products and activities such as those required by the U.S. Department of Defense [DoD-STD-2167A] for mission critical software. It might be based upon a coherent software development and maintenance methodology, such as that described in "Life-Cycle Support in the Ada Environment" [McDermid 1984]. The latter expresses the view that, "The purpose of a tool is to support a method by automating some aspect of the use or application of the method."

3.2.3 APSE Viewed as an Information Management System

In this view primary emphasis is placed on how project information is stored, retrieved, manipulated, and maintained over the entire life cycle. A key issue is the set of

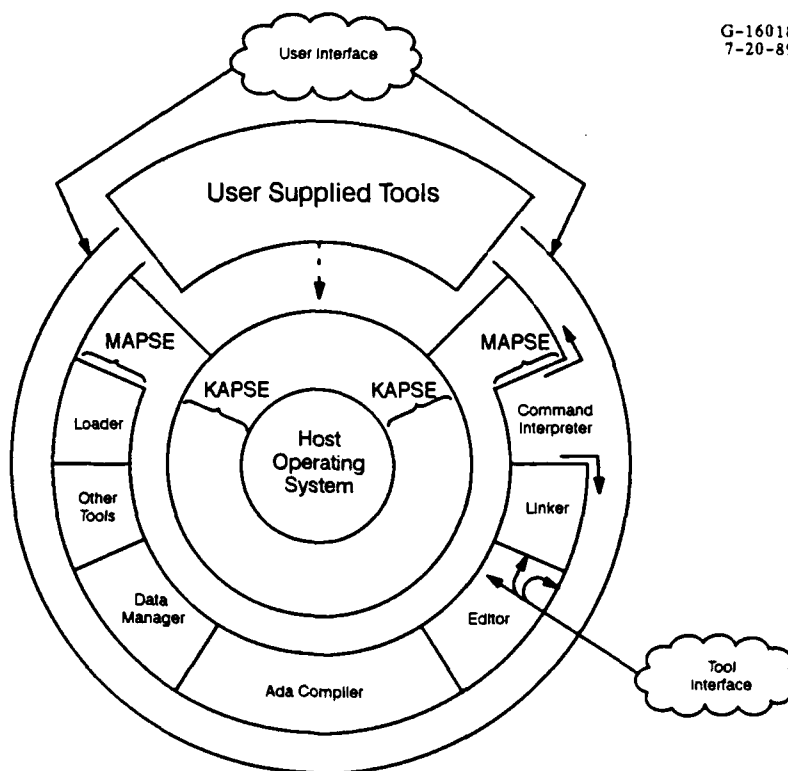


Figure 3.2-1 Stoneman APSE Architecture

interfaces between tools and the project database [Houghton 1987]. The database may be considered a tool that is used by other tools; in this case, the database itself is the interface between the other tools. Another key issue is the control of access to information, and how this is affected by data structure models — such as hierarchical, relational, transactional, etc. McDermid [McDermid 1985] envisions three generations of environments as follows. The first generation would be a conventional collection of Ada tools tied to a Unix-like environment. The second generation would be based on a database schema that provides appropriate life cycle support, but has a static, non-adaptive structure. The third generation would provide a dynamic, adaptive information management system that can encapsulate any underlying database schema.

Another key set of issues to be addressed are those specific to the Ada compilation process. A unique feature of the Ada language is that inter-unit dependencies exist at compilation time. Source code units can be compiled separately, but not independently. The implications of this feature are discussed further in Section 3.3.1 under Integrity.

3.2.4 APSE Viewed as a User-Oriented, Interactive System

This view emphasizes user interfaces and human factors. According to the authors of a survey paper [Houghton 1987], "Many systems that measure poorly in terms of human factors, including most traditional operating systems, have withstood the test of time. . . . The real users are the systems programmers or gurus." (But, in the future) "Software engineering environments . . . should not require a system expert as an interface between the software engineer and the environment." A conference overview paper [Henderson 1987] sees "a trend toward the use of graphics." The desire for graphical interactive workstations is echoed in a paper on personal development systems [Gutz 1981] with the words, "time sharing is an idea whose time is gone." An important set of issues concerns the different roles of various users of the system, different modes of use for each, and how the system orients its support to the current user/mode. An influential set of concepts has come from the world of artificial intelligence, such as the "incremental enrichment" style of LISP program development [Barstow 1981].

3.2.5 APSE Viewed as a Knowledge-Based Expert System

In this view components of an APSE (or conceivably the entire APSE) are created as expert systems based on the past experience of human experts in specific domains. One issue of IEEE Expert [IEEE Expert 1986] contains a collection of papers which address this possibility. A paper in this collection [Zuallkernan 1986] outlines some steps to determine the feasibility of this approach and treats the specific area of software testing as a case study. This view may be considered a special, advanced case of the preceding view, where the style of user-interaction is that of an "expert assistant."

3.2.6 APSE Viewed as a Stable Framework

This view has been advocated by C.M. McKay of U. Houston Clear Lake [McKay 1987], and is motivated by the need to support large, complex, non-stop, distributed, long-lived systems such as the NASA Space Station. The APSE is viewed as a stable framework to which new or improved tools can be added over time without interfering with or invalidating previous work. The framework is defined in terms of standard phases and deliverables, in one dimension, and stable interface sets separating tools and various classes of objects, in the other dimension. Ideally, the stability of the framework and its interface sets will allow both the flight system and its support environment to evolve incrementally as reliable, maintainable systems.

E&V Reference Manual, Version 2.0

Another view, similarly motivated, is illustrated in Fig. 3.2-2. This figure has been adapted from the "STARS Consolidated Development Plan" [STARS 1988]; it was supplied to the E&V Task by Unisys, one of the three STARS prime contractors. It depicts an adaptable environment architecture based on layered portable interfaces. The top interface of the Virtual Layer, for example, is intended to be independent of the underlying hardware. It could be specified as the CAIS-A standard, the IEEE POSIX/Ada interface, or an interface specifically engineered for the STARS program. Components of the Object Structure Layer use the services of the Virtual Layer. Many of these components will make use of industry standard interfaces such as GKS, X, and SQL. Standardization of these interface layers permit the APSE to evolve in a stable way.

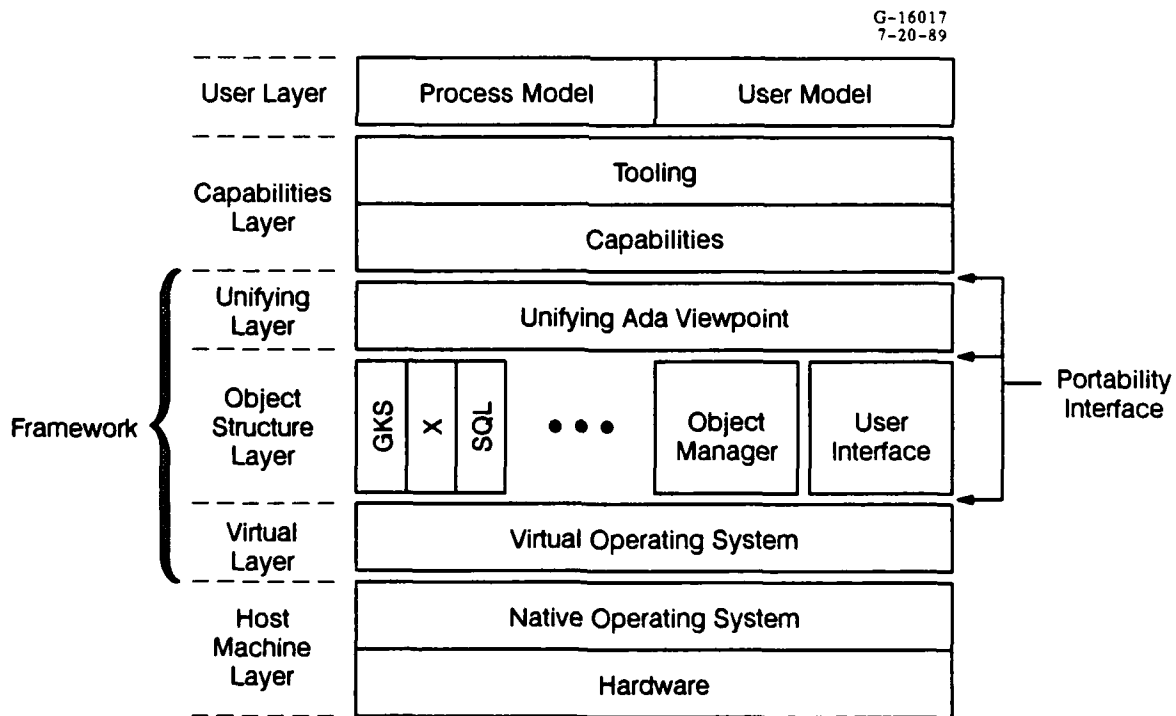


Figure 3.2-2 STARS Environment Architecture

3.3 KEY ATTRIBUTES OF WHOLE APSES

This section provides a brief discussion of what appear to be some of the key whole-APSE attributes. The discussion is organized in accordance with the attribute taxonomy [Attributes 6] given in Chapter 6 of this manual, which employs three top-level "acquisition concern" categories:

- Performance Attributes (6.1)
- Design Attributes (6.2)
- Adaptation Attributes (6.3).

Under each of these is a second-level set of "quality factors", such as Efficiency, Integrity, etc. These are further decomposed into "criteria", some of which apply to more than one quality factor. All the criteria attributes are listed alphabetically in Section 6.4 of the Attributes Index.

The definitions given in Chapter 6 are very component-oriented. For example, the definition of Efficiency [Efficiency 6.1.1] includes the words, "The extent to which a component fulfills its purpose using a minimum of computing resources." A whole-APSE version of the definition of Efficiency might read, "The extent to which an APSE supports life cycle activities using a minimum of computing and development-team resources." This same kind of language modification is used throughout the discussion to follow.

The selection of attributes highlighted below has been influenced by discussions within the E&V Team and by the following papers (some of which make reference to many other papers): "Toward Integrated Software Development Environments" [Wasserman 1981b], "Software Development Environment Issues as Related to Ada" [Notkin 1981], "Essential Properties of IPSEs" [Lehman 1987], and "Characteristics and Functions of Software Engineering Environments: An Overview" [Houghton 1987].

A quotation from another paper by Wasserman [Wasserman 1981a] perhaps best sets the stage for this discussion of key whole-APSE attributes. He says, "The goal is to create an environment that not only enhances developer productivity but also supports the creation of superior products."

3.3.1 Performance Attributes

[Efficiency 6.1.1] — As stated above, efficiency, in the whole-APSE context, is measured in terms of resources used by an entire team in performing an entire group of activities

or major "chunk" of work during development of a software product. This attribute is clearly related to the overall project goal of team productivity.

[Integrity 6.1.2] — This attribute deals with the extent to which access to the software environment or other data is controlled, especially for the purposes of monitoring status, preserving integrity of different versions, and controlling changes. It is an attribute of an important set of management functions that greatly influence team productivity and product quality. A critical aspect of this attribute concerns the treatment of the unique Ada feature (discussed in Section 3.2.3) associated with inter-unit dependencies. In managing Ada program library units strict rules apply to the order of compilation. As a project is developed, obsolete units need to be recompiled, compilation order changes, and Ada closure sets may be redefined. Therefore, the APSE's management of Ada source and object modules should be assessed with these special requirements in mind.

[Usability/Anomaly Management 6.1.5/6.4.2]

and

[Usability/Cost 6.1.5/6.4.11]

and

[Usability/Maturity 6.1.5/6.4.18] — These are the aspects of usability that naturally concern managers and controllers of facility investment resources. Anomaly management affects the probability that an APSE will be functionally ready at some specified point in time. It must operate, of course, on available hardware. Absence of such availability in a timely way would naturally be disastrous.

Cost includes the cost of purchase or lease, installation, user assistance, and maintenance. Its importance is self-evident.

Maturity is the extent to which an APSE has been used in the development of deliverable software by typical users and to which the feedback from that use has been reflected in improvements.

[Usability/Operability 6.1.5/6.4.20]

and

[Usability/Training 6.1.5/6.4.36] — These represent the human-factors aspects of usability such as ease-of-use, ease-of-learning, on-line help features, and consistency of

interfaces. This set of attributes will have a major influence not only on long-term productivity, but on the early acceptance of an APSE by individuals and the team as a whole. The resulting impact on motivation and team spirit can be crucial.

3.3.2 Design Attributes

[Correctness/Completeness 6.2.1/6.4.9] — This is the extent to which an APSE supports the complete set of operations necessary to perform its intended function, which is to provide full support to a development team across an entire product life cycle. This attribute could be interpreted in a minimal way, listing only truly essential (MAPSE) functions necessary for a particular project. Alternatively, it could be interpreted as a list of functions desired to provide the capability to produce high-quality products.

[Maintainability/Self-Descriptiveness 6.2.2/6.4.28] — In the whole-APSE context the aspect of this attribute to be emphasized concerns the underlying database or schema used to store and retrieve project data. In one of the papers cited above [Lehman 1987] this attribute was called data-structuredness. The Ada-Europe document "Selecting an Ada Environment" [Lyons 1986] says, "It is now generally recognized that the totality of information that a project must store, consists not only of individual entities containing data but also of the relationships between them, such as the facts that a particular object file has been compiled from a particular source file, that a source file implements a particular specification, or that an error report relates to a particular release of a system."

[Verifiability, Testability 6.2.3] — This is the extent to which an APSE facilitates evaluation of its own performance, so that productivity and quality metrics can be gathered and analyzed. It is through the use of this attribute that the extensibility (see below) of an APSE can be exploited effectively in a continual process of improvement.

3.3.3 Adaptation Attributes

[Expandability/Augmentability 6.3.1/6.4.4] — This is the extent to which an APSE facilitates the addition of new capabilities in response to needs that go beyond its original requirements. The new capabilities could include new functions, expanded data capacity, or new types of data relationships. The word used to describe this feature in several of the papers cited above is extensibility. Given the current state of the art and rapid pace of change of environment

technology, the presence of this quality (however it is achieved) appears necessary in order to provide a path for evolutionary improvements.

[Interoperability/Commonality 6.3.2/6.4.7] — This is the ability of APSEs to exchange database objects and their relationships without conversion of formats, and the use of interface standards (e.g., CAIS [@CAIS]) to facilitate such exchanges.

[Transportability 6.3.4] — This is the extent to which an APSE supports the movement of software components to or from another APSE without change in functionality or reprogramming, and the use of interface standards (e.g., CAIS) to facilitate such movements. Another whole-APSE attribute that may be important is the extent to which the entire APSE can be transferred to a different host/operating system, based on its use of standardized interfaces to the underlying system.

3.4 APPROACHES TO WHOLE-APSE E&V

This section provides a brief discussion of approaches to whole-APSE assessment, including both evaluation of performance and quality, and validation of conformance to standards. References to Guidebook sections and other documents are included, in cases where there are known examples of existing assessors or assessment products under development. Such assessors can be categorized generally as either "tools" or "aids." [E&V Tools and Aids 1987] A more refined breakdown is the following:

- Benchmarks and Test Suites (Tools)
- Checklists and Questionnaires (Aids)
- Structured Experiments (Aids that may use Tools)
- Decision Aids (Aids that may use Tools and other Aids).

These are discussed, in turn, in the following four subsections.

3.4.1 Benchmarks and Test Suites

Benchmarks are standard tests used to measure the execution, performance or acceptability of an APSE function or set of functions. A test suite is an organized collection of such

tests. The Ada Compiler Validation Capability [ACVC 1989] is a test suite designed to test conformance of an Ada compiler to the formal definition of the language [DoD 1983]. A prototype compiler performance evaluation test suite was generated by the Institute for Defense Analysis [GB: 5.2], and a more carefully engineered set known as the ACEC or Ada Compiler Evaluation Capability [GB: 5.3] has been developed, under an E&V Task contract, by the Boeing Company. Another collection of Ada compiler performance tests has been gathered by the ACM SIGAda Performance Issues Working Group [PIWG 1987]. Results of these tests, run on a number of commercial compilers, will be published in the open literature.

3.4.2 Checklists and Questionnaires

Checklists and questionnaires are used to gather data from vendors or users of tools and APSEs. Examples of such data might include specification parameters, design features, historical information, typical usage scenarios, implementation strategies, enhancement plans or desires, and problem reports. An early example, in Ada terms, of such a questionnaire was one applied to the evaluation of the ROLM Ada Work Center [Castor 1983].

A whole-APSE-oriented example of the use of questionnaires is the book "Selecting an Ada Environment" [Lyons 1986], which is synopsized in the E&V Guidebook [GB: 4.9]. The book is organized around background discussions of various topics followed by appropriate questions addressing each topic. The E&V Guidebook [GB 13.1] provides an example of a single APSE Characterization Form that can be used to capture and organize a summary of the capabilities and features of an APSE. The Guidebook also contains many examples of checklists devoted to individual topics.

3.4.3 Structured Experiments

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

3.4.4 Decision Aids

Decision aids allow a user to assess an APSE from a particular point of view. Decision aids may combine the results of a number of tests, questionnaires, and/or monitored experiments, each of which is weighted according to its value for the view being considered [GB 3].

4. LIFE CYCLE ACTIVITIES

This chapter deals with the life cycle activities served by the support environment. The foundation for this discussion is DoD-STD-2167A [DoD-STD-2167A]. The divisions of the life cycle that are chosen to represent the progression of activities depend on the view that a person has of the APSE. If one thinks of the APSE as only providing an environment for developing software, then the activity groups can be limited to the following six:

- Software Requirements Analysis
- Preliminary Design
- Detailed Design
- Coding and CSU Testing
- CSC Integration and Testing
- CSCI Testing.

However, if one sees the APSE as providing support across the entire system life cycle (Integrated Project Support Environment, or IPSE, rather than APSE) [Whole APSE Issues 3.], then the following five activity groups should be added as well:

- System Concepts
- System Requirements Analysis/Design
- System Integration and Testing
- Operational Testing and Evaluation
- Change Requirements.

Of the above five activity groups, the first two precede the six software development phases and the next two follow the six groups in the overall system life cycle. The last activity group, change requirements, is one that may be started in parallel with any of the other groups and addresses the issues of enhancement, error correction, and modification.

Also, a global life cycle 'activity group' is introduced. This is not a true portion of the life cycle, but represents a group of activities that may occur anywhere across the entire life cycle. An example of a global function is general purpose text editing.

Chapter 4 of DoD-STD-2167A lists the general requirements for the life cycle activities. These requirements fall into the following activity classes:

- Software Development Management
- Software Engineering
- Formal Qualification Testing
- Software Product Evaluations
- Software Configuration Management
- Transitioning to Software Support.

These activities are used as a second-level of classification under each top-level life cycle, activity division of this chapter.

Although this chapter is structured around the life cycle activities of DoD-STD-2167A, the information is equally applicable to software life cycles that do not fit that model. The following may be considered to be a relatively generic view of the activities encompassed in a software life cycle:

- Requirements Engineering
- Design Engineering
- Implementation
- Maintenance.

This is a generic segmentation since, regardless of the software engineering approach taken, it is necessary to: 1) decide what to build, 2) decide how to build it, 3) build it, and 4) support it for its useful life. Different life cycle models combine and structure these activities in different ways, but each activity is usually included. In a sense, they are the invariants of software engineering.

Mapping the DoD-STD-2167A life cycle activities to the generic set introduced above results in the groupings shown in Table 4-1. In the generic list of life cycle activities, each is assumed to incorporate the analysis of its products. This can include requirements analysis, design reviews, code walkthroughs, and dynamic analysis or conventional testing activities. Therefore, no separate testing activity is listed. If an additional activity is desired to highlight testing or integration activities, the mapping that results is straightforward.

Table 4-1 Life Cycle Activities Mapping

GENERIC LIFE CYCLE ACTIVITIES	DoD-STD-2167A LIFE CYCLE ACTIVITIES
Requirements Engineering	System Concepts System Requirements Analysis/Design Software Requirements Analysis
Design Engineering	Preliminary Design Detailed Design
Implementation	Coding and CSU Testing CSC Integration and Testing CSCI Testing System Integration and Testing Operational Testing and Evaluation
Maintenance	Change Requirements

Regardless of the life cycle of interest (e.g., Spiral Model [Boehm 1988] or rapid prototyping), it should be possible to map its activities to the generic set and then follow the mapping to DoD-STD-2167A to locate relevant information in this chapter. The second-level of classification can be a further guide to the functions to be expected to support management, engineering, testing, quality assurance, configuration management, and preparation for maintenance.

Figure 4-1 shows the relationships between the life cycle activities and the other elements in the Reference Manual. Each of the life cycle activity groups are described in the following sections and under each section are six subsections that deal with each of the activity classes. The functions specified for each section are those that are typically used in that portion of the life cycle.

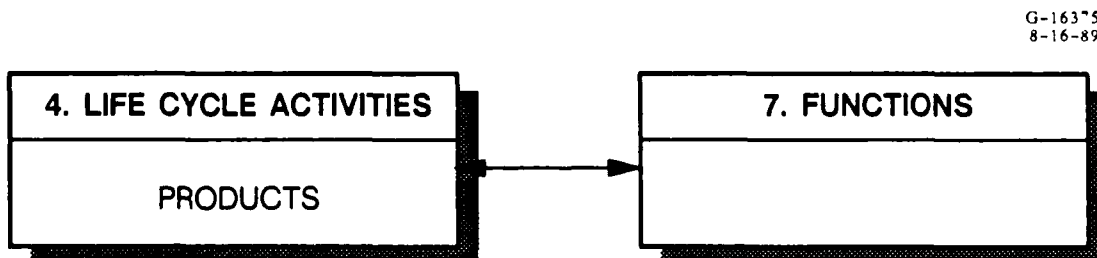


Figure 4-1 Life Cycle Activity Relationships

4.1 SYSTEM CONCEPTS

4.1.1 System Development Management

Cross References:

Products:

Functions:

4.1.2 System Engineering

Cross References:

Products:

Functions:

[Requirements Simulation
Requirements Prototyping
Simulation and Modeling

7.3.2.1,
7.3.2.2,
7.3.2.3]

4.1.3 Formal Qualification Testing

Cross References:

Products:

Functions:

4.1.4 System Product Evaluations

Cross References:

Products:

Functions:

4.1.5 Configuration Management

Cross References:

Products:

Functions:

4.1.6 Transitioning to System Support

Cross References:

Products:

Functions:

4.2 SYSTEM REQUIREMENTS ANALYSIS/DESIGN

4.2.1 System Development Management

Cross References:

Products:	
[Software Development Plan	(SDP)]
Functions:	
[Predefined and User-Defined Forms	7.1.2.3]

4.2.2 System Engineering

Cross References:

Products:	
[System/Segment Specification	(SSS),
Prime Item Development Specification	(PIDS),
Critical Item Development Specification	(CIDS),
System/Segment Design Document	(SSDD),
Software Requirements Specification	(SRS),
Interface Requirements Specification	(IRS)]
Functions:	
[Predefined and User-Defined Forms	7.1.2.3,
Systems Requirements	7.1.6.1,
Requirements to Natural Language	7.1.6.3,
Quality Measurement	7.3.1.9,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Maintainability Analysis	7.3.1.18,
Requirements Simulation	7.3.2.1,
Requirements Prototyping	7.3.2.2]

4.2.3 Formal Qualification Testing

Cross References:

Products:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Documentation Management	7.2.1.2,
Testability Analysis	7.3.1.7,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17]

4.2.4 System Product Evaluations

Cross References:

Products:

Functions:

[Consistency Checking	7.3.1.13,
Auditing	7.3.1.22]

4.2.5 Configuration Management

Cross References:

Products:

Functions:

4.2.6 Transitioning to System Support

Cross References:

Products:

[Computer Resources Integrated Support Document	(CRISD)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.3 SOFTWARE REQUIREMENTS ANALYSIS

4.3.1 System Development Management

Cross References:

Products:

[Software Development Plan (SDP)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Resource Estimation 7.2.2.5,
Tracking 7.2.2.6]

4.3.2 Software Engineering

Cross References:

Products:

[Software Requirements Specification (SRS),
Interface Requirements Specification (IRS)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Software Requirements 7.1.6.2,
Requirements to Natural Language 7.1.6.3,
Program Library Management 7.2.1.7,
Data Flow Analysis 7.3.1.3,
Functional Analysis 7.3.1.4,
Traceability Analysis 7.3.1.6,
Quality Measurement 7.3.1.9,
Maintainability Analysis 7.3.1.18,
Requirements Simulation 7.3.2.1,
Requirements Prototyping 7.3.2.2]

4.3.3 Formal Qualification Testing

Cross References:

Products:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Documentation Management	7.2.1.2,
Cross Reference	7.3.1.17]

4.3.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Testability Analysis	7.3.1.7,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Auditing	7.3.1.22]

4.3.5 Configuration Management

Cross References:

Products:

Functions:

[Specification Management	7.2.1.6]

4.3.6 Transitioning to Software Support

Cross References:

Products:

[Computer Resources Integrated Support Document	(CRISD)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.4 PRELIMINARY DESIGN

4.4.1 Software Development Management

Cross References:

Products:

[Software Development Plan (SDP)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Resource Estimation 7.2.2.5,
Tracking 7.2.2.6]

4.4.2 Software Engineering

Cross References:

Products:

[Software Design Document (SDD),
Interface Design Document (IDD)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Preliminary Design 7.1.6.4,
Design Generation 7.1.7.1,
Documentation Management 7.2.1.2,
Program Library Management 7.2.1.7,
Interface Analysis 7.3.1.5,
Testability Analysis 7.3.1.7,
Quality Measurement 7.3.1.9,
Complexity Measurement 7.3.1.10,
Reusability Analysis 7.3.1.14,
Maintainability Analysis 7.3.1.18,
Invocation Analysis 7.3.1.19,
Structured Walkthrough 7.3.1.21,
Auditing 7.3.1.22,
Type Analysis 7.3.1.27,
Units Analysis 7.3.1.28,
Simulation and Modeling 7.3.2.3,
Design Prototyping 7.3.2.4]

4.4.3 Formal Qualification Testing

Cross References:

Products:

[Software Test Plan (STP)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Documentation Management 7.2.1.2,
Test Condition Analysis 7.3.1.8]

4.4.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Requirements Reconstruction 7.1.7.2,
Completeness Checking 7.3.1.12,
Consistency Checking 7.3.1.13,
Cross Reference 7.3.1.17,
Scanning 7.3.1.20,
Auditing 7.3.1.22]

4.4.5 Configuration Management

Cross References:

Products:

Functions:

[Specification Management 7.2.1.6]

4.4.6 Transitioning to Software Support

Cross References:

Products:

[Computer Resources Integrated Support Document	(CRISD),
Computer System Operator's Manual	(CSOM),
Software User's Manual	(SUM)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.5 DETAILED DESIGN

4.5.1 Software Development Management

Cross References:

Products:

[Software Development Plan (SDP)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Resource Estimation 7.2.2.5,
Tracking 7.2.2.6]

4.5.2 Software Engineering

Cross References:

Products:

[Software Design Document	(SDD),
Interface Design Document	(IDD),
Software Development File	(SDF)]

Functions:

[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Detailed Design	7.1.6.5,
Design Generation	7.1.7.1,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Program Library Management	7.2.1.7,
Test Data Management	7.2.1.8,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Reusability Analysis	7.3.1.14,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Invocation Analysis	7.3.1.19,
Structured Walkthrough	7.3.1.21,
Auditing	7.3.1.22,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
Design Prototyping	7.3.2.4,
Formal Verification	7.3.3]

4.5.3 Formal Qualification Testing

Cross References:

Products:
[Software Test Description (STD)]

Functions:
[Predefined and User-Defined Forms 7.1.2.3,
Documentation Management 7.2.1.2,
Test Conditional Analysis 7.3.1.8,
Cross Reference 7.3.1.17]

4.5.4 Software Product Evaluations

Cross References:

Products:

Functions:
[Requirements Reconstruction 7.1.7.2,
Completeness Checking 7.3.1.12,
Consistency Checking 7.3.1.13,
Cross Reference 7.3.1.17,
Scanning 7.3.1.20,
Auditing 7.3.1.22]

4.5.5 Configuration Management

Cross References:

Products:

Functions:
[Specification Management 7.2.1.6]

4.5.6 Transitioning to Software Support

Cross References:

Products:

[Computer Resources Integrated Support Document	(CRISD),
Computer System Operator's Manual	(CSOM),
Software User's Manual	(SUM),
Software Programmer's Manual	(SPM),
Firmware Support Manual	(FSM)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.6 CODING AND CSU TESTING

4.6.1 Software Development Management

Cross References:

Products:
[Software Development Plan (SDP)]

Functions:
[Predefined and User-Defined Forms 7.1.2.3,
Resource Estimation 7.2.2.5,
Tracking 7.2.2.6]

4.6.2 Software Engineering

Cross References:

Products:
[Software Development File (SDF)]

Functions:
[Data Editing 7.1.1.2,
Predefined and User-Defined Forms 7.1.2.3,
Assembling 7.1.6.6,
Compilation 7.1.6.7,
Conversion 7.1.6.8,
Macro Expansion 7.1.6.9,
Structure Preprocessing 7.1.6.10,
Body Stub Generation 7.1.6.11,
Preamble Generation 7.1.6.12,
Linking/Loading 7.1.6.13,
Interpretation 7.1.6.14,
Software and System Test Communications 7.1.6.15,
Program Generation 7.1.7.3,
Test Harness Generation 7.1.7.7,
Database Management 7.2.1.1,
Documentation Management 7.2.1.2,
File Management 7.2.1.3,
Program Library Management 7.2.1.7,

4.6.2 Software Engineering (Continued)

Functions:

Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,
I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,
Comparison	7.3.1.1,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reusability Analysis	7.3.1.14,
Syntax and Semantics Checking	7.3.1.15,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Structured Walkthrough	7.3.1.21,
Auditing	7.3.1.22,
Error Checking	7.3.1.23,
Statistical Analysis	7.3.1.24,
Statistical Profiling	7.3.1.25,
Structure Checking	7.3.1.26,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
I/O Specification Analysis	7.3.1.29,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Debugging	7.3.2.5,
Executable Assertion Checking	7.3.2.6,
Constraint Evaluation (Contention)	7.3.2.7,
Coverage/Frequency Analysis	7.3.2.8,
Mutation Analysis	7.3.2.9,
Symbolic Execution	7.3.2.10,
Regression Testing	7.3.2.11,
Emulation	7.3.2.12,
Tuning	7.3.2.13,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Path and Domain Selection	7.3.2.18,
Formal Verification	7.3.3]

4.6.3 Formal Qualification Testing

Cross References:

Products:
 [Software Test Description (STD)]

Functions:

4.6.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Source Reconstruction	7.1.7.4,
Decompilation	7.1.7.5,
Disassembling	7.1.7.6,
Comparison	7.3.1.1,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Scanning	7.3.1.20,
Auditing	7.3.1.22,
Sizing Analysis	7.3.1.30,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.6.5 Configuration Management

Cross References:

Products:

Functions:

4.6.6 Transitioning to Software Support

Cross References:

Products:

[Computer System Operator's Manual	(CSOM),
Software User's Manual	(SUM),
Software Programmer's Manual	(SPM),
Firmware Support Manual	(FSM)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.7 CSC INTEGRATION AND TESTING

4.7.1 Software Development Management

Cross References:

Products:	
[Software Development Plan	(SDP)]
Functions:	
[Predefined and User-Defined Forms	7.1.2.3,
Resource Estimation	7.2.2.5,
Tracking	7.2.2.6]

4.7.2 Software Engineering

Cross References:

Products:	
[Software Development File	(SDF)]
Functions:	
[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Assembling	7.1.6.6,
Compilation	7.1.6.7,
Conversion	7.1.6.8,
Macro Expansion	7.1.6.9,
Structure Preprocessing	7.1.6.10,
Body Stub Generation	7.1.6.11,
Preamble Generation	7.1.6.12,
Linking/Loading	7.1.6.13,
Interpretation	7.1.6.14,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Program Library Management	7.2.1.7,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,

4.7.2 Software Engineering (Continued)

Functions:

I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,
Comparison	7.3.1.1,
Data Flow Analysis	7.3.1.3,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Error Checking	7.3.1.23,
Statistical Analysis	7.3.1.24,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
I/O Specification Analysis	7.3.1.29,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Debugging	7.3.2.5,
Executable Assertion Checking	7.3.2.6,
Coverage/Frequency Analysis	7.3.2.8,
Mutation Analysis	7.3.2.9,
Regression Testing	7.3.2.11,
Resource Utilization	7.3.2.12,
Emulation	7.3.2.13,
Timing Analysis	7.3.2.14,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Problem Report Analysis	7.3.4]

4.7.3 Formal Qualification Testing

Cross References:

Products:

[Software Test Description (STD)]

Functions:

[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,
I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,
Comparison	7.3.1.1,
Test Condition Analysis	7.3.1.8,
Correctness Checking	7.3.1.11,
Cross Reference	7.3.1.17,
Auditing	7.3.1.22,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Executable Assertion Checking	7.3.2.6,
Real Time Analysis	7.3.2.17]

4.7.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Requirements Reconstruction	7.1.7.2,
Source Reconstruction	7.1.7.4,
Decompilation	7.1.7.5,
Disassembling	7.1.7.6,
Comparison	7.3.1.1,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Scanning	7.3.1.20,
Auditing	7.3.1.22,
Sizing Analysis	7.3.1.30,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.7.5 Configuration Management

Cross References:

Products:

Functions:

4.7.6 Transitioning to Software Support

Cross References:

Products:

[Computer System Operator's Manual	(CSOM),
Software User's Manual	(SUM),
Software Programmer's Manual	(SPM),
Firmware Support Manual	(FSM)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3]

4.8 CSCI TESTING

4.8.1 Software Development Management

Cross References:

Products:	
[Software Development Plan	(SDP)]
Functions:	
[Predefined and User-Defined Forms	7.1.2.3,
Resource Estimation	7.2.2.5,
Tracking	7.2.2.6]

4.8.2 Software Engineering

Cross References:

Products:	
[Software Development File	(SDF)]
Functions:	
[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Assembling	7.1.6.6,
Compilation	7.1.6.7,
Conversion	7.1.6.8,
Macro Expansion	7.1.6.9,
Structure Preprocessing	7.1.6.10,
Preamble Generation	7.1.6.12,
Linking/Loading	7.1.6.13,
Interpretation	7.1.6.14,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Program Library Management	7.2.1.7,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,

4.8.2 Software Engineering (Continued)

Functions:

I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,
Import/Export	7.2.3.6,
Comparison	7.3.1.1,
Data Flow Analysis	7.3.1.3,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Error Checking	7.3.1.23,
Statistical Analysis	7.3.1.24,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
I/O Specification Analysis	7.3.1.29,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Debugging	7.3.2.5,
Executable Assertion Checking	7.3.2.6,
Coverage/Frequency Analysis	7.3.2.8,
Mutation Analysis	7.3.2.9,
Regression Testing	7.3.2.11,
Resource Utilization	7.3.2.12,
Emulation	7.3.2.13,
Timing Analysis	7.3.2.14,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Problem Report Analysis	7.3.4]

4.8.3 Formal Qualification Testing

Cross References:

Products:

[Software Test Description	(STD),
Software Test Report	(STR)]

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,
I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,
Comparison	7.3.1.1,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Statistical Analysis	7.3.1.24,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Executable Assertion Checking	7.3.2.6,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.8.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Sizing Analysis	7.3.1.30,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.8.5 Configuration Management

Cross References:

Products:
[Version Description Document (VDD)]

Functions:
[Predefined and User-Defined Forms 7.1.2.3]

4.8.6 Transitioning to Software Support

Cross References:

Products:
[Computer System Operator's Manual (CSOM),
Software User's Manual (SUM),
Software Programmer's Manual (SPM),
Firmware Support Manual (FSM)]

Functions:
[Predefined and User-Defined Forms 7.1.2.3]

4.9 SYSTEM INTEGRATION AND TESTING

4.9.1 Software Development Management

Cross References:

Products:

Functions:

4.9.2 Software Engineering

Cross References:

Products:

[Software Development File (SDF)]

Functions:

[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Assembling	7.1.6.6,
Compilation	7.1.6.7,
Conversion	7.1.6.8,
Macro Expansion	7.1.6.9,
Structure Preprocessing	7.1.6.10,
Preamble Generation	7.1.6.12,
Linking/Loading	7.1.6.13,
Interpretation	7.1.6.14,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Program Library Management	7.2.1.7,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,
I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,

4.9.2 Software Engineering (Continued)

Functions:

Import/Export	7.2.3.6,
Comparison	7.3.1.1,
Data Flow Analysis	7.3.1.3,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Error Checking	7.3.1.23,
Statistical Analysis	7.3.1.24,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
I/O Specification Analysis	7.3.1.29,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Debugging	7.3.2.5,
Executable Assertion Checking	7.3.2.6,
Coverage/Frequency Analysis	7.3.2.8,
Mutation Analysis	7.3.2.9,
Regression Testing	7.3.2.11,
Resource Utilization	7.3.2.12,
Emulation	7.3.2.13,
Timing Analysis	7.3.2.14,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Problem Report Analysis	7.3.4]

4.9.3 Formal Qualification Testing

Cross References:

Products:

[Software Test Description	(STD),
Software Test Report	(STR)]

Functions:

[Software and System Test Communications	7.1.6.15,
Data and Error Logging	7.2.1.11,
Runtime Environment	7.2.3.5,
Statistical Profiling	7.3.1.25,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Timing Analysis	7.3.2.14]

4.9.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Sizing Analysis	7.3.1.30,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.9.5 Configuration Management

Cross References:

Products:

[Version Description Document

(VDD)]

Functions:

[Predefined and User-Defined Forms

7.1.2.3]

4.9.6 Transitioning to System Support

Cross References:

Products:

Functions:

4.10 OPERATIONAL TESTING AND EVALUATION

4.10.1 Software Development Management

Cross References:

Products:

Functions:

4.10.2 Software Engineering

Cross References:

Products:

[Software Development File (SDF)]

Functions:

[Data Editing	7.1.1.2,
Predefined and User-Defined Forms	7.1.2.3,
Assembling	7.1.6.6,
Compilation	7.1.6.7,
Conversion	7.1.6.8,
Macro Expansion	7.1.6.9,
Structure Preprocessing	7.1.6.10,
Preamble Generation	7.1.6.12,
Linking/Loading	7.1.6.13,
Interpretation	7.1.6.14,
Software and System Test Communications	7.1.6.15,
Program Generation	7.1.7.3,
Test Harness Generation	7.1.7.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Program Library Management	7.2.1.7,
Test Data Management	7.2.1.8,
Data and Error Logging	7.2.1.11,
Status Display	7.2.1.12,
I/O Support	7.2.3.2,
Runtime Environment	7.2.3.5,

4.10.2 Software Engineering (Continued)

Functions:

Import/Export	7.2.3.6,
Comparison	7.3.1.1,
Data Flow Analysis	7.3.1.3,
Interface Analysis	7.3.1.5,
Testability Analysis	7.3.1.7,
Test Condition Analysis	7.3.1.8,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Error Checking	7.3.1.23,
Statistical Analysis	7.3.1.24,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
I/O Specification Analysis	7.3.1.29,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Random Test Generation	7.3.1.32,
Debugging	7.3.2.5,
Executable Assertion Checking	7.3.2.6,
Coverage/Frequency Analysis	7.3.2.8,
Mutation Analysis	7.3.2.9,
Regression Testing	7.3.2.11,
Resource Utilization	7.3.2.12,
Emulation	7.3.2.13,
Timing Analysis	7.3.2.14,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17,
Problem Report Analysis	7.3.4]

4.10.3 Formal Qualification Testing

Cross References:

Products:

[Software Test Description	(STD),
Software Test Report	(STR)]

Functions:

[Software and System Test Communications	7.1.6.15,
Data and Error Logging	7.2.1.11,
Runtime Environment	7.2.3.5,
Statistical Profiling	7.3.1.25,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.10.4 Software Product Evaluations

Cross References:

Products:

Functions:

[Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Auditing	7.3.1.22,
Sizing Analysis	7.3.1.30,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14]

4.10.5 Configuration Management

Cross References:

Products:

Functions:

[Evaluation Results Management

7.2.1.9]

4.10.6 Transitioning to System Support

Cross References:

Products:

Functions:

4.11 CHANGE REQUIREMENTS

4.11.1 System Development Management

Cross References:

Products:

Functions:

4.11.2 System Engineering

Cross References:

Products:

[Engineering Change Proposal (ECP)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3,
Change Impact Analysis 7.3.5.1]

4.11.3 Formal Qualification Testing

Cross References:

Products:

Functions:

4.11.4 System Product Evaluations

Cross References:

Products:

Functions:

4.11.5 Configuration Management

Cross References:

Products:

[Specification Change Notice (SCN)]

Functions:

[Predefined and User-Defined Forms 7.1.2.3]

4.11.6 Transitioning to System Support

Cross References:

Products:

Functions:

4.12 GLOBAL

4.12.1 System Development Management

Cross References:

Products:

Functions:

[Encryption	7.1.6.18,
Decryption	7.1.6.19,
Database (Object) Management	7.2.1.1,
Documentation Management	7.2.1.2,
File Management	7.2.1.3,
Electronic Mail	7.2.1.4,
Electronic Conferencing	7.2.1.5,
Performance Monitoring	7.2.1.10,
Cost Estimation	7.2.2.1,
Quality Specification	7.2.2.2,
Scheduling	7.2.2.3,
Work Breakdown Structure	7.2.2.4,
Risk Analysis	7.2.2.9,
Access Control	7.2.3.7]

4.12.2 System Engineering

Cross References:

Products:

Functions:

[Text Editing	7.1.1.1,
Graphics Editing	7.1.1.3,
MIL-STD Format	7.1.2.1,
Table of Contents	7.1.2.2,
On-Line Assistance Processing	7.1.3,
Sort/Merge	7.1.4,
Graphics Generation	7.1.5,
Compression	7.1.6.16,
Expansion	7.1.6.17,
Command Language Processing	7.2.3.1,
Input/Output Support	7.2.3.2,
Kernel	7.2.3.3,
Math/Statistics	7.2.3.4,
Import/Export	7.2.3.6,
Job Scheduling	7.2.3.8,
Resource Management	7.2.3.9]

4.12.3 Formal Qualification Testing

Cross References:

Products:

Functions:

4.12.4 System Product Evaluations

Cross References:

Products:

Functions:

[Comparison
Spelling Checking

7.3.1.1,
7.3.1.2]

4.12.5 Configuration Management

Cross References:

Products:

Functions:

[Configuration Management

7.2.2.7]

4.12.6 Transitioning to System Support

Cross References:

Products:

Functions:

5. APSE TOOL CATEGORIES

This chapter deals with APSEs, toolsets, and tools. E&V technology is always applied to APSEs or their components and thus this index will be a natural starting point for many users of the RM. Additional discussions of APSE definitions and views are provided in Sections 3.1 and 3.2, respectively. The following sections describe the APSE and its components and relate the components to functions as shown in Fig. 5-1. The relationships between tools and functions given in this chapter are typical (or traditional) relationships. The real capabilities should be determined by the tool specifications, marketing claims, or the like.

There is also a relationship between tools and attributes. Some attributes are related to qualities of the function(s) performed (such as completeness) by the software, while other attributes relate to non-functional aspects (such as modularity) of the software. Thus, to determine how to assess a tool, both the tool-function relationships and the attributes must be explored to identify the metrics to be used.

APSEs are the highest level of toolset. APSEs should contain all the tools and toolsets needed to support the full spectrum of project activities. APSEs, like toolsets, have characteristics that may be assessed as a whole [Whole APSE Assessment Issues 3]. Toolsets are one or more tools which are intimately coupled and support a related set of functions within the APSE. Each section in this chapter describes a toolset category that might be found in an APSE. The section titles are:

G-16376
8-11-89

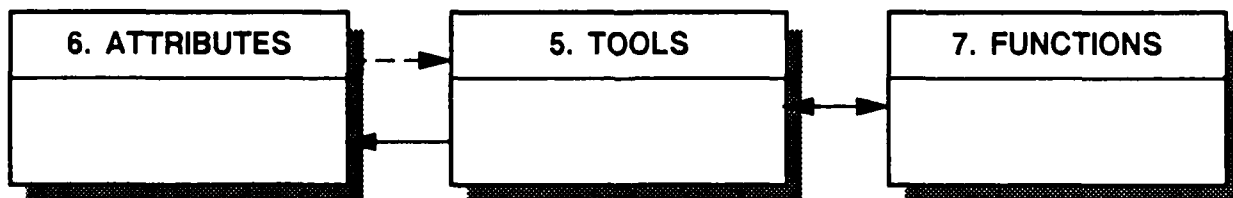


Figure 5-1 Tool Relationships

E&V Reference Manual, Version 2.0

- Computer Management System
- Information Management System
- Tool Support Components
- Distributed APSE Support
- Project Management System
- Desktop System
- Configuration Management System
- Document Generation System
- Requirements/Design Support
- Distributed System Development and Runtime Support
- Implementation Support
- Compilation System
- Target Code Generation and Analysis System
- Test System.

The subsections within each section give the tool categories that comprise the toolset. The complete, two-level tool taxonomy may be seen by examining the Table of Contents under Chapter 5. Not all toolsets are necessary for an APSE; likewise, for a specific toolset, all tools may not always be present.

5.1 COMPUTER MANAGEMENT SYSTEM

Description:

Those tools needed to access, use, and maintain the hardware and software which consists of the APSE and the system on which it runs.

5.1.1 Command Language Interface

Cross References:

Functions:

[Command Language Processing 7.2.3.1]

5.1.2 Archive, Backup, and Retrieval System

Cross References:

Functions:

[Compression 7.1.6.16,
Expansion 7.1.6.17,
Import/Export 7.2.3.6]

5.1.3 Security System

Cross References:

Functions:

[Encryption 7.1.6.18,
Decryption 7.1.6.19,
Kernel 7.2.3.3,
Access Control 7.2.3.7]

5.1.4 Job Scheduler

Cross References:

Functions:

[Kernel	7.2.3.3,
Runtime Environment	7.2.3.5,
Job Scheduling	7.2.3.8]

5.1.5 Resource Controller

Cross References:

Functions:

[Kernel	7.2.3.3,
Runtime Environment	7.2.3.5,
Resource Management	7.2.3.9]

5.1.6 Import/Export System

Cross References:

Functions:

[Import/Export	7.2.3.6]

5.1.7 On-Line Assistance

Cross References:

Functions:
[On-Line Assistance Processing 7.1.3]

5.1.8 Input and Output Services

Cross References:

Functions:
[Input/Output Support 7.2.3.2]

5.1.9 Performance Monitor

Cross References:

Functions:
[Performance Monitoring 7.2.1.10]

5.2 INFORMATION MANAGEMENT SYSTEM

Description:

Those tools needed to control the information flow during the development of software systems. This includes the organization, accession, modification, dissemination, and processing of any associated information.

5.2.1 File Manager

Cross References:

Functions:
[File Management 7.2.1.3]

5.2.2 Database Manager

Cross References:

Functions:
[Sort/Merge 7.1.4,
Database Management 7.2.1.1]

5.3 TOOL SUPPORT COMPONENTS

Description:

Those tools needed to provide interfaces between other tools and host system services.

5.3.1 Virtual Operating System

Cross References:

Functions:

[Database (Object) Management	7.2.1.1,
File Management	7.2.1.3,
Input/Output Support	7.2.3.2,
Kernel	7.2.3.3,
Runtime Environment	7.2.3.5,
Import/Export	7.2.3.6,
Access Control	7.2.3.7,
Job Scheduling	7.2.3.8,
Resource Management	7.2.3.9]

5.3.2 Window Manager

Cross References:

Functions:

[Input/Output Support	7.2.3.2,
Kernel	7.2.3.3]

5.3.3 Language Bindings to Standard Interface Specification Implementations

Cross References:

Functions:

5.3.4 I/O Pipes

Cross References:

Functions:	
[Input/Output Support	7.2.3.2]

5.3.5 RAM Cache

Cross References:

Functions:	
[Input/Output Support	7.2.3.2,
Emulation	7.3.2.13]

5.4 DISTRIBUTED APSE SUPPORT

Description:

Those tools needed to provide the cooperative communication in supporting the development of mission critical software at diverse geographical locations.

5.5 PROJECT MANAGEMENT SYSTEM

Description:

Those tools needed to plan, develop, and maintain an applications system.

5.5.1 Cost Estimator

Cross References:

Functions:
[Cost Estimation 7.2.2.1]

5.5.2 Size Estimator

Cross References:

Functions:
[Sizing Analysis 7.3.1.30,
Resource Utilization 7.3.2.12]

5.5.3 Scheduler

Cross References:

Functions:
[Scheduling 7.2.2.3]

5.5.4 Work Breakdown Structure Editor

Cross References:

Functions:
[Work Breakdown Structure 7.2.2.4]

5.5.5 Resource Estimator

Cross References:

Functions:
[Resource Estimation 7.2.2.5]

5.5.6 Tracking

Cross References:

Functions:
[Tracking 7.2.2.6]

5.5.7 Problem Report Analyzer

Cross References:

Functions:
[Problem Report Analysis 7.3.4]

5.5.8 Change Impact Analyzer

Cross References:

Functions:
[Change Impact Analysis 7.3.5.1]

5.5.9 Analysis and Reporting

Cross References:

Functions:
[Tracking 7.2.2.6,
Risk Analysis 7.2.2.9]

5.6 DESKTOP SYSTEM

Description:

Those tools needed to do the every-day, administrative tasks of business.

5.6.1 Spreadsheet

Cross References:

Functions:

[Predefined and User-Defined Forms
Math/Statistics

7.1.2.3,
7.2.3.4]

5.6.2 Calculator

Cross References:

Functions:

[Math/Statistics

7.2.3.4]

5.6.3 Address/Phone Book

Cross References:

Functions:

[Sort/Merge
Electronic Mail
Electronic Conferencing

7.1.4,
7.2.1.4,
7.2.1.5]

5.6.4 Electronic Mail

Cross References:

Functions:

[Sort/Merge

Electronic Mail

7.1.4,

7.2.1.4]

5.6.5 Electronic Conferencing

Cross References:

Functions:

[Electronic Conferencing

7.2.1.5]

5.6.6 Calendar

Cross References:

Functions:

[Scheduling

7.2.2.3]

5.6.7 Dictionary/Thesaurus

Cross References:

Functions:

[Sort/Merge

Comparison

Spelling Checking

7.1.4,

7.3.1.1,

7.3.1.2]

5.7 CONFIGURATION MANAGEMENT SYSTEM

Description:

Those tools needed to control the contents of software systems. This includes monitoring the status, preserving the integrity of released and developed versions, and controlling the effects of changes throughout the lifetime of the software system.

5.7.1 Configuration Identification

Cross References:

Functions:
[Configuration Management 7.2.2.7]

5.7.2 Configuration Control

Cross References:

Functions:
[Program Library Management 7.2.1.7,
Configuration Management 7.2.2.7]

5.7.3 Configuration Status Accounting

Cross References:

Functions:
[Tracking 7.2.2.6,
Configuration Management 7.2.2.7]

5.7.4 Version Control

Cross References:

Functions:

[Program Library Management
Configuration Management

7.2.1.7,
7.2.2.7]

5.7.5 History

Cross References:

Functions:

[Tracking
Configuration Management

7.2.2.6,
7.2.2.7]

5.8 DOCUMENT GENERATION SYSTEM

Description:

Those tools needed to develop and produce documents.

5.8.1 Document Manager

Cross References:

Functions:

[Document Management 7.2.1.2]

5.8.2 Word Processor

Cross References:

Functions:

[Text Editing 7.1.1.1,
Table of Contents 7.1.2.2,
Predefined and User-Defined Forms 7.1.2.3,
Syntax And Semantics Checking 7.3.1.15]

5.8.3 Spelling Checker

Cross References:

Functions:

[Spelling Checking 7.3.1.2]

5.8.4 Graphics Generator

Cross References:

Functions:

[Graphics Editing	7.1.1.3,
Graphics Generation	7.1.5]

5.8.5 Formatter

Cross References:

Functions:

[MIL-STD Format	7.1.2.1,
Table of Contents	7.1.2.2,
Predefined and User-Defined Forms	7.1.2.3]

5.9 REQUIREMENTS/DESIGN SUPPORT

Description:

Those tools needed for the definition, specification, and design of software.

5.9.1 Definition Language Processor

Cross References:

Functions:

[Text Editing	7.1.1.1,
System Requirements	7.1.6.1,
Software Requirements	7.1.6.2]

5.9.2 Specification Language Processor

Cross References:

Functions:

[Text Editing	7.1.1.1,
Preliminary Design	7.1.6.4,
Detailed Design	7.1.6.5,
Design Generation	7.1.7.1,
Specification Management	7.2.1.6,
Requirements Simulation	7.3.2.1]

5.9.3 Enterprise Model

Cross References:

Functions:

[Graphics Editing	7.1.1.3,
Graphics Generation	7.1.5,
System Requirements	7.1.6.1,
Database Management	7.2.1.1,
Simulation and Modeling	7.3.2.3]

5.9.4 Data Model/Dictionary

Cross References:

Functions:

[Text Editing	7.1.1.1,
Graphics Editing	7.1.1.3,
Graphics Generation	7.1.5,
Software Requirements	7.1.6.2,
Preliminary Design	7.1.6.4,
Detailed Design	7.1.6.5,
Design Generation	7.1.7.1,
Database Management	7.2.1.1,
Specification Management	7.2.1.6,
Traceability Analysis	7.3.1.6,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Cross Reference	7.3.1.17,
Type Analysis	7.3.1.27,
Simulation and Modeling	7.3.2.3]

5.9.5 Process Model

Cross References:

Functions:

[Graphics Editing	7.1.1.3,
Graphics Generation	7.1.5,
System Requirements	7.1.6.1,
Software Requirements	7.1.6.2,
Preliminary Design	7.1.6.4,
Detailed Design	7.1.6.5,
Design Generation	7.1.7.1,
Database Management	7.2.1.1,
Specification Management	7.2.1.6,
Interface Analysis	7.3.1.5,
Traceability Analysis	7.3.1.6,
Correctness Checking	7.3.1.11,
Completeness Checking	7.3.1.12,
Consistency Checking	7.3.1.13,
Reusability Analysis	7.3.1.14,
Cross Reference	7.3.1.17,
Invocation Analysis	7.3.1.19,
Simulation and Modeling	7.3.2.3]

5.9.6 Simulators

Cross References:

Functions:

[Graphics Generation	7.1.5,
Requirements Simulation	7.3.2.1,
Simulation and Modeling	7.3.2.3]

5.9.7 Prototyping Tools

Cross References:

Functions:

[Text Editing	7.1.1.1,
Graphics Editing	7.1.1.3,
Graphics Generation	7.1.5,
Preliminary Design	7.1.6.4,
Design Generation	7.1.7.1,
Program Generation	7.1.7.3,
Requirements Prototyping	7.3.2.2,
Simulation and Modeling	7.3.2.3,
Design Prototyping	7.3.2.4,
Emulation	7.3.2.13]

5.9.8 Design Library Manager

Cross References:

Functions:

[Requirements Reconstruction	7.1.7.2,
Database Management	7.2.1.1,
Program Library Management	7.2.1.7,
Quality Assessment	7.2.2.8,
Interface Analysis	7.3.1.5,
Reusability Analysis	7.3.1.14]

5.10 DISTRIBUTED SYSTEMS DEVELOPMENT AND RUNTIME SUPPORT

Description:

Those tools needed to support software development for distributed processing systems, and to provide runtime support for distributed processing systems.

5.11 IMPLEMENTATION SUPPORT

Description:

Those tools needed to construct the source code for software. The source code being constructed could be either for an applications program or for another tool in the APSE (meta-tool).

5.11.1 Applications (Code) Generator

Cross References:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Macro Expansion	7.1.6.9,
Body Stub Generation	7.1.6.11,
Preamble Generation	7.1.6.12,
Program Generation	7.1.7.3,
Interface Analysis	7.3.1.5,
Invocation Analysis	7.3.1.19]

5.11.2 Database Schema Generator

Cross References:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Structure Preprocessing	7.1.6.10]

5.11.3 Report Generator

Cross References:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Program Generation	7.1.7.3]

5.11.4 Screen Generator

Cross References:

Functions:

[Predefined and User-Defined Forms	7.1.2.3,
Graphics Generation	7.1.5,
Program Generation	7.1.7.3]

5.11.5 Syntax-Directed (Language-Sensitive) Editor

Cross References:

Functions:

[Text Editing	7.1.1.1,
Predefined and User-Defined Forms	7.1.2.3,
Macro Expansion	7.1.6.9,
Syntax and Semantics Checking	7.3.1.15]

5.11.6 Data Definition Language Processor

Cross References:

Functions:

[Database Management	7.2.1.1]

5.11.7 Data Manipulation Language Processor

Cross References:

Functions:

[Database Management	7.2.1.1]

5.11.8 Reusable Components Library

Cross References:

Functions:

[Requirements Reconstruction	7.1.7.2,
Database Management	7.2.1.1,
Program Library Management	7.2.1.7,
Quality Assessment	7.2.2.8,
Interface Analysis	7.3.1.5,
Reusability Analysis	7.3.1.14]

5.12 COMPILATION SYSTEM

Description:

For the purposes of this document, the compilation system is defined as those APSE components which are Ada-specific: the compiler, the code generator, the program library management system, and the runtime system. All components required for validation are found here.

5.12.1 Translating Editor

Cross References:

Functions:	
[Text Editing	7.1.1.1,
Compilation	7.1.6.7,
Syntax And Semantics Checking	7.3.1.15]

5.12.2 Compiler (Front End)

Cross References:

Functions:	
[Compilation	7.1.6.7,
Syntax And Semantics Checking	7.3.1.15]

5.12.3 Compiler (Code Generator — Back End)

Cross References:

Functions:	
[Compilation	7.1.6.7]

5.12.4 Interpreter

Cross References:

Functions:
[Interpretation 7.1.6.14]

5.12.5 Program Library Manager

Cross References:

Functions:
[Program Library Management 7.2.1.7]

5.12.6 Runtime System

Cross References:

Functions:
[Input/Output Support 7.2.3.2,
Runtime Environment 7.2.3.5]

5.13 TARGET CODE GENERATION AIDS AND ANALYSIS TOOLSET

Description:

Those tools needed to provide visibility into target processes during program execution. In addition, the tools are used in the case where the host computer is also the target computer.

5.13.1 Host-Target System Cross-Assembler

Cross References:

Functions:

[Assembler
Conversion

7.1.6.6,
7.1.6.8]

5.13.2 Host-Based Target Linker

Cross References:

Functions:

[Linking/Loading

7.1.6.13]

5.13.3 Host-Based Target Loader

Cross References:

Functions:

[Linking/Loading

7.1.6.13]

5.13.4 Host-Based Target System Instruction-Level Simulator

Cross References:

Functions:
[Simulation and Modeling 7.3.2.3]

5.13.5 Host-Based Target System Instruction-Level Emulator

Cross References:

Functions:
[Emulation 7.3.2.13]

5.13.6 Host-Based Target Code Symbolic Debugger

Cross References:

Functions:
[Debugging 7.3.2.5]

5.13.7 Host-to-Target Downloader

Cross References:

Functions:
[Import/Export 7.2.3.6]

5.13.8 Target-to-Host Uploader

Cross References:

Functions:
[Import/Export 7.2.3.6]

5.14 TEST SYSTEM

Description:

Those tools needed to support and facilitate the planning, development, execution, evaluation, and documentation of tests of software.

5.14.1 Static Analyzers

Cross References:

Functions:

[Data Flow Analysis	7.3.1.3,
Functional Analysis	7.3.1.4,
Interface Analysis	7.3.1.5,
Traceability Analysis	7.3.1.6,
Testability Analysis	7.3.1.7,
Quality Measurement	7.3.1.9,
Complexity Measurement	7.3.1.10,
Completeness Measurement	7.3.1.12,
Consistency Checking	7.3.1.13,
Reusability Analysis	7.3.1.14,
Syntax and Semantics Checking	7.3.1.15,
Reachability Analysis	7.3.1.16,
Cross Reference	7.3.1.17,
Maintainability Analysis	7.3.1.18,
Invocation Analysis	7.3.1.19,
Scanning	7.3.1.20,
Structured Walkthrough	7.3.1.21,
Auditing	7.3.1.22,
Structure Checking	7.3.1.26,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28,
Formal Verification	7.3.3]

5.14.2 Tool Building Services

Cross References:

Functions:
[Program Generation 7.1.7.3]

5.14.3 Test Building Services

Cross References:

Functions:
[Test Condition Analysis 7.3.1.8,
I/O Specification Analysis 7.3.1.29,
Random Test Generation 7.3.1.32,
Executable Assertion Checking 7.3.2.6,
Constraint Evaluation 7.3.2.7,
Coverage/Frequency Analysis 7.3.2.8,
Mutation Analysis 7.3.2.9,
Symbolic Execution 7.3.2.10,
Path and Domain Selection 7.3.2.18]

5.14.4 Test Description and Preparation Services

Cross References:

Functions:
[Data Editing 7.1.1.2,
Body Stub Generation 7.1.6.11,
Software and System Test Communications 7.1.6.15,
Test Data Management 7.2.1.8,
Comparison 7.3.1.1]

5.14.5 Test Execution Services

Cross References:

Functions:

[Test Harness Generation	7.1.7.7,
Data and Error Logging	7.2.1.11,
Quality Measurement	7.3.1.9,
Regression Testing	7.3.2.11,
Emulation	7.3.2.13]

5.14.6 Test Analysis Services

Cross References:

Functions:

[Evaluation Results Management	7.2.1.9,
Status Display	7.2.1.12,
Correctness Checking	7.3.1.11,
Cross Reference	7.3.1.17,
Statistical Analysis	7.3.1.24,
Statistical Profiling	7.3.1.25,
Sizing Analysis	7.3.1.30,
Data Reduction and Analysis	7.3.1.31,
Resource Utilization	7.3.2.12,
Timing Analysis	7.3.2.14,
Tuning	7.3.2.15,
Reliability Analysis	7.3.2.16,
Real Time Analysis	7.3.2.17]

5.14.7 Decision Support Services

Cross References:

Functions:

[Error Checking	7.3.1.23,
Problem Report Analysis	7.3.4,
Change Impact Analysis	7.3.5.1]

6.

ATTRIBUTES

Attributes are the characteristics of tools or "whole APSEs" which the E&V technology user evaluates (or validates) to make assessments and comparisons. Therefore, the attributes are integral to finding E&V technology. This manual uses a hierarchy of software attributes derived from one presented in a report by the Rome Air Development Center [RADC 1985]. The focus of the RADC report is planning and designing quality into application software throughout the software life cycle. Nevertheless, it provides an excellent framework for understanding all the issues that surround software quality and is recommended for those users seeking more details into this complex process. The RADC report is therefore chosen as a basis for this section of the E&V Reference Manual even though the focus here is on system and support software and also on specifying and assessing attributes for software that is already developed. The remainder of this introductory section is used to summarize some of the issues that must be dealt with in understanding the software attributes. These are:

- The attribute hierarchy
- The functional dependence of attributes
- Guidance in the selection of attributes
- Attribute interrelationships.

The first two levels of the attribute hierarchy are shown in Table 6-1. The highest level of the hierarchy (performance, design, and adaptation) shows three broad categories of acquisition concerns to potential users with regard to software. The next level shows quality factors which are user-oriented terms, each representing an aspect of software (or tool) quality. The quality factors can themselves be decomposed into various criteria as shown in Table 6-2. The criteria are software-oriented terms representing software characteristics. The degree to which these characteristics are present in the software is an indication of the degree of presence of a quality factor. The criteria can support more than one of the software quality factors at the next higher level.

Table 6-1 Top Level Attribute Hierarchy

G-04769

ACQUISITION CONCERN	USER CONCERN	QUALITY FACTOR
PERFORMANCE- HOW WELL DOES IT FUNCTION?	HOW WELL DOES IT UTILIZE A RESOURCE? HOW SECURE IS IT? WHAT CONFIDENCE CAN BE PLACED IN WHAT IT DOES? HOW WELL WILL IT PERFORM UNDER ADVERSE CONDITIONS? HOW EASY IS IT TO USE?	EFFICIENCY INTEGRITY RELIABILITY SURVIVABILITY USABILITY
DESIGN- HOW WELL IS IT DESIGNED?	HOW WELL DOES IT CONFORM TO THE REQUIREMENTS? HOW EASY IS IT TO REPAIR? HOW EASY IS IT TO VERIFY ITS PERFORMANCE?	CORRECTNESS MAINTAINABILITY VERIFIABILITY, TESTABILITY
ADAPTATION- HOW ADAPTABLE IS IT?	HOW EASY IS IT TO EXPAND, CHANGE, OR UPGRADE ITS CAPABILITY OR PERFORMANCE? HOW EASY IS IT TO INTERFACE WITH ANOTHER SYSTEM? HOW EASY IS IT TO CONVERT FOR USE IN ANOTHER APPLICATION? HOW EASY IS IT TO TRANSPORT TO ANOTHER ENVIRONMENT?	EXPANDABILITY, FLEXIBILITY INTEROPERABILITY REUSABILITY TRANSPORTABILITY

The quality criteria can also be decomposed into metrics which are software-oriented details of the software characteristics. These metrics represent specific questions, checklists, or other tests that are used to determine the extent to which the tool has that characteristic. The metrics must often be tailored to the particular function(s) that the software performs. Other metrics do not deal with functionality, but with aspects of the software that are independent of the function(s) performed.

Table 6-2 Complete Attribute Hierarchy

G-16184
8-10-89

Acquisition Concern	Acquisition Concern		Performance 6.1					Design 6.2			Adaptation 6.3			
	Factor	Criterion/Selection	1. Efficiency	2. Integrity	3. Reliability	4. Survivability	5. Usability	1. Correctness	2. Maintainability	3. Verifiability	1. Expandability	2. Interoperability	3. Reusability	4. Transportability
Performance	Accuracy	6.4.1			X									
	Anomaly management (fault or error tolerance, robustness)	6.4.2			X	X								
	Autonomy	6.4.5				X								
	Capacity	6.4.6					X							
	Communication effectiveness	6.4.8	X											
	Cost	6.4.11					X							
	Distributedness	6.4.12				X								
	Maturity	6.4.18			X		X							
	Operability (communicativeness)	6.4.20					X							
	Power	6.4.21					X							
	Processing (execution) effectiveness	6.4.22	X											
	Reconfigurability	6.4.24				X								
	Required configuration	6.4.26					X							
	Storage effectiveness	6.4.31	X											
	System accessibility	6.4.32		X										
	Traning	6.4.36					X							
Design	Completeness	6.4.9						X						
	Consistency	6.4.10						X	X					
	Traceability	6.4.35						X						
	Visibility (test availability)	6.4.38							X	X				
Adaptation	Application independence	6.4.3											X	
	Augmentability	6.4.4									X			
	Commonality (data and communication)	6.4.7										X		
	Document accessibility	6.4.13											X	
	Functional overlap	6.4.14										X		
	Functional scope	6.4.15											X	
	Generality	6.4.16									X		X	
	Rehostability (independence)	6.4.25										X	X	X
	Retargetability (independence)	6.4.27									X	X	X	X
	System clarity	6.4.33											X	
	System compatability	6.4.34										X		
	Virtuality	6.4.37									X			
General	Granularity	6.4.17				X	X		X	X	X		X	
	Modularity	6.4.19				X			X	X	X	X	X	X
	Proprietary rights	6.4.23							X		X			
	Self-descriptiveness	6.4.28							X	X	X		X	X
	Simplicity	6.4.29			X				X	X	X		X	
	Software production vehicle(s)	6.4.30							X		X		X	X

E&V Reference Manual, Version 2.0

The RADC report not only defines the software quality factors, criteria, and metrics, it also provides guidance for their selection and use. In particular, there is guidance for selecting attributes based on:

- System characteristics
- Complementary factors
- Shared criteria
- Attribute interrelationships.

The desired software quality factors should be tailored to the characteristics of the software being built and the development environment. In fact, some of the tools selected for the development environment will probably be based on the system characteristics. For example, if a tool or environment is to have a long life cycle, the E&V user should be concerned with expandability and maintainability. Likewise, if the system being built is a real-time application, the developer should consider acquiring tools that can assess the efficiency of the system.

Another consideration when selecting quality factors to be assessed is the effect of low quality levels among complementary factors. Four factors are complementary to most other factors and should influence the selection of important factors. Table 6-3 shows the complementary factors:

- Reliability
- Correctness
- Maintainability
- Verifiability.

Quality levels for the other specified factors are difficult to measure accurately when there are low quality levels for any of the four complementary factors. For example, if a high quality level for reliability is specified, high quality levels for correctness and verifiability should also be specified. This is because if the scores for reliability are high, but the software is incorrect or difficult to verify, the true reliability may be low due to incorrect software or uncertainty in verification. The complementary quality factors show that the attributes are not independent. Any project,

Table 6-3 Complementary Software Quality Factors

G-04768

COMPLEMENTARY QUALITY FACTOR QUALITY FACTOR SPECIFIED	E F F I C I E N C Y	I N T E G R I T Y	R E L I A B I L I T Y	S U R V I V A B I L I T Y	U S A B I L I T Y	C O R R E C T N E S S	M A I N T A I N A B I L I T Y	V E R I F I A B I L I T Y	E X P A N D A B I L I T Y	I N T E R O P E R A B I L I T Y	R E U S A B I L I T Y	T R A N S P O R T A B I L I T Y
EFFICIENCY												
INTEGRITY			*			*		*				
RELIABILITY						*		*				
SURVIVABILITY			*			*		*				
USABILITY			*			*	*	*				
CORRECTNESS												
MAINTAINABILITY			*			*						
VERIFIABILITY, TESTABILITY			*			*	*					
EXPANDABILITY, FLEXIBILITY			*			*	*	*				
INTEROPERABILITY			*			*	*	*				
REUSABILITY			*			*	*	*				
TRANSPORTABILITY			*			*	*	*				

* = DEPENDENCY

regardless of the type of system or application, should consider the complementary factors in the quality specifications.

The criteria that are attributes of more than one quality factor are shared criteria. For example, Table 6-2 shows simplicity is a criterion for five of the factors. The beneficial effect of this is that these attributes are built into the software only once; likewise, assessment of these

attributes need only be done once. Therefore, costs associated with specifying factors that share common criteria are generally less than costs associated with specifying factors that do not share criteria.

Assigning more than one quality factor to an APSE component can have either a beneficial or an adverse effect, depending on the combination of factors that have been specified. Some factors have criteria that conflict with another factor; and some have criteria that cooperate with another factor. Attribute criteria affecting other factors are shown in Table 6-4. Attributes that are basic criteria of a factor are identified with an "x"; criteria that are in a positive or beneficial relationship with another factor are identified with a "+"; and criteria that are in a negative or adverse relationship with another factor are identified with a "-". For example, operability is a criterion of usability and is shown to have a beneficial relationship with maintainability. The assertion is that the operability of usable software aids in software maintenance, even though it is not an essential characteristic of maintainability. The implication is that the effort to maintain software will be less if usability is also a specified quality. An example of a criterion with an adverse relationship is anomaly management. Anomaly management is a criterion of reliability and is shown to have a conflicting relationship with efficiency. The assertion is that the additional code required to perform anomaly management increases the runtime and requires additional memory, thus decreasing the potential efficiency. This implies that efficient use of resources will be more difficult to achieve if reliability is also a specified quality factor. Possible solutions to this conflict include:

- Budget and schedule to try to achieve high goals for both factors
- Lowering goals for one or the other factor, including decreasing emphasis on the specific criterion that conflicts and increasing emphasis on those that do not conflict
- Allocating more resources to the host system (e.g., more processing power or more memory) and possibly decreasing emphasis on high software quality goals.

Figure 6-1 shows how the attributes are related to other elements of the E&V Reference Manual and to elements in the E&V Guidebook. The various attributes that are useful in defining assessment objectives for APSEs or APSE components are described in this chapter.

E&V Reference Manual, Version 2.0

Table 6-4 Beneficial and Adverse Effects of Criteria on Software Quality Factors

G-16185
8-10-89

Acquisition Concern	Acquisition Concern		Performance 6.1					Design 6.2			Adaptation 6.3			
	Factor	Criterion/Selection	1. Efficiency	2. Integrity	3. Reliability	4. Survivability	5. Usability	1. Correctness	2. Maintainability	3. Verifiability	1. Expandability	2. Interoperability	3. Reusability	4. Transportability
Performance	Accuracy	6.4.1	-		X									
	Anomaly management (fault or error tolerance, robustness)	6.4.2	-		X	X	+							
	Autonomy	6.4.5				X								
	Capacity	6.4.6					X				+			
	Communication effectiveness	6.4.8	X						-	-				-
	Cost	6.4.11					X							
	Distributedness	6.4.12		-		X					+			
	Maturity	6.4.18			X		X							
	Operability (communicativeness)	6.4.20	-				X		+	+				
	Power	6.4.21					X							
	Processing (execution) effectiveness	6.4.22	X						-	-				-
	Reconfigurability	6.4.24				X			+		-		-	-
	Required configuration	6.4.26					X				+		+	+
	Storage effectiveness	6.4.31	X						-	-				-
	System accessibility	6.4.32	-	X										
	Training	6.4.36					X							
Design	Completeness	6.4.9						X	+				+	
	Consistency	6.4.10						X	X	+	+		+	
	Traceability	6.4.35						X	+	+	+		+	
	Visibility (test availability)	6.4.38							X	X				
Adaptation	Application independence	6.4.3											X	+
	Augmentability	6.4.4									X			
	Commonality (data and communication)	6.4.7	-									X		
	Document accessibility	6.4.13		-			+		+				X	
	Functional overlap	6.4.14										X		
	Functional scope	6.4.15											X	
	Generality	6.4.16	-	-	-	-					X	-	X	
	Rehostability (independence)	6.4.25	-								X	X	X	X
	Retargetability (independence)	6.4.27	-								X	X	X	X
	System clarity	6.4.33											X	
	System compatibility	6.4.34		-								X		
	Virtuality	6.4.37	-								X			
General	Granularity	6.4.17				X	X		X	X	X		X	
	Modularity	6.4.19	-			X			X	X	X	X	X	X
	Proprietary rights	6.4.23							X	X	X			
	Self-descriptiveness	6.4.28	-						X	X	X		X	X
	Simplicity	6.4.29	-		X				X	X	X		X	
	Software production vehicle(s)	6.4.30							X		X		X	X

X = Basic Relationship + = Positive Effect - = Negative Effect Blank = None or application dependent

G-16377
8-11-89

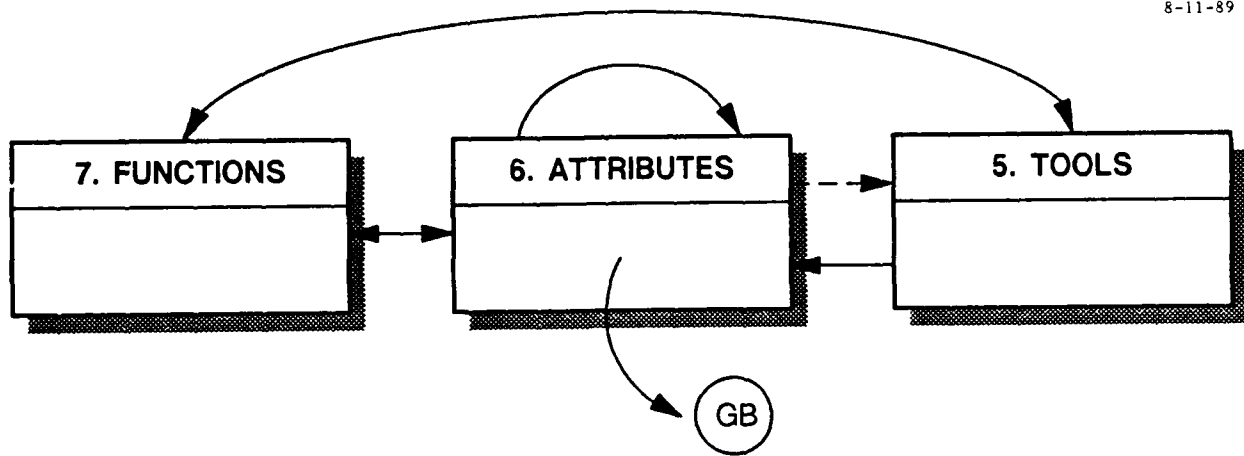


Figure 6-1 Attribute Relationships

6.1 PERFORMANCE

Description:

Performance quality factors deal both with the ability of the software to function and with error occurrences that affect software functioning. Low quality factors predict poor software performance. [@RADC 1985]

Cross References:

Software Quality Factors:

[Efficiency	6.1.1,
Integrity	6.1.2,
Reliability	6.1.3,
Survivability	6.1.4,
Usability	6.1.5]

6.1.1 Efficiency

Description:

The ratio of actual utilization of the system resources to optimum utilization. [@GIT 1987]

The extent to which a component fulfills its purpose using a minimum of computing resources. Of course, many of the ways of coding efficiently are not necessarily efficient in the sense of being cost-effective, since transportability, maintainability, etc., may be degraded as a result. [@DACS 1979]

Cross References:

Acquisition Concern:

[Performance

6.1]

Software-Oriented Criteria:

[Communication Effectiveness

6.4.8,

Processing Effectiveness

6.4.22,

Storage Effectiveness

6.4.31]

Complementary Software Quality Factors:

Cooperating Criteria:

Conflicting Criteria:

[Accuracy

6.4.1,

Anomaly Management

6.4.2,

Commonality

6.4.7,

Generality

6.4.16,

Modularity

6.4.19,

Operability

6.4.20,

Reconfigurability

6.4.24,

Rehostability

6.4.25,

Retargetability

6.4.27,

Self-Descriptiveness

6.4.28,

Simplicity

6.4.29,

System Accessibility

6.4.32,

Virtuality

6.4.37]

6.1.2 Integrity

Description:

The probability that the system will perform without failure and will protect the system and data from unauthorized access. [@GIT 1987] The extent to which unauthorized access to or modification of software or data can be controlled in a computer system. [@IEEE 1983]

Cross References:

Acquisition Concern: [Performance	6.1]
Software-Oriented Criteria: [System Accessibility	6.4.32]
Complementary Software Quality Factors: [Reliability	6.1.3,
Correctness	6.2.1,
Verifiability, Testability	6.2.3]
Cooperating Criteria:	
Conflicting Criteria: [Distributedness	6.4.12,
Document Accessibility	6.4.13,
Generality	6.4.16,
System Compatibility	6.4.34]

6.1.3 Reliability

Description:

The probability that the system will perform as intended under stated conditions for a specified period of time. [@GIT 1987]

Cross References:

Acquisition Concern: [Performance	6.1]
Software-Oriented Criteria: [Accuracy	6.4.1,
Anomaly Management	6.4.2,
Maturity	6.4.18,
Simplicity	6.4.29]
Complementary Software Quality Factors: [Correctness	6.2.1,
Verifiability, Testability	6.2.3]
Cooperating Criteria:	
Conflicting Criteria: [Generality	6.4.16]

6.1.4 Survivability

Description:

The extent to which the software will perform and support critical functions without failures within a specified time period when a portion of the system is inoperable. Survivability deals with the continued performance of software (e.g., in a degraded mode) even when a portion of the system has failed. [@RADC 1985]

Cross References:

Acquisition Concern: [Performance	6.1]
Software-Oriented Criteria: [Anomaly Management	6.4.2,
Autonomy	6.4.5,
Distributedness	6.4.12,
Granularity	6.4.17,
Modularity	6.4.19,
Reconfigurability	6.4.24]
Complementary Software Quality Factors: [Reliability	6.1.3,
Correctness	6.2.1,
Verifiability, Testability	6.2.3]
Cooperating Criteria:	
Conflicting Criteria: [Generality	6.4.16]

6.1.5 Usability

Description:

Extent to which resources required to acquire, install, learn, operate, prepare input for, and interpret output of a component are minimized.

Cross References:

Acquisition Concern:
[Performance

6.1]

Software-Oriented Criteria:

[Capacity

6.4.6,

Cost

6.4.11,

Granularity

6.4.17,

Maturity

6.4.18,

Operability

6.4.20,

Power

6.4.21,

Required Configuration

6.4.26,

Training

6.4.36]

Complementary Software Quality Factors:

[Reliability

6.1.3,

Correctness

6.2.1,

Maintainability

6.2.2,

Verifiability, Testability

6.2.3]

Cooperating Criteria:

[Anomaly Management

6.4.2,

Document Accessibility

6.4.13]

Conflicting Criteria:

6.2 DESIGN

Description:

Design quality factors deal mainly with software failure and correction. Low quality levels usually result in repeating a portion of the development process (e.g., redesign, recode, reverify); hence the term design. [RADC 1985]

Cross References:

Software Quality Factors:
[Correctness
Maintainability
Verifiability, Testability

6.2.1,
6.2.2,
6.2.3]

6.2.1 Correctness

Description:

The extent to which the system conforms to its specifications and standards. [GIT 1987]

(1) The extent to which software is free from design defects and from coding defects; that is fault free. (2) The extent to which software meets its specified requirements. [IEEE 1983]

Cross References:

Acquisition Concern:
[Design

6.2]

Software-Oriented Criteria:
[Completeness
Consistency
Traceability

6.4.9,
6.4.10,
6.4.35]

Complementary Software Quality Factors:

Cooperating Criteria:

Conflicting Criteria:

6.2.2 Maintainability

Description:

The ease of effort for locating and fixing software failures within a specified time period.
[@RADC 1985] The probability that the system can be restored to a specified condition within a specified amount of time. [@GIT 1987]

Cross References:

Acquisition Concern:	
[Design	6.2]
Software-Oriented Criteria:	
[Consistency	6.4.10,
Granularity	6.4.17,
Modularity	6.4.19,
Proprietary Rights	6.4.23,
Self-Descriptiveness	6.4.28,
Simplicity	6.4.29,
Software Production Vehicle(s)	6.4.30,
Visibility	6.4.38]
Complementary Software Quality Factors:	
[Reliability	6.1.3,
Correctness	6.2.1]
Cooperating Criteria:	
[Completeness	6.4.9,
Document Accessibility	6.4.13,
Operability	6.4.20,
Reconfigurability	6.4.24,
Traceability	6.4.35]
Conflicting Criteria:	
[Communication Effectiveness	6.4.8,
Processing Effectiveness	6.4.22,
Storage Effectiveness	6.4.31]

6.2.3 Verifiability, Testability

Description:

The extent to which the specified system operation and performance determine the conditions and criteria for tests. [GIT 1987] (1) The extent to which software facilitates both the establishment of test criteria and the evaluation of the software with respect to those criteria. (2) The extent to which the definition of requirements facilitates analysis of the requirements to establish test criteria. [IEEE 1983]

Cross References:

Acquisition Concern:	
[Design	6.2]
Software-Oriented Criteria:	
[Granularity	6.4.17,
Modularity	6.4.19,
Self-Descriptiveness	6.4.28,
Simplicity	6.4.29,
Visibility	6.4.38]
Complementary Software Quality Factors:	
[Reliability	6.1.3,
Correctness	6.2.1,
Maintainability	6.2.2]
Cooperating Criteria:	
[Consistency	6.4.10,
Operability	6.4.20,
Traceability	6.4.35]
Conflicting Criteria:	
[Communication Effectiveness	6.4.8,
Processing Effectiveness	6.4.22,
Storage Effectiveness	6.4.31]

6.3 ADAPTATION

Description:

Adaptation quality factors deal mainly with using software beyond its original requirements, such as extending or expanding capabilities and adapting for use in another application or environment. Low quality levels predict relatively high costs for new software use. [@RADC 1985]

Cross References:

Software Quality Factors:	
[Expandability, Flexibility	6.3.1,
Interoperability	6.3.2,
Reusability	6.3.3,
Transportability	6.3.4]

6.3.1 Expandability, Flexibility

Description:

The extent to which the system capability or performance can be increased by enhancing current functions or adding new functions (expandability). The extent to which system purpose, functions, or data can be changed to satisfy other specified requirements (flexibility). [@GIT 1987] The extent to which a component allows new capabilities to be added and existing capabilities to be easily tailored to user needs. [@DACS 1979]

Cross References:

Acquisition Concern:	
[Adaptation	6.3]
Software-Oriented Criteria:	
[Augmentability	6.4.4,
Generality	6.4.16,
Granularity	6.4.17,
Modularity	6.4.19,
Proprietary Rights	6.4.23,
Retargetability	6.4.27,
Self-Descriptiveness	6.4.28,
Simplicity	6.4.29,
Software Production Vehicle(s)	6.4.30,
Virtuality	6.4.37]
Complementary Software Quality Factors:	
[Reliability	6.1.3,
Correctness	6.2.1,
Maintainability	6.2.2,
Verifiability, Testability	6.2.3]
Cooperating Criteria:	
[Capacity	6.4.6,
Consistency	6.4.10,
Distributedness	6.4.12,
Required Configuration	6.4.26,
Traceability	6.4.35]
Conflicting Criteria:	
[Reconfigurability	6.4.24]

6.3.2 Interoperability

Description:

The probability that two or more systems can exchange information under stated conditions and use the information that has been exchanged. [@GIT 1987] The ability of APSEs to exchange database objects and their relationships in forms usable by components and user programs without conversion. Interoperability is measured by the degree to which this exchange can be accomplished without conversion. [@E&V Requirements]

Cross References:

Acquisition Concern:
[Adaptation

6.3]

Software-Oriented Criteria:

[Commonality
Functional Overlap
Modularity
Rehostability
Retargetability
System Compatibility

6.4.7,
6.4.14,
6.4.19,
6.4.25,
6.4.27,
6.4.34]

Complementary Software Quality Factors:

[Reliability
Correctness
Maintainability
Verifiability, Testability

6.1.3,
6.2.1,
6.2.2,
6.2.3]

Cooperating Criteria:

[Generality

6.4.16]

Conflicting Criteria:

6.3.3 Reusability

Description:

The extent to which a component can be adapted for use in another application. [@RADC 1985]

Cross References:

Acquisition Concern:
[Adaptation

6.3]

Software-Oriented Criteria:

[Application Independence

6.4.3,

Document Accessibility

6.4.13,

Functional Scope

6.4.15,

Generality

6.4.16,

Granularity

6.4.17,

Modularity

6.4.19,

Rehostability

6.4.25,

Retargetability

6.4.27,

Self-Descriptiveness

6.4.28,

Simplicity

6.4.29,

Software Production Vehicle(s)

6.4.30,

System Clarity

6.4.33]

Complementary Software Quality Factors:

[Reliability

6.1.3,

Correctness

6.2.1,

Maintainability

6.2.2,

Verifiability, Testability

6.2.3]

Cooperating Criteria:

[Completeness

6.4.9,

Consistency

6.4.10,

Required Configuration

6.4.26,

Traceability

6.4.35]

Conflicting Criteria:

[Reconfigurability

6.4.24]

6.3.4 Transportability

Description:

The extent to which a component can be adapted for use in another environment (e.g., different host or target hardware, operating system, APSE). [@RADC 1985] The ability of a component to be installed on a different APSE without change in functionality. Transportability is measured in the degree to which this installation can be accomplished without reprogramming. [@E&V Requirements]

Cross References:

Acquisition Concern:	
[Adaptation	6.3]
Software-Oriented Criteria:	
[Modularity	6.4.19,
Rehostability	6.4.25,
Retargetability	6.4.27,
Self-Descriptiveness	6.4.28,
Software Production Vehicle(s)	6.4.30]
Complementary Software Quality Factors:	
[Reliability	6.1.3,
Correctness	6.2.1,
Maintainability	6.2.2,
Verifiability, Testability	6.2.3]
Cooperating Criteria:	
[Application Independence	6.4.3,
Required Configuration	6.4.26]
Conflicting Criteria:	
[Communication Effectiveness	6.4.8,
Processing Effectiveness	6.4.22,
Reconfigurability	6.4.24,
Storage Effectiveness	6.4.31]

6.4 SOFTWARE-ORIENTED CRITERIA

6.4.1 Accuracy

Description:

A quantitative measure of the magnitude of error, preferably expressed as a function of the relative error, a high value of this measure corresponding to a small error. Error is defined as a discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition. [@IEEE 1983]

Cross References:

Software Quality Factors:
[Reliability 6.1.3]

Beneficial Quality Factors:

Adverse Quality Factors:
[Efficiency 6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5]

6.4.2 Anomaly Management, Fault or Error Tolerance, Robustness

Description:

Those characteristics of software which provide for continuity of operations under and recovery from non-nominal conditions. Such conditions include: improper input data, computational failures, hardware faults, device errors, communication errors, and node or communication failures. [RADC 1985] The protection of a component from itself, user errors, and system errors. The ability to recover and provide meaningful diagnostics in the event of unforeseen situations. A robust routine will avoid failing for input values where the desired output is well-defined, but the intermediate results of a straightforward implementation may cause the routine to fail. [E&V Requirements]

Cross References:

Software Quality Factors:	
[Reliability	6.1.3,
Survivability	6.1.4]
Beneficial Quality Factors:	
[Usability	6.1.5]
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
*Compilation	7.1.6.7,
@GB: ARTEWG Catalogue of Ada Runtime	
Implementation Dependencies	5.10;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4;
Runtime Environment	7.2.3.5,
@GB: Runtime Support System Questionnaire	5.14]
*NOTE: Normally, the concern of evaluation is the quality of a tool. However, this evaluation technique focuses on the <u>products</u> of the tool rather than the tool itself.	

6.4.3 Application Independence

Description:

Those characteristics of software which determine its non-dependency on database system, microcode, computer architecture, data structures, and algorithms. [@RADC 1985]

Cross References:

Software Quality Factors:
[Reusability 6.3.3]

Beneficial Quality Factors:
[Transportability 6.3.4]

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5;
Whole APSE Assessment Issues 3.,
@GB: APSE Customization Questionnaire 13.4]

6.4.4 Augmentability

Description:

Those characteristics of software which provide for expansion of capability for functions and data. Metrics include: data storage expansion, computational extensibility, channel extensibility, and design extensibility. [@RADC 1985]

Cross References:

Software Quality Factors:
[Expandability, Flexibility 6.3.1]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues 3.,	
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Systems Requirements 7.1.6.1,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements 7.1.6.2,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design 7.1.6.4,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Compilation 7.1.6.7,	
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management 7.2.2.7	
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing 7.2.3.1,	
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.5 Autonomy

Description:

Those characteristics of software which determine its non-dependency on interfaces and functions. Metrics include: interface complexity and self-sufficiency. [@RADC 1985]

Cross References:

Software Quality Factors:
[Survivability 6.1.4]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5]

6.4.6 Capacity

Description:

The upper and lower limits of the functions implemented by a tool. For example, the maximum and minimum quantities of input data for a tool to operate. [@E&V Requirements]

Cross References:

Software Quality Factors:
 [Usability 6.1.4]
 Beneficial Quality Factors:
 [Expandability, Flexibility 6.3.1]
 Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues 3.,	
@GB: APSE Characterization	13.1;
Whole APSE Assessment Issues 3.,	
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Systems Requirements 7.1.6.1,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements 7.1.6.2,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design 7.1.6.4,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Compilation 7.1.6.7,	
@GB: IDA Benchmarks	5.2;
Compilation 7.1.6.7,	
@GB: MITRE Benchmark Generator Tool (BGT)	5.6;
Compilation 7.1.6.7,	
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management 7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing 7.2.3.1,	
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.7 Commonality (Data and Communication)

Description:

Those characteristics of software which provide for the use of interface standards for protocols, routines, and data representations. [RADC 1985] The set of assumptions made by the component and made about the component by the remaining components and the system in which it appears. Software components have control, data, and service interfaces. [DACS 1979]

Cross References:

Software Quality Factors:	
[Interoperability	6.3.2]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues	3.,
@GB: Distributed APSE Questionnaire	12.1;
Whole APSE Assessment Issues	3.,
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Whole APSE Assessment Issues	3.,
@GB: APSE Customization Questionnaire	13.4;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.8 Communication Effectiveness

Description:

Those characteristics of the software which provide for minimum utilization of communications resources in performing functions. [@RADC 1985]

Cross References:

Software Quality Factors:
[Efficiency 6.1.1]

Beneficial Quality Factors:

Adverse Quality Factors:
[Maintainability 6.2.2,
Verifiability, Testability 6.2.3,
Transportability 6.3.4]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5;
Runtime Environment 7.2.3.5,
@GB: Runtime Support System Questionnaire 5.14]

6.4.9 Completeness

Description:

The extent to which a component provides the complete set of operations necessary to perform a function. For example: are all the inputs and outputs clearly defined for each operation; does each data element have a clearly defined meaning, format, and source; and are all operations available and working as expected? [E&V Requirements]

Cross References:

Software Quality Factors:	
[Correctness	6.2.1]
Beneficial Quality Factors:	
[Maintainability	6.2.2,
Reusability	6.3.3]
Adverse Quality Factors:	

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues	3.,
@GB: APSE Characterization	13.1;
Whole APSE Assessment Issues	3.,
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Text Editing	7.1.1.1,
@GB: Text Editing Capabilities Checklist	99.1;
Text Editing	7.1.1.1,
@GB: Language-Sensitive Editing Capabilities Checklist	99.2;
Systems Requirements	7.1.6.1,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Systems Requirements	7.1.6.1,
@GB: Time-Critical Applications Support Checklist	9.5;
Software Requirements	7.1.6.2,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements	7.1.6.2,
@GB: Time-Critical Applications Support Checklist	9.5;

E&V Reference Manual, Version 2.0

Preliminary Design	7.1.6.4,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Preliminary Design	7.1.6.4,	
@GB: Time-Critical Applications Support Checklist		9.5;
Detailed Design	7.1.6.5,	
@GB: Time-Critical Applications Support Checklist		9.5;
Assembling	7.1.6.6,	
@GB: Cross Development System Support Questionnaire		13.3;
Compilation	7.1.6.7,	
@GB: ACVC		5.1;
Compilation	7.1.6.7,	
@GB: Cross Development System Support Questionnaire		13.3;
Linking/Loading	7.1.6.13,	
@GB: Cross Development System Support Questionnaire		13.3;
Database Management	7.2.1.1,	
@GB: Database Management Checklist		15.2;
File Management	7.2.1.3,	
@GB: File Management Checklist		15.1;
Electronic Mail	7.2.1.4,	
@GB: Electronic Mail Checklist		15.3;
Program Library Management	7.2.1.7,	
@GB: Program Library Management Checklist		5.9;
Performance Monitoring	7.2.1.10,	
@GB: Performance Monitoring Checklist		99.3;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Capabilities Checklist		10.1;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Kernel	7.2.3.3,	
@GB: CIVC		8.1;
Runtime Environment	7.2.3.5,	
@GB: ARTEWG Runtime Environment Taxonomy		5.11;
Runtime Environment	7.2.3.5,	
@GB: Runtime Support System Questionnaire		5.14;
Import/Export	7.2.3.6,	
@GB: Import/Export Capabilities Checklist		6.3;
Analysis	7.3,	
@GB: Testing Capabilities Checklist		7.1;
Syntax and Semantics Checking	7.3.1.15,	
@GB: Language-Sensitive Editing Capabilities Checklist		99.2;
Requirements Prototyping	7.3.2.2,	
@GB: Requirements Prototyping Capabilities Checklist		9.2;
Simulation and Modeling	7.3.2.3,	
@GB: Instruction-Level Simulation Checklist		6.8;

E&V Reference Manual, Version 2.0

Simulation and Modeling	7.3.2.3,	
@GB: Simulation and Modeling Capabilities Checklist		9.3;
Simulation and Modeling	7.3.2.3,	
@GB: Cross Development System Support Questionnaire		13.3;
Debugging	7.3.2.5,	
@GB: Debugging Capabilities Checklist		6.5;
Debugging	7.3.2.5,	
@GB: Cross Development System Support Questionnaire		13.3;
Emulation	7.3.2.13,	
@GB: Cross Development System Support Questionnaire		13.3;
Timing Analysis	7.3.2.14,	
@GB: Timing Analysis Capabilities Checklist		6.6;
Timing Analysis	7.3.2.14,	
@GB: Cross Development System Support Questionnaire		13.3;
Real-Time Analysis	7.3.2.17,	
@GB: Real-Time Analysis Capabilities Checklist		6.7]

6.4.10 Consistency

Description:

Those characteristics of software which provide for uniform design and implementation techniques and notation. Metrics include the use of conventions for: design representations, calling sequences, communication protocols, error handling, object references, coding style, data representations, naming conventions, global data, and multiple object copies. [@RADC 1985]

Cross References:

Software Quality Factors:	
[Correctness	6.2.1,
Maintainability	6.2.2]
Beneficial Quality Factors:	
[Verifiability, Testability	6.2.3,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3]
Adverse Quality Factors:	

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues	3.,
@GB: Distributed APSE Questionnaire	12.1;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.11 Cost

Description:

The cost of a complete component or the costs of features of a component. The cost of a component may vary depending on delivery with source code or object code only (for example). Other cost considerations are installation, user assistance, and maintenance support. [@E&V Requirements]

Cross References:

Software Quality Factors:
 [Usability 6.1.5]
 Beneficial Quality Factors:
 Adverse Quality Factors:

Guidance References:

[@GB: Vendor Evaluation Questionnaire	99.7;
@GB: Cost Questionnaire	99.9;
Whole APSE Assessment Issues	3.,
@GB: APSE Characterization	13.1;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.12 Distributedness

Description:

Those characteristics of software which determine the degree to which software functions are geographically or logically separated within the system. Metrics include: data at different nodes or on different storage devices, data accessibility at different nodes, alternative data sources, functional redundancy, distributed control, and minimum number of nodes.
[@RADC 1985]

Cross References:

Software Quality Factors: [Survivability	6.1.4]
Beneficial Quality Factors: [Expandability, Flexibility	6.3.1]
Adverse Quality Factors: [Integrity	6.1.2]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
Whole APSE Assessment Issues	3.,
@GB: APSE Customization Questionnaire	13.4]

6.4.13 Document Accessibility

Description:

Those characteristics of software which provide for easy access to software and selective use of its components. Metrics include: ready access to documents, clarity and simplicity of documents, depiction of control and data flow, finding aids, separation by function, complete description of requirements and limitations, and access to the source code or runtime version. [@RADC 1985]

Cross References:

Software Quality Factors:
[Reusability 6.3.3]

Beneficial Quality Factors:
[Usability 6.1.5,
Maintainability 6.2.2]

Adverse Quality Factors:
[Integrity 6.1.2]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: Vendor Evaluation Questionnaire	99.7;
Compilation 7.1.6.7,	
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management 7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing 7.2.3.1,	
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.14 Functional Overlap

Description:

Those characteristics of software which provide common functions to both systems.
[@RADC 1985]

Cross References:

Software Quality Factors:
[Interoperability 6.3.2]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues 3.,	
@GB: Distributed APSE Questionnaire	12.1]

6.4.15 Functional Scope

Description:

Those characteristics of software which provide commonality of functions among applications. Metrics include: function specificity, function commonality, and function selective usability. [@RADC 1985]

Cross References:

Software Quality Factors:
[Reusability 6.3.3]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5;
Runtime Environment 7.2.3.5,
@GB: Runtime Support System Questionnaire 5.14]

6.4.16 Generality

Description:

Those characteristics of software which provide breadth to the functions performed with respect to the application. Metrics include: the number of calling modules, modules that perform only a single process (external input, external output, algorithm), modules free of machine-dependent operations, limitations on data volume, and limitations on data values. [RADC 1985]

Cross References:

Software Quality Factors:	
[Expandability, Flexibility	6.3.1,
Reusability	6.3.3]
Beneficial Quality Factors:	
[Interoperability	6.3.2]
Adverse Quality Factors:	
[Efficiency	6.1.1,
Integrity	6.1.2,
Reliability	6.1.3,
Survivability	6.1.4]

Guidebook References:

[RADC: Software Quality Metric Worksheets	99.5;
Whole APSE Assessment Issues	3.,
RADC: APSE Customization Questionnaire	13.4;
Compilation	7.1.6.7,
RADC: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
RADC: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
RADC: Command Language Interpreter	
Assessment Questionnaire	99.4;
Runtime Environment	7.2.3.5,
RADC: Runtime Support System Questionnaire	5.14]

6.4.17 Granularity

Description:

The degree to which a component has separate limited functions that are composable, user selectable, and communicate through a common database. [@E&V Requirements]

Cross References:

Software Quality Factors:

[Survivability	6.1.4,
Usability	6.1.5,
Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools	99.6;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4;
Runtime Environment	7.2.3.5,
@GB: Runtime Support System Questionnaire	5.14]

6.4.18 Maturity

Description:

The extent to which a component has been used in the development of deliverable software by typical users and to which the feedback from that use has been reflected in modifications to the component. [@E&V Requirements]

Cross References:

Software Quality Factors:

[Reliability

6.1.3,

Usability

6.1.5]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools	99.6;
@GB: Vendor Evaluation Questionnaire	99.7;
@GB: Maturity Questionnaire	99.10;
Whole APSE Assessment Issues	3.,
@GB: APSE Characterization	13.1;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.19 Modularity

Description:

The extent to which software is composed of discrete components such that a change to one component has minimal impact on other components. [IEEE 1983] The extent to which a component is implemented in a hierarchical structure in which identifiable functions are isolated in separate compilation units. Modular software is characterized by low coupling between units (restrictions on the number and types of parameters) and high cohesion within units (singularity of purpose).

Cross References:

Software Quality Factors:	
[Survivability	6.1.4,
Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Expandability, Flexibility	6.3.1,
Interoperability	6.3.2,
Reusability	6.3.3,
Transportability	6.3.4]

Beneficial Quality Factors:

Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
Whole APSE Assessment Issues	3.,
@GB: APSE Customization Questionnaire	13.4;
Runtime Environment	7.2.3.5,
@GB: Runtime Support System Questionnaire	5.14]

6.4.20 Operability, Communicativeness

Description:

Those characteristics of software which determine operations and procedures concerned with operation of software and which provide useful inputs and outputs which can be assimilated. Metrics include: the user/component dialog, error reporting and response procedures, debugging facilities, ergonomic considerations, event logging or monitoring facilities, transparency of implementation details, and input and output options. [RADC 1985] The user/component dialog established to control the execution of the component. This is driven by the set of assumptions made by the component and made about the component by the persons who use it.

Cross References:

Software Quality Factors:	
[Usability	6.1.5]
Beneficial Quality Factors:	
[Maintainability	6.2.2,
Verifiability, Testability	6.2.3]
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets		99.5;
@GB: SEI Assessment of Software Engineering Tools		99.6;
Whole APSE Assessment Issues	3.,	
@GB: Distributed APSE Questionnaire		12.1;
Whole APSE Assessment Issues	3.,	
@GB: APSE Characterization		13.1;
Whole APSE Assessment Issues	3.,	
@GB: Ada-Europe Ada Environment Questionnaires		13.2;
Whole APSE Assessment Issues	3.,	
@GB: APSE Customization Questionnaire		13.4;
Compilation	7.1.6.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Command Language Processing	7.2.3.1,	
@GB: Command Language Interpreter		
Assessment Questionnaire		99.4]

6.4.21 Power

Description:

The extent to which a component has capabilities, such as default options and wild card operations, that contribute to the effectiveness of the user. [@E&V Requirements]

Cross References:

Software Quality Factors:
[Usability 6.1.5]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues 3.,	
@GB: APSE Characterization	13.1;
Whole APSE Assessment Issues 3.,	
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Text Editing 7.1.1.1,	
@GB: Text Editing Capabilities Checklist	99.1;
Text Editing 7.1.1.1,	
@GB: Language-Sensitive Editing Capabilities Checklist	99.2;
Preliminary Design 7.1.6.4,	
@GB: SEI Design Support Equipment	9.1;
Detailed Design 7.1.6.5,	
@GB: SEI Design Support Equipment	9.1;
Assembling 7.1.6.6,	
@GB: Assembling Checklist	6.1;
Compilation 7.1.6.7,	
@GB: Compilation Checklist	5.8;

E&V Reference Manual, Version 2.0

Compilation	7.1.6.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Linking/Loading	7.1.6.13,	
@GB: Linking/Loading Checklist		6.2;
Database Management	7.2.1.1,	
@GB: Database Management Checklist		15.2;
File Management	7.2.1.3,	
@GB: File Management Checklist		15.1;
Electronic Mail	7.2.1.4,	
@GB: Electronic Mail Checklist		15.3;
Program Library Management	7.2.1.7,	
@GB: Program Library Management Checklist		5.9;
Configuration Management	7.2.2.7,	
@GB: SEI Configuration Management Experiment		10.2;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Command Language Processing	7.2.3.1,	
@GB: Command Language Interpreter Assessment Questionnaire		99.4;
Analysis	7.3,	
@GB: Testing Capabilities Checklist		7.1;
Syntax and Semantics Checking	7.3.1.15,	
@GB: Language-Sensitive Editing Capabilities Checklist		99.2;
Dynamic Analysis	7.3.2,	
@GB: SEI Unit Testing and Debugging Experiment		7.2;
Debugging	7.3.2.5,	
@GB: Debugging Capabilities Checklist		6.5;
Debugging	7.3.2.5,	
@GB: SEI Unit Testing and Debugging Experiment		7.2;
Emulation	7.3.2.13,	
@GB: Emulation Capabilities Checklist		6.4]

6.4.22 Processing (Execution) Effectiveness

Description:

Those characteristics of the software which provide for minimum utilization of processing resources in performing functions. Metrics include algorithmic processing effectiveness and data usage effectiveness. [@RADC 1985] The choice between alternative algorithms based on those taking the least amount of time. [@DACS 1979]

Cross References:

Software Quality Factors:	
[Efficiency	6.1.1]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Transportability	6.3.4]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues	3.,
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Systems Requirements	7.1.6.1,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements	7.1.6.2,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design	7.1.6.4,
@GB: SEI Design Support Experiment	9.1;
Preliminary Design	7.1.6.4,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Detailed Design	7.1.6.5,
@GB: SEI Design Support Experiment	9.1;

E&V Reference Manual, Version 2.0

*Compilation	7.1.6.7,	
@GB: IDA Benchmarks		5.2;
*Compilation	7.1.6.7,	
@GB: ACEC		5.3;
*Compilation	7.1.6.7,	
@GB: PIWG Benchmark Tests		5.4;
*Compilation	7.1.6.7,	
@GB: University of Michigan Benchmark Tests		5.5;
*Compilation	7.1.6.7,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Compilation	7.1.6.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Configuration Management	7.2.2.7,	
@GB: SEI Configuration Management Experiment		10.2;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Command Language Processing	7.2.3.1,	
@GB: Command Language Interpreter Assessment Questionnaire		99.4;
Dynamic Analysis	7.3.2,	
@GB: SEI Unit Testing and Debugging Experiment		7.2;
Debugging	7.3.2.5,	
@GB: SEI Unit Testing and Debugging Experiment		7.2]

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

6.4.23 Proprietary Rights

Description:

Restrictions on the release, distribution, or use of a component. This includes so called "data rights" restrictions. [@E&V Requirements]

Cross References:

Software Quality Factors:
[Maintainability 6.2.2,
Expandability, Flexibility 6.3.1]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools 99.6;
@GB: Vendor Evaluation Questionnaire 99.7;
@GB: Licensing Issues Questionnaire 99.11;
Whole APSE Assessment Issues 3.,
@GB: Ada-Europe Ada Environment Questionnaires 13.2;
Compilation 7.1.6.7,
@GB: Compiler Assessment Questionnaire 5.12;
Configuration Management 7.2.2.7,
@GB: Configuration Management Assessment Questionnaire 10.3;
Command Language Processing 7.2.3.1,
@GB: Command Language Interpreter
Assessment Questionnaire 99.4]

6.4.24 Reconfigurability

Description:

Those characteristics of software which provide for continuity of system operation when one or more processors, storage units, or communication links fail. Metrics include ensuring communication continuity, maintaining the integrity of data values, restoring failed communication links, and replicating critical data at multiple nodes. [@RADC 1985]

Cross References:

Software Quality Factors: [Survivability	6.1.4]
Beneficial Quality Factors: [Maintainability	6.2.2]
Adverse Quality Factors: [Efficiency	6.1.1,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3,
Transportability	6.3.4]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5]

6.4.25 Rehostability

Description:

Those characteristics of software which determine its non-dependency on software environment (computing system, operating system, utilities, input, output routines, libraries). [@RADC 1985: Independence] The ability of an APSE component to be installed on a different host or different operating system with needed reprogramming localized to the KAPSE or machine dependencies. Included in this attribute is conformance to any existing pertinent interface standards such as the CAIS. [@E&V Requirements]

Cross References:

Software Quality Factors:	
[Interoperability	6.3.2,
Reusability	6.3.3,
Transportability	6.3.4]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: Host and Target Questionnaire	14.1;
@GB: Machine-Specific Characteristics Questionnaire	14.2;
@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management	7.2.2.7,
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing	7.2.3.1,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.26 Required Configuration

Description:

The amount and types of hardware or software facilities needed for the operation of a component. This includes primary and secondary storage and any other required resources. [@E&V Requirements]

Cross References:

Software Quality Factors:
[Usability 6.1.5]

Beneficial Quality Factors:
[Expandability, Flexibility 6.3.1,
Reusability 6.3.3,
Transportability 6.3.4]

Adverse Quality Factors:

Guidebook References:

[@GB: Required Configuration Questionnaire		99.8;
Whole APSE Assessment Tools	3.,	
@GB: Distributed APSE Questionnaire		12.1;
Whole APSE Assessment Issues	3.,	
@GB: APSE Characterization		13.1;
Whole APSE Assessment Issues	3.,	
@GB: Ada-Europe Ada Environment Questionnaires		13.2;
Whole APSE Assessment Issues	3.,	
@GB: APSE Customization Questionnaire		13.4;
Compilation	7.1.6.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Command Language Processing	7.2.3.1,	
@GB: Command Language Interpreter		
Assessment Questionnaire		99.4]

6.4.27 Retargetability

Description:

The ability of an APSE component to accomplish its function with respect to another target. The component may require modification. [@E&V Requirements] The retargetability of a component is usually determined by the software and machine independence of the outputs generated by the component.

Cross References:

Software Quality Factors:	
[Expandability, Flexibility	6.3.1,
Interoperability	6.3.2,
Reusability	6.3.3,
Transportability	6.3.4]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: Host and Target Questionnaire	14.1;
@GB: Machine-Specific Characteristics Questionnaire	14.2;
@GB: RADC Software Quality Metric Worksheets	99.5;
Whole APSE Assessment Issues	3.,
@GB: APSE Customization Questionnaire	13.4;
*Compilation	7.1.6.7,
@GB: ARTEWG Catalogue of Ada	
Runtime Implementation Dependencies	5.10;
Compilation	7.1.6.7,
@GB: Compiler Assessment Questionnaire	5.12;
Runtime Environment	7.2.3.5,
@GB: Runtime Support System Questionnaire	5.14]

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

6.4.28 Self-Descriptiveness

Description:

Those characteristics of software which provide explanation of the implementation of functions. Metrics include: quantity and effectiveness of comments and the descriptiveness of the implementation language (such as meaningful variable names and formatting that reflects the structure of code). [@RADC 1985] The technical data, including on-line, documentation, listings, and printouts, which serve the purpose of elaborating the design or details of a component. [@DACS 1979]

Cross References:

Software Quality Factors:	
[Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3,
Transportability	6.3.4]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Efficiency	6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
Systems Requirements	7.1.6.1,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements	7.1.6.2,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design	7.1.6.4,
@GB: NADC/SFS CASE Tools Evaluation	9.4]

6.4.29 Simplicity

Description:

Those characteristics of software which provide for definition and implementation of functions in the most non-complex and understandable manner. Metrics include: design structure, use of a structured language, data and control flow complexity, coding simplicity, specificity, and Halstead's level of difficulty measure. [@RADC 1985] The lack of complication of a system or system component, determined by such factors as the number and intricacy of interfaces, the number and intricacy of conditional branches, the degree of nesting, the types of data structures, and other system characteristics. [@IEEE 1983]

Cross References:

Software Quality Factors:

[Reliability	6.1.3,
Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3]

Beneficial Quality Factors:

Adverse Quality Factors:

[Efficiency	6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6]

6.4.30 Software Production Vehicle(s)

Description:

The methodology(ies), language(s), technique(s), and software tool(s) used to produce the software related to a component. [@E&V Requirements]

Cross References:

Software Quality Factors:

[Maintainability	6.2.2,
Expandability, Flexibility	6.3.1,
Reusability	6.3.3,
Transportability	6.3.4]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: SEI Assessment of Software Engineering Tools	99.6;
@GB: Vendor Evaluation Questionnaire	99.7;
@GB: Software Production Vehicle(s) Questionnaire	99.12]

6.4.31 Storage Effectiveness

Description:

Those characteristics of the software which provide for minimum utilization of storage resources in performing functions. Metrics include: the use of virtual storage, dynamic reallocation of memory, and the avoidance of redundant storage of data. [RADC 1985] The choice between alternative source code constructions based on those taking the minimum number of words of object code or in which the information-packing ... is high. [DACS 1979]

Cross References:

Software Quality Factors:	
[Efficiency	6.1.1]
Beneficial Quality Factors:	
Adverse Quality Factors:	
[Maintainability	6.2.2,
Verifiability, Testability	6.2.3,
Transportability	6.3.4]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Preliminary Design	7.1.6.4,
@GB: SEI Design Support Experiment	9.1;
Detailed Design	7.1.6.5,
@GB: SEI Design Support Experiment	9.1;
*Compilation	7.1.6.7,
@GB: IDA Benchmarks	5.2;
*Compilation	7.1.6.7,
@GB: ACEC	5.3;
*Compilation	7.1.6.7,
@GB: PIWG Benchmark Tests	5.4;
*Compilation	7.1.6.7,
@GB: ARTEWG Catalogue of Ada	
Runtime Implementation Dependencies	5.10;

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

E&V Reference Manual, Version 2.0

Compilation	7.1.6.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Configuration Management	7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Command Language Processing	7.2.3.1,	
@GB: Command Language Interpreter		
Assessment Questionnaire		99.4]

6.4.32 System Accessibility

Description:

Those characteristics of software which provide for control and audit of access to the software and data. [@RADC 1985] Hardware or software features, operating procedures, or management procedures designed to permit authorized access and prevent unauthorized access to a computer system. [@IEEE 1983]

Cross References:

Software Quality Factors:
[Integrity 6.1.2]

Beneficial Quality Factors:

Adverse Quality Factors:
[Efficiency 6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Issues 3.,	
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Runtime Environment 7.2.3.5,	
@GB: Runtime Support System Questionnaire	5.14]

6.4.33 System Clarity

Description:

Those characteristics of software which provide for clear description of program structure in a non-complex and understandable manner. Metrics include: interface complexity, program flow complexity, application functional complexity, communication complexity, and structure clarity. [@RADC 1985]

Cross References:

Software Quality Factors:
[Reusability 6.3.3]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
Systems Requirements	7.1.6.1,
@GE: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements	7.1.6.2,
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design	7.1.6.4,
@GB: NADC/SPS CASE Tools Evaluation	9.4]

6.4.34 System Compatibility

Description:

Those characteristics of software which provide the hardware, software, and communication compatibility of two systems. Metrics include: communication compatibility, data compatibility, hardware compatibility, software compatibility, and access to documentation for the other systems. [@RADC 1985]

Cross References:

Software Quality Factors:
[Interoperability 6.3.2]

Beneficial Quality Factors:

Adverse Quality Factors:
[Integrity 6.1.2]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
Whole APSE Assessment Tools 3.,	
@GB: Distributed APSE Questionnaire	12.1;
Whole APSE Assessment Issues 3.,	
@GB: APSE Customization Questionnaire	13.4;
Compilation 7.1.6.7,	
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management 7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing 7.2.3.1,	
@GB: Command Language Interpreter Assessment Questionnaire	99.4;
Runtime Environment 7.2.3.5,	
@GB: Runtime Support System Questionnaire	5.14]

6.4.35 Traceability

Description:

Those characteristics of software which provide a thread of origin from the implementation to the requirements with respect to the specified development envelope and operational environment. [@RADC 1985]

Cross References:

Software Quality Factors:
[Correctness 6.2.1]

Beneficial Quality Factors:
[Maintainability 6.2.2,
Verifiability, Testability 6.2.3,
Expandability, Flexibility 6.3.1,
Reusability 6.3.3]

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5;
@GB: SEI Assessment of Software Engineering Tools 99.6]

6.4.36 Training

Description:

Those characteristics of software which provide transition from current operation and provide initial familiarization. [RADC 1985] The extent to which training and other user help is available from the vendor of a component or from the component itself, including on-line, documentation, listings, and printouts, which serve the purpose of providing operating instructions for using the component to obtain desired results. [DACS 1979]

Cross References:

Software Quality Factors:
 [Usability 6.1.5]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[RADC Software Quality Metric Worksheets	99.5;
@GB: SEI Assessment of Software Engineering Tools	99.6;
@GB: Vendor Evaluation Questionnaire	99.7;
Whole APSE Assessment Issues 3.,	
@GB: Ada-Europe Ada Environment Questionnaires	13.2;
Systems Requirements 7.1.6.1,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Software Requirements 7.1.6.2,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Preliminary Design 7.1.6.4,	
@GB: NADC/SPS CASE Tools Evaluation	9.4;
Compilation 7.1.6.7,	
@GB: Compiler Assessment Questionnaire	5.12;
Configuration Management 7.2.2.7,	
@GB: Configuration Management Assessment Questionnaire	10.3;
Command Language Processing 7.2.3.1,	
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

6.4.37 Virtuality

Description:

Those characteristics of software which present a system that does not require user knowledge of the physical, logical, or topological characteristics. [@RADC 1985]

Cross References:

Software Quality Factors:
[Expandability, Flexibility 6.3.1]

Beneficial Quality Factors:

Adverse Quality Factors:
[Efficiency 6.1.1]

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5]

6.4.38 Visibility, Test Availability

Description:

Those characteristics of software which provide status monitoring of the development and operation. [@RADC 1985] The availability of tests that verify the correctness or effectiveness of a component function or feature. These tests may also verify proper response for an incorrect input or technique. Metrics include: unit testing, integration testing, and CSCI testing. [@E&V Requirements]

Cross References:

Software Quality Factors:
[Maintainability 6.2.2,
Verifiability, Testability 6.2.3]

Beneficial Quality Factors:

Adverse Quality Factors:

Guidebook References:

[@GB: RADC Software Quality Metric Worksheets 99.5;
@GB: SEI Assessment of Software Engineering Tools 99.6;
Compilation 7.1.6.7,
@GB: Compiler Assessment Questionnaire 5.12;
Configuration Management 7.2.2.7,
@GB: Configuration Management Assessment Questionnaire 10.3;
Command Language Processing 7.2.3.1,
@GB: Command Language Interpreter
Assessment Questionnaire 99.4]

FUNCTIONS

This chapter provides a functional taxonomy used for characterizing the functional capabilities of a tool, toolset, or APSE. The top two levels of the taxonomy are as follows:

- Transformation
 - Editing
 - Formatting
 - On-Line Assistance Processing
 - Sort/Merge
 - Graphics Generation
 - Translation
 - Synthesis
- Management
 - Information Management
 - Project Management
 - Computer System Management
- Analysis
 - Static Analysis
 - Dynamic Analysis
 - Formal Verification
 - Problem Report Analysis
 - Change Request Analysis.

The complete, three-level, taxonomy may be seen by examining the Table of Contents under Chapter 7.

This type of functional classification scheme was first described in the NBS Taxonomy report [Houghton 1983] and later elaborated in the SEE Taxonomy [Kean 1985]. The E&V Team has followed the general scheme suggested in the above-referenced documents, but with some further elaboration and modifications designed to make the taxonomy compatible with the rest of the multiple indexing scheme used in other parts of this manual.

Figure 7-1 shows how the functions are related to other elements of the E&V Reference Manual and to elements in the E&V Guidebook. The various functions that are useful in defining assessment objectives for APSEs or APSE components are described in this chapter. The relationships between tools and functions given in this chapter are typical (or traditional) relationships. The real capabilities should be determined by the tool specifications, marketing claims, or the like.

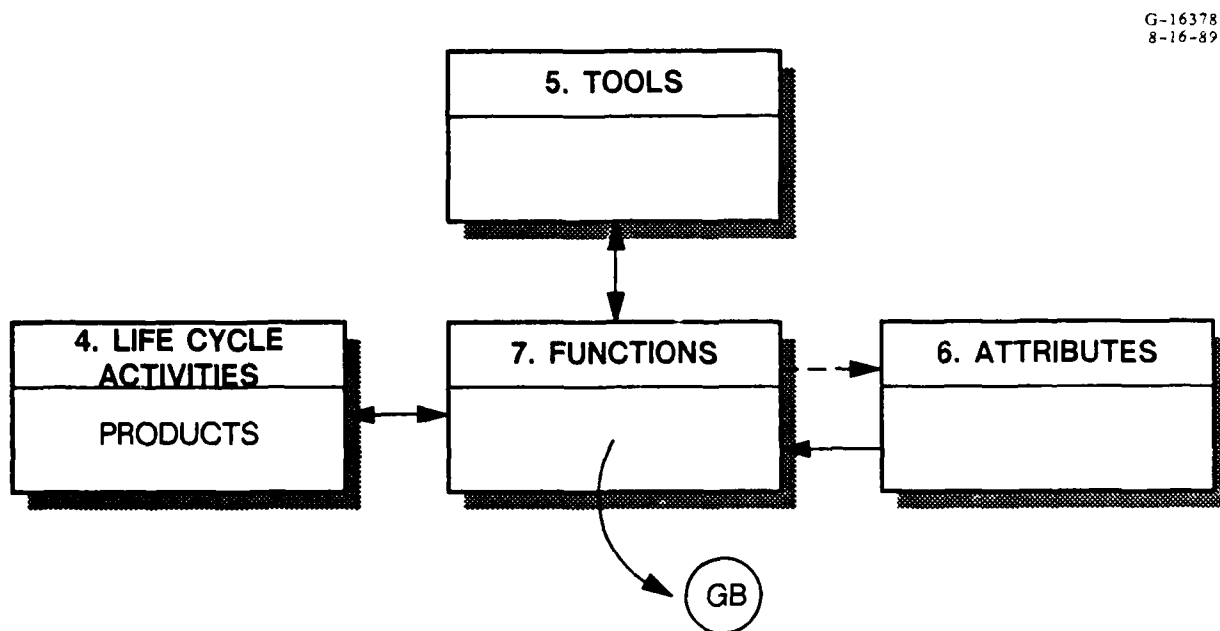


Figure 7-1 Function Relationships

7.1 TRANSFORMATION

Description:

Transformation features describe how the subject is manipulated to accommodate the user's needs. They describe what transformations take place as input to the tool is processed. [Kean 1985]

7.1.1 Editing

Description:

Selective revision of computer-resident data. [Kean 1985]

7.1.1.1 Text Editing

Description:

Editing capabilities provided for textual data. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.2]

Tools:

[Word Processor	5.8.2,
Definition Language Processor	5.9.1,
Specification Language Processor	5.9.2,
Data Model/Dictionary	5.9.4,
Prototyping Tools	5.9.7,
Syntax-Directed (Language-Sensitive) Editor	5.11.5,
Translating Editor	5.12.1]

Guidebook References:

[Completeness	6.4.9,	
@GB: Text Editing Capabilities Checklist		99.1;
Completeness	6.4.9,	
@GB: Language-Sensitive Editing Capabilities Checklist		99.2;
Power	6.4.21,	
@GB: Text Editing Capabilities Checklist		99.1;
Power	6.4.21,	
@GB: Language-Sensitive Editing Capabilities Checklist		99.2]

7.1.1.2 Data Editing

Description:

Editing capabilities provided for data in internal (machine) format.

Cross References:

Life Cycle Activities:

[Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Description and Preparation Services	5.14.4]

Guidebook References:

7.1.1.3 Graphics Editing

Description:

Editing capabilities provided for graphical data. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Global	4.12.2]

Tools:	
[Graphics Generator	5.8.4,
Enterprise Model	5.9.3,
Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Prototyping Tools	5.9.7]

Guidebook References:

7.1.2 Formatting

Description:

Arranging data according to predefined and/or user-defined conventions. [Kean 1985]

7.1.2.1 MIL-STD Format

Description:

The formatting of documents as specified by the MIL-STD(s).

Cross References:

Life Cycle Activities:
[Global

4.12.2]

Tools:
[Formatter

5.8.5]

Guidebook References:

7.1.2.2 Table of Contents

Description:

The production of a table of contents by examining the contents to be placed within the document.

Cross References:

Life Cycle Activities:
[Global

4.12.2]

Tools:
[Word Processor
Formatter

5.8.2,
5.8.5]

Guidebook References:

7.1.2.3 Predefined and User-Defined Forms

Description:

The ability to call up or define standard formats such as those that are mentioned in DoD-STD-2167A.

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.1 – 4.2.3, 4.2.6,
Software Requirements Analysis	4.3.1 – 4.3.3, 4.3.6,
Preliminary Design	4.4.1 – 4.4.3, 4.4.6,
Detailed Design	4.5.1 – 4.5.3, 4.5.6,
Coding and CSU Testing	4.6.1, 4.6.2, 4.6.6,
CSC Integration and Testing	4.7.1 – 4.7.3, 4.7.6,
CSCI Testing	4.8.1 – 4.8.3, 4.8.5, 4.8.6,
System Integration and Testing	4.9.2, 4.9.5,
Operation Testing and Evaluation	4.10.2,
Change Requirements	4.11.2, 4.11.5]

Tools:

[Spreadsheet	5.6.1,
Word Processor	5.8.2,
Formatter	5.8.5,
Applications (Code) Generator	5.11.1,
Database Schema Generator	5.11.2,
Report Generator	5.11.3,
Screen Generator	5.11.4,
Syntax-Directed (Language-Sensitive) Editor	5.11.5]

Guidebook References:

7.1.3 On-Line Assistance Processing

Description:

User interface that is part of the input/output process of a programming support environment (e.g., command assistance, assistance, on-line tutoring, definition assistance, etc.).
[@Kean 1985]

Cross References:

Life Cycle Activities: [Global	4.12.2]
Tools: [On-Line Assistance	5.1.7]

7.1.4 Sort/Merge

Description:

The process of arranging data in a specific order (e.g., alphabetical order). [@Kean 1985]

Cross References:

Life Cycle Activities: [Global	4.12.2]
Tools: [Database Manager	5.2.2,
Address/Phone Book	5.6.3,
Electronic Mail	5.6.4,
Dictionary/Thesaurus	5.6.7]

7.1.5 Graphics Generation

Description:

The input, construction, storage, retrieval, manipulation, alteration, and analysis of pictorial data (e.g., generation of system architectures, software designs, financial analysis, maps, graphs, etc.). [Kean 1985]

Cross References:

Life Cycle Activities:
[Global 4.12.2]

Tools:
[Graphics Generator 5.8.4,
Enterprise Model 5.9.3,
Data Model/Dictionary 5.9.4,
Process Model 5.9.5,
Simulators 5.9.6,
Prototyping Tools 5.9.7,
Screen Generator 5.11.4]

7.1.6 Translation

Description:

The conversion from one language form to another. [Kean 1985]

7.1.6.1 Systems Requirements

Description:

The processing of formal systems requirements statements into an internal database representation for subsequent use. [@Kean 1985]

Cross References:

Life Cycle Activities:
[System Requirements Analysis/Design 4.2.2]

Tools:
[Definition Language Processor 5.9.1,
Enterprise Model 5.9.3,
Process Model 5.9.5]

Guidebook References:

[Augmentability	6.4.4,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Capacity	6.4.6,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness	6.4.9,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness	6.4.9,	
@GB: Time-Critical Applications Support Checklist		9.5;
Processing Effectiveness	6.4.22,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Self Descriptiveness	6.4.28,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
System Clarity	6.4.33,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Training	6.4.36,	
@GB: NADC/SPS CASE Tools Evaluation		9.4]

7.1.6.2 Software Requirements

Description:

The processing of formal software requirements statements into an internal database representation for subsequent use. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Software Requirements Analysis 4.3.2]

Tools:
[Definition Language Processor 5.9.1,
Data Model/Dictionary 5.9.4,
Process Model 5.9.5]

Guidebook References:

[Augmentability	6.4.4,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Capacity	6.4.6,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness:	6.4.9,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness	6.4.9,	
@GB: Time-Critical Applications Support Checklist		9.5;
Processing Effectiveness	6.4.22,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Self Descriptiveness	6.4.28,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
System Clarity	6.4.33,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Training	6.4.36,	
@GB: NADC/SPS CASE Tools Evaluation		9.4]

7.1.6.3 Requirements to Natural Language

Description:

The transformation of formal requirements language constructs into English language text.
[@Kean 1985]

Cross References:

Life Cycle Activities:	
[System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.2]

Tools:

Guidebook References:

7.1.6.4 Preliminary Design

Description:

The processing of formal preliminary design statements into an internal database representation for subsequent use. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.2]
Tools:	
[Specification Language Processor	5.9.2,
Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Prototyping Tools	5.9.7]

Guidebook References:

[Augmentability	6.4.4,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Capacity	6.4.6,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness	6.4.9,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Completeness	6.4.9,	
@GB: Time-Critical Applications Support Checklist		9.5;
Power	6.4.21,	
@GB: SEI Design Support Experiment		9.1;
Processing Effectiveness	6.4.22,	
@GB: SEI Design Support Experiment		9.1;
Processing Effectiveness	6.4.22,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Self Descriptiveness	6.4.28,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Storage Effectiveness	6.4.31,	
@GB: SEI Design Support Experiment		9.1;
System Clarity	6.4.33,	
@GB: NADC/SPS CASE Tools Evaluation		9.4;
Training	6.4.36,	
@GB: NADC/SPS CASE Tools Evaluation		9.4]

7.1.6.5 Detailed Design

Description:

The processing of formal detailed design statements into an internal database representation for subsequent use. [Kean 1985]

Cross References:

Life Cycle Activities:
[Detailed Design 4.5.2]

Tools:
[Specification Language Processor 5.9.2,
Data Model/Dictionary 5.9.4,
Process Model 5.9.5]

Guidebook References:

[Completeness	6.4.9,	
@GB: Time-Critical Applications Support Checklist		9.5;
Power	6.4.21,	
@GB: SEI Design Support Experiment		9.1;
Processing Effectiveness	6.4.22,	
@GB: SEI Design Support Experiment		9.1;
Storage Effectiveness	6.4.31,	
@GB: SEI Design Support Experiment		9.1]

7.1.6.6 Assembling

Description:

Translating a program expressed in an assembly language into object code. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Host-Target System Cross-Assembler	5.13.1]

Guidebook References:

[Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3;
Power	6.4.21,	
@GB: Assembling Checklist		6.1]

7.1.6.7 Compilation

Description:

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

Cross References:

Life Cycle Activities:

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

Tools:

[Translating Editor	5.12.1,
Compiler (Front End)	5.12.2,
Compiler (Code Generator — Back End)	5.12.3]

Guidet ook References:

*[Anomaly Management	6.4.2,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Anomaly Management	6.4.2,	
@GB: Compiler Assessment Questionnaire		5.12;
Augmentability	6.4.4,	
@GB: Compiler Assessment Questionnaire		5.12;
Capacity	6.4.6,	
@GB: IDA Benchmarks		5.2;
Capacity	6.4.6,	
@GB: MITRE Benchmark Generator Tool		5.6;
Capacity	6.4.6,	
@GB: Compiler Assessment Questionnaire		5.12;

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

E&V Reference Manual, Version 2.0

Commonality	6.4.7,	
@GB: Compiler Assessment Questionnaire		5.12;
Completeness	6.4.9,	
@GB: ACVC		5.1;
Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3;
Consistency	6.4.10,	
@GB: Compiler Assessment Questionnaire		5.12;
Cost	6.4.11,	
@GB: Compiler Assessment Questionnaire		5.12;
Document Accessibility	6.4.13,	
@GB: Compiler Assessment Questionnaire		5.12;
Generality	6.4.16,	
@GB: Compiler Assessment Questionnaire		5.12;
Granularity	6.4.17,	
@GB: Compiler Assessment Questionnaire		5.12;
Maturity	6.4.18,	
@GB: Compiler Assessment Questionnaire		5.12;
Operability	6.4.20,	
@GB: Compiler Assessment Questionnaire		5.12;
Power	6.4.21,	
@GB: Compilation Checklist		5.8;
Power	6.4.21,	
@GB: Compiler Assessment Questionnaire		5.12;
*Processing Effectiveness	6.4.22,	
@GB: IDA Benchmarks		5.2;
*Processing Effectiveness	6.4.22,	
@GB: ACEC		5.3;
*Processing Effectiveness	6.4.22,	
@GB: PIWG Benchmark Tests		5.4;
*Processing Effectiveness	6.4.22,	
@GB: University of Michigan Benchmark Tests		5.5;
*Processing Effectiveness	6.4.22,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Processing Effectiveness	6.4.22,	
@GB: Compiler Assessment Questionnaire		5.12;
Proprietary Rights	6.4.23,	
@GB: Compiler Assessment Questionnaire		5.12;

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

E&V Reference Manual, Version 2.0

Rehostability	6.4.25,	
@GB: Compiler Assessment Questionnaire		5.12;
Required Configuration	6.4.26,	
@GB: Compiler Assessment Questionnaire		5.12;
*Retargetability	6.4.27,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Retargetability	6.4.27,	
@GB: Compiler Assessment Questionnaire		5.12;
*Storage Effectiveness	6.4.31,	
@GB: IDA Benchmarks		5.2;
*Storage Effectiveness	6.4.31,	
@GB: ACEC		5.3;
*Storage Effectiveness	6.4.31,	
@GB: PIWG Benchmark Tests		5.4;
*Storage Effectiveness	6.4.31,	
@GB: ARTEWG Catalogue of Ada Runtime Implementation Dependencies		5.10;
Storage Effectiveness	6.4.31,	
@GB: Compiler Assessment Questionnaire		5.12;
System Compatibility	6.4.34,	
@GB: Compiler Assessment Questionnaire		5.12;
Training	6.4.36,	
@GB: Compiler Assessment Questionnaire		5.12;
Visibility	6.4.38,	
@GB: Compiler Assessment Questionnaire		5.12]

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

7.1.6.8 Conversion

Description:

Modifying an existing program to enable it to operate with similar functional capabilities in a different environment. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Host-Target System Cross-Assembler	5.13.1]

Guidebook References:

7.1.6.9 Macro Expansion

Description:

Augmenting instructions in a source language with user defined sequences of instructions in the same source language. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Applications (Code) Generator	5.11.1,
Syntax-Directed (Language-Sensitive) Editor	5.11.5]

Guidebook References:

7.1.6.10 Structure Preprocessing

Description:

Translating a computer program with structured constructs into its equivalent without structured constructs. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:	
[Database Schema Generator	5.11.2]

Guidebook References:

7.1.6.11 Body Stub Generation

Description:

The creation of null bodies for specifications requiring bodies whose corresponding bodies have yet to be defined. [Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2]

Tools:

[Applications (Code) Generator	5.11.1,
Test Description and Preparation Services	5.14.4]

Guidebook References:

7.1.6.12 Preamble Generation

Description:

Generating the preamble for a main program that contains parameters. [Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Applications (Code) Generator	5.11.1]

Guidebook References:

7.1.6.13 Linking/Loading

Description:

The creation of a load/executable module on the host machine from one or more independently translated object modules or load modules by resolving cross-references among the object modules, and possibly relocating elements. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Host-Based Target Linker	5.13.2,
Host-Based Target Loader	5.13.3]

Guidebook References:

[Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3;
Power	6.4.21,	
@GB: Linking/Loading Checklist		6.2]

7.1.6.14 Interpretation

Description:

The translation of a source program into some intermediate data structure, then executing the algorithm by carrying out each operation given in the intermediate structure. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Interpreter	5.12.4]

Guidebook References:

7.1.6.15 Software and System Test Communications

Description:

The exchange or translation of information or data from one level of test for use in constructing test data or scenarios for other levels of test. For example, given a set of test data from a unit test, constructing system test inputs to drive the unit in an identical manner.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2, 4.9.3,
Operational Testing and Evaluation	4.10.2, 4.10.3]

Tools:

[Test Description and Preparation Services	5.14.4]

Guidebook References:

7.1.6.16 Compression

Description:

The translation of data into a more densely-packed format for the purposes of saving space on the storage media and/or reducing transmission time. The process may have to be reversed in order for the data to be processed again.

Cross References:

Life Cycle Activities:
[Global 4.12.2]

Tools:
[Archive, Backup, and Retrieval System 5.1.2]

Guidebook References:

7.1.6.17 Expansion

Description:

The translation of data from a densely-packed format into a format that is more natural for the purpose of processing the data.

Cross References:

Life Cycle Activities:
[Global 4.12.2]

Tools:
[Archive, Backup, and Retrieval System 5.1.2]

Guidebook References:

7.1.6.18 Encryption

Description:

The translation of data into a form in which it is impossible to understand the meaning of the data until the process has been reversed.

Cross References:

Life Cycle Activities:
[Global 4.12.1]

Tools:
[Security System 5.1.3]

Guidebook References:

7.1.6.19 Decryption

Description:

The translation of data from a form in which it is impossible to understand the meaning of the data into a form in which the data can be understood.

Cross References:

Life Cycle Activities:
[Global

4 [2.1]

Tools:
[Security System

5.1.3]

Guidebook References:

7.1.7 Synthesis

Description:

The generation of programs according to predefined rules from a program specification or intermediate language. [@Kean 1985]

7.1.7.1 Design Generation

Description:

The translation or interpretation used to construct program designs.

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.2,
Detailed Design	4.5.2]

Tools:	
[Specification Language Processor	5.9.2,
Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Prototyping Tools	5.9.7]

Guidebook References:

7.1.7.2 Requirements Reconstruction

Description:

The extraction of software requirements statements from a program design language.

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.4,
Detailed Design	4.5.4,
CSC Integration and Testing	4.7.4]

Tools:	
[Design Library Manager	5.9.8,
Reusable Components Library	5.11.8]

Guidebook References:

7.1.7.3 Program Generation

Description:

The translation or interpretation used to construct computer programs (e.g., language translator generator, syntax analyzer generator, code generator generator, environment definition generator, user interface generator, etc.). [Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Prototyping Tools	5.9.7,
Applications (Code) Generator	5.11.1,
Report Generator	5.11.3,
Screen Generator	5.11.4,
Tool Building Services	5.14.2]

Guidebook References:

7.1.7.4 Source Reconstruction

Description:

The extraction of lexical and structure attributes from an intermediate language. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.4,
CSC Integration and Testing	4.7.4]

Tools:

Guidebook References:

7.1.7.5 Decompilation

Description:

The translation of machine code sequences into a higher level language (i.e., back into the same language from which the machine code sequences were generated). [Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.4,
CSC Integration and Testing	4.7.4]

Tools:

Guidebook References:

7.1.7.6 Disassembling

Description:

The translation of machine code sequences into assembly language sequences. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.4,
CSC Integration and Testing	4.7.4]

Tools:

Guidebook References:

7.1.7.7 Test Harness Generation

Description:

The construction of an environment for running software component tests, linking test capabilities into an integrated toolset to perform specific tests, accepting program inputs, simulating missing components, comparing actual outputs with expected outputs to determine correctness, and reporting discrepancies. [DeMillo 1987]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Execution Services	5.14.5]

Guidebook References:

7.2 MANAGEMENT

Description:

Features that aid the management or control of system/software development. [@Kean 1985]

7.2.1 Information Management

Description:

The organization, accessing, modification, dissemination, and processing of information that is associated with the development of a software system. [@Kean 1985]

7.2.1.1 Database (Object) Management

Description:

Managing a collection of interrelated data stored together with controlled redundancy to serve one or more applications and independent of the programs using the data. [Kean 1985]

Cross References:

Life Cycle Activities:

[Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.1]

Tools:

[Database Manager	5.2.2,
Virtual Operating System	5.3.1,
Enterprise Model	5.9.3,
Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Design Library Manager	5.9.8,
Data Definition Language Processor	5.11.6,
Data Manipulation Language Processor	5.11.7,
Reusable Components Library	5.11.8]

Guidebook References:

[Completeness	6.4.9,	
@GB: Database Management Checklist		15.2;
Power	6.4.21,	
@GB: Database Management Checklist		15.2]

7.2.1.2 Documentation Management

Description:

The development and control of software documentation. [@Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.3,
Software Requirements Analysis	4.3.3,
Preliminary Design	4.4.2, 4.4.3,
Detailed Design	4.5.2, 4.5.3,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.1]

Tools:

[Document Manager	5.8.1]

Guidebook References:

7.2.1.3 File Management

Description:

Providing and controlling access to files associated with the development of software.
[@Kean 1985]

Cross References:

Life Cycle Activities:	
[Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.1]

Tools:	
[File Manager	5.2.1,
Virtual Operating System	5.3.1]

Guidebook References:

[Completeness	6.4.9,	
@GB: File Management Checklist		15.1;
Power	6.4.21,	
@GB: File Management Checklist		15.1]

7.2.1.4 Electronic Mail

Description:

The process of receiving/sending messages from/to other system users. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.1]

Tools:
[Address/Phone Book
Electronic Mail

5.6.3,
5.6.4]

Guidebook References:

[Completeness
@GB: Electronic Mail Checklist

6.4.9,

15.3;

Power
@GB: Electronic Mail Checklist

6.4.21,

15.3]

7.2.1.5 Electronic Conferencing

Description:

The concurrent on-line correspondence between two or more users. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.1]

Tools:
[Address/Phone Book
Electronic Conferencing

5.6.3,
5.6.5]

Guidebook References:

7.2.1.6 Specification Management

Description:

The control of requirements specifications. Specification management features are somewhat methodology dependent because associated with them are requirements languages with formal procedures for their use. [Kean 1985]

Cross References:

Life Cycle Activities:

[Software Requirements Analysis	4.3.5,
Preliminary Design	4.4.5,
Detailed Design	4.5.5]

Tools:

[Specification Language Processor	5.9.2,
Data Model/Dictionary	5.9.4,
Process Model	5.9.5]

Guidebook References:

7.2.1.7 Program Library Management

Description:

The creation, manipulation, display, and deletion of the various components of a program library. A program library is a repository for all program information (e.g., source programs, object programs, executable programs, documentation, data, etc.). [Kean 1985]

Cross References:

Life Cycle Activities:

[Software Requirements Analysis	4.3.2,
Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Configuration Control	5.7.2,
Version Control	5.7.4,
Design Library Manager	5.9.8,
Reusable Components Library	5.11.8,
Program Library Manager	5.12.5]

Guidebook References:

[Completeness	6.4.9,	
@GB: Program Library Management Checklist		5.9;
Power	6.4.21,	
@GB: Program Library Management Checklist		5.9]

7.2.1.8 Test Data Management

Description:

The development and control of software test data. [@Kean 1985]

Cross References:

Life Cycle Activities:

[Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Description and Preparation Services	5.14.4]

Guidebook References:

7.2.1.9 Evaluation Results Management

Description:

The cataloguing and maintenance of the results from the operational test and evaluation of the product. [Kean 1985]

Cross References:

Life Cycle Activities:
[Operational Testing and Evaluation 4.10.5]

Tools:
[Test Analysis Services 5.14.6]

Guidebook References:

7.2.1.10 Performance Monitoring

Description:

The monitoring of the performance characteristics of the finished product. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.1]

Tools:
[Performance Monitor

5.1.9]

Guidebook References:

[Completeness
@GB: Performance Monitoring Checklist

6.4.9,

99.3]

7.2.1.11 Data and Error Logging

Description:

The automatic capture of information describing system state changes and anomalies or exceptions that occur during test execution.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2, 4.9.3,
Operational Testing and Evaluation	4.10.2, 4.10.3]

Tools:

[Test Execution Services	5.14.5]

Guidebook References:

7.2.1.12 Status Display

Description:

The presentation of information concerning test execution, software operation, and system state.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.2.2 Project Management

Description:

The management of a system/software development project. [@Kean 1985]

7.2.2.1 Cost Estimation

Description:

The process of determining the amount of labor necessary for the completion of a task, the amount and potential costs of computer time required, etc., prior to and during a project's lifetime. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global 4.12.1]

Tools:
[Cost Estimator 5.5.1]

Guidebook References:

7.2.2.2 Quality Specification

Description:

The selection and prioritization of quality factors required for a given application; availability of techniques to apply trade-off analyses (quality factor vs. quality factor, and quality factor vs. cost) and quantify quality requirements. Potential quality factors include: efficiency, integrity, reliability, survivability, usability, correctness, maintainability, verifiability/testability, expandability/flexibility, interoperability, reusability, and transportability. [Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.1]

Tools:

Guidebook References:

7.2.2.3 Scheduling

Description:

The process of identifying tasks, products to be delivered, delivery dates, personnel needed to complete tasks, etc. for a development project. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global 4.12.1]

Tools:
[Scheduler 5.5.3,
Calendar 5.6.6]

Guidebook References:

7.2.2.4 Work Breakdown Structure

Description:

The enumeration of all work activities in hierarchic refinements of detail that divides work to be done into short, manageable tasks with quantifiable inputs, outputs, schedules, and assigned responsibilities. [Kean 1985]

Cross References:

Life Cycle Activities:
[Global 4.12.1]

Tools:
[Work Breakdown Structure Editor 5.5.4]

Guidebook References:

7.2.2.5 Resource Estimation

Description:

The estimation of resources attributed to an entity. [@Kean 1985]

Cross References:

Life Cycle Activities:

[Software Requirements Analysis	4.3.1,
Preliminary Design	4.4.1,
Detailed Design	4.5.1,
Coding and CSU Testing	4.6.1,
CSC Integration and Testing	4.7.1,
CSCI Testing	4.8.1]

Tools:

[Resource Estimator	5.5.5]

Guidebook References:

7.2.2.6 Tracking

Description:

The tracking of the development of an entity through the software life cycle. [Kean 1985]

Cross References:

Life Cycle Activities:

[Software Requirements Analysis	4.3.1,
Preliminary Design	4.4.1,
Detailed Design	4.5.1,
Coding and CSU Testing	4.6.1,
CSC Integration and Testing	4.7.1,
CSCI Testing	4.8.1]

Tools:

[Tracking	5.5.6,
Analysis and Reporting	5.5.9,
Configuration Status Accounting	5.7.3,
History	5.7.5]

Guidebook References:

7.2.2.7 Configuration Management

Description:

The establishment of baselines for configuration items, the control of changes to these baselines, and the control of releases to the operational environment. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global 4.12.5]

Tools:
[Configuration Identification 5.7.1,
Configuration Control 5.7.2,
Configuration Status Accounting 5.7.3,
Version Control 5.7.4,
History 5.7.5]

Guidebook References:

[Augmentability 6.4.4,
@GB: Configuration Management Assessment Questionnaire 10.3;
Capacity 6.4.6,
@GB: Configuration Management Assessment Questionnaire 10.3;
Commonality 6.4.7,
@GB: Configuration Management Assessment Questionnaire 10.3;
Completeness 6.4.9,
@GB: Configuration Management Capabilities Checklist 10.1;
Completeness 6.4.9,
@GB: Configuration Management Assessment Questionnaire 10.3;
Consistency 6.4.10,
@GB: Configuration Management Assessment Questionnaire 10.3;
Cost 6.4.11,
@GB: Configuration Management Assessment Questionnaire 10.3;
Document Accessibility 6.4.13,
@GB: Configuration Management Assessment Questionnaire 10.3;
Generality 6.4.16,
@GB: Configuration Management Assessment Questionnaire 10.3;

E&V Reference Manual, Version 2.0

Granularity	6.4.17,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Maturity	6.4.18,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Operability	6.4.20,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Power	6.4.21,	
@GB: SEI Configuration Management Experiment		10.2;
Power	6.4.21,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Processing Effectiveness	6.4.22,	
@GB: SEI Configuration Management Experiment		10.2;
Processing Effectiveness	6.4.22,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Proprietary Rights	6.4.23,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Rehostability	6.4.25,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Storage Effectiveness	6.4.31,	
@GB: Configuration Management Assessment Questionnaire		10.3;
System Compatibility	6.4.34,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Training	6.4.36,	
@GB: Configuration Management Assessment Questionnaire		10.3;
Visibility	6.4.38,	
@GB: Configuration Management Assessment Questionnaire		10.3]

7.2.2.8 Quality Assessment

Description:

The use of field data to determine the achieved level of quality in deployed software systems.
A verification that the specified quality requirements have been met. [Kean 1985]

Cross References:

Life Cycle Activities:

Tools:

[Design Library Manager

Reusable Components Library

5.9.8,

5.11.8]

Guidebook References:

7.2.2.9 Risk Analysis

Description:

The process of determining the amount of risk associated with the completion of a task, prior to and during a project's lifetime. Risk may be defined as the probability of missing a deadline, probability of overrunning a budget, probability of technological failure, and the like. Also includes the analysis of the operational system, for example, mean time between failures (MTBF) and number of remaining errors.

Cross References:

Life Cycle Activities: [Global	4.12.1]

Tools: [Analysis and Reporting	5.5.9]

Guidebook References:

7.2.3 Computer System Management

Description:

The management of hardware/software architectures to support the life cycle software engineering environment. Such services include: creating, scheduling, and removing tasks; switching the processor among tasks; sending messages between tasks; providing direct and import/export access; managing distributed systems; sending files from one host machine to another on a network, etc. [Kean 1985]

7.2.3.1 Command Language Processing

Description:

The processing of command language constructs into functions performed by the operating system. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Global	4.12.2]
Tools:	
[Command Language Interface	5.1.1]

Guidebook References:

[Anomaly Management	6.4.2,
Augmentability	6.4.4,
Capacity	6.4.6,
Commonality	6.4.7,
Consistency	6.4.10,
Cost	6.4.11,
Document Accessibility	6.4.13,
Generality	6.4.16,
Granularity	6.4.17,
Maturity	6.4.18,
Operability	6.4.20,
Power	6.4.21,
Processing Effectiveness	6.4.22,
Proprietary Rights	6.4.23,
Rehostability	6.4.25,
Required Configuration	6.4.26,
Storage Effectiveness	6.4.31,
System Compatibility	6.4.34,
Training	6.4.36,
Visibility	6.4.38,
@GB: Command Language Interpreter	
Assessment Questionnaire	99.4]

7.2.3.2 Input/Output Support

Description:

The services for accessing standard I/O devices such as disks, tapes, terminals, and printers from within a program.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.2]

Tools:

[Input and Output Services	5.1.8,
Virtual Operating System	5.3.1,
Window Manager	5.3.2,
I/O Pipes	5.3.4,
RAM Cache	5.3.5,
Runtime System	5.12.6]

Guidebook References:

7.2.3.3 Kernel

Description:

The functions which provide access for tools and application programs to services of the native operating system.

Cross References:

Life Cycle Activities:
[Global 4.12.2]

Tools:
[Security System 5.1.3,
Job Scheduler 5.1.4,
Resource Controller 5.1.5,
Virtual Operating System 5.3.1,
Window Manager 5.3.2]

Guidebook References:

[Completeness 6.4.9, 8.1;
@GB: CIVC
Completeness 6.4.9, 8.2]
@GB: Tool Support Interface Evaluation

7.2.3.4 Math/Statistics

Description:

The services for supporting standard math and statistical operations.

Cross References:

Life Cycle Activities:
[Global

4.12.2]

Tools:
[Spreadsheet
Calculator

5.6.1,
5.6.2]

Guidebook References:

7.2.3.5 Runtime Environment

Description:

The functions that can be expected to be found in the runtime libraries for Ada implementations. "It should be noted that the dividing line between the predefined runtime support library on the 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. Not addressed here are elements of a runtime library that exist because of special characteristics of the underlying computing resource." [ARTEWG 1988]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2, 4.9.3,
Operational Testing and Evaluation	4.10.2, 4.10.3]

Tools:

[Job Scheduler	5.1.4,
Resource Controller	5.1.5,
Virtual Operating System	5.3.1,
Runtime System	5.12.6]

Guidebook References:

[Anomaly Management	6.4.2,	
@GB: Runtime Support System Questionnaire		5.14;
Communication Effectiveness	6.4.8,	
@GB: Runtime Support System Questionnaire		5.14;
Completeness	6.4.9,	
@GB: ARTEWG Runtime Environment Taxonomy		5.11;
Completeness	6.4.9,	
@GB: Runtime Support System Questionnaire		5.14;
Functional Scope	6.4.15,	
@GB: Runtime Support System Questionnaire		5.14;

E&V Reference Manual, Version 2.0

Generality	6.4.16,	
@GB: Runtime Support System Questionnaire		5.14;
Granularity	6.4.17,	
@GB: Runtime Support System Questionnaire		5.14;
Modularity	6.4.19,	
@GB: Runtime Support System Questionnaire		5.14;
Retargetability	6.4.27,	
@GB: Runtime Support System Questionnaire		5.14;
System Accessibility	6.4.32,	
@GB: Runtime Support System Questionnaire		5.14;
System Compatibility	6.4.34,	
@GB: Runtime Support System Questionnaire		5.14]

7.2.3.6 Import/Export

Description:

The services for communicating objects between various computer systems or networks.

Cross References:

Life Cycle Activities:

[CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.2]

Tools:

[Archive, Backup, and Retrieval System	5.1.2,
Import/Export System	5.1.6,
Virtual Operating System	5.3.1,
Host-to-Target Downloader	5.13.7,
Target-to-Host Uploader	5.13.8]

Guidebook References:

[Completeness	6.4.9,	
@GB: Import/Export Checklist		6.3]

7.2.3.7 Access Control

Description:

The process of controlling access to various components of the environment.

Cross References:

Life Cycle Activities:
[Global

4.12.1]

Tools:
[Security System
Virtual Operating System

5.1.3,
5.3.1]

Guidebook References:

7.2.3.8 Job Scheduling

Description:

The process of controlling the execution of the programs.

Cross References:

Life Cycle Activities:
[Global

4.12.2]

Tools:
[Job Scheduler
Virtual Operating System

5.1.4,
5.3.1]

Guidebook References:

7.2.3.9 Resource Management

Description:

The process of managing computing resources such as main and secondary memory and I/O channels to processes on the system.

Cross References:

Life Cycle Activities:
[Global 4.12.2]

Tools:
[Resource Controller 5.1.5,
Virtual Operating System 5.3.1]

Guidebook References:

7.3 ANALYSIS

Description:

The features that provide an examination of a substantial whole to determine both qualitative and quantitative properties. [@Kean 1985]

Guidebook References:

[Completeness	6.4.9,	
@GB: Testing Capabilities Checklist		7.1;
Power	6.4.21,	
@GB: Testing Capabilities Checklist		7.1]

7.3.1 Static Analysis

Description:

Static analysis features specify operations on the subject without regard to the executability of the subject. They describe the manner in which the subject is analyzed. [@Kean 1985]

7.3.1.1 Comparison

Description:

The determination and assessment of similarities between two or more items. [@Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2 - 4.7.4,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2,
Global	4.12.4]
Tools:	
[Dictionary/Thesaurus	5.6.7,
Test Description and Preparation Services	5.14.4]

Guidebook References:

7.3.1.2 Spelling Checking

Description:

The identification of incorrectly spelled words. [@Kean 1985]

Cross References:

Life Cycle Activities:
[Global

4.12.4]

Tools:
[Dictionary/Thesaurus
Spell Checker

5.6.7,
5.8.3]

Guidebook References:

7.3.1.3 Data Flow Analysis

Description:

The analysis of data transformation paths to identify anomalies and specify paths to be exercised during testing. [DeMillo 1987]

Cross References:

Life Cycle Activities:

[Software Requirements Analysis	4.3.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.4 Functional Analysis

Description:

The analysis of formally stated requirements to determine their consistency and completeness. [Kean 1985]

Cross References:

Life Cycle Activities:
[Software Requirements Analysis 4.3.2]

Tools:
[Static Analyzers 5.14.1]

Guidebook References:

7.3.1.5 Interface Analysis

Description:

The checking of interfaces between program elements for consistency and adherence to predefined rules and/or axioms. [Kean 1985]

Cross References:

Life Cycle Activities:

[Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Process Model	5.9.5,
Design Library Manager	5.9.8,
Applications (Code) Generator	5.11.1,
Reusable Components Library	5.11.8,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.6 Traceability Analysis

Description:

The checking for internal consistency within the software requirements specifications.
[@Kean 1985]

Cross References:

Life Cycle Activities:
[Software Requirements Analysis 4.3.2]

Tools:
[Data Model/Dictionary 5.9.4,
Process Model 5.9.5,
Static Analyzers 5.14.1]

Guidebook References:

7.3.1.7 Testability Analysis

Description:

The quantitative measurement of the extent to which a component facilitates the establishment of verification criteria and supports evaluation of its performance.

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.3,
Software Requirements Analysis	4.3.4,
Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.8 Test Condition Analysis

Description:

The analysis of formal requirements language statements to determine data values to be examined and mechanisms to be used in the verification of test results. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.3,
Detailed Design	4.5.2, 4.5.3,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:	
[Test Building Services	5.14.3]

Guidebook References:

7.3.1.9 Quality Measurement

Description:

The quantitative measurement of specified quality factors for use during the evaluation of software products (and prediction of software quality) at key milestones during development. Factors to be analyzed include: efficiency, integrity, reliability, survivability, usability, correctness, maintainability, verifiability/testability, expandability/flexibility, interoperability, reusability, and transportability. [@Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.2,
Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Static Analyzers	5.14.1,
Test Execution Services	5.14.5]

Guidebook References:

7.3.1.10 Complexity Measurement

Description:

The determination of how complicated an entity (e.g., routine, program, system, etc.) is by evaluating some number of associated characteristics. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:	
[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.11 Correctness Checking

Description:

The determination of agreement between the component as coded and the programming specification (algorithmic correctness).

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.12 Completeness Checking

Description:

The assessment of whether or not an entity has all its parts present and if those parts are fully developed. [Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.4,
Preliminary Design	4.4.4,
Detailed Design	4.5.4,
Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2 - 4.8.4,
System Integration and Testing	4.9.2, 4.9.4,
Operational Testing and Evaluation	4.10.2, 4.10.4]

Tools:

[Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.13 Consistency Checking

Description:

The determination of whether or not an entity is internally consistent in the sense that it is consistent with its specification. [Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.2 - 4.2.4,
Software Requirements Analysis	4.3.4,
Preliminary Design	4.4.4,
Detailed Design	4.5.4,
Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2, 4.8.4,
System Integration and Testing	4.9.2, 4.9.4,
Operational Testing and Evaluation	4.10.2, 4.10.4]

Tools:

[Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.14 Reusability Analysis

Description:

The quantitative measurement of specified reusability factors for use during the evaluation of software products (and prediction of software reusability).

Cross References:

Life Cycle Activities:

[Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2]

Tools:

[Process Model	5.9.5,
Design Library Manager	5.9.8,
Reusable Components Library	5.11.8,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.15 Syntax and Semantics Checking

Description:

The detection of errors in the syntax and semantics in the use of a formal language.

Cross References:

Life Cycle Activities:
[Coding and CSU Testing 4.6.2]

Tools:
[Word Processor 5.8.2,
Syntax-Directed (Language-Sensitive) Editor 5.11.5,
Translating Editor 5.12.1,
Compiler (Front End) 5.12.2,
Static Analyzers 5.14.1]

Guidebook References:

[Completeness 6.4.9,
@GB: Language-Sensitive Editing Capabilities Checklist 99.2;
Power 6.4.21,
@GB: Language-Sensitive Editing Capabilities Checklist 99.2]

7.3.1.16 Reachability Analysis

Description:

The detection of sections of code that cannot be executed because of the structure of the code unit that contains it.

Cross References:

Life Cycle Activities:

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

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.17 Cross Reference

Description:

The referencing of entities to other entities by logical means. [Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.3,
Software Requirements Analysis	4.3.3, 4.3.4,
Preliminary Design	4.4.4,
Detailed Design	4.5.2 - 4.5.4,
Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2 - 4.7.4,
CSCI Testing	4.8.2, 4.8.4,
System Integration and Testing	4.9.2, 4.9.4,
Operational Testing and Evaluation	4.10.2, 4.10.4]

Tools:

[Data Model/Dictionary	5.9.4,
Process Model	5.9.5,
Static Analyzers	5.14.1,
Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.18 Maintainability Analysis

Description:

The quantitative measurement of specified maintainability factors for use during the evaluation of software products (and prediction of software maintainability).

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.2,
Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2, 4.8.4,
System Integration and Testing	4.9.2, 4.9.4,
Operational Testing and Evaluation	4.10.2, 4.10.4]

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.19 Invocation Analysis

Description:

The analysis of a system/software design for the purpose of determining calling relationships between elements. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.2,
Detailed Design	4.5.2]

Tools:	
[Process Model	5.9.5,
Applications (Code) Generator	5.11.1,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.20 Scanning

Description:

The examination of entities sequentially to identify key areas or structure. [@Kean 1985]

Cross References:

Life Cycle Activities:

[Preliminary Design

4.4.4,

Detailed Design

4.5.4,

Coding and CSU Testing

4.6.4,

CSC Integration and Testing

4.7.4]

Tools:

[Static Analyzers

5.14.1]

Guidebook References:

7.3.1.21 Structured Walkthrough

Description:

The interactive display of source code (or its design) with the capability for branching to subordinate program modules/units. This feature automates code reading such that a walkthrough of a computer program can be performed on a video terminal. [@Kean 1985]

Cross References:

Life Cycle Activities:

[Preliminary Design

4.4.2,

Detailed Design

4.5.2,

Coding and CSU Testing

4.6.2]

Tools:

[Static Analyzers

5.14.1]

Guidebook References:

7.3.1.22 Auditing

Description:

The conducting of an examination to determine whether or not predefined rules have been followed. [Kean 1985]

Cross References:

Life Cycle Activities:

[System Requirements Analysis/Design	4.2.4,
Software Requirements Analysis	4.3.4,
Preliminary Design	4.4.2, 4.4.4,
Detailed Design	4.5.2, 4.5.4,
Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2 - 4.7.4,
CSCI Testing	4.8.2, 4.8.4,
System Integration and Testing	4.9.2, 4.9.4,
Operational Testing and Evaluation	4.10.2, 4.10.4]

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.23 Error Checking

Description:

The determination of discrepancies, their importance, and/or their cause. [Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Decision Support Services	5.14.7]

Guidebook References:

7.3.1.24 Statistical Analysis

Description:

The performance of statistical data collection and analysis. [@Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.25 Statistical Profiling

Description:

The analysis of a computer program to determine statement types, number of occurrences of each statement type, and the percentage of each statement type in relation to the complete program. [Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
System Integration and Testing	4.9.3,
Operational Testing and Evaluation	4.10.3]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.26 Structure Checking

Description:

The detection of structural flaws within a program (e.g., improper loop nestings, unreferenced labels, unreachable statements, and statements with no successors). Flaws detected by this function are not illegal or erroneous, but do constitute bad style. [Kean 1985]

Cross References:

Life Cycle Activities:
[Coding and CSU Testing

4.6.2]

Tools:
[Static Analyzers

5.14.1]

Guidebook References:

7.3.1.27 Type Analysis

Description:

The evaluation of whether or not the domain of values attributed to an entity are properly and consistently defined. [Kean 1985]

Cross References:

Life Cycle Activities:

[Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Data Model/Dictionary	5.9.4,
Static Analyzers	5.14.1]

Guidebook References:

7.3.1.28 Units Analysis

Description:

The determination of whether or not the units or physical dimensions attributed to an entity are properly defined and consistently used. [Kean 1985]

Cross References:

Life Cycle Activities:

[Preliminary Design	4.4.2,
Detailed Design	4.5.2,
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Static Analyzers	5.14.1]

Guidebook References:

7.3.1.29 I/O Specification Analysis

Description:

The analysis of the input and output specifications in a program usually for the generation of test data. [Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Test Building Services	5.14.3]

Guidebook References:

7.3.1.30 Sizing Analysis

Description:

The quantitative measurement of the maximum amount of primary (and secondary) storage required to run a program.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2, 4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2 - 4.8.4,
System Integration and Testing	4.9.2 - 4.9.4,
Operational Testing and Evaluation	4.10.2 - 4.10.4]

Tools:

[Size Estimator	5.5.2,
Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.31 Data Reduction and Analysis

Description:

The extraction and processing of key information from test execution logs to determine correctness or appropriateness of system behavior during test execution.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2, 4.9.3,
Operational Testing and Evaluation	4.10.2, 4.10.3]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.3.1.32 Random Test Generation

Description:

The construction of test data sets by randomly choosing a subset of all possible input values. The distribution may be arbitrary, or it may attempt to accurately reflect the distribution of inputs in the application environment. [DeMillo 1987]

Cross References:

Life Cycle Activities:

[CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Building Services	5.14.3]

Guidebook References:

7.3.2 Dynamic Analysis

Description:

Dynamic analysis features specify operations that are determined during or after execution takes place. Dynamic analysis features differ from those classified as static by virtue of the fact that they require some form of symbolic or machine execution. They describe the techniques used by the tool to derive meaningful information about a program's execution behavior. [Kean 1985]

Guidebook References:

[Power	6.4.21,	
@GB: SEI Unit Testing and Debugging Experiment		7.2;
Processing Effectiveness	6.4.22,	
@GB: SEI Unit Testing and Debugging Experiment		7.2]

7.3.2.1 Requirements Simulation

Description:

The execution of code enhanced requirements statements to examine functional interfaces and performance. [Kean 1985]

Cross References:

Life Cycle Activities:	
[System Concepts	4.1.2,
System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.2]
Tools:	
[Specifications Language Processor	5.9.2,
Simulators	5.9.6]

Guidebook References:

7.3.2.2 Requirements Prototyping

Description:

The rapid construction of critical functions of a system early in the life cycle for the purpose of understanding the requirements. During this activity, these results are intended to be thrown away. [Kean 1985]

Cross References:

Life Cycle Activities:

[System Concepts	4.1.2,
System Requirements Analysis/Design	4.2.2,
Software Requirements Analysis	4.3.2]

Tools:

[Prototyping Tools	5.9.7]

Guidebook References:

[Completeness	6.4.9,	
@GB: Requirements Prototyping Capabilities Checklist		9.2]

7.3.2.3 Simulation and Modeling

Description:

The representation of selected characteristics of the behavior of one physical or abstract system by another system (e.g., the representation of physical phenomena by means of operations performed by a computer system, the representation of operations of a computer system by those of another computer system, etc.). [Kean 1985]

Cross References:

Life Cycle Activities:
[System Concepts
Preliminary Design

4.1.2,
4.4.2]

Tools:
[Enterprise Model
Data Model/Dictionary
Process Model
Simulators
Prototyping Tools
Host-Based Target System
Instruction-Level Simulators

5.9.3,
5.9.4,
5.9.5,
5.9.6,
5.9.7,
5.13.4]

Guidebook References:

[Completeness	6.4.9,	
@GB: Instruction-Level Simulation Checklist		6.8;
Completeness	6.4.9,	
@GB: Simulation and Modeling Capabilities Checklist		9.3;
Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3]

7.3.2.4 Design Prototyping

Description:

The rapid construction of critical functions of a system early in the life cycle for the purpose of understanding the requirements. During this activity, the results will be retained for the purpose of incrementally developing the product. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Preliminary Design	4.4.2,
Detailed Design	4.5.2]

Tools:	
[Prototyping Tools	5.9.7]

Guidebook References:

7.3.2.5 Debugging

Description:

The process of locating, analyzing, and correcting suspected faults in a program. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Host-Based Target Code Symbolic Debugger	5.13.6]

Guidebook References:

[Completeness	6.4.9,	
@GB: Debugging Capabilities Checklist		6.5;
Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3;
Power	6.4.21,	
@GB: Debugging Capabilities Checklist		6.5;
Power	6.4.21,	
@GB: SEI Unit Testing and Debugging Experiment		7.2;
Processing Effectiveness	6.4.22,	
@GB: SEI Unit Testing and Debugging Experiment		7.2]

7.3.2.6 Executable Assertion Checking

Description:

The checking of user-embedded statements that assert relationships between elements of a program. An assertion is a logical expression that specifies a condition or relation among the program variables. Checking may be performed with symbolic or run-time data. [Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2, 4.8.3,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Building Services	5.14.3]

Guidebook References:

7.3.2.7 Constraint Evaluation (Contention)

Description:

The generation and/or solution of path input or output constraints for determining test input or proving programs correct. [Kean 1985]

Cross References:

Life Cycle Activities:
[Coding and CSU Testing 4.6.2]

Tools:
[Test Building Services 5.14.3]

Guidebook References:

7.3.2.8 Coverage/Frequency Analysis

Description:

The determination and assessment of measures associated with the invocation of program structural elements to determine the adequacy of a test run. Coverage analysis is useful when attempting to execute each statement, branch, path, or program. [Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Test Building Services	5.14.3]

Guidebook References:

7.3.2.9 Mutation Analysis

Description:

The application of test data to a program and its "mutants" (i.e., programs that contain one or more likely errors) in order to determine test data adequacy. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Test Building Services	5.14.3]

Guidebook References:

7.3.2.10 Symbolic Execution

Description:

The reconstruction of the logic and computations along a program path by executing the path with symbolic rather than actual values of data. [Kean 1985]

Cross References:

Life Cycle Activities:
[Coding and CSU Testing 4.6.2]

Tools:
[Test Building Services 5.14.3]

Guidebook References:

7.3.2.11 Regression Testing

Description:

The rerunning of test cases which a program has previously executed correctly in order to detect errors spawned by changes or corrections made during software development and maintenance. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:	
[Test Execution Services	5.14.5]

Guidebook References:

7.3.2.12 Resource Utilization

Description:

The analysis of resource utilization associated with system hardware or software. [Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2 - 4.8.4,
System Integration and Testing	4.9.2 - 4.9.4,
Operational Testing and Evaluation	4.10.2 - 4.10.4]

Tools:

[Size Estimator	5.5.2,
Test Analysis Services	5.14.6]

Guidebook References:

7.3.2.13 Emulation

Description:

The imitation of all or part of one computer system by another, primarily by hardware, so that the imitating computer system accepts the same data, executes the same programs, and achieves the same results as the imitated system. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[RAM Cache	5.3.5,
Prototyping Tools	5.9.7,
Host-Based Target System	
Instruction-Level Emulator	5.13.5,
Test Execution Services	5.14.5]

Guidebook References:

[Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3;
Power	6.4.21,	
@GB: Emulation Capabilities Checklist		6.4]

7.3.2.14 Timing Analysis

Description:

The reporting of actual CPU, wall-clock, or other times associated with parts of a program.
[@Kean 1985]

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.4,
CSC Integration and Testing	4.7.2, 4.7.4,
CSCI Testing	4.8.2 - 4.8.4,
System Integration and Testing	4.9.2 - 4.9.4,
Operational Testing and Evaluation	4.10.2 - 4.10.4]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

[Completeness	6.4.9,	
@GB: Timing Analysis Capabilities Checklist		6.6;
Completeness	6.4.9,	
@GB: Cross Development System Support Questionnaire		13.3]

7.3.2.15 Tuning

Description:

The activity of optimizing parts of a program which account for significant amounts of resources. The optimization follows the determination of a "performance profile" which is the identification of the parts of a program which use the most resources.

Cross References:

Life Cycle Activities:

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

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.3.2.16 Reliability Analysis

Description:

The determination of the ability of an item to perform a required function under stated conditions for a stated period of time. [@Kean 1985]

Cross References:

Life Cycle Activities:

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

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

7.3.2.17 Real-Time Analysis

Description:

The determination of the behavior of a program which must recognize, interpret, and respond to external, (usually) asynchronous events at speeds which allow the system to handle each event in a specified amount of time.

Cross References:

Life Cycle Activities:

[Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2, 4.7.3,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Test Analysis Services	5.14.6]

Guidebook References:

[Completeness	6.4.9,	
@GB: Real-Time Analysis Checklist		6.7]

7.3.2.18 Path and Domain Selection

Description:

The construction of test data sets by partitioning the program input space into path domains (those subsets of the input space that cause execution of each path) and selecting data points from the domains that correspond to the paths of interest. Test data points are also selected on and near the boundaries of the domains to reveal path selection errors that may exist in the program under test. [DeMillo 1987]

Cross References:

Life Cycle Activities:
[Coding and CSU Testing 4.6.2]

Tools:
[Test Building Services 5.14.3]

Guidebook References:

7.3.3 Formal Verification

Description:

The use of rigorous mathematical techniques to prove the consistency between an algorithmic solution and a rigorous, complete specification of the intent of the solution.
[@Kean 1985]

Cross References:

Life Cycle Activities:
[Detailed Design
Coding and CSU Testing

4.5.2,
4.6.2]

Tools:
[Static Analyzers

5.14.1]

7.3.4 Problem Report Analysis

Description:

The analysis of problem reports for the purpose of determining the validity of the reported problem and a corrective action. [Kean 1985]

Cross References:

Life Cycle Activities:

[CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2,
Operational Testing and Evaluation	4.10.2]

Tools:

[Problem Report Analyzer	5.5.7,
Decision Support Services	5.14.7]

7.3.5 Change Request Analysis

Description:

The analysis of change requests to determine the necessity of the change, technical/economic impacts, and approach to accomplishing the change. [Kean 1985]

7.3.5.1 Change Impact Analysis

Description:

The ability to determine, for a proposed support/enhancement operation, the impact of proposed changes to the software system. For example, changes could be specified at the requirements, design, or code levels and, utilizing the traceability mechanisms which link elements of various life cycle activities, the impact(s) of the changes could be identified. Impact(s) would include those to requirements, design, code, test cases/test data, and associated documentation. [Kean 1985]

Cross References:

Life Cycle Activities:	
[Change Requirements	4.11.2]
Tools:	
[Change Impact Analyzer	5.5.8,
Decision Support Services	5.14.7]

Guidebook References:

APPENDIX A

CITATIONS

- [ACVC 1989] "Ada Compiler Validation Procedures, Version 2.0," Ada Joint Program Office, May 1989.
- [ARTEWG 1988] "A Framework for Describing Ada Runtime Environments," Proposed by Ada Runtime Environments Working Group (SIGAda), Ada Letters, Volume VIII, Number 3, May/June 1988, pp. 51-68.
- [Barstow 1981] D.R. Barstow, and H.E. Shrobe, "Observations on Interactive Programming Environments," [IEEE 1981], 286-301, 1981.
- [Boehm 1988] B.W. Boehm, "A Spiral Model of Software Development and Enhancement," Computer, Volume 21, Number 5, May 1988.
- [Buxton 1980] [DoD 1980].
- [CAIS] [DoD 1986].
- [Castor 1983] V.L. Castor, "Criteria for the Evaluation of ROLM Corporation's Ada Work Center," Wright-Patterson AFB, 10 January 1983.
- [DACS 1979] The DACS Glossary, A Bibliography of Software Engineering Terms, RADC/COED, Griffiss AFB, NY, October 1979, DTIC Number AD A127 689.
- [Dart 1987] S.A. Dart, R.J. Ellison, P.H. Feiler, and A.N. Habermann, "Software Development Environments," Software Engineering Institute, Technical Report CMU/SEI-87-TR-24, November 1987, DTIC Number AD A200 542.
- [DeMillo 1987] R.A. DeMillo, W.M. McCracken, R.J. Martin, and J.F. Passafiume, "Software Testing and Evaluation," Benjamin Cummings Publishing Company, 1987.
- [DoD-STD-2167A] Defense System Software Development, U.S. Department of Defense, 29 February 1988, DTIC Number AD A198 825.
- [DoD 1980] J.N. Buxton, "Requirements for Ada Programming Support Environments — STONEMAN," U.S. Department of Defense, February 1980, DTIC Number AD A100 404.
- [DoD 1983] ANSI/MIL-STD-1815A-1983, Reference Manual for the Ada Programming Language, U.S. Department of Defense, 17 February 1983, DTIC Number AD A131 511.
- [DoD 1986] "Military Standard Common Ada Programming Support Environment (APSE) Interface Set (CAIS)," U.S. Department of Defense, DoD-STD-1838, 9 October 1986, DTIC Number AD A157 589.

E&V Reference Manual, Version 2.0

- [E&V Classification] "E&V Classification Schema Report", AFWAL/AAAF, Wright-Patterson AFB, OH, Version 1.0, 15 June 1987.
- [E&V Guidebook] [@GB].
- [E&V Plan] [@E&V Report 1984: E&V Plan A.].
- [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, DTIC Number AD A153 609.
- [E&V Report 1987] Evaluation and Validation (E&V) Team Public Report, Volume III, Air Force Wright Aeronautical Laboratories, Wright-Patterson AFB, September 1987, DTIC Number AD A196 164.
- [E&V Requirements][@E&V Report 1984: E&V Requirements B.].
- [E&V Workshop 1984] V.L. Castor, et al., "Ada Programming Support Environment (APSE) Evaluation and Validation (E&V) Workshop Report," Institute for Defense Analysis, December 1984, DTIC Number AD A156 667.
- [GB] "E&V Guidebook," WRDC/AAAF, Wright-Patterson AFB, OH, Version 2.0, September 1989, DTIC Number pending.
- [GIT 1987] "USDR&E (T&E) Software Test and Evaluation Project: Software Test and Evaluation Manual, Volume II, Guidelines for Software Test and Evaluation in the Department of Defense," Georgia Institute of Technology, GIT-SERC-87/03, 25 February 1987, DTIC Number AD A189 204.
- [Gutz 1981] S. Gutz, A.I. Wasserman, and M.J. Spier, "Personal Development Systems for the Professional Programmer," Computer, April 1981.
- [Henderson 1987] P.B. Henderson, "Software Development/Programming Environments," ACM Software Engineering Notes, Volume 12, Number 1, January 1987.
- [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, December 1982, Issued February 1983.
- [Houghton 1987] R.C. Houghton, Jr., and D.R. Wallace, "Characteristics and Functions of Software Engineering Environments," ACM Software Engineering Notes, Volume 12, Number 1, January 1987.
- [IEEE 1981] Tutorial: Software Development Environments, ed. A.I. Wasserman, IEEE Catalog No. EHO 187-5, 1981.

E&V Reference Manual, Version 2.0

- [IEEE 1983] "IEEE Standard Glossary of Software Engineering Terminology," The Institute of Electrical and Electronics Engineers, Inc., ANSI/IEEE Std 729-1983, 18 February 1983.
- [IEEE Expert 1986] IEEE Expert, Winter 1986.
- [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, DTIC Number AD B096 355.
- [Lehman 1987] M.M. Lehman, and W.M. Turski, "Essential Properties of IPSEs," ACM Software Engineering Notes, Volume 12, Number 1, January 1987.
- [Lyons 1986] "Selecting an Ada Environment," eds. T.G.L. Lyons, and J.C.D. Nissen, Ada-Europe Environments Working Group, Cambridge University Press, 1986.
- [NBS Taxonomy] [@Houghton 1983].
- [McDermid 1984] J. McDermid, and K. Ripken, "Life Cycle Support in the Ada Environment," Cambridge University Press, 1984.
- [McDermid 1985] Integrated Project Support Environments, ed. J. McDermid, Peter Peregrinus (IEEE), 1985.
- [McKay 1987] C.W. McKay, "A Proposed Framework for the Tools and Rules to Support the Life Cycle of the Space Station Program," Proceedings of the IEEE Compass '87 Conference, June 1987.
- [Notkin 1981] D.S. Notkin, and A.N. Habermann, "Software Development Environment Issues as Related to Ada," [@IEEE 1981], 107-133, 1981.
- [PIWG 1987] Ada Slices, Official Newsletter of the ACM SIGAda User's Committee Performance Issues Working Group, "PIWG," 5 March 1987.
- [RADC 1985] T.P. Bowen, G.B. Wigle, and J.T. Tsai, "Specification of Software Quality Attributes Software Quality Specification Guidebook and Software Quality Evaluation Guidebook," Rome Air Development Center, Griffiss AFB, RADC-TR-85-37, Volumes II and III (of three), February 1985, DTIC Number AD A153 989.
- [SEE Taxonomy] [@Kean 1985].
- [STARS 1988] "STARS Consolidated Technical Development Plan," November 1988.
- [STONEMAN] [@DoD 1980].
- [Texas Instruments 1985] The APSE Interactive Monitor, Texas Instruments, Slide Presentation to the E&V Team, 5 September 1985.

E&V Reference Manual, Version 2.0

- [Wasserman 1981a] A.I. Wasserman, "The Ecology of Software Development Environments," [IEEE 1981].
- [Wasserman 1981b] A.I. Wasserman, "Toward Integrated Software Development Environments," [IEEE 1981].
- [Zuolkernan 1986] I. Zuolkernan, W.T. Tsai, and D. Volovik, "Expert Systems and Software Engineering: Ready for Marriage?," IEEE Expert, Winter, 1986.

APPENDIX B

GLOSSARY

B.1 ACRONYMS AND ABBREVIATIONS

This section provides a list of all acronyms and abbreviations used in this manual, with the associated meanings.

ANSI	American National Standards Institute
APSE	Ada Programming Support Environment
ARTEWG	Ada Runtime Environments Working Group (SIGAda)
CAIS	Common APSE Interface Set
CB7	Commerce Business Daily
CIDS	Critical Item Development Specification
CRISD	Computer Resources Integrated Support Document
CSC	Computer Software Component
CSCI	Computer Software Configuration Item
CSOM	Computer System Operator's Manual
CSU	Computer Software Unit
DACS	Data and Analysis Center for Software
DoD	Department of Defense
ECP	Engineering Change Proposal
E&V	Evaluation and Validation
FSM	Firmware Support Manual
GB	E&V Guidebook
HWCI	Hardware Configuration Item
IDD	Interface Design Document
IEEE	Institute of Electrical and Electronics Engineers
IPSE	Integrated Project Support Environment
IRS	Interface Requirements Specification
MCCS	Mission-Critical Computer System
PIDS	Prime Item Development Specification
PIWG	Performance Issues Working Group (SIGAda)
PSE	Programming Support Environment
RADC	Rome Air Development Center
RFP	Request for Proposal
RM	E&V Reference Manual
SCN	Specification Change Notice
SDD	Software Design Document
SDE	Software Development Environment
SDF	Software Development File

E&V Reference Manual, Version 2.0

SDP	Software Development Plan
SEE	Software Engineering Environment
SIGAda	Special Interest Group for Ada of the Association for Computing Machinery (ACM)
SOW	Statement of Work
SPM	Software Programmer's Manual
SPS	Software Product Specification
SRS	Software Requirements Specification
SSDD	System/Segment Design Document
SSS	System/Segment Specification
STARS	Software Technology for Adaptable, Reliable Systems
STD	Software Test Description
STP	Software Test Plan
STR	Software Test Report
SUM	Software User's Manual
T&E	Test and Evaluation
USDR&E	Under Secretary of Defense-Research and Engineering
VDD	Version Description Document
WBS	Work Breakdown Structure

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), all explicit references, the table of contents, the composite index, and the reference appendix. 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        ::= <text lacking special characters>
designator_list ::= designator [ "," designator ]*
```

designator ::= leads "." | leads ["." digits]*
leads ::= digits b caps
digits ::= ('0'-'9')
caps ::= ('A'-'Z')

C.2 INDEXES

The index format of the Reference System builds on the reference specifications above, and is derived from MIL-STD-1815A.

index ::= [entry]#
entry ::= topic [redirector] [reference_list]
topic ::= phrase ["(" designator_list ")"]
redirector ::= "[" "see" ["also"] ":" phrase_list "]"
phrase_list ::= phrase ["," phrase]*

C.3 FORMAL CHAPTERS

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

C.3.1 Chapter Components

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

descriptor ::= designator phrase [description]
description ::= "Description:" text
text ::= < prose text >

functions ::= "Functions:" reference_list

products ::= "Products:" reference-list

gb_references ::= "Guidebook:" reference_list

life_cycle_activities ::= "Life Cycle Activities:" reference_list

tools ::= "Tools:" reference_list

quality_factors ::= "Software Quality Factors:"
reference_list

acquisition_concern ::= "Acquisition Concern:"
reference_list

software_criteria ::= "Software-Oriented Criteria:"
reference_list

complementary_factors ::= "Complementary Software Quality Factors:"
reference_list

cooperating_criteria ::= "Cooperating Criteria:"
reference_list

conflicting_criteria ::= "Conflicting Criteria:"
reference_list

beneficial_factors ::= "Beneficial Quality Factors:"
reference_list

adverse_factors ::= "Adverse Quality Factors:"
reference_list

C.3.2 Chapter Entries

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

life_cycle_activity ::= descriptor management engineering testing evaluation
configuration transition

management	::= descriptor products functions
engineering	::= descriptor products functions
testing	::= descriptor products functions
evaluation	::= descriptor products functions
configuration	::= descriptor products functions
transition	::= descriptor products functions
apse	::= descriptor [toolset]*
toolset	::= descriptor [tool]*
tool	::= descriptor functions
attribute	::= descriptor [concern]* [factor]* [criterion]*
concern	::= descriptor quality_factors
factor	::= descriptor acquisition_concern software_criteria complementary_factors cooperating_criteria conflicting_criteria
criterion	::= descriptor quality_factors beneficial_factors adverse_factors gb_references
function	::= descriptor [life_cycle_activities tools] [gb references]

C.3.3 Formal Chapter Ordering

The formal portion of the RM is found in chapters four through seven. Each of the classes of chapter entries is found in a distinct chapter.

```
formal_chapters ::= [ life_cycle_activity ] *
                  [ apse ]
```

[attribute]*

[function]*

C.4 TABLE OF CONTENTS

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

table_of_contents ::= [chapter]* index

chapter ::= designator phrase designator_page

designator_page ::= designator "-" digits

index ::= "Index" digits

C.5 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]*

citations ::= key body "."

key ::= "[" text "]"

body ::= text b key

INDEX

Access	
[Access Management	7.2.37;
Document Accessibility	6.4.13,
System Accessibility	6.4.32]
Accuracy	6.4.1
Activities, Life Cycle	4.
Ada Programming Support Environment	
[see: APSE]	
Adaptation	6.3
[Adaptation Attributes (of Whole APSEs)]	3.3.3]
Address/Phone Book	5.6.3
Analysis	
[APSE Tool Categories	
Analysis and Reporting	5.5.9,
Target Code Generation and Analysis System	5.13;
Functions:	7.3,
Change Impact Analysis	7.3.5.1,
Change Request Analysis	7.3.5,
Coverage/Frequency Analysis	7.3.2.8,
Data Flow Analysis	7.3.1.3,
Data Reduction and Analysis	7.3.1.31,
Dynamic Analysis	7.3.2,
Functional Analysis	7.3.1.4,
Interface Analysis	7.3.1.5,
Invocation Analysis	7.3.1.19,
I/O Specification Analysis	7.3.1.29,
Maintainability Analysis	7.3.1.18,
Mutation Analysis	7.3.2.9,
Problem Report Analysis	7.3.4,
Reachability Analysis	7.3.1.16,
Real Time Analysis	7.3.2.17,
Reliability Analysis	7.3.2.16,
Reusability Analysis	7.3.1.14,
Risk Analysis	7.2.2.9,
Sizing Analysis	7.3.1.30,
Static Analysis	7.3.1,
Statistical Analysis	7.3.1.24,
Test Condition Analysis	7.3.1.8,
Testability Analysis	7.3.1.7,

Timing Analysis	7.3.2.14,
Traceability Analysis	7.3.1.6,
Type Analysis	7.3.1.27,
Units Analysis	7.3.1.28;
Test Analysis Services	5.14.6]
Anomaly Management	6.4.2
Application	
[Application Independence	6.4.3;
Applications (Code) Generator	5.11.1]
APSE	
[see also: Environment]	
[APSE Tool Categories	5.,
Distributed APSE Support	5.4;
Life Cycle Activities	4.;
Whole APSE Assessment Issues	3.,
Approaches to Whole-APSE E&V	3.4,
APSE Definitions and Alternate Names	3.1,
APSE Viewed as a Collection of Tools	3.2.1,
APSE Viewed as a Knowledge-Based Expert System	3.2.5,
APSE Viewed as a Methodology-Support System	3.2.2,
APSE Viewed as a Stable Framework	3.2.6,
APSE Viewed as a User-Oriented, Interactive System	3.2.4,
APSE Viewed as an Information Management System	3.2.3,
Kernel APSE (KAPSE)	3.2.1,
Key Attributes of Whole APSEs	3.3,
Minimal APSE (MAPSE)	3.2.1,
Views of an APSE	3.2]
Archive, Backup, and Retrieval System	5.1.2
Assemble	
[Assembler	5.12.4;
Assembling	7.1.6.6;
Host-Target System Cross-Assembler	5.13.1]
Assertion Checking, Executable	7.3.2.6
Assistance	
[see: On-Line Assistance]	
Attributes	6.
[Central Role of Attributes and Functions	2.3;
Key Attributes of Whole APSEs	3.3]
Auditing	7.3.1.22
Augmentability	6.4.4
Autonomy	6.4.5

E&V Reference Manual, Version 2.0

Backup, and Retrieval System, Archive,	5.1.2
Benchmarks and Test Suites	3.4.1
Bindings to Standard Interface Specification Implementations, Language	5.3.3
Body Stub Generation	7.1.6.11
Calculator	5.6.2
Calendar	5.6.6
Capacity	6.4.6
Change	
[see also: Product: Engineering Change Proposal, Specification Change Notice]	
[Change Impact Analysis	7.3.5.1;
Change Impact Analyzer	5.5.8;
Change Request Analysis	7.3.5;
Change Requirements	4.11]
Checklists and Questionnaires	3.4.2
CIDS	
[see: Product: Critical Item Development Specification]	
Classification Schema	2.1
Code	
[Applications (Code) Generator	5.11.1;
Coding and CSU Testing	4.6;
Compiler (Code Generator — Back End)	5.12.3;
Target Code Generation and Analysis System	5.13]
Command Language	
[Command Language Processing	7.2.3.1;
Command Language Interface	5.1.1]
Commonality (Data and Communication)	6.4.7
Communicate	
[Commonality (Data and Communication)	6.4.7,
Communication Effectiveness	6.4.8,
Operability (Communicativeness)	6.4.20;
Software and System Test Communications	7.1.6.15]
Comparison	7.3.1.1
Compile	
[Compilation	7.1.6.7;
Compilation System:	5.12,

E&V Reference Manual, Version 2.0

Compiler (Code Generator — Back End)	5.12.3
Compiler (Front End)	5.12.2]
Completeness	6.4.9
[Completeness Checking]	7.3.1.12]
Complexity Measurement	7.3.1.10
Component	
[APSE Tool Categories	5.,
Reusable Components Library	5.11.8,
Tool Support Components	5.3]
Compression	7.1.6.16
Computer	
[see also: Product: Computer Resources Integrated Support Document]	
[Computer Management System	5.1;
Computer System Management	7.2.3]
Concepts, System	4.1
Conferencing	
[see: Electronic]	
Configuration	
[Configuration Management	7.2.2.7;
Life Cycle Activities: Configuration Management	4.x.5;
Configuration Management System	5.7,
Configuration Control	5.7.2,
Configuration Identification	5.7.1,
Configuration Status Accounting	5.7.3;
Required Configuration	6.4.26]
Consistency	6.4.10
[Consistency Checking]	7.3.1.13]
Constraint Evaluation	7.3.2.7
Contention	
[see: Constraint]	
Conversion	7.1.6.8
Correctness	6.2.1
[Correctness Checking]	7.3.1.11]
Cost	6.4.11
[Cost Estimator	5.5.1;
Cost Estimation	7.2.2.1]
Coverage/Frequency Analysis	7.3.2.8

E&V Reference Manual, Version 2.0

CRISD	
[see: Product: Computer Resources Integrated Support Document]	
Cross Reference	7.3.1...
CSC Integration and Testing	4.7
CSCI Testing	4.8
CSOM	
[see: Product: Computer System Operator's Manual]	
CSU Testing, Coding, and	4.6
Data	
[Data and Error Logging	7.2.1.11;
Data Definition Language Processor	5.11.6;
Data Editing	7.1.1.2,
Data Flow Analysis	7.3.1.3;
Data Manipulation Language Processor	5.11.7,
Data Model/Dictionary	5.9.4;
Data Reduction and Analysis	7.3.1.31,
Database Management	7.2.1.1;
Database Manager	5.2.2,
Database Schema Generator	5.11.2;
Test Data Management	7.2.1.8]
Debug	
[Debugging	7.3.2.5;
Host-Based Target Code Symbolic Debugger	5.13.6]
Decision	
[Decision Aids	3.4.4;
Decision Support Services	5.14.7]
Decompilation	7.1.7.5
Decryption	7.1.6.19
Definition Language Processor	5.9.1
Design	6.2
[see also: Product: Interface Design Document,	
Software Design Document,	
System/Segment Design Document]	
[Design Attributes (of Whole APSEs)	3.3.2;
Design Generation	7.1.7.1;
Design Library Manager	5.9.8;
Design Prototyping	7.3.2.4;
Detailed Design	4.5;
Detailed Design Translation	7.1.6.5;
Preliminary Design	4.4;

E&V Reference Manual, Version 2.0

Preliminary Design Translation	7.1.6.4;
Requirements/Design Support	5.9;
System Requirements Analysis/Design	4.2]
Desktop System	5.6
Detailed Design	
[see: Design]	
Development	
[see also: Methodology]	
[Distributed System Development and Runtime Support	5.10;
Life Cycle Activities: System Development Management	4.x.1]
Dictionary	
[Data Model/Dictionary	5.9.4,
Dictionary/Thesaurus	5.6.7]
Disassembling	7.1.7.6
Display, Status	7.2.1.12
Distributed	
[Distributed APSE Support	5.4,
Distributed System Development and Runtime Support	5.10;
Distributedness	6.4.12]
Document	
[see also: Product]	
[Document Accessibility	6.4.13;
Document Generation System	5.8,
Document Manager	5.8.1;
Documentation Management	7.2.1.2]
Domain Selection, Path and	7.3.2.18
Downloader, Host-to-Target	5.13.7
Dynamic Analysis	7.3.2
E&V	
[Approaches to Whole-APSE E&V	3.4;
The Need for E&V Technology	1.3]
ECP	
[see: Product: Engineering Change Proposal]	
Edit	
[Editing:	7.1.1,
Data Editing	7.1.1.2,
Graphics Editing	7.1.1.3,
Text Editing	7.1.1.1;
Syntax-Directed (Language-Sensitive) Editor	5.11.5,

E&V Reference Manual, Version 2.0

Translating Editor	5.12.1,
Word Processor	5.4.2]
Efficiency	6.1.1
Electronic	
[APSE Tool Categories:	
Electronic Conferencing	5.6.5,
Electronic Mail	5.6.4;
Functions:	
Electronic Conferencing	7.2.1.5,
Electronic Mail	7.2.1.4]
Emulate	
[Emulation	7.3.2.13;
Host-Based Target System Instruction-Level Emulator	5.13.5]
Encryption	7.1.6.18
Engineering, System	4.x.2
Enterprise Model	5.9.3
Environment	
[Ada Programming Support Environment (APSE)	3.1,
Framing Environment	3.1,
General Environment	3.1,
Integrated Project Support Environment (IPSE)	3.1,
Language-Centered Environment	3.1,
Method-Based Environment	3.1,
Programming Environment	3.1,
Programming Support Environment (PSE)	3.1,
Software Development Environment (SDE)	3.1,
Software Engineering Environment (SEE)	3.1,
Structure-Oriented Environment	3.1,
Toolkit Environment	3.1]
Error	
[Data and Error Logging	7.2.1.11,
Error Checking	7.3.1.23;
Error Tolerance (Anomaly Management)	6.4.2]
Estimate	
[Cost Estimation	7.2.2.1;
Cost Estimator	5.5.1;
Resource Estimation	7.2.2.5;
Resource Estimator	5.5.5,
Size Estimator	5.5.2]
Evaluation	
[see also: E&V]	
[Constraint Evaluation	7.3.2.7,

E&V Reference Manual, Version 2.0

Evaluation Results Management	7.2.1.9;
Life Cycle Activities: System Product Evaluations	4.x.4,
Operational Testing And Evaluation	4.10]
Execute	
[Executable Assertion Checking	7.3.2.6;
Processing (Execution) Effectiveness	6.4.22;
Symbolic Execution	7.3.2.10;
Test Execution Services	5.14.5]
Expand	
[Expandability	6.3.1;
Expansion	7.1.6.17,
Macro Expansion	7.1.6.9]
Experiments, Structured	3.4.3
Expert System, APSE Viewed as a Knowledge-Based	3.2.5
Export System, Import/	5.1.6
[see: Import/Export]	
Fault Tolerance	6.4.2
File	
[File Management	7.2.1.3;
File Manager	5.2.1]
Flexibility	6.3.1
Forms, Predefined and User-Defined	7.1.2.3
Formal	
[Formal Verification	7.3.3;
Life Cycle Activities: Formal Qualification Testing	4.x.3]
Format	
[Formatter	5.8.5;
Formatting:	7.1.2,
MIL-STD Format	7.1.2.1,
Table of Contents	7.1.2.2]
Framework, APSE Viewed as a Stable	3.2.6
Frequency Analysis, Coverage/	7.3.2.8
FSM	
[see: Product: Firmware Support Manual]	
Function	
[Functional Analysis	7.3.1.4;
Functional Overlap	6.4.14,
Functional Scope	6.4.15;
Functions	7.]

E&V Reference Manual, Version 2.0

Generality	6.4.16
Generate	
[Body Stub Generation	
Design Generation	7.1.6.11,
Document Generation System	7.1.7.1;
Graphics Generation	5.8;
Graphics Generator	7.1.5;
Preamble Generation	5.8.4;
Program Generation	7.1.6.12,
Random Test Generation	7.1.7.3,
Target Code Generation and Analysis System	7.3.1.32;
Test Harness Generation	5.13;
	7.1.7.7]
Global	4.12
Granularity	6.4.17
Graphics	
[Graphics Editing	
Graphics Generation	7.1.1.3,
Graphics Generator	7.1.5;
	5.8.4]
Harness Generation, Test	7.1.7.7
Help	
[see: On-Line Assistance]	
History	5.7.5
I/O	
[I/O Pipes	
I/O Specification Analysis	5.3.4;
Input and Output Services	7.3.1.29;
Input/Output Support	5.1.8;
	7.2.3.2]
IDD	
[see: Product: Interface Design Document]	
Impact Analysis, Change	7.3.5.1
Implementation Support	5.11
Import/Export	
[Import/Export System	7.2.3.6
	5.1.6]
Input	
[see: I/O]	
Index	
[Index Relationships	
Using the Reference Indexes Directly	2.4;
	2.3.1]

Information Management

[APSE Viewed as an Information Management System 3.2.3;
 Functions: Information Management 7.2.1;
 Information Management System 5.2]

Integration

[CSC integration And Testing 4.7,
 System Integration And Testing 4.9]

Integrity

6.1.2

Interactive System, APSE Viewed as a User-Oriented

3.2.4

Interface

[see also: Product: Interface Design Document,
 Interface Requirements Specification]
 [Command Language Interface 5.1.1;
 Interface Analysis 7.3.1.5;
 Language Bindings to Standard Interface
 Specification Implementations 5.3.3;
 Layered Portable Interfaces 3.2.6]

Interoperability

6.3.2

Interpret

[Interpretation 7.1.6.14;
 Interpreter 5.12.5]

Invocation Analysis

7.3.1.19

IPSE

[see also: APSE, Environment]
 [Life Cycle Activities 4.]

IRS

[see: Product: Interface Requirements Specification]

Job

[Job Scheduler 5.1.4;
 Job Scheduling 7.2.3.8]

Kernel

7.2.3.3

Language

[see also: Command Language]
 [Data Definition Language Processor 5.11.6,
 Data Manipulation Language Processor 5.11.7,
 Definition Language Processor 5.9.1;
 Requirements to Natural Language Translation 7.1.6.3;
 Specification Language Processor 5.9.2,
 Syntax-Directed (Language-Sensitive) Editor 5.11.5]

Library

[Design Library Manager 5.9.8;
 Program Library Management 7.2.1.7;

E&V Reference Manual, Version 2.0

Program Library Manager	5.12.6,
Reusable Components Library	5.11.8]
Life Cycle Activities	4.
Link	
[Host-Based Target Linker	5.13.2;
Linking/Loading	7.1.6.13]
Load	
[Host-Based Target Loader	5.13.3;
Linking/Loading	7.1.6.13]
Logging, Data and Error	7.2.1.1
Macro Expansion	7.1.6.9
Mail	
[see: Electronic]	
Maintainability	6.2.2
[Maintainability Analysis	7.3.1.18]
Management	
[APSE Tool Categories:	
Computer Management System	5.1,
Configuration Management System	5.7,
Information Management System	5.2,
Project Management System	5.5;
APSE Viewed as an Information Management System	3.2.3;
Attributes:	
Anomaly Management	6.4.2;
Functions:	7.2,
Access Management	7.2.3.7,
Computer System Management	7.2.3,
Configuration Management	7.2.2.7,
Database Management	7.2.1.1,
Documentation Management	7.2.1.2,
Evaluation Results Management	7.2.1.9,
File Management	7.2.1.3,
Information Management	7.2.1,
Program Library Management	7.2.1.7,
Project Management	7.2.2,
Resource Management	7.2.3.9,
Specification Management	7.2.1.6,
Test Data Management	7.2.1.8;
Life Cycle Activities:	
Configuration Management	4.x.5,
Software Development Management	4.x.1]
Math/Statistics	7.2.3.4

E&V Reference Manual, Version 2.0

Maturity	6.4.18
Measurement	
[Complexity Measurement	7.3.1.10,
Quality Measurement	7.3.1.9]
Merge, Sort/	7.1.4
Methodology-Support System, APSE Viewed as a	3.2.2
Model	
[Data Model/Dictionary	5.9.4,
Enterprise Model	5.9.3,
Process Model	5.9.5;
Simulation And Modeling	7.3.2.3]
Modularity	6.4.19
Monitor	
[Performance Monitor	5.1.9;
Performance Monitoring	7.2.1.10]
Mutation Analysis	7.3.2.9
Natural Language Translation, Requirements to	7.1.6.3
On-Line Assistance	
[On-Line Assistance Processing	7.1.3;
On-Line Assistance	5.1.7]
Operability	6.4.20
Operating System, Virtual	5.3.1
Operational Testing and Evaluation	4.10
Output	
[see: I/O]	
Path and Domain Selection	7.3.2.18
Performance	6.1
[Performance Attributes (of Whole APSEs)	3.3.1;
Performance Monitor	5.1.9;
Performance Monitoring	7.2.1.10]
Phone Book, Address/	5.6.3
PIDS	
[see: Product: Prime Item Development Specification]	
Power	6.4.21
Preamble Generation	7.1.6.12

Preliminary Design

[see: Design]

Problem Report

[Problem Report Analysis

7.3.4;

Problem Report Analyzer

5.5.7]

Process

[Process Model

5.9.5;

Processing (Execution) Effectiveness

6.4.22]

Product

[Computer Resources Integrated Support Document (CRISD):

Detailed Design

4.5.6,

Preliminary Design

4.4.6,

Software Requirements Analysis

4.3.6,

System Requirements Analysis/Design

4.2.6;

Computer System Operator's Manual (CSOM):

Coding and CSU Testing

4.6.6,

CSC Integration and Testing

4.7.6,

CSCI Testing

4.8.6,

Detailed Design

4.5.6,

Preliminary Design

4.4.6;

Critical Item Development Specification (CIDS):

System Requirements Analysis/Design

4.2.2;

Engineering Change Proposal (ECP):

Change Requirements

4.11.2;

Firmware Support Manual (FSM):

Coding and CSU Testing

4.6.6,

CSC Integration and Testing

4.7.6,

CSCI Testing

4.8.6,

Detailed Design

4.5.6;

Interface Design Document (IDD):

CSCI Testing

4.8.2,

Detailed Design

4.5.2,

Preliminary Design

4.4.2;

Interface Requirements Specification (IRS):

Software Requirements Analysis

4.3.2,

System Requirements Analysis/Design

4.2.2;

Life Cycle Activities: System Product Evaluations

4.x.4;

Prime Item Development Specification (PIDS):

System Requirements Analysis/Design

4.2.2;

Software Design Document (SDD):

CSCI Testing

4.8.2,

Detailed Design

4.5.2,

Preliminary Design

4.4.2;

E&V Reference Manual, Version 2.0

Software Development File (SDF):	
Coding and CSU Testing	4.6.2,
CSC Integration and Testing	4.7.2,
CSCI Testing	4.8.2,
Detailed Design	4.5.2;
Software Development Plan (SDP):	
Coding and CSU Testing	4.6.1,
CSC Integration and Testing	4.7.1,
CSCI Testing	4.8.1,
Detailed Design	4.5.1,
Preliminary Design	4.4.1,
Software Requirements Analysis	4.3.1,
System Requirements Analysis/Design	4.2.1;
Software Product Specification (SPS):	
CSCI Testing	4.8.2,
System Integration and Testing	4.9.2;
Software Programmer's Manual (SPM):	
Coding and CSU Testing	4.6.6,
CSC Integration and Testing	4.7.6,
CSCI Testing	4.8.6,
Detailed Design	4.5.6;
Software Requirements Specification (SRS):	
Software Requirements Analysis	4.3.2,
System Requirements Analysis/Design	4.2.2;
Software Test Description (STD):	
Coding and CSU Testing	4.6.3,
CSC Integration and Testing	4.7.3,
CSCI Testing	4.8.3,
Detailed Design	4.5.3;
Software Test Plan (STP):	
Preliminary Design	4.4.3;
Software Test Report (STR):	
CSCI Testing	4.8.3;
Software User's Manual (SUM):	
Coding and CSU Testing	4.6.6,
CSC Integration and Testing	4.7.6,
CSCI Testing	4.8.6,
Detailed Design	4.5.6,
Preliminary Design	4.4.6;
Specification Change Notice (SCN):	
Change Requirements	4.11.5;
System/Segment Design Document (SSDD):	
System Requirements Analysis/Design	4.2.2;
System/Segment Specification (SSS):	
System Requirements Analysis/Design	4.2.2;

E&V Reference Manual, Version 2.0

Version Description Document (VDD):	
CSCI Testing	4.8.5,
System Integration and Testing	4.9.5]
Program	
[see also: Product: Software Programmer's Manual]	
[Program Generation	7.1.7.3,
Program Library Management	7.2.1.7;
Program Library Manager	5.12.6]
Project Management	7.2.2
[Project Management System	5.5]
Proprietary Rights	6.4.23
Prototype	
[Design Prototyping	7.3.2.4;
Prototyping Tools	5.9.7;
Requirements Prototyping	7.3.2.2]
Qualification Testing, Formal	4.x.3
Quality	
[Quality Measurement	7.3.1.9;
Quality Analyzer	5.2.2;
Quality Assessment	7.2.2.8,
Quality Specification	7.2.2.2]
Questionnaires, Checklists and	3.4.2
RAM Cache	5.3.5
Random Test Generation	7.3.1.32
Reachability Analysis	7.3.1.16
Real Time Analysis	7.3.2.17
Reconfigurability	6.4.24
Reduction and Analysis, Data	7.3.1.31
Reference	
[Cross Reference	7.3.1.17;
Structure and Use of the Reference Manual	2.]
Regression Testing	7.3.2.11
Rehostability	6.4.25
Reliability	6.1.3
[Reliability Analysis	7.3.2.16]
Report	
[see also: Product: Software Test Report]	
[Analysis and Reporting	5.5.9;

E&V Reference Manual, Version 2.0

Change Report Analysis	7.3.5,
Problem Report Analysis	7.3.4;
Report Generator	5.11.3]
Requirements	
[see also: Product: Interface Requirements Specification, Software Requirements Specification]	
[Change Requirements	4.11;
Required Configuration	6.4.26;
Requirements/Design Support	5.9;
Requirements Prototyping	7.3.2.2.
Requirements Reconstruction	7.1.7.2,
Requirements Simulation	7.3.2.1,
Requirements to Natural Language Translation	7.1.6.3;
Software Requirements Analysis	4.3;
Software Requirements Translation	7.1.6.1;
System Requirements Analysis/Design	4.2;
System Requirements Translation	7.1.6.2]
Resource	
[see also: Product: Computer Resources Integrated Support Document]	
[Resource Controller	5.1.5;
Resource Estimation	7.2.2.5;
Resource Estimator	5.5.5;
Resource Management	7.2.3.9,
Resource Utilization	7.3.2.12]
Retargetability	6.4.27
Retrieval System, Archive, Backup, and	5.1.2
Reusable	
[Reusable Components Library	5.11.8;
Reusability	6.3.3;
Reusability Analysis	7.3.1.14]
Risk Analysis	7.2.2.9
Robustness	6.4.2
Runtime	
[Distributed System Development and Runtime Support	5.10;
Runtime Environment	7.2.3.5;
Runtime System	5.12.7]
Scanning	7.3.1.20
Schedule	
[Job Scheduler	5.1.4;
Job Scheduling	7.2.3.8;
Scheduler	5.5.3;
Scheduling	7.2.2.3]

Schema Generator, Database	5.11.2
SCN [see: Product: Specification Change Notice]	
Screen Generator	5.11.4
SDD [see: Product: Software Design Document]	
SDF [see: Product: Software Development File]	
SDP [see: Product: Software Development Plan]	
Security System	5.1.3
Selection, Path and Domain	7.3.2.18
Self-Descriptiveness	6.4.28
Simplicity	6.4.29
Simulate [Host-Based Target System Instruction-Level Simulator Requirements Simulation Simulation and Modeling Simulators]	5.13.4; 7.3.2.1, 7.3.2.3; 5.9.6]
Size [Size Estimator Sizing Analysis]	5.5.2; 7.3.1.30]
Software [Software and System Test Communications Software-Oriented Criteria: Software Production Vehicle(s) Software Requirements Analysis Software Requirements Translation]	7.1.6.15; 6.4, 6.4.30; 4.3; 7.1.6.2]
Sort/Merge	7.1.4
Source Reconstruction	7.1.7.4
Specification [see also : Product: Critical Item Development Specification, Interface Requirements Specification, Prime Item Development Specification, Software Product Specification, Software Requirements Specification, System/Segment Specification]	
[I/O Specification Analysis]	7.3.1.29;

E&V Reference Manual, Version 2.0

Language Bindings to Standard Interface	
Specification Implementations	5.3.3;
Quality Specification	7.2.2.2;
Specification Language Processor	5.9.2;
Specification Management	7.2.1.6]
Spelling	
[Spell Checker	5.8.3;
Spelling Checking	7.3.1.2]
SPM	
[see: Product: Software Programmer's Manual]	
Spreadsheet	5.6.1
SPS	
[see: Product: Software Product Specification]	
SRS	
[see: Product: Software Requirements Specification]	
SSDD	
[see: Product: System/Segment Design Document]	
SSS	
[see: Product: System/Segment Specification]	
Static	
[Static Analysis	7.3.1;
Static Analyzers	5.14.1]
Statistic	
[Math/Statistics	7.2.3.4,
Statistical Analysis	7.1.3.24,
Statistical Profiling	7.1.3.25]
Status Display	7.2.1.12
STD	
[see: Product: Software Test Description]	
Storage Effectiveness	6.4.31
STP	
[see: Product: Software Test Plan]	
STR	
[see: Product: Software Test Report]	
Structure	
[Object Structure Layer	3.2.6;
Structure and Use of the Reference Manual	2.;
Structure Checking	7.3.1.26,
Structure Preprocessing	7.1.6.10]

E&V Reference Manual, Version 2.0

SUM

[see: Product: Software User's Manual]

Support

[see also: Product: Computer Resources Integrated Support Document]

[APSE Tool Categories]

Distributed APSE Support	5.4,
Distributed System Development and Runtime Support	5.10,
Implementation Support	5.11,
Requirements/Design Support	5.9,
Tool Support Components	5.3;
Life Cycle Activities: Transitioning to System Support	4.x.6]

Survivability

6.1.4

Symbolic Execution

7.3.2.10

Syntax

[Syntax-Directed (Language-Sensitive) Editor	5.11.5;
Syntax And Semantics Checking	7.3.1.15]

Synthesis

7.1.7

System

[see also: Product: Computer System Operator's Manual,
System/Segment Design Document,
System/Segment Specification]

[Distributed System Development and Runtime Support	5.10;
Life Cycle Activities:	
System Concepts	4.1,
System Development Management	4.x.1,
System Engineering	4.x.2,
System Integration and Testing	4.9,
System Product Evaluations	4.x.4,
System Requirements Analysis/Design	4.2,
Transitioning To System Support	4.x.6;
Software and System Test Communications	7.1.6.15;
System Accessibility	6.4.32,
System Clarity	6.4.33,
System Compatibility	6.4.34;
System Requirements Translation	7.1.6.1]

Table of Contents Formatting

7.1.2.2

Target

[Target Code Generation and Analysis System	5.13,
Host-Based Target Code Symbolic Debugger	5.13.6,
Host-Based Target Linker	5.13.2,
Host-Based Target Loader	5.13.3,
Host-Based Target System Instruction-Level Emulator	5.13.5,

E&V Reference Manual, Version 2.0

Host-Based Target System Instruction-Level Simulator	5.13.4,
Host-Target System Cross-Assembler	5.13.1,
Host-to-Target Downloader	5.13.7,
Target-to-Host Uploader	5.13.8]
Test	
[see also: Product: Software Test Description, Software Test Plan, Software Test Procedure, Software Test Report]	
[Life Cycle Activities:	
Coding and CSU Testing	4.6,
CSC Integration and Testing	4.7,
CSCI Testing	4.8,
Formal Qualification Testing	4.x.3,
Operational Testing and Evaluation	4.10,
System Integration and Testing	4.9;
Random Test Generation	7.3.1.32,
Regression Testing	7.3.2.11,
Software and System Test Communications	7.1.6.15;
Test Analysis Services	5.14.6,
Test Building Services	5.14.3;
Test Condition Analysis	7.3.1.8,
Test Data Management	7.2.1.8;
Test Description and Preparation Services	5.14.4,
Test Execution Services	5.14.5;
Test Harness Generation	7.1.7.7;
Test Suites, Benchmarks and	3.4.1;
Test System	5.14;
Testability (Verifiability)	6.2.3;
Testability Analysis	7.3.1.7]
Text Editing	7.1.1.1
Thesaurus, Dictionary/	5.6.7
Timing Analysis	7.3.2.14
Tool	
[APSE Tool Categories	5.,
Tool Building Services	5.14.2,
Tool Support Components	5.3;
APSE Viewed as a Collection of Tools	3.2.1]
Traceability Analysis	7.3.1.6
Tracking	
[Function	7.2.2.6;
Tool	5.5.6]

E&V Reference Manual, Version 2.0

Training	6.4.36
Transitioning to System Support	4.x.6
Translate	
[Translating Editor	5.12.1;
Translation	7.1.6]
Transportability	6.3.4
Tuning	7.3.2.15
Type Analysis	7.3.1.27
Unit	
[Coding and CSU Testing	4.6;
Units Analysis	7.3.1.28]
Uploader, Target-to-Host	5.13.8
Usability	6.1.5
Validation	
[see: E&V]	
VDD	
[see: Product: Version Description Document]	
Verifiability	6.2.3
Verification, Formal	7.3.3
Version Control	5.7.4
Virtual	
[Virtual Layer	3.2.6;
Virtual Operating System	5.3.1;
Virtuality	6.4.37]
Visibility	6.4.38
Walkthrough, Structured	7.3.1.21
Whole APSE	
[see: APSE]	
Window Manager	5.3.2
Word Processor	5.8.2
Work Breakdown Structure	
[Function	7.2.2.4;
Work Breakdown Structure Editor	5.5.4]