

IEEE ABBET™

IEEE Trial-Use Standard for A Broad-Based Environment for Test (ABBET)

Overview and Architecture

Sponsor

IEEE Standards Coordinating Committee 20
Abbreviated Test Language for All Systems (ATLAS)

Approved 8 December 1998

IEEE-SA Standards Board

Abstract: The overall concept of A Broad-Based Environment for Test (ABBET)™ is defined, and mandatory requirements for implementation of ABBET are specified. The elements of ABBET and their interrelationships are described. Guidelines and requirements governing the elements of the ABBET set of standards and guides are established, and common terms to be used throughout the set are defined.

Keywords: ABBET™, product test, diagnostics, software framework

The Institute of Electrical and Electronics Engineers, Inc.
345 East 47th Street, New York, NY 10017-2394, USA

Copyright © 1999 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 24 June 1999. Printed in the United States of America.

Print: ISBN 0-7381-1545-2 SH94721
PDF: ISBN 0-7381-1546-0 SS94721

No part of this publication may be reproduced in any form, in an electronic retrieval system or otherwise, without the prior written permission of the publisher.

IEEE Standards documents are developed within the IEEE Societies and the Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. Members of the committees serve voluntarily and without compensation. They are not necessarily members of the Institute. The standards developed within IEEE represent a consensus of the broad expertise on the subject within the Institute as well as those activities outside of IEEE that have expressed an interest in participating in the development of the standard.

Use of an IEEE Standard is wholly voluntary. The existence of an IEEE Standard does not imply that there are no other ways to produce, test, measure, purchase, market, or provide other goods and services related to the scope of the IEEE Standard. Furthermore, the viewpoint expressed at the time a standard is approved and issued is subject to change brought about through developments in the state of the art and comments received from users of the standard. Every IEEE Standard is subjected to review at least every five years for revision or reaffirmation. When a document is more than five years old and has not been reaffirmed, it is reasonable to conclude that its contents, although still of some value, do not wholly reflect the present state of the art. Users are cautioned to check to determine that they have the latest edition of any IEEE Standard.

Comments for revision of IEEE Standards are welcome from any interested party, regardless of membership affiliation with IEEE. Suggestions for changes in documents should be in the form of a proposed change of text, together with appropriate supporting comments.

Interpretations: Occasionally questions may arise regarding the meaning of portions of standards as they relate to specific applications. When the need for interpretations is brought to the attention of IEEE, the Institute will initiate action to prepare appropriate responses. Since IEEE Standards represent a consensus of all concerned interests, it is important to ensure that any interpretation has also received the concurrence of a balance of interests. For this reason, IEEE and the members of its societies and Standards Coordinating Committees are not able to provide an instant response to interpretation requests except in those cases where the matter has previously received formal consideration.

Comments on standards and requests for interpretations should be addressed to:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
P.O. Box 1331
Piscataway, NJ 08855-1331
USA

Note: Attention is called to the possibility that implementation of this standard may require use of subject matter covered by patent rights. By publication of this standard, no position is taken with respect to the existence or validity of any patent rights in connection therewith. The IEEE shall not be responsible for identifying patents for which a license may be required by an IEEE standard or for conducting inquiries into the legal validity or scope of those patents that are brought to its attention.

Authorization to photocopy portions of any individual standard for internal or personal use is granted by the Institute of Electrical and Electronics Engineers, Inc., provided that the appropriate fee is paid to Copyright Clearance Center. To arrange for payment of licensing fee, please contact Copyright Clearance Center, Customer Service, 222 Rosewood Drive, Danvers, MA 01923 USA; (978) 750-8400. Permission to photocopy portions of any individual standard for educational classroom use can also be obtained through the Copyright Clearance Center.

Introduction

[This introduction is not part of IEEE Std 1226-1998, IEEE Trial-Use Standard for A Broad-Based Environment for Test (ABBET), Overview and Architecture.]

The benefits of using standards in test-related applications have long been recognized. The scope for standardization extends from low-level standards associated with test instrument control to high-level standards associated with specifying tests in an implementation-independent manner.

In the 1960s, Aeronautical Radio, Incorporated (ARINC) started the development of the Abbreviated Test Language for Avionics Systems (ATLAS). In 1976, management of the ATLAS standard was passed to the IEEE, and the ATLAS acronym was changed to Abbreviated Test Language for All Systems to reflect its broader field of applications.

Within the IEEE, development of ATLAS and ATLAS-related standards was vested in an ad hoc committee, which later became the IEEE Standards Coordinating Committee 20 (SCC20). In the mid-1980s, SCC20 broadened the scope of its activities to embrace other standards projects related to test and diagnosis, including Automatic Test Program Generation (ATPG), Test Equipment Description Language (TEDL), Artificial Intelligence Exchange and Service Tie to All Test Environments (AI-ESTATE), and A Broad Based Environment for Test (ABBET).

This standard provides the framework for a family of standards providing software interface specifications for test-related applications and environments. This family incorporates object-oriented technology and information modeling to specify programming language-independent interfaces between elements within a wide variety of test environments, including built-in test systems (BIT), automatic test systems (ATS), and manual test environments. These interfaces are specified using the ISO 10303-11:1994 EXPRESS information modeling language and the Object Management Group (OMG) Interface Definition Language (IDL).

This standard defines the ABBET architecture primarily through the definition of a Core Test Information Model (CTIM) and a test foundation framework (TFF). The CTIM defines the basic information entities required in any test application and provides a vehicle for formally modeling behavior within a test environment. The TFF defines a class hierarchy corresponding to these basic information entities and provides several methods within each class to enable basic operations to be performed on these entities. Component standards within the ABBET architecture define subclasses from the TFF specific to particular requirements within the test environment.

At the time this standard was completed, the ABBET Subcommittee had the following membership.

Gary L. Hardenburg, Chair

Gayle Matysek, Secretary

Kirk Thompson, Architecture Working Group Chair

Guy Adam	William Brit Frank	Chuck Oliverio
James D. Baker	Joe Gile	Leslie A. Orlidge
Brad Balfour	Roger Goldberg	Dave Pawson
Jean-Yves Barbier	Jerry Grasso	Edward G. Perkins
Darrel Bartz	Arnold M. Greenspan	Jean Pouilly
Anne Bautista	Greg Griffes	John Powell
Aldo Bernal	John Halper	William Pritchett
Michael Blair	Arthur Hann	Jeffrey Rajhel
Stephen M. Bowen	Robert G. Hayes	Narayanan Ramachanran
Greg Bowman	Leonard Haynes	William E. Reeves
Alan L. Bridges	Bob Hillman	John Riley
David M. Brooks	Ashley M. B. Hulme	William Rodriguez
Ron Brooks	Ulla Jacobsson	David Roggendorff
Dale Brown	Andrew M. Joskowski	Robert Rolfe
Theodor Calin	Frank X. Kearns	Eric Sacher
Larry Carpenter	David Kidd	Eric Saenz
E. Elaine Cashar	Lars-Olov Larsson	Howard Savage
Richard D. Churcher	Hans Lecluyse	Michael Scalise
George Clarius	Robert Leif	Helmut Scheibenzuber
Christopher Connelly	James L. Lewis	Rickey Schippang
James Cottrell	Terry Lindemann	Michael L. Seavey
Bob Cram	Alfred Lowenstein	John W. Sheppard
Kent Cross	Richard J. Maguire	Lee A. Shombert
Robert J. Curry	Michael Malesich	Randy Simpson
David Dahlbacka	Michael Martin	Philip T. Smith
Bernard Dathy	Harry McGuckin	Butch Sneade
Danny C. Davis	Doug McNabb	Joseph J. Stanco
Jake Dawson	Martin Meth	Joseph L. Stautberg
Jeff Dean	James M. Mills	Richard Tarzaiki
Somnath Deb	James Moore	Jack Taylor
Gerald DePasquale	Charlie Morris	J. Richard Weger
Bernd Dinklage	David S. Morris	Daniel H. Weiss
Keith Ellis	William P. Neblett	William A. Whitaker
Michael T. Ellis	Tom Newton	David Williamson
Horst Ey	David L. Nichols	Will W. Young
Rob Fletcher	Nick Nicholson	Tak Yuen
Stephen Fortier	Hans Obermeier	Jehuda Ziegler
Ken Fox		Dan Zimmermann

In addition, the following persons made significant contributions to this standard:

Maria Bostic	Chris Gorringer	Roy Oishi
Terry Campbell	T. Winston Griffin	Catherine Ozenfant
Dez Cass	James Hanna	Richard Patrick
Jack Cross	Gary L. Hinkle	James Reeder
Christine Fisher	Glenn Laflin	William Russel
Al Gilman	John Leuthe	David Smith
John Goddard	Peter Lord	Mark Thullen
Sharon Goodall	John Odak	Roger L. Williams

The following members of the balloting committee voted on this standard:

Guy Adam
James D. Baker
Jean-Yves Barbier
Jeffrey S. Barkley
Aldo Bernal
Stephen M. Bowen
Greg Bowman
Alan L. Bridges
David M. Brooks
E. Elaine Cashar
Richard D. Churcher
William Colmer
Robert J. Curry
David Dahlbacka
Bernard Dathy
Danny C. Davis
James Allen Dawson
Somnath Deb
Bernd Dinklage
Michael T. Ellis
Anthony Fitzpatrick
William Brit Frank
Arnold M. Greenspan
Gary L. Hardenburg
Vance Harwood
Raymond L. Hasenstab

Robert G. Hayes
Mike Heilman
John E. Heiser
Wesley Higaki
Gary L. Hinkle
Catherine Hogans
Ashley M. B. Hulme
Gary M. Jenkins
Andrew M. Joskowski
Frank X. Kearns
Paul B. Knese
James L. Lewis
Alfred Lowenstein
Richard J. Maguire
Michael A. Malesich
Gayle Matysek
Robert L. McGarvey
James M. Mills
Mukund U. Modi
David S. Morris
William P. Neblett
Richard E. Neese
David L. Nichols
Hans Obermeier
Roy Oishi
Leslie A. Orlidge
Edward G. Perkins

Dennis C. Peterson
Alain Plassart
Jeffrey Rajhel
William E. Reeves
William Rodriguez
David Roggendorff
David E. Rolince
Eric Sacher
Helmut Scheibenzuber
Warren M. Scheinin
Michael L. Seavey
John W. Sheppard
Lee A. Shombert
William R. Simpson
Philip T. Smith
Joseph J. Stanco
Robert Stasonis
Joseph L. Stautberg
Kirk D. Thompson
Ronald Waxman
J. Richard Weger
Daniel H. Weiss
Lynn Wheelwright
William A. Whitaker
Roger L. Williams
Will W. Young

When the IEEE-SA Standards Board approved this standard on 8 December 1998, it had the following membership:

Richard J. Holleman, Chair

Donald N. Heirman, Vice Chair

Judith Gorman, Secretary

Satish K. Aggarwal
Clyde R. Camp
James T. Carlo
Gary R. Engmann
Harold E. Epstein
Jay Forster*
Thomas F. Garrity
Ruben D. Garzon

James H. Gurney
Jim D. Isaak
Lowell G. Johnson
Robert Kennelly
E. G. "Al" Kiener
Joseph L. Koepfinger*
Stephen R. Lambert
Jim Logothetis
Donald C. Loughry

L. Bruce McClung
Louis-François Pau
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Hans E. Weinrich
Donald W. Zipse

*Member Emeritus

Kim Breitfelder
IEEE Standards Project Editor

Contents

1.	Overview.....	1
1.1	Scope.....	1
1.2	Purpose.....	1
1.3	Application.....	2
2.	References.....	2
3.	Definitions, acronyms, and abbreviations.....	3
3.1	Definitions.....	3
3.2	Acronyms.....	8
4.	ABBET overview.....	8
4.1	Background.....	8
4.2	Purpose.....	9
4.3	ABBET and the product life cycle.....	9
4.4	ABBET framework.....	14
4.5	Specification techniques	15
4.6	ABBET component standards.....	17
5.	ABBET architecture.....	17
5.1	Scope.....	17
5.2	Fundamental concepts.....	18
5.3	Architecture levels	19
5.4	Relation to other ABBET standards	21
6.	Conformance.....	22
6.1	Interfaces.....	22
6.2	Implementation	23
7.	CTIM.....	23
7.1	CTIM scope	23
7.2	Overview	24
7.3	CTIM outline	25
7.4	Applying the CTIM.....	25
7.5	CTIM and the ABBET domain.....	27
7.6	Relationship to the TFF	28
7.7	EXPRESS source	28
8.	TFF.....	29
8.1	Overview	29
8.2	TFF structure.....	30
8.3	TFF architecture.....	32
8.4	TestObject management.....	36

Annex A (normative) TFF interface definition file listings	38
Annex B (normative) CTIM schema.....	41
Annex C (informative) Bibliography	50

IEEE ABBET™

IEEE Trial-Use Standard for A Broad-Based Environment for Test (ABBET)

Overview and Architecture

1. Overview

The set of A Broad-Based Environment for Test (ABBET) standards and guides was developed by the ABBET Subcommittee of the IEEE Standards Coordinating Committee 20 (SCC20) to serve as standards for product test. ABBET specifies a comprehensive environment for integrating design data, test strategies and requirements, test procedures, test results management, and test system implementations. The ABBET standards consist of this base standard and a series of component standards. A number of guides are associated with the ABBET standards.

1.1 Scope

This standard defines the overall concept of ABBET and specifies mandatory requirements for its implementation. This standard is the base document for the set of standards and guides that together comprise ABBET, and, as such, spans its complete capability.

The scope includes various test-related activities over a product's total life cycle and the key information that is produced and used by these activities. ABBET supports a general test methodology that can be applied to various types of products (electronic, mechanical, etc.), various levels of integration (system, subsystem, module, etc.), and various test applications (design verification test, manufacturing test, maintenance test, etc.).

1.2 Purpose

ABBET is a set of software interface standards for the product test domain that are intended to facilitate the transition of test information and test implementation. These software interface standards are defined to

support software component portability, reusability, exchangeability, and interoperability, and to serve as targets for test-related software development tools.

This standard describes the elements of ABBET and their interrelationships and defines common terms to be used throughout the set. The purpose of this standard is to ensure that the elements in the ABBET set of standards and guides fit together as a cohesive whole.

1.3 Application

This standard provides an overview of the ABBET set of standards and guides for developing the following:

- ABBET-conformant test systems
- ABBET-based test program sets (TPSs) and test objects
- Design data for use in test
- ABBET environment for tools
- Shared usage of maintenance data and test results

Anticipated users of the ABBET set of standards include the following:

- Unit under test (UUT) developers
- UUT maintainers
- TPS developers
- TPS maintainers
- Automatic test equipment (ATE) system developers
- ATE system maintainers
- Test instrument developers
- Developers of ABBET-based tools and systems
- Developers of prime mission equipment that use the supported UUT as a component

2. References

This standard shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision will be considered for incorporation into this standard.

ANSI/ISO 9899-1990 (Reaff 1997), Programming Languages—C.¹

IEEE Std 100-1996, The IEEE Standard Dictionary of Electrical and Electronics Terms, Sixth Edition.²

IEEE Std 1232-1995, IEEE Trial-Use Standard for Artificial Intelligence and Expert System Tie to Automatic Test Equipment (AI-ESTATE): Overview and Architecture.

ISO 10303-11:1994, Industrial automation systems and integration—Product data representation and exchange—Part 11: Description methods: The EXPRESS language reference manual.³

¹ANSI publications are available from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

²IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, P.O. Box 1331, Piscataway, NJ 08855-1331, USA (<http://www.standards.ieee.org/>).

³ISO publications are available from the ISO Central Secretariat, Case Postale 56, 1 rue de Varembe, CH-1211, Genève 20, Switzerland/Suisse (<http://www.iso.ch/>). ISO publications are also available in the United States from the Sales Department, American National Standards Institute, 11 West 42nd Street, 13th Floor, New York, NY 10036, USA (<http://www.ansi.org/>).

ISO 10303-21:1994, Industrial automation systems and integration—Product data representation and exchange—Part 21: Implementation methods: Clear text encoding of the exchange structure.

ISO DIS 14750, Information technology — Open Distributed Processing — Interface Definition Language

ISO/IEC DIS 14882:1998, Programming languages—C++.

3. Definitions, acronyms, and abbreviations

The terms defined in this clause are specified herein to support readability and clarity of this standard. In the event a term is explicitly redefined, or further defined, in a component standard, the component standard's definition shall be normative for that component standard.

3.1 Definitions

3.1.1 Abbreviated Test Language for All Systems (ATLAS): A standard abbreviated English language used in the preparation and documentation of test procedures or test programs that can be implemented either manually or with automatic or semiautomatic test equipment.

3.1.2 A Broad Based Environment for Test (ABBET)TM: A standards activity of SCC20 to develop a set of standards intended to support an open test environment.

3.1.3 adapter: A device or series of devices designed to provide a compatible connection between the test subject and the test equipment. *Synonyms:* **interface device; interface test adapter; test adapter.**

3.1.4 anomaly: Deviation from the normal behavior of a test subject. Faults (e.g., output stuck high, gain low) and manufacturing defects (e.g., missing or incorrect components, incorrectly installed components) are kinds of anomalies.

3.1.5 automatic test equipment (ATE): Equipment that is designed to conduct analysis of functional or static parameters to evaluate the degree of performance degradation and that may be designed to perform fault isolation of unit malfunctions.

3.1.6 automatic test system (ATS): Includes the automatic test equipment (ATE) as well as all support equipment, software, test programs, and adapters.

3.1.7 behavior: A formal representation of the characteristics that describe the operation, function, relationships, control, or static properties of a test entity.

3.1.8 binding: A defined mapping from the syntax and semantics of a software specification language into the syntax and semantics of a general purpose programming language. The purpose of such mapping to allow either a human or a machine to translate a program specified in the former language into a compilable and executable program in the latter.

3.1.9 built-in test (BIT): An integral capability of the test subject used to provide self-test capability.

3.1.10 built-in test equipment (BITE): Hardware included solely for the built-in test function.

3.1.11 class: A template for the creation of an object instance. The class defines the properties of an object.

3.1.12 classification: A grouping of objects on the basis of common characteristics. This is the normal method for grouping objects within an object-oriented programming environment.

3.1.13 common services: Data type and functional declarations defining an application procedural interface used to access and manipulate numerical and digital data, physical data units, mathematical functions and constants, unit conversion factors, and functions used by two or more interfaces.

3.1.14 container class: A class that defines an object that holds other objects.

3.1.15 context: Reflects the intended scope of a set of tests. Examples of context include manufacturing process test, maintenance test, design verification test, screening test, etc.

3.1.16 Core Test Information Model (CTIM): The fundamental information entities required to describe tests, test specifications, test requirements, and other test entities within the ABBETTM domain. The CTIM contains the significant types from which the component standards may be derived.

3.1.17 corrective action: Intended to eliminate anomalies. Corrective actions include repair, replacement, calibration, alignment, and other services. *See also:* **maintenance**.

3.1.18 diagnosis: The conclusion(s) resulting from tasks, tests, observations, or other information.

3.1.19 diagnostic controller: The agent (this could be from an expert/reasoner system or from an operator) that invokes test procedures in the sequence required to achieve test goals.

3.1.20 diagnostic knowledge: Provides the information required to support the diagnostic process. This knowledge defines the relationships between possible test outcomes and anomalies that may cause these outcomes.

3.1.21 diagnostic process: A structured combination of tasks, tests, observations, and other information used to localize a fault or faults.

3.1.22 encapsulation: The grouping of data and operations upon that data into a single object.

3.1.23 entity: A distinct thing, object, or concept. (*The Artificial Intelligence Dictionary*, Ellen Thro, MICROTREND Books, San Marco, CA.)

3.1.24 error logging: The recording of an error condition detected during the execution of a service.

3.1.25 framework: A collection of classes created specifically to serve the needs of an application area.

3.1.26 functional partitioning: The logical separation of system or unit elements along interfaces that define and isolate these elements on the basis of function or purpose.

3.1.27 functional test: A test that is intended to verify that a test subject is behaving as specified.

3.1.28 Go/NoGo test: Terms referring to the condition or state of operability of a unit that can only have two outcomes, Go, functioning properly, or NoGo, not functioning properly. (Adapted from MIL-STD-1309D)

3.1.29 handler: A program or routine that performs or controls one task (e.g., error detection).

3.1.30 historical data: All relevant information available concerning the product, tests, and test equipment. This includes test observations (raw measurement data), derived test outcomes (i.e., LO, HI, GO), diagnostic conclusions derived from test results and the knowledge base, test subject mission and configuration history, test resources mission and history, etc.

3.1.31 information modeling: An information model is a formal description of types of ideas, facts, and processes, which together form a model of a portion of interest of the real world and which provides an explicit set of interpretation rules. *See also:* **corrective action**. (*Information modeling the EXPRESS way*, D. Schenck and P. Wilson, New York: Oxford University Press, 1994.)

3.1.32 instrument: A device whose purpose is usually the generation or measurement of a class of signal.

3.1.33 integrated diagnostics: A structured process that maximizes the effectiveness of diagnostics by integrating testability, automatic and manual testing, training and technical information to provide a cost effective capability to detect and isolate faults with the ultimate goal of maximizing equipment availability and minimizing total user cost. (AUTOTESTCON '92, "The Application of Integrated Diagnostics Techniques to Heavy Duty Vehicle Testing," Martin D. Brazet, GE Automated Systems Dept. Burlington, MA, p. 31.)

3.1.34 interface: A shared boundary that specifies the interconnection between two units or systems, hardware or software. In hardware, the specification includes the type, quantity, and function of the interconnection circuits and the type and form of signals to be interchanged via those circuits. In software, the specification includes the object type and, where necessary, the name or instance handle of specific objects copied or shared between the two systems.

3.1.35 Interface Definition Language (IDL): A machine-compilable language used to describe interfaces that clients call and implementations provide. IDL provides a neutral way to define an interface. [IDL is an Object Management Group (OMG) product.]

3.1.36 interface device: *See:* **adapter.**

3.1.37 interface test adapter: *See:* **adapter.**

3.1.38 knowledge-based test: A test based in part on previously acquired information. (Adapted from MIL-STD-1309D)

3.1.39 language-independent specification: The format for describing services that is not tied to any specific computer language.

3.1.40 maintenance: Activity intended to keep equipment (hardware) or programs (software) in satisfactory working condition, including replacements, adjustments, repairs, software/firmware updates, and program improvements. Maintenance can be preventive or corrective. (Adapted from MIL-STD-1309D)

3.1.41 manual testing: Testing that requires a human to execute some or all of a test procedure.

3.1.42 mapping: Process of correspondence between the elements of one set and the elements of another set.

3.1.43 method: A property of a class that defines a specific behavior.

3.1.44 object: A member (instance) of a class that encapsulates the data (state) and the behavior (methods) of the object. (*The Artificial Intelligence Dictionary*, Ellen Thro, MICROTREND Books, San Marco, CA.)

3.1.45 Object Management Group (OMG): Organization of computer manufacturers, software developers, communications organizations, and computer users established to promote open object-oriented computer architectures and standards.

3.1.46 object repository: An area for the storage of objects, like a library.

3.1.47 observation: The raw data acquired by executing a test procedure. It represents the observed characteristics of a specific signal (e.g., the voltage peak of a sinusoid wave form), the observed characteristics of the environment (e.g., the ambient temperature), or the derived value of product characteristics (e.g., the measured value of gain).

3.1.48 open architecture: An architecture from which a system can be assembled from multiple vendor-supplied interface components. The resulting system can execute applications written by arbitrary independent vendors and can be extended by users other than the original supplier.

3.1.49 operation: An action defined by a procedure.

3.1.50 parametric testing: Testing of a test subject's ability to function correctly within acceptable tolerance when all values are varied within specified limits. (Adapted from MIL-STD-1309D)

3.1.51 portability: The ease with which software can be transferred from one system or environment to another. A relative measure of effort, inversely proportional to the level of modification required for software to be transferred from one system or environment to another.

3.1.52 process: Sequence of operations performed in and by the equipment in which a variable is to be controlled.

3.1.53 product characteristic: An observable attribute of a product. This includes functional, physical, and performance characteristics (e.g., gain and bandwidth of an amplifier).

3.1.54 resource manager: A process or activity that initializes and manages the resources in a system.

3.1.55 response data: The information sensed from a test subject as the result of an applied stimulus.

3.1.56 run-time test object (RTO): Contains the procedures utilized and data necessary to execute a test on a test subject within a specified context.

3.1.57 schema: The structure or framework used to define a data record. This includes each field's name, type, shape, dimension, and mapping.

3.1.58 sense signal: The response taken or measured from a test subject.

3.1.59 sequencer: An object that controls the execution flow of programs.

3.1.60 service: Operation or run-time call whose behavior and interface are standardized. *See also:* **method**.

3.1.61 signal: **(A)** The behavior controlled or observed by a test resource. **(B)** A visual, audible, or other indication used to convey information.

3.1.62 software reuse: The reapplication of previously developed software or information associated with that software.

3.1.63 stimulus: Any physical or electrical input applied to a test subject intended to produce a measurable response. (Adapted from MIL-STD-1309D)

3.1.64 stimulus data: The information associated with stimuli.

3.1.65 test: An observed activity that may be caused to occur (e.g., stimulus-response) in order to obtain information about the behavior of a test subject.

3.1.66 testability: A design characteristic that allows the status (operable, inoperable, or degraded) of an item to be determined and the isolation of faults within the item to be performed in a timely and efficient manner. (Adapted from MIL-STD-1309D)

3.1.67 test adapter: *See:* **adapter**.

3.1.68 test application framework (TAF): A structure for organizing the test objects related to a specific testing requirement.

3.1.69 test asset: An assemblage of instruments, interconnect devices, supporting software, and manual procedures that enable one or more test objectives to be achieved. *See also:* **automatic test system.**

3.1.70 test control: The functionality that directs and facilitates the execution of tests and the collection of data.

3.1.71 test foundation framework (TFF): A comprehensive set of object classes that supports the use of product data, the development of test programs, and the utilization of diagnostic data elements. The classes defined in the TFF are based wholly on the fundamental functionality required for test.

3.1.72 test method: A specification that defines the algorithm, procedures, and required controllable inputs and potential behavior (nominal or anomalous) of a test object.

3.1.73 test object: Any object defined for use within the domain of test representing an encapsulated view of a test method with interfaces to a test system.

3.1.74 test outcome: A mapping from an observation to one of a set of discrete possibilities.

3.1.75 test procedure: The implementation of a test method.

3.1.76 test program (TP): A program specifically intended for the testing of a test subject.

3.1.77 test program set (TPS): An assembly of items necessary to test a test subject on a piece of automatic test equipment (ATE). This includes the electrical, mechanical, instructional, and logical decision elements. The individual elements of the TPS are the TP, the adapter, and the TPS documentation (TPSD).

3.1.78 test requirement: A specification of the test methods and test conditions needed to evaluate and diagnose a test subject.

3.1.79 test specification: A document that defines the tests to be performed on a test subject to verify conformance with its performance specification, without reference to any specific test equipment or test method.

3.1.80 test strategy: (A) The arrangement of specific tester types to achieve optimum throughput and diagnostic capability at the least possible cost given the fault spectrum, process yield, production rate, and product mix for a particular environment. (Adapted from MIL-STD-1309D) **(B)** A selection of test methods to achieve some diagnostic test within execution time and test resource constraints.

3.1.81 test subject: The specific product design that is the focus of attention or target for the development of tests and diagnostics.

3.1.82 user interface: The part of the application that permits the user and application to communicate with each other to perform certain tasks.

3.1.83 virtual instrument software architecture (VISA): The general name given to the VPP (VXI Plug & Play) 4 Specification and its associated architecture. The architecture consists of two main VISA components: the VISA resource manager and the VISA instrument control resources.

3.2 Acronyms

ABBET	A Broad-Based Environment for Test
AI-ESTATE	Artificial Intelligence Exchange and Service Tie to All Test Environments
API	Application Program Interface
ATE	automatic test equipment
ATLAS	Abbreviated Test Language for All Systems
ATS	automatic test system
BIT	built-in test
BITE	built-in test equipment
C/ATLAS	Common/Abbreviated Test Language for All Systems
CORBA	Common Object Request Broker Architecture
CTIM	Core Test Information Model
EDIF	Electronics Design Interchange Format
FRU	field replaceable unit
IDL	Interface Definition Language
IDEF	ICAM (Integrated Computer-Aided Manufacturing) Definition
OMG	Object Management Group
RTO	run-time test object
TAF	test application framework
TFF	test foundation framework
TP	test program
TPS	test program set
TPSD	test program set documentation
TRIM	test resource information model
UUT	unit under test
VHDL	VHSIC (Very high speed integrated circuit) Hardware Description Language
VISA	Virtual Instrument Software Architecture
VPP	VXI Plug & Play
VXI	VME (Versa Module Eurocard) Extension for Instrumentation

4. ABBET overview

4.1 Background

The need for improved testing environments is caused by market and technology pressures. There are market pressures to reduce a new product's time-to-market and to reduce maintenance costs. At the same time, testing environments must respond to continuously evolving and increasingly complex technology used in modern products. Test technology needs to keep pace with these pressures.

The lack of accepted industry standards for test related information is one factor that has limited the availability and use of improved test environments. Computer-aided design (CAD) environments and exchange standards for design data have helped designers to cope with these pressures, but testing environments have lagged behind.

Computer-aided test environments offer the potential to efficiently access and manipulate design and test information. For example, product test information from earlier life cycle phases can be preserved in order to eliminate rediscovery efforts in later phases. Libraries of reusable test procedures can be provided for various types of products. Test equipment interfaces, capabilities, and control characteristics can also be accessed from libraries and used to adapt existing tests to other environments with different test resources.

Not all tests are performed using automatic test equipment (ATE). ABBET also supports the use of built-in test (BIT) and test applications intended to help individuals who perform manual tests.

4.2 Purpose

The primary purpose of ABBET is to specify standards for test environments that encompass the total product life cycle. ABBET defines an integrated set of test related information and services that support the information and implementation needs of test environments for testing applications. ABBET is intended to accomplish the following objectives:

- a) Facilitate the communication, sharing, and reuse of product design and test information for the purpose of testing the product.
- b) Facilitate TPS portability and interoperability.
- c) Facilitate instrument interchangeability.
- d) Facilitate the development, integration, and use of test software and test software development tools.
- e) Support the application of integrated diagnostics. (This is accomplished through standardized diagnostic models for products and information transfers related to diagnostics at various levels.)
- f) Specify a modular architecture for test software based on a framework that supports reusable software assets.
- g) Specify standard Application Program Interfaces (APIs) for test-related services that are either embedded within a product itself or on ATE and other maintenance aids.
- h) Specify standard APIs in a way that allows conforming software products to be implemented in various programming languages.
- i) Enable enhanced capabilities for test tools and environments.
- j) Allow for the use of existing standards in an ABBET-compliant environment. (This may be accomplished by the use of integration protocols that define a mapping between the information provided by those standards and the information specified in the ABBET standard.)

4.3 ABBET and the product life cycle

Five reasons for performing tests follow:

- *Design verification.* Verify that a product design meets its functional performance specifications.
- *Product verification.* Verify that a product meets its functional and structural performance specifications.
- *Product calibration.* Adjust the characteristics of a product to meet specified tolerances or performance criteria. Calibration tests are often incorporated as part of a verification process.
- *Diagnosis.* Identify a fault condition in the event a product verification test fails or a problem is reported.
- *Process control.* Acquire information about a process used to produce, operate, or maintain a product. For example, monitor and maintain proper environmental conditions during a manufacturing process.

These reasons motivate the creation of test assets. Test assets can be manually or automatically controlled; extend from simple inspection to complex functional processing; be built in or applied by external means; or be applied on entire systems or constituent subassemblies. The decisions regarding the test assets depend upon the selected life-cycle test strategies. All tests produce potentially useful data that can be used for evaluation as dictated by either technology or policy. ABBET supports the useful exchange of test information among product life-cycle phases while supporting the different test-related technologies present in each.

The product life cycle shown in the top portion of Figure 1 is typical of product development, deployment, operation, and maintenance operations. The life cycle phases are shown from left to right across the diagram. Activities performed during each phase are shown as circles on the diagram, with the test activities being high-lighted. The three life cycle phases that have test activities are as follows:

- *Design.* The product proceeds from a concept to a working prototype or preproduction model. The characteristics of the model are tested to verify compliance with specified design and performance criteria.
- *Manufacture.* The product is fabricated and made available for distribution and use. Tests are used to assure that the product is properly assembled prior to delivery.
- *Support.* The product is performing its intended function. Tests are used to monitor the product's condition during normal operation, to support periodic maintenance activities, and to diagnose faults in trouble situations.

Figure 1 also indicates that products frequently have multiple levels of assembly (e.g., circuit card, sub-system, system) and test activities occur at all levels within each life-cycle phase.

The lower portion of Figure 1 identifies five categories of test information associated with every test application.

- *Design data.* This includes characteristics of the test subject that must be known before tests can be defined. It can include both static structural properties and dynamic behavior.
- *Test strategy.* This includes the targeted set of test subject faults, specifications for a set of individual tests, and diagnostic knowledge. The targeted set of test subject faults is the set of defects that the test application is expected to detect if present. Each individual test is characterized by a measured test subject characteristic, its acceptable range of values, and the method used for making the measurement. Diagnostic knowledge correlates test outcomes with the test subject faults that they reveal. The test strategy within a specific test context contributes to the test requirements that must be defined and met.
- *Test control.* This consists of manual or automatic test procedures for the test subject.
- *Test resources.* This includes descriptions of test equipment configurations, capabilities, and control characteristics.
- *Test and maintenance information.* This consists of historical data collected during performance of tests. It can include measured values for each individual test, recommended corrective actions, and maintenance actions that were performed. This collected information can be used within and between various phases of the product life cycle to assure proper maturation of the testability and diagnostics of a product, as well as the product itself

The lower portion of Figure 1 also shows several paths where improved flow of test information can reduce the overall cost of test. The solid arrows represent information flow dependencies during the development process and the subsequent feedback of actual test results within a single test application. Test information flows reasonably well within a single test application as indicated by the solid arrows. However, this information is rarely captured in a form that other, related test applications can readily access and use. The dotted arrows indicate that frequently the transfer of test information from one life-cycle phase to another and from one level of system integration to another is nearly nonexistent.

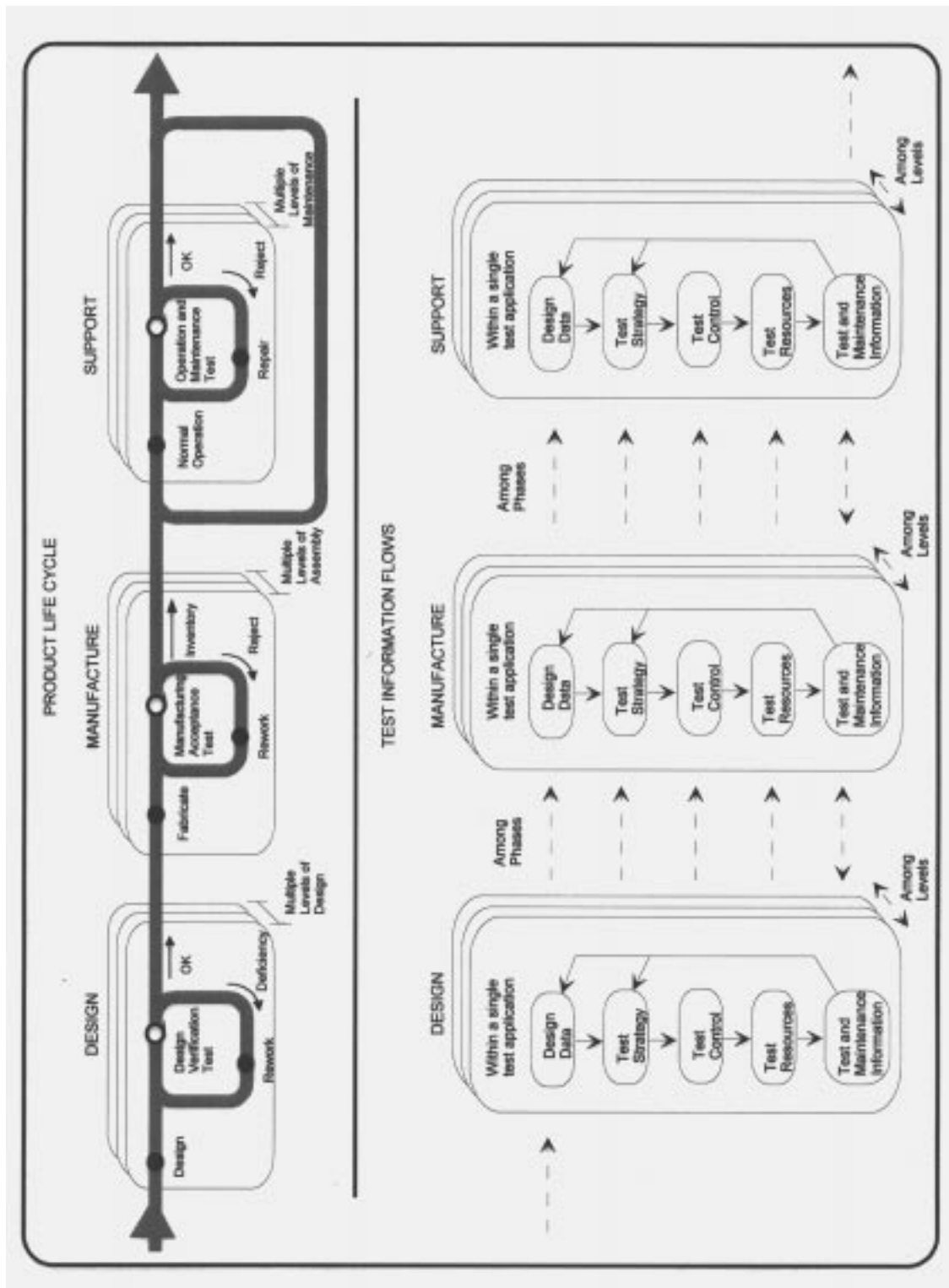


Figure 1—Product life cycle and test information flows