



IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior

IEEE Computer Society

Sponsored by the
Software and Systems Engineering Standards Committee

1175.4TM

IEEE
3 Park Avenue
New York, NY 10016-5997, USA
17 April 2009

IEEE Std 1175.4TM-2008

IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior

Sponsor

**Software and Systems Engineering Standards Committee
of the
IEEE Computer Society**

Approved 10 December 2008

IEEE-SA Standards Board

Abstract: A reference model that provides a common interpretation basis by which tools can express and communicate the observable features of system/software behavior to users and to other tools is presented. This standard specifies a Conceptual Metamodel for understanding and describing the causal behavior for a system. The purpose of this Conceptual Metamodel is to express causal behavior and compositions of causal behavior in a model that integrates all observable operational features of a system into one behavior specification. This Conceptual Metamodel is useful for analyzing systems, for constructing particular system behavior models, and for using those models in the specification, design, and evaluation of engineered systems. It provides the necessary semantic elements for describing general hardware/software systems, including hardware-only, software-only, or mixed system components, and it allows these different types of components to be treated in a consistent manner, providing a basis for representing a wide variety of systems.

Keywords: behavior model, behavior specification, Computer-Aided Software Engineering (CASE) tools, engineering model, model-based testing, modeling principles, requirements, software specification, specification-based testing, system, system testing

The Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York, NY 10016-5997, USA

Copyright © 2009 by the Institute of Electrical and Electronics Engineers, Inc.
All rights reserved. Published 17 April 2009. Printed in the United States of America.

IEEE and POSIX are registered trademarks in the U.S. Patent & Trademark Office, owned by the Institute of Electrical and Electronics Engineers, Incorporated.

Capability Maturity Model and CMMI are registered in the U.S. Patent and Trademark Office by Carnegie Mellon University.

PDF: ISBN 978-0-7381-5861-7 STD95857
Print: ISBN 978-0-7381-5862-4 STDPD95857

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. The IEEE develops its standards through a consensus development process, approved by the American National Standards Institute, which brings together volunteers representing varied viewpoints and interests to achieve the final product. Volunteers are not necessarily members of the Institute and serve without compensation. While the IEEE administers the process and establishes rules to promote fairness in the consensus development process, the IEEE does not independently evaluate, test, or verify the accuracy of any of the information or the soundness of any judgments contained in its standards.

Use of an IEEE Standard is wholly voluntary. The IEEE disclaims liability for any personal injury, property or other damage, of any nature whatsoever, whether special, indirect, consequential, or compensatory, directly or indirectly resulting from the publication, use of, or reliance upon this, or any other IEEE Standard document.

The IEEE does not warrant or represent the accuracy or content of the material contained herein, and expressly disclaims any express or implied warranty, including any implied warranty of merchantability or fitness for a specific purpose, or that the use of the material contained herein is free from patent infringement. IEEE Standards documents are supplied “**AS IS**.”

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.

In publishing and making this document available, the IEEE is not suggesting or rendering professional or other services for, or on behalf of, any person or entity. Nor is the IEEE undertaking to perform any duty owed by any other person or entity to another. Any person utilizing this, and any other IEEE Standards document, should rely upon his or her independent judgment in the exercise of reasonable care in any given circumstances or, as appropriate, seek the advice of a competent professional in determining the appropriateness of a given IEEE standard.

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 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. A statement, written or oral, that is not processed in accordance with the IEEE-SA Standards Board Operations Manual shall not be considered the official position of IEEE or any of its committees and shall not be considered to be, nor be relied upon as, a formal interpretation of the IEEE. At lectures, symposia, seminars, or educational courses, an individual presenting information on IEEE standards shall make it clear that his or her views should be considered the personal views of that individual rather than the formal position, explanation, or interpretation of the IEEE.

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. Comments on standards and requests for interpretations should be submitted to the following address:

Secretary, IEEE-SA Standards Board
445 Hoes Lane
Piscataway, NJ 08854
USA

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; +1 978 750 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 1175.4-2008, IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior.

The 1175™ family of standards

NOTE—References to “P1175.X” in this standard refer to members of the 1175 family of standards that were not yet approved at the time that this standard was published.^a

This standard is a member of the 1175 family of IEEE standards. The members of this family include the following:

Standard number	Title
IEEE Std 1175.1™-2002 [B4] * †	IEEE Guide for CASE Tool Interconnections—Classification and Description
IEEE Std 1175.2™-2006 [B5]	IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections
IEEE Std 1175.3™-2004 [B6]	IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying Software Behavior
IEEE Std 1175.4™-2008	IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior
IEEE P1175.5™ ‡ **	Draft Standard for Computer-Aided Software Engineering (CASE) Tool Interconnections—Reference Data Metamodel for System Behavior Specifications

* The numbers in brackets correspond to those of the bibliography in Annex A.

† IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854-1331, USA (<http://standards.ieee.org/>).

‡ This IEEE standards project was not approved by the IEEE-SA Standards Board at the time this publication went to press. For information about obtaining a draft, contact the IEEE.

** The title and description of IEEE P1175.5 are current as of the IEEE Std 1175.4-2008 publication date. Because the P1175.5 draft was not yet approved as of March 2009, this information is subject to change. For the most current P1175.5 information, please consult IEEE Xplore at <http://ieeexplore.ieee.org/xpl/standards.jsp>.

This family of standards replaces IEEE Std 1175-1991 [B3].^b IEEE Std 1175-1991 was advanced to a full-use standard in 1994. It covered a number of closely related subjects, and the scope of material contained was able to serve a number of divergent interests.

This family of standards restructures and substantially augments the material in IEEE Std 1175-1991. It has been divided into several individually useful documents in order to facilitate its use by different communities of interest. These guides, recommended practices, and standards generally address issues involved in characterizing the kinds of interconnections that exist between a computing system tool and its environment. Although particularly intended to address the implementation and use of Computer-Aided Software Engineering (CASE) tools, the discussion of interconnections in this family actually has wider applicability to computing system tools in general, beyond only CASE tools.

^a Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement this standard.

^b Although approved in 1991, IEEE Std 1175-1991 was actually published in 1992 and is sometimes found referenced as IEEE Std 1175-1992. It appears in the standards numerical listing on the IEEE Xplore Web site (<http://ieeexplore.ieee.org/>) as IEEE Std 1175-1992, with the title IEEE Trial-Use Standard Reference Model for Computing System Tool Interconnections. In 1994, the term trial-use was removed from the title when the standard was approved for full-use status. The 1994 version, which was identical to the 1992 publication except for the title and minor editorial corrections, is not available on the IEEE Web site.

Four kinds of interconnections with a computing system tool are addressed: interconnections with organizations, users, platforms, and other computing system tools. Consideration of interconnections is important to understanding, selecting, implementing, and using computing system tools. Also, although many computing system tools do not need to communicate behavior descriptions of subject systems, their creators need to develop such behavior descriptions for the tools themselves.

A brief summary overview of each of the members of this family of standards is given in the following paragraphs. A more complete overview is available in IEEE Std 1175.1-2002 [B4], which provides an integrated overview of the members of the 1175 family of standards, and it describes the fundamental concepts that provide a basis for organizing the material.

IEEE Std 1175.1-2002, IEEE Guide for CASE Tool Interconnections— Classification and Description

IEEE Std 1175.1-2002 [B4] is a guide to the IEEE 1175 family of standards. It describes how these standards are intended to be used to accomplish the effective integration of computing system tools into a productive engineering environment and sets forth the fundamental concepts on which these standards are based. These concepts establish the integrating framework for the other members of this family of standards. IEEE Std 1175.1-2002 describes the scope of application of each member standard, the various issues addressed in each standard, and the interrelationships among the members of the 1175 family of standards.

IEEE Std 1175.2-2006, IEEE Recommended Practice for CASE Tool Interconnection—Characterization of Interconnections

The IEEE Std 1175.2-2006 [B5] recommended practice presents four contexts for a computing system tool's interconnections that offer insight into the operational problems of interconnecting computing system tools with their environment. This recommended practice establishes recommended collections of standard contextual attributes describing relationships between a computing system tool and its organizational deployment, its human user, its executable platform, and its peer tools, as illustrated in Figure a. These contextual attributes are of the “news-story” form that includes: who, what, when, where, and why. The values of these contextual attributes are references to organizational, industrial, and professional standards. By assisting users to reach a clear understanding of the context of operation for a computing system tool, this recommended practice contributes to the effective implementation and application of computing system tools.

IEEE Std 1175.3-2004, IEEE Standard for CASE Tool Interconnections— Reference Model for Specifying Software Behavior

IEEE Std 1175.3-2004 [B6] is an expansion of Part 3 of IEEE Std 1175-1991. It focuses specifically on a common set of modeling concepts found in commercial CASE tools for describing the operational behavior of a software product, and it provides a formal, logical model for describing this behavior. IEEE Std 1175.3-2004 also defines a Semantic Transfer Language (STL) for communicating software behavior descriptions from one tool to another. A notable feature of the STL is its design for human readability, which makes STL text files suitable for use in software design reviews by users unfamiliar with computing system tool diagramming notations. In addition, the design of the STL syntax readily permits analysts to prepare and edit STL descriptions using a text editor or word processor.

To permit backward compatibility with Part 3 of IEEE Std 1175-1991, IEEE Std 1175.3-2004 makes no changes to the STL syntax or to the rules for conformance to this syntax as originally defined in that

standard. However, some aspects of the 1991 syntax that were previously left as user-defined have now been specified in order to increase the consistency and reliability with which the STL may be used for exchanging software specification information. In addition, improvements have been made in how the STL syntax is defined and explained. Finally, the STL Interconnection Profile has been replaced with more straightforward, “user-friendly” tabular and comma-separated-value formats to define a Tool Interconnection Profile that can serve the same purpose as the original form of the profile.

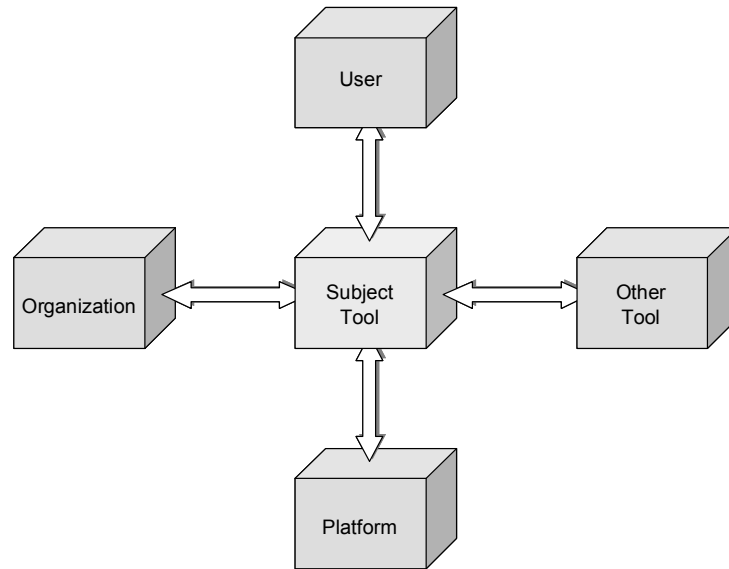


Figure a

IEEE Std 1175.4-2008, IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior

IEEE Std 1175.4-2008 encompasses the description of the types of the computing systems supported by IEEE Std 1175.3-2004 [B6], but it goes further, providing a basis for representing a wider variety of systems. Specifically, IEEE Std 1175.4-2008 provides the necessary semantic elements for describing general hardware/software systems, including hardware-only, software-only, or mixed system components, and it allows these different types of components to be treated in a consistent manner.

IEEE P1175.5, Draft Standard for Computer-Aided Software Engineering (CASE) Tool Interconnections—Reference Data Metamodel for System Behavior Specifications

This standard defines a Data Metamodel for system behavior specifications. Figure b illustrates one use for such a behavior specification metamodel in the context of supporting information transfer from one user’s tool to another user’s tool. The Data Metamodel provides explicit definitions of typed data elements, information representations, and relationships with which behavior models for subject systems can be instantiated. These elements, representations, and relations serve to reify the Conceptual Metamodel for system behavior specification described in IEEE Std 1175.4-2008.

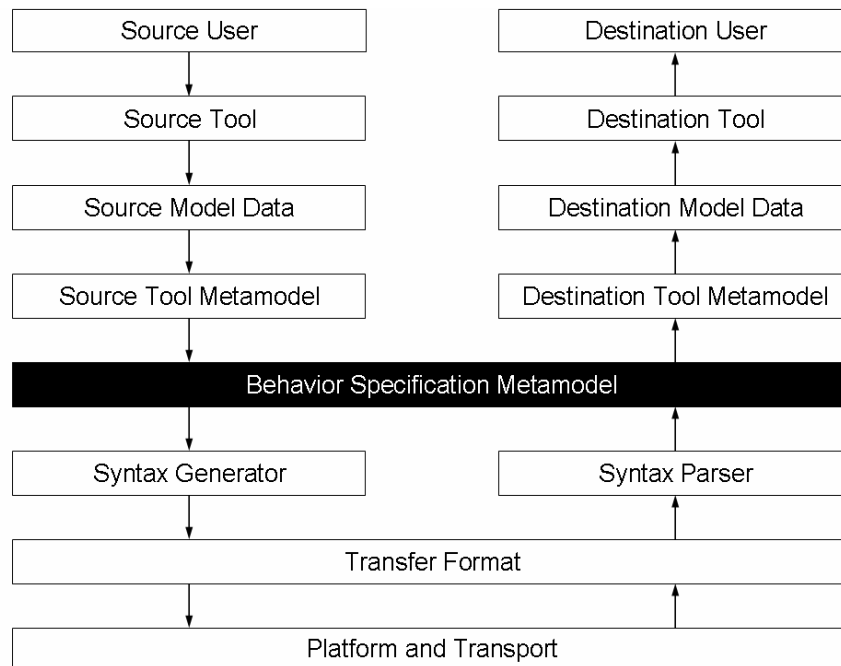


Figure b

When multiple tools are being used to describe a system, each may maintain its own information metamodel. However, as depicted in Figure b, to share information about a subject system, each tool must map its own individual metamodel into a common Behavior Specification Metamodel.

Notice to users

Laws and regulations

Users of these documents should consult all applicable laws and regulations. Compliance with the provisions of this standard does not imply compliance to any applicable regulatory requirements. Implementers of the standard are responsible for observing or referring to the applicable regulatory requirements. IEEE does not, by the publication of its standards, intend to urge action that is not in compliance with applicable laws, and these documents may not be construed as doing so.

Copyrights

This document is copyrighted by the IEEE. It is made available for a wide variety of both public and private uses. These include both use, by reference, in laws and regulations, and use in private self-regulation, standardization, and the promotion of engineering practices and methods. By making this document available for use and adoption by public authorities and private users, the IEEE does not waive any rights in copyright to this document.

Updating of IEEE documents

Users of IEEE standards should be aware that these documents may be superseded at any time by the issuance of new editions or may be amended from time to time through the issuance of amendments, corrigenda, or errata. An official IEEE document at any point in time consists of the current edition of the document together with any amendments, corrigenda, or errata then in effect. In order to determine whether a given document is the current edition and whether it has been amended through the issuance of amendments, corrigenda, or errata, visit the IEEE Standards Association Web site at <http://ieeexplore.ieee.org/xpl/standards.jsp>, or contact the IEEE at the address listed previously.

For more information about the IEEE Standards Association or the IEEE standards development process, visit the IEEE-SA Web site at <http://standards.ieee.org>.

Errata

Errata, if any, for this and all other standards can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/updates/errata/index.html>. Users are encouraged to check this URL for errata periodically.

Interpretations

Current interpretations can be accessed at the following URL: <http://standards.ieee.org/reading/ieee/interp/index.html>.

Patents

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 is not responsible for identifying Essential Patent Claims for which a license may be required, for conducting inquiries into the legal validity or scope of Patents Claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance, if any, or in any licensing agreements are reasonable or non-discriminatory. Users of this standard are expressly advised that determination of the validity of any patent rights, and the risk of infringement of such rights, is entirely their own responsibility. Further information may be obtained from the IEEE Standards Association.

Participants

At the time this standard was submitted to the IEEE-SA Standards Board for approval, the 1175.4 Working Group had the following membership:

Carl A. Singer, *Chair*

Jimi E. Arvidsson
Bakul Banerjee
Abby Beifeld
Peter L. Eirich
Subramanya R. Jois

Paul C. Jorgensen
Sohel M. Khan
Dwayne L. Knirk
Mohamed Ashraf Kottilungal
Horace H. Lawrence

Álvaro F. C. Medeiros
Lou F. Pinto
Robert M. Poston
Subrato Sensharma
Robert M. Wessely

The following members of the individual balloting committee voted on this standard. Balloters may have voted for approval, disapproval, or abstention.

Butch Anton
Angela Anuszewski
Bakul Banerjee
Zulema Belyeu
Pieter Botman
Juan Carreon
Lanna Carruthers
Lawrence Catchpole
Keith Chow
S. Claassen
Paul Croll
Geoffrey Darnton
Thomas Dineen
Sourav Dutta
Yaacov Fenster

Allan Gillard
Randall Groves
John Harauz
Mark Henley
Werner Hoelzl
Atsushi Ito
Mark Jaeger
Dwayne L. Knirk
Thomas Kurihara
Susan Land
Richard Martin
William Milam
James Moore
Rajesh Murthy

Ulrich Pohl
Robert Robinson
Rey Robles
Randall Safier
Bartien Sayogo
Robert Schaaf
David J. Schultz
Stephen Schwarm
Carl A. Singer
Luca Spotorno
Thomas Starai
Walter Struppler
Vincent Tume
Paul Work
Oren Yuen

When the IEEE-SA Standards Board approved this standard on 10 December 2008, it had the following membership:

Robert M. Grow, *Chair*
Thomas Prevost, *Vice Chair*
Steve M. Mills, *Past Chair*
Judith Gorman, *Secretary*

Victor Berman
Richard DeBlasio
Andy Drozd
Mark Epstein
Alexander Gelman
William R. Goldbach
Arnold M. Greenspan
Kenneth S. Hanus

Jim Hughes
Richard H. Hulett
Young Kyun Kim
Joseph L. Koepfinger*
John Kulick
David J. Law
Glenn Parsons

Ronald C. Petersen
Chuck Powers
Narayanan Ramachandran
Jon Walter Rosdahl
Anne-Marie Sahazizian
Malcolm V. Thaden
Howard L. Wolfman
Don Wright

*Member Emeritus

Also included are the following nonvoting IEEE-SA Standards Board liaisons:

Satish K. Aggarwal, *NRC Representative*
Michael Janezic, *NIST Representative*

Jennie Steinhagen
IEEE Standards Program Manager, Document Development

Malia Zaman
IEEE Standards Program Manager, Technical Program Development

Contents

1. Overview	1
1.1 Scope	1
1.2 Purpose	1
1.3 Applicability	2
1.4 Engineering models and metamodels	3
1.5 Audience	4
1.6 Conformance	4
1.7 Organization of this standard	5
2. Normative references	5
3. Special terms	6
4. System behavior model requirements	8
4.1 Application scope	9
4.2 Declarative form	10
4.3 Predictive ability	11
4.4 Test creation ability	14
4.5 Life cycle roles	15
4.6 Common interpretation	16
5. System behavior metamodels	16
5.1 Developing the Conceptual and Data Metamodels for behavior modeling	16
5.2 Using the Conceptual and Data Metamodels for behavior modeling	18
6. Conceptual Metamodel foundations	19
6.1 Agent of behavior—Unit	19
6.2 Percepts of unit behavior—Boundary interfaces	24
6.3 Concepts of unit behavior—Behavior patterns	30
6.4 Observables—Base quantities	53
6.5 Composition of unit behavior—Build structures	56
6.6 Prediction of unit behavior	63
7. Conceptual Metamodel requirements	64
7.1 Requirements for metamodel organization	64
7.2 Requirements for representing Boundary Interfaces	71
7.3 Requirements for representing Behavior Patterns	72
7.4 Requirements for representing Base Quantities	75
7.5 Requirements for representing Build Structures	78
Annex A (informative) Bibliography	80
Annex B (informative) Comparison of the system behavior model with other engineering models	81

IEEE Standard for CASE Tool Interconnections—Reference Model for Specifying System Behavior

IMPORTANT NOTICE: *This standard is not intended to ensure safety, security, health, or environmental protection in all circumstances. Implementers of the standard are responsible for determining appropriate safety, security, environmental, and health practices or regulatory requirements.*

This IEEE document is made available for use subject to important notices and legal disclaimers. These notices and disclaimers appear in all publications containing this document and may be found under the heading “Important Notice” or “Important Notices and Disclaimers Concerning IEEE Documents.” They can also be obtained on request from IEEE or viewed at <http://standards.ieee.org/IPR/disclaimers.html>.

1. Overview

1.1 Scope

Most inter-tool data transfer standards deal with protocol and syntax of the transfer, with a shared semantic basis assumed. This standard provides an explicitly defined metamodel (and meta-metamodel) for specifying system and software behavior. It defines a semantic basis of observables that allows each tool, whatever its own internal ontology, to communicate facts about the behavior of a subject system as precisely as the tool’s metamodel allows. Conventional tool model elements are reduced into simpler, directly observable fact statements about system behavior. This metamodel is much expanded over the original metamodel for software behavior in Part 3 of IEEE Std 1175™-1991 [B3].¹

1.2 Purpose

This reference model provides a common interpretation basis by which tools may express and communicate the observable features of system/software behavior to users and to other tools. Tools incorporating this metamodel in their import/export facilities enable engineers to interconnect best-in-class analysis and specification tools for integrated problem solving. Another feature of this metamodel is that it

¹ The numbers in brackets correspond to those of the bibliography in Annex A.

provides a specification that is directly testable. Finally, the provision of an explicit meta-metamodel enables tool builders to extend the reference metamodel for particular purposes.

1.3 Applicability

A fundamental notion in this standard is that the engineering specifications (technical descriptions) of a system are models of that system. Each model characterizes some aspect of the system from a particular viewpoint to serve particular engineering needs. Functional and physical elements associated with a particular aspect of the system are described and organized into models of all necessary aspects of the subject system.

The aspect of systems for which the model described in this standard may be used is the causal (stimulus-response) behavior and affective (functional or historical dependence) behavior of systems. (These behaviors are discussed in more detail in 6.3.1.) The class of systems for which the model described in this standard may be used are engineered systems that interact dynamically and deterministically with some part of the external world in which they exist, for effecting changes in that external world. Such systems have material existence, they have measurable static and dynamic properties, and they consume energy to provide their effects.

The class of engineered system addressed by this standard includes those that take action based on their sense of properties or events in their environment. Such systems monitor their environment and predictably perform expected actions based on the environment they perceive. Their only causal behaviors might be an OFF-ON state change and an ON-OFF state change controlled by a power switch. Examples might include a patient-monitoring system, an intrusion detection system, or an automatic home coffee-maker. These are primarily passive systems that do not rely on external agents to stimulate their performance.

This class of engineered system also includes those that do not take action unless directed to do so by a stimulus from their environment. Examples would include an information system storing input data, responding to edits, and producing reports on request, or a remotely commanded sensor system that senses phenomena in the local environment, stores selected digital data streams, transmits data records, or purges old data records. These are reactive systems that do rely on external agents to stimulate their performance.

Numerous models and modeling frameworks are currently used to understand, develop, and document the detailed structural and transformational relationships in the design of such systems. The interaction behavior, which is the only perceptual basis for fitting the subject system into its environment and verifying that the subject system instance conforms to that basis, is sometimes lost or difficult to reconstruct from those models.

This standard specifies a Conceptual Metamodel for understanding and describing the causal and affective behavior for a system. While many tabular, graphical, and animated representations of this information can be used for human presentation and interaction within documents, drawings, and tool media, characterizing those representations is not within the scope of this standard.

This standard does not address other models of systems that facilitate the management, development, production, and use of engineered systems. Such models include reliability and availability models, safety models, security models, physical models of form and fit, cost and schedule estimation models, technology readiness models, failure and fault models, manufacturability and production models, and environmental assault and survivability models. It is a system engineering concern to understand the partitioning of system aspects among such models, the relationships between these separate models, and the relationships between these models and the subject system.

1.4 Engineering models and metamodels

Engineering models can be characterized by what elements of the system are selected for description in the model, by the forms and patterns used for model description, and by the kinds of inferences that can be made from the model.² Within the scope of those selected elements, forms, and patterns, the model becomes a representation (to some degree of fidelity) of the actual system. With the addition of appropriate methods of evaluation and prediction, the model can become a useful engineering tool.

The dependability of such a tool requires standardization of modeling forms and patterns, an unambiguous relationship to identifiable system elements, repeatable inferences from the model, and a coherent interpretation of predictions. The set of syntactic and semantic rules that define these things is called a *metamodel*.

Models have a purpose that is served by the choice of descriptive elements in the model, the questions that can be posed to and answered with the model, and the methods by which inferences are made. A model can be expressed in a variety of representations. The choice of representation depends on that representation's capability to represent the things and relationships being modeled, the adequacy of inference mechanisms for extracting information from the model, and the ease of creating the model and extracting inferences.

The purpose for a system behavior model is to represent the net, externally observable behavior of a system. It is thus centered on the interactions of a subject system with its environment. The intended use of this model is to understand the services it provides to its external environment and the obligations of the external environment that enable successful operation of the system. This is a pure black-box specification of a system that is fully distinguished from its environment.

The reference model in this standard is a particular Conceptual Metamodel for the description of system behavior, viewing system behavior as the delivery of application value into an application domain through the aggregate of all system responses to internal and external stimuli.³ It is focused on the development and expression of the total mapping of sequences of stimulus interactions to sequences of response interactions brought about by the operation of a subject system.

This reference model contains a small number of fundamental concepts abstracted directly from the observable instances and patterns of system behavior. When cast into a corresponding Data Metamodel, these concepts and their relationships provide the basis for interpreting a data model of system behavior and for inferring the possible sequences of observable behaviors of the subject system.

A system behavior model focuses on the interactions that occur between elements in the application and the subject system. It models the ways in which the system senses properties and events in the application domain and the ways in which the system causes desired changes in that domain. Generally, a system behavior model does not contain explicit identification of every mapping of possible stimulus sequence to resulting response sequence. Rather, it defines the totality of patterns that encompass all possible occurrences.

System behaviors are a part of *system requirements*, as they state requirements on the system's interaction with the application domain. System requirements could also be called *product requirements*, *system specifications*, or *product specifications*.

Things to be modeled in the system's interaction with the application domain include the following:

- Selection of relevant external interactions that are shared with the application domain
- Selection of temporal granularity (modeling in “point time” or in “continuous time”)

² The system modeling perspective for a tool is described in 5.2 of IEEE Std 1175.2™-2006 [B5].

³ This interpretation of behavior is adapted from definition 2.b for *behavior* found in Dictionary.com Unabridged (v 1.1).

- Representation of selected interactions in terms of physical, logical, and temporal characteristics
- Causal relationships between occurrences of stimulus interactions and response interactions
- Affective relationships between input and output interaction contents
- Affective relationships sequences of behaviors

Prior to the existence of a system, these are behavior specifications to be met. Following development of a system that purports to comply with the model, these provide one basis for testing and accepting the system (through verification).

A common failure in system behavior modeling is the assumption that the only stimulus and input interactions that occur are those that are desired. Neither the model nor the system will be robust (predictable) if this assumption is invalid.

Annex B provides a brief description of several other engineering models commonly used in system development and describes their relationship with the system behavior model. The following models are addressed:

- Stakeholder requirements model
- System use model
- System mechanism model
- System implementation model
- System verification model
- System validation model

1.5 Audience

The audience for this standard consists of systems analysts, system designers, system modelers, product specifiers, product architects, product designers, product design verifiers, specification validators, software testers, simulation developers, and specialists in the verification and validation of models and simulations.

1.6 Conformance

The reference model for specifying system behavior is described in a Conceptual Metamodel, as explained in Clause 5. The Conceptual Metamodel requires a Data Metamodel as the tool by which a system modeler can express, document, store, and analyze a behavior model for a particular system. A Data Metamodel enables modelers to construct a model of some system's behavior in accord with the Conceptual Metamodel.

The various observables, relationships, and fact types identified in the Conceptual Metamodel (see 5.1) constitute the core requirements to be satisfied by a Data Metamodel. That is, any proposed Data Metamodel shall provide an explicit representation of the Conceptual Metamodel elements so that model facts can be recorded and manipulated.

A Data Metamodel conforming to this Conceptual Metamodel shall provide representations and structures of data elements, data values, and expressions for recording and communicating the conceptual elements identified in Clause 7. It shall also provide model verification rules to be applied over populated data structures for evaluating completeness and consistency of a particular system behavior model with respect to the Conceptual Metamodel.

1.7 Organization of this standard

This document contains seven clauses and two annexes. The following annotations provide a roadmap to the contents:

Clause 1 describes the purpose and scope of this standard and also provides conformance criteria. It distinguishes system behavior models from other types of engineering models.

Clause 2 lists the references necessary for implementing this standard.

Clause 3 provides definitions of terms.

Clause 4 identifies the requirements to be met by a reference model for system behavior, and some of the capabilities it would provide.

Clause 5 distinguishes the Conceptual Metamodel defined in this standard from the Data Metamodel defined elsewhere, and it describes the complementary roles of each in providing the reference model for specifying system behavior.

Clause 6 identifies and elaborates three general principles on which the Conceptual Metamodel worldview in this standard is based and explains the nature and role of each element in the Conceptual Metamodel. It provides an inference concept for predicting observable unit behavior. It also provides a composition structure for relating behaviors of an assembly of units to behaviors of the units composing that assembly.

Clause 7 summarizes the elements and relationships within the Conceptual Metamodel for system behavior. This clause also describes how this standard is to be applied.

Annex A provides informative references.

Annex B provides a brief description of several other engineering models.

2. Normative references

The following referenced documents are indispensable for the application of this document (i.e., they must be understood and used, so each referenced document is cited in text and its relationship to this document is explained). For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments or corrigenda) applies.

The following glossary standard shall be used, when applicable, for tool or software terms not defined in this standard.

IEEE Std 610.12TM, IEEE Standard Glossary of Software Engineering Terminology.^{4, 5}

⁴ IEEE publications are available from the Institute of Electrical and Electronics Engineers, 445 Hoes Lane, Piscataway, NJ 08854-1331, USA (<http://standards.ieee.org/>).

⁵ The standards or products referred to in this clause are trademarks of the Institute of Electrical and Electronics Engineers, Inc.

3. Special terms

For the purposes of this document, the following terms and definitions apply. *The Authoritative Dictionary of IEEE Standards Terms* [B2] or IEEE Std 610.12⁶ should be referenced for terms not defined in this subclause.

Other relevant terms defined in Clause 2 of IEEE Std 610.12 include: **CASE**, **interface**, **semantics**, **software tool**, and **syntax**.

The following additional definitions describe specific terms as used within the context of this standard.

3.1 action: A statement of causal and affective relationships in a behavior model linking particular stimulus interactions to particular response interactions and changes within a unit under a certain set of conditions on a unit's lifeline.

3.2 affective relationship: A functional dependency between prior input interaction occurrences and later output interaction occurrences in a behavior pattern.

3.3 assembly: A collection of units in which ports on different units are compatibly interconnected so behaviors of one unit can influence behaviors of another unit by means of interactions occurring through interconnected ports.

NOTE—An *assembly* may be regarded as behaviorally equivalent with a unit. The ports of the unit are just those ports of the assembly units that are not internally interconnected. The behavior of the unit is a composition of the shared behaviors of the assembly units.⁷

3.4 available interaction: An interaction that is allowed by one unit and controlled by another unit.

3.5 behavior: The aggregate of all unit responses to internal and external stimuli. *See also:* **behavior pattern**.

3.6 behavior pattern: A relationship that maps a sequence of stimulus interactions to a sequence of response interactions. *See also:* **affective relationship**, **causal relationship**.

3.7 behavior state: A state which represents the partitioning of all unit behavior patterns into (possibly overlapping) subsets of behavior patterns that can be elicited from a unit at a particular point in time.

3.8 boundary: A collection of all interface ports between a unit and its environment, characterized by the input and output interactions that it allows.

3.9 causal relationship: An existence dependency between the stimulus of a behavior and the various responses that are caused to occur.

3.10 condition: A logical predicate involving one or more behavior model elements.

NOTE—A *condition* predicate may be a statement about some characteristic of a model element, a statement about a relationship between model elements, or a logical combination of such statements.

⁶ Information on references can be found in Clause 2.

⁷ Notes in text, tables, and figures of a standard are given for information only and do not contain requirements needed to implement the standard.

3.11 coordinated interactions: A pair of interactions in which the first interaction is generated by a unit as a stimulus to some other unit, in expectation that a second interaction will be generated as a response from that other unit (or even a third unit) to benefit the initiating unit for a particular purpose. *See also:* **obligation**.

NOTE—One unit controls the initiation of this exchange, subordinating the other unit(s) to itself. Such a subordinating exchange is possible even if the subordinated unit generated the stimulus that triggered the initiating unit's behavior.

3.12 event: A singular moment in time at which some perceptible phenomenological change (energy, matter, or information) occurs at the port of a unit.

3.13 guarantee condition: A statement of the constraints that will be satisfied by output interaction occurrences and the next property state as a result of the occurrence of a particular behavior pattern.

3.14 guard condition: A statement of the circumstances (input interaction occurrences and property state) that allow a stimulus to cause the occurrence of a particular behavior pattern.

3.15 interaction: An identity of phenomena existing over some period of time at the interface between two units, caused by one unit and affecting the other unit. *See also:* **available interaction, coordinated interactions, response, stimulus**.

NOTE—The identity is expressed in relevant phenomenological terms. Generally, an interaction identity can be categorized as energy transfer, matter transfer, or information transfer.

3.16 interaction alias: A relationship in a build specification that matches an interaction instance in one port with an interaction instance in another port.

NOTE—The *interaction alias* is used to indicate identity or compatibility of matching interactions at two ports of a port couple or a port alias. In a port alias, both interactions have the same direction. In a port coupling, the interactions have opposite directions. This provides an element of “model bookkeeping” necessary for permitting models developed independently for different units to be integrated into a unified composite model.

3.17 lifeline: Any historical sequence of behaviors through which a unit is manipulated by external stimuli.

3.18 obligation: A collaboration pattern of interaction between two units used when one unit is not independently capable of providing its service behavior without the assistance of the other unit.

3.19 port: A n interface on a unit's boundary that allows a subset of the interactions that can be exchanged with another unit.

NOTE—Interactions at a *port* have a common phenomenological basis by which they are exchanged. Occurrences of interactions in one direction are serialized at a port.

3.20 port alias: a replacement relationship in a build specification that identifies a port of one unit with a port of a subunit and indicates that interactions at the two ports can be paired identically or compatibly.

NOTE—A *port alias* is used to indicate identity or compatibility of two ports at different levels of assembly in which the port in a lower level unit of assembly serves as a realization of a port a higher level unit of assembly level. This provides an element of “model bookkeeping” necessary for permitting models developed independently for different units to be integrated into a unified composite model. Compatibility criteria are satisfied by valid pairings.

3.21 port couple: An interconnection relationship in a build specification that identifies a port of one unit with a port of another (structural peer) unit and indicates that interactions output from one are input to the other, and vice versa.

NOTE—A *port couple* defines a one-to-one matching of the two ports defining opposite sides of a particular interface. Ports match identically if all output interaction occurrences at one port match are identical with all input interaction occurrences accepted at the other port. Ports match compatibly if interaction occurrences satisfy specific compatibility criteria.

3.22 property: A measurable phenomenological characteristic of an interaction.

3.23 property state: A state comprising properties associated with a unit that are defined by functional dependence on the properties of past interaction occurrences and that affect the properties of future output interaction occurrences.

NOTE—A particular *property state* of the unit is defined at a point in time by the values of the internal state properties.

3.24 response: A reaction of the unit to a specific stimulus, depending on the current conditions on the unit's lifeline.

NOTE—A *response* might be the occurrence of internally controlled interaction(s) affecting the environment (external response), or it might be the occurrence of an internally controlled change to the unit's state (internal response).

3.25 state: A characteristic of a unit (usually composite in structure and multi-dimensional) expressing the cumulative effect of all previous unit interaction occurrences, in terms of which the delayed affective relationships of the unit can be evaluated. *See also:* **behavior state, property state.**

NOTE—A *state* is a shorthand representation for the unit's interaction occurrence history. Ascribing "state" to a unit implies that it has a capability to modify future behaviors as a result of past interactions with its environment.

3.26 state invariant condition: A statement of constraints or relations that can be used to distinguish a subset of particular property states that all satisfy the condition.

3.27 stimulus: Whatever causes a unit to exhibit an occurrence of a behavior pattern in the unit's repertoire; something causing or regarded as causing a response.

NOTE—A *stimulus* might be the occurrence of an externally controlled interaction affecting the unit (external stimulus), or it might be the occurrence of an internally controlled event (internal stimulus).

3.28 subject unit: A unit that is the subject of a behavior modeling.

NOTE—A *subject unit* is segregated from a collection of units that constitute the environment in which its behaviors are solicited and exhibited.

3.29 unit: A distinguishable architectural unit with individual identity, boundary, and behavior that is observable through interactions with other such units.

NOTE—A *unit* may be a system, a subsystem, a system element, an assembly of components, or a primitive component in a system.

4. System behavior model requirements

As defined in this standard, the reference model for system behavior satisfies the following requirements:

- Application scope: the reference model is capable of specifying behaviors for a useful range of interesting systems.
- Declarative form: the reference model is restricted to the specification of causal and affective behavior effects, without the specification of mechanisms for producing that behavior.

- Predictive ability: the reference model supports the inference of required and allowed behaviors from the declarative patterns in the model.
- Test ability: the reference model expresses all behavior in terms of observable interactions.
- Life cycle role: the reference model serves specific life cycle purposes in the process areas of requirements definition, technical design, and verification.
- Common interpretation: the reference model provides a context of interpretation that allows multiple users (and tools) to make similar inferences for questions about the subject system behavior.

This clause addresses the needs and constraints associated with each of these requirements.

4.1 Application scope

The reference model for causal and affective behavior presented in this standard describes reactive systems that exchange information, energy, and matter with other systems in their operational environment. The systems of concern in this standard have physical realizations and exhibit repeatable, or at least predictable, behaviors that are used to bring about specific intended effects in their environment. Such systems provide value into an application domain through the exchange of interactions between the subject system and other hardware, software, or bioware systems.

While modern systems are realized with considerable complexity, their specification descriptions are often biased to model some system features in far more detail than others. Risk considerations drive the focus onto the primary value-producing areas of complexity in the system. Furthermore, the adoption of *de facto* design and implementation standards could minimize the level of concern for other feature areas. For real-world systems, a *most-like* determination suggests the use of particular specification models with particular focus areas.

- The value of information management systems tends to focus on the organization and storage of their data content and the business and editing rules for preventing the acceptance of invalid information. The external interactions of these systems are stylized through the definition of screen sets and report sets.
- The value of control systems tends to focus on the detection of complex structures of input signals, the interpretation of certain inputs within the sequence of previous inputs, and the timed or timely generation of complex structures of output signals. The data content of these systems is often of the form of parameter sets and logs.
- The value of simulation systems tends to focus on the accurate rendition of output data streams with respect to specified model parameters and initial conditions.
- The value of desktop office tools tends to focus on the construction and manipulation of virtual objects (e.g., lists, documents, diagrams), with storage, recall, and transfer to various output media.
- The value of networks tends to focus on rapid, uncorrupted point-to-point information transfer through complex node and link topologies, with extensive messaging choreography to manage end-to-end reliability and protection.

While focused and detailed specification models address specific features, they usually omit other features altogether. Specification frameworks have emerged to identify the more inclusive set of issues that should be included in specification. The purpose of the Conceptual Metamodel defined in this standard is to express causal and affective behavior and compositions of such behavior in a model that integrates all observable operational features of a system into one behavior specification. In practical use, this specification should be complemented with the additional detail available in other specialized models, but it provides the framework in which those additional details fit and have an observable meaning.

The stakeholder requirements identify what accomplishments are to be wrought in the application domain through the operation of the subject system. The system behavior model captures the pre-design analyses and decisions about how the subject system will interact with that application domain, and what rules need be satisfied by those interactions so that it will have its intended effect. Developing and recording the description of this causal and affective behavior provides several value-added opportunities for assurance.

- Constructing a causal and affective behavior model for a subject system encourages perception of the subject system in its physical environment. Often, important design bases are understood earlier in a life cycle when accounting for all interaction ports, multiplicities of architectural units, potential interactions and functional concurrency, and timing and duration expectations.
- Combining a detailed causal and affective behavior specification with assumptions of the environment, it is possible to investigate whether the specified subject system can be expected to actually satisfy the stakeholder requirements.
- Comparing an assembly of component causal and affective behaviors against the net causal and affective behavior of the assembly provides evidence for soundness of decomposition; it can also elucidate assumptions that need confirmation and ambiguities that need resolution.
- Organizing the required observables of system operation into a causal and affective behavior model provides a direct basis for inferring interaction streams to stimulate, sensitize, probe, and observe all the expected behaviors of the subject system. This allows the development of test cases and test suites in which test cases are traceable by construction to the specifications. The behavior specification can provide a direct, countable reference base for arguing specification-based adequacy of the test suite.

The normative content in this standard is neutral with respect to the scale of the object whose behavior and structure is modeled. Rather than using terms such as *system*, *subsystem*, or *component*, the object of attention is abstracted to *architectural unit*, or simply *unit*. A unit is always embedded in some operational context or environment and exhibits behavior through its interactions with that environment. A unit also may have a composition in terms of interconnected units whose individual behaviors aggregate to some extent as the behavior of the assembly.

The reference model is focused on the specification of a unit in isolation, with the identification of all the behavior patterns by which it delivers its capabilities into a context of other units.

4.2 Declarative form

A behavior specification model identifies the exchanges of matter, energy, and information between the subject system and its external environment, and it specifies the rules and relationships among these exchanges so that system behavior satisfies the stakeholder requirements. (Practically, system behavior can intentionally satisfy only the stakeholder requirements as understood and documented by the stakeholders.) It is inappropriate for the behavior specification model to describe a mechanism for these behaviors, but it should identify what behaviors the subject system delivers into the application domain through specified interactions, relationships, and properties. Thus, the behavior modeling approach should be declarative. Many different mechanisms may be designed to produce the specified behaviors, differing in their satisfaction of any number of other design criteria. The chosen mechanisms would themselves be described and characterized by a variety of mechanism models. Design constraints are not part of a behavior model.

From the perspective of the application domain, the system behavior model is imperative. Its specified interactions and rules of behavior provide a means by which the requirements in that domain can be made true. With respect to the subject system itself, however, the behavior model defines and describes only what is visible in the application domain. It does not describe any mechanism for generating that behavior, only how to recognize the behaviors.

The behavior model is to be restrictively interpreted. It is a model for which some scope of behaviors is selected for inclusion. For those selected behaviors, it is a description of what is controllable and

observable in the subject system. A permissive interpretation, on the other hand, would assume that whatever is not prohibited is allowed; such a specification would describe only constraints. A permissive interpretation has an unbounded scope of verification and cannot be the basis for engineering assurance in the development of the subject system.

The behavior model is declarative in that its meaning is inferred from the elements in that model, their definitions, and their relationships. There is no meaning implied by any sequencing of the model content.

4.3 Predictive ability

A system interacts dynamically with some part of the external world in which it exists to effect changes in that external world. Let $E(t_0)$ and $S(t_0)$ describe the environment and the system, respectively, at time t_0 .

The environment E includes objects that exist even in the absence of the system S , as well as objects that might exist because of the system. From a stakeholder requirements viewpoint, there is some collection of environment objects that includes those objects that interact directly with system S , as well as other objects that either indirectly affect S or are indirectly affected by S . From a behavior specification viewpoint, all that is perceivable in the environment E is the set of interactions that pass between E and S .

As interactions between E and S occur over a period of time, both the environment and the system change, from $E(t_0)$ to $E(t_1)$ and from $S(t_0)$ to $S(t_1)$, respectively. Figure 1 illustrates the synchronous evolution of environment and system on parallel lifelines as they interact through a collection of shared phenomena P .

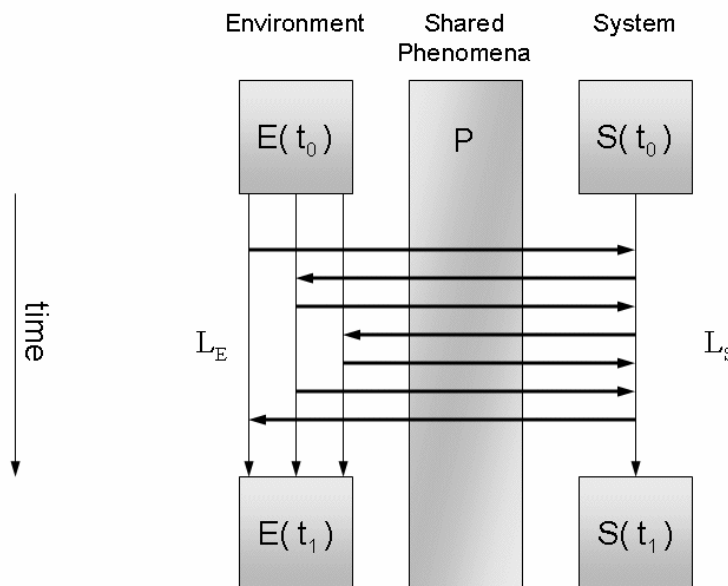


Figure 1—Representing temporal transformations in the world

The environment E evolves under the influence of the interactions from S in response to the interactions issued from E . Denoting the evolutionary lifeline path of E by L_E , the situation can be expressed as an evolutionary transformation of $E(t_0)$ to $E(t_1)$ (as denoted in Figure 1):

$$L_E : E(t_0) \rightarrow E(t_1)$$

(The transformation L_E is actually a composite of transforms across the lifelines of the objects in E .) A stakeholder requirements model aims precisely at identifying and describing those objects included in E

and at defining particular lifeline trajectory points such as $E(t_0)$ that exist there. The system use model then aims at characterizing some of the transformations L_E that could be used to connect those identifiable points.

The system S evolves under the influence of particular sequences of particular interactions from E . Denoting the evolutionary lifeline path of S by L_S , the situation can be expressed as an evolutionary transformation of $S(t_0)$ to $S(t_1)$ (also denoted in Figure 1):

$$L_S : S(t_0) \rightarrow S(t_1)$$

The behavior model for the system aims at identifying and describing those interactions that are possible through phenomena in P and at defining the required cause-effect relationships that can be used to steer the evolution of $E(t_0)$ to $E(t_1)$. Any particular L_S for the system results from a composition of particular behaviors in particular sequences. There is considerable freedom in the specification of both the L_S trajectories and the basic behaviors in them, even when there are constraints on the L_E trajectories. Such freedom should be exploited to optimize other behavior-related properties of the system, to include reliability, safety, and testability.

Figure 1 shows that a specific L_E and L_S are the same, since both are expressed as the same sequence of interactions. However, the set of all L_E is not the same as the set of all L_S . In an idealized world, the two sets would be the same. In a real world, they cannot be the same. The difference is due to the difference between what is desired and what is possible in the possible interactions between E and S . The set of possible L_S differs from the set of desired L_E due to possible “errors” in the interactions and interaction sequences emanating from E . (These possibilities include erroneous interaction content, erroneous interaction timing, erroneous ordering of interactions, and missing interactions.) A comprehensive analysis would consider both the desired set of L_E as well as a “corrupted” set of L_E to assemble the complete set of interactions and sequences that could be expected in the set of L_S .

NOTE 1—Interaction errors and corruption are real world issues. Expanded functionality is required in environment and system objects to discriminate, reject, respond, and recover from their presence. The behavior specification should not assume the absence of these effects, so the metamodel must provide for representing these issues.

Similarly, the actual complete set of L_E differs from the set of desired L_S due to possible errors in the interactions and interaction sequences emanating from S . Thus, objects in E should be prepared to evolve under the possible, even if unexpected, interactions and sequences from S .

In many interaction sequences, there is a kind of dance in which an environment object and the system alternate turns in generating interactions to one another. This has the effect of synchronizing the two unit lifelines as both units experience the interactions concurrently with the other at specific points in time. The time synchronization between the two is private, and it provides a simple untimed protocol for providing system effects into the environment. This is inadequate when interactions are required at specific times or on specific interval measures, or when the two units are required to synchronize with other environment object lifelines. More generally, some form of common time measurement is necessary.

It is assumed that a notion of time, continuous to some granularity, can be ascribed to the system as it evolves through its lifeline. Units of time can be marked or noted and the interval of time passage can be determined by various means. Some external interaction behaviors usually exist for synchronizing the unit’s timekeeping capability with time points in the unit’s environment. In addition, the system might have nonbehavioral requirements for stability and lifetime of timing capability.

From a modeling perspective, this assumption allows the notion of temporally synchronized traversal of lifelines by environment objects and by the system.

It is a complex task to analyze the set of all required L_E that fulfill the stakeholder requirements in E , and it is a different complex task to determine for S the complete collection of causal and affective behaviors from which the corresponding L_S can be composed.

System behavior is the aggregate of all system responses to internal and external stimuli. It is a mapping of sequences of stimulus interactions to sequences of response interactions. The goal of the reference behavior model is to enable the description of all causal and affective behaviors.

NOTE 2—The practicality of defining a software program as an explicit mapping of input sequences to output sequences with no local variables (data state) was demonstrated in the Lucid programming language (Wadge and Ashcroft [B8]).

To describe a particular system behavior requires the following:

- The description of a particular $S(t_0)$
- The representation of a particular L_S
- The prediction of $S(t_1)$ from $S(t_0)$ and L_S

Correspondingly, the reference model must provide the following:

- A means to express a model $M(t_0)$ corresponding with $S(t_0)$
- A means for composing declarative cause-effect patterns into an evolutionary path L_M in M that corresponds with an evolutionary path L_S in S
- An inference mechanism that exhibits the following relationship:

$$L_M : M(t_0) \rightarrow M(t_1)$$

Figure 2 illustrates this correspondence relationship between the predictive capability of the reference model M and a system S .

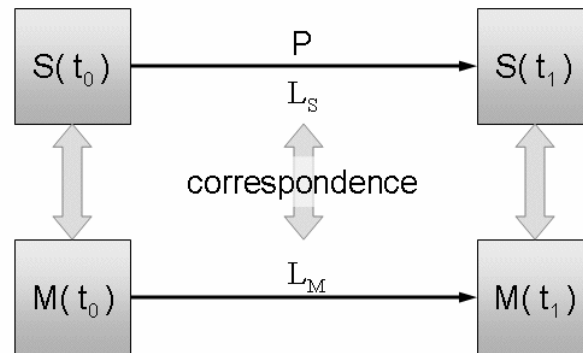


Figure 2—Correspondence between model behavior and system behavior

The role of this standard in developing system behavior models and in establishing the correspondence of such models with real or envisioned systems is addressed in Clause 5.

The representation of all cause-effect patterns of the system and a mechanism for inferring allowed evolutionary paths L_S is central to the reference model. These patterns are described as relationships between the interactions that occur through the shared phenomena P . These interactions are observables in E as they originate or terminate in the environment. The sequences and content of these interactions drive any particular evolutionary path. Clause 6 provides a more detailed description of the observables that have representations in M .

NOTE 3—The reference model in this standard is useful in practical ways, but it is not intended to be theoretically comprehensive. It is not required to be adequate for establishing global assertions about all possible behavior sequences.

Establishing a predictive capability requires that the behavior model must provide for representing the system at a particular time. It is an essential modeling problem to relate elements of the system S with elements in a model M . One purpose of the reference model definition is to identify and specify classes of elements in S and the corresponding classes of elements in M that represent them.

4.4 Test creation ability

Testing a system's behavior requires establishing known initial conditions $S(t_0)$, providing a specified sequence of stimuli and other inputs, observing the resulting system responses, and finally confirming that the system satisfies expected final conditions $S(t_1)$. This is a classical black-box viewpoint (with the advantage that it is applicable at multiple scales in extent and in time). A suite of such tests provides evidence that an implemented system realizes all the behaviors specified for meeting the stakeholder requirements. (These tests do not provide evidence that the stakeholder requirements themselves are satisfied by use of the system.)

Effective testing has two needs. The first need is for a surrogate to replace the environment E that is capable of generating stimulus interactions in P for S to use. The surrogate should also be capable of capturing response interactions in P that S produces. [In some cases, the surrogate might have additional capabilities for establishing the initial conditions $S(t_0)$ and for determining the final conditions $S(t_1)$.]

The second need is for test cases that specify appropriate initial conditions, sequences of stimuli to be applied, expected responses to be produced, and final conditions of the system. Based on the degree of refinement of a system behavior model, a sophisticated sampling strategy might be used to create an extensive set of such test cases. This approach to test case design is frequently called *model-based testing*. When the behavior model is part of the specification of the system to be realized, this may also be called *specification-based testing*.

The fundamental idea is to derive tests from the behavior model and to apply them against the realized system. There is no assumption about the structures and elements used to realize the system. The focus, rather, is to determine whether the realized system behaviors are equivalent with those behaviors intended for use in composing L_S , and ultimately, L_E . The tests are normally at the granularity of individual behaviors or small collections of individual behaviors. The test granularity should be commensurate with the controllability and observability properties of the system.

- *Controllability* refers to what the subject system can be directed to do by its stimuli (this might include control of its internal state as well as the control of its responses).
- *Observability* refers to what can be inferred about the subject system from observations of its responses.

Figure 3 illustrates this use of model-based test case specifications as system tests.

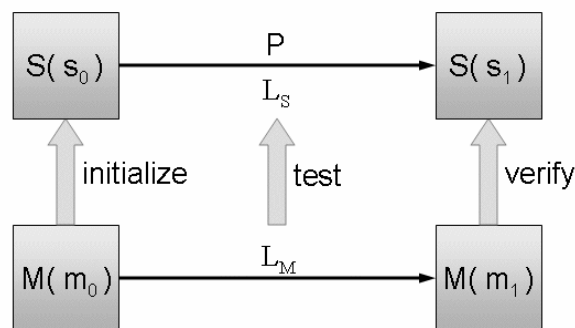


Figure 3—Model-based testing relationships

Initially, s_0 is the state of the system corresponding to the initial state of the model m_0 . Similarly, s_1 is the final state of the system corresponding to the final state of the model m_1 . The sequence of input and output interactions in L_S are defined in accord with those interaction sequences in L_M that move m_0 to m_1 in the model.

The use of the behavior model in this context depends explicitly on the predictive ability as required in 4.3. That ability implies that the behavior model contains all the information needed for the generation of representative sequences of individual behavior occurrences.

4.5 Life cycle roles

The declarative statements in the system behavior model M provide a “behavior specification” for the subject system, in that they provide a set of assertions that are to be true of the behaviors exhibited by that implementation.

The system behavior specification represents the earliest design decisions for a system, namely, decisions about how the system can interact with the application domain to bring into being the results described in the stakeholder requirements.

Prior to the implementation of a system S , that behavior specification can be analyzed for satisfaction of the stakeholder requirements in the expected usage contexts. As it identifies all behaviors to be supplied by system, it is the bridge between the stakeholder requirements and a technical solution. This specification might be validated against stakeholder requirements models and system usage models for the subject system.

During development, the system design can be analyzed to ensure that all specified behaviors can be produced and to understand the role of the various system design elements in producing them.

The system behavior specification also provides one basis for evaluating an implemented system that purports to comply with it. The behavior specification provides an explicit coverage target for final verification as it provides a list of behaviors expressed directly in testable (observable) terms.

- First, the specification can serve as a constructive test design basis for building a comprehensive suite of test situations, for parts, assemblies, and the complete system. The specification-based testing objective is to provide evidence of completeness and correctness over an arguable sampling of all application domain interaction sequences and behaviors. (In contrast, exploratory and context-based testing, following most probable error guidance, can be highly valuable during development for the detection of design defects, especially when product specifications are notional or incomplete.)
- Second, the model inference mechanisms can serve the development of expected system responses to those situations (test oracle). Subjecting the system to these tests and comparing its actual responses with the expected responses demonstrates explicitly the degree of compliance of the implemented system behavior with the intended system behavior.

The design process develops a physical and functional hierarchy, with specified interfaces between design elements. As at the system level, behavior models provide appropriate “behavior specifications” for parts as well as assemblies in the design, again providing assertions that are to be true of the behaviors exhibited by those design elements. These specifications comprise a portion of the specification tree that expresses the system design, and they provide a basis for behavior evaluation of realized parts and assemblies during system construction. (A more detailed discussion of the relevant process areas is found in Technical Report CMU/SEI-2002-TR-011 [B7].)

Following acceptance of parts, assemblies, and the system, these behavior specifications provide essential information to the collection of system reference and maintenance manuals.

4.6 Common interpretation

A behavior model that has an ambiguous interpretation with respect to a subject system is worse than no model at all. What is required is a common basis for model interpretation among people and tools that need to share an understanding of the subject system. Having such a “shared interpretation” means that each person or tool can make similar inferences about the subject system from the descriptive content of the model and the standard inference mechanism. Similarity between such inferences is the test for demonstrating a shared interpretation.

As described in 4.3, the reference model in this standard expresses the cause-effect patterns of system behavior as relationships between the interactions that can occur between the system and its environment through the shared phenomena *P*. Because these interactions are observables in the system’s environment, they provide an unambiguous vocabulary of “physical” referents with which to interpret the model directly into observables. The descriptions of observables in the system behavior model include the following:

- Detailed descriptions of the controlling and observable interactions
- Identification of content and property relationship patterns among the controlling and observable interactions
- Causal and affective relationship patterns between controlling and observable interactions

Not all reference model concepts are directly observable. There are also *indirect observables*. An indirect observable may be called an *inferred* or a *derived* observable. An indirect observable is not knowable by actual observation, but could be determined from other direct observations. Inference without direct observation may be regarded as imputed reality. The most common imputed reality is called the *state* of the unit. Such imputations are accounting shorthand for the history of interaction occurrences that affect the unit’s direct observables. Such bookkeeping practice localizes the effect of historically distributed interaction occurrences into a single current referent. (See 6.3.7 and 6.3.8.)

Clause 6 elaborates the classes of model observables and their basic characteristics.

5. System behavior metamodels

5.1 Developing the Conceptual and Data Metamodels for behavior modeling

An overview of the use of the system behavior reference model for the modeling of a subject system, real or envisioned, is illustrated in Figure 4.