IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

Sponsor

Software Engineering Standards Committee
of the
IEEE Computer Society

Approved 21 September 2000

IEEE-SA Standards Board

Abstract: This recommended practice addresses the activities of the creation, analysis, and sustainment of architectures of software-intensive systems, and the recording of such architectures in terms of architectural descriptions. A conceptual framework for architectural description is established. The content of an architectural description is defined. Annexes provide the rationale for key concepts and terminology, the relationships to other standards, and examples of usage.
Keywords: architectural description, architecture, software-intensive system, stakeholder concerns, system stakeholder, view, viewpoint

---

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

Print: ISBN 0-7381-2518-0 SH94869
PDF: ISBN 0-7381-2519-9 SS94869

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.

IEEE is the sole entity that may authorize the use of certification marks, trademarks, or other designations to indicate compliance with the materials set forth herein.

Introduction

(This introduction is not part of IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*.)

It has long been recognized that "architecture" has a strong influence over the life cycle of a system. In the past, hardware-related architectural aspects were dominant, whereas software-related architectural integrity, when it existed, often was to be sacrificed first in the course of system development. Today, software-intensive systems are pervasive. The cost of software development and the increasing complexity of software systems have changed the relative balance. Software technology is maturing rapidly. The practice of system development can benefit greatly from adherence to architectural precepts.

However, the concepts of architecture have not been consistently defined and applied within the life cycle of software-intensive systems. Despite significant industrial and research activity in this area, there is no single, accepted framework for codifying architectural thinking, thereby facilitating the common application and evolution of available and emerging architectural practices.

The IEEE Architecture Planning Group (APG) was formed in August 1995 to address this need. The APG was chartered by the IEEE Software Engineering Standards Committee (SESC) to set a direction for incorporating architectural thinking into IEEE standards. The result of the APG's deliberations was to recommend an IEEE activity with the following goals:

— To define useful terms, principles and guidelines for the consistent application of architectural precepts to systems throughout their life cycle

— To elaborate architectural precepts and their anticipated benefits for software products, systems, and aggregated systems (i.e., "systems of systems")

— To provide a framework for the colilection and consideration of architectural attributes and related information for use in IEEE standards

— To provide a useful road map for the incorporation of architectural precepts in the generation, revision, and application of IEEE standards

In April 1996 SESC created the Architecture Working Group (AWG) to implement those recommendations.

Participants

At the time this recommended practice was completed, the Architecture Working Group had the following membership.

Basil Sherlund, *Chair*
Ronald L. Wade, *Vice Chair*
David Emery, *Vice Chair for Liaison*
Rich Hilliard, *Secretary and Editor*

Frank C. Belz
S. Jeromy Carriere
Mark Dehlin
Verlynda Dobbs
Nancy Eickelmann
Walter J. Ellis
William Gess
Vladan V. Jovanovic
Judith S. Kerner
Dwayne L. Knirk
Ron Kohl
Alexei E. Kotov
Philippe Kruchten
Simon Liu
Mark W. Maier
Bill McMullen
Fatima Mili
Glenn Plonk
Peter T. Poon
Dave Rayford
Ann Reedy
Ira Sachs
Thomas F. Saunders
M. Wayne Shiveley
Louise Tamres

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

Edward A. Addy
Mario R. Barbacci
Edward E. Bartlett
Leo Beltracchi
Frank C. Belz
Richard E. Biehl
Edward R. Byrne
Lawrence Catchpole
Leslie Chambers
Keith Chow
Antonio M. Cicu
Sylvain Clermont
Rosemary Coleman
Darrell Cooksey
Paul R. Croll
Gregory T. Daich
Bostjan K. Derganc
Perry R. DeWeese
Verlynda Dobbs
Franz D. Engelmann
William Eventoff
Jonathan H. Fairclough
John W. Fendrich
John H. Fowler
Eva Freund
Andrew Gabb
Juan Garbajosa-Sopena
Hiranmay Ghosh
Marilyn Ginsberg-Finner
Julio Gonzalez-Sanz
Lewis Gray
L. M. Gunther
John Harauz
Herbert Hecht
Mark Henley
John W. Horch
George Jackelen
Frank V. Jorgensen
Vladan V. Jovanovic
William S. Junk
Judith S. Kerner
Thomas M. Kurihara
Helmut Kurzdorfer
Kyoung-In Kwon
J. Dennis Lawrence
Thomas B. Lindahl
Jim J. Logan
Henry A. Malec
Tomoo Matsubara
Ian R. McChesney
James W. Moore
Frederick I. Moxley
Francisco Navas Plano
Pavol Navrat
Gerald L. Ourada
Alex Polack
R. Razouk
Annette D. Reilly
Helmut Sandmayr
Frederico Sousa Santos
Robert J. Schaaf
Carl A. Singer
Richard S. Sky
Mitchell W. Smith
Fred J. Strauss
Sandra Swearingen
Toru Takeshita
Richard H. Thayer
Booker Thomas
Leonard L. Tripp
Theodore J. Urbanowicz
Tom Vaiskunas
Ronald L. Wade
Randall K. Walters
Larry L. Wear
Charles J. Wertz
Scott A. Whitmire
Paul R. Work
Natalie C. Yopconka
Geraldine Zimmerman

When the IEEE-SA Standards Board approved this standard on 21 September 2000, it had the following membership:

Donald N. Heirman, *Chair*
James T. Carlo, *Vice Chair*
Judith Gorman, *Secretary*

Satish K. Aggarwal
Mark D. Bowman
Gary R. Engmann
Harold E. Epstein
H. Landis Floyd
Jay Forster*
Howard M. Frazier
Ruben D. Garzon
James H. Gurney
Richard J. Holleman
Lowell G. Johnson
Robert J. Kennelly
Joseph L. Koepfinger*
Peter H. Lips
L. Bruce McClung
Daleep C. Mohla
James W. Moore
Robert F. Munzner
Ronald C. Petersen
Gerald H. Peterson
John B. Posey
Gary S. Robinson
Akio Tojo
Donald W. Zipse

*Member Emeritus

Also included is the following nonvoting IEEE-SA Standards Board liaison:

Alan Cookson, *NIST Representative*
Donald R. Volzka, *TAB Representative*

Noelle D. Humenick
IEEE Standards Project Editor

Contents

IEEE Recommended Practice for Architectural Description of Software-Intensive Systems

1. Overview

1.1 Scope

This recommended practice addresses the architectural description of software-intensive systems. A *software-intensive system* is any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole.

The scope of this recommended practice encompasses those products of system development that capture architectural information. This includes architectural descriptions that are used for the following:

a) Expression of the system and its evolution
b) Communication among the system stakeholders
c) Evaluation and comparison of architectures in a consistent manner
d) Planning, managing, and executing the activities of system development
e) Expression of the persistent characteristics and supporting principles of a system to guide acceptable change
f) Verification of a system implementation's compliance with an architectural description
g) Recording contributions to the body of knowledge of software-intensive systems architecture

1.2 Purpose

The purpose of this recommended practice is to facilitate the expression and communication of architectures and thereby lay a foundation for quality and cost gains through standardization of elements and practices for architectural description.

Despite significant efforts to improve engineering practices and technologies, software-intensive systems continue to present formidable risks and difficulties in their design, construction, deployment, and evolution. Recent attempts to address these difficulties have focused on the earliest period of design decision-making and evaluation, increasingly referred to as the *architectural level* of system development. The phrases *archi-*

tectural level and *architecture* are widely, if imprecisely, used. Their use reflects acceptance of an architectural metaphor in the analysis and development of software-intensive systems. A key premise of this metaphor is that important decisions may be made early in system development in a manner similar to the early decision-making found in the development of civil architecture projects.

Many innovations are resulting from this attention to the architectural level, among them architectural description languages and associated tools and environments; architectural frameworks, models, and patterns; and techniques for architectural analysis, evaluation, and architecture-based reuse. While these efforts differ considerably in important aspects, sufficient commonality exists to warrant the development of a recommended practice to codify their common elements.

These innovations are occurring, and maturing, rapidly within many research and application communities, and they reflect differing interests, influences, insights, and intentions. There is a general consensus on the importance of the *architectural level of systems development*, and that that level consists of early decision-making about overall design structure, goals, requirements, and development strategies. However, there has not yet emerged any reliable consensus on a precise definition of a system's *architecture*, i.e., how it should be described, what uses such a description may serve, or where and when it should be defined. The boundaries and relationships between architectural trends and practices, and other practices; and between architectural technology and other technology, are not yet widely recognized.

In such situations, progress often depends on mediating influences. Potential adopters of architectural practices and technology need a frame of reference within which to address implementation and adoption decisions. Technology developers need a frame of reference within which to communicate the motivating concepts of their technology, and to accumulate and appreciate feedback from early adoption.

To these ends, this recommended practice is intended to reflect generally accepted trends in practices for architectural description and to provide a technical framework for further evolution in this area. Furthermore, it establishes a conceptual framework of concepts and terms of reference within which future developments in system architectural technology can be deployed. This recommended practice codifies those elements on which there is consensus; specifically the use of multiple views, reusable specifications for models within views, and the relation of architecture to system context.

1.3 Intended users

The principal class of users for this recommended practice comprises stakeholders in system development and evolution, including the following:

- Those that use, own, and acquire the system (users, operators, and acquirers, or *clients*)
- Those that develop, describe, and document architectures (architects)
- Those that develop, deliver, and maintain the system (architects, designers, programmers, maintainers, testers, domain engineers, quality assurance staff, configuration management staff, suppliers, and project managers or *developers*)
- Those who oversee and evaluate systems and their development (chief information officers, auditors, and independent assessors)

A secondary class of users of this recommended practice comprises those involved in the enterprise-wide infrastructure activities that span multiple system developments, including methodologists, process and process-improvement engineers, researchers, producers of standards, tool builders, and trainers.

1.4 Conformance to this recommended practice

An architectural description conforms to this recommended practice if that description meets the requirements listed in Clause 5.

2. References

This recommended practice shall be used in conjunction with the following publications. When the following standards are superseded by an approved revision, the revision shall apply.

IEEE Std 610.12–1990, IEEE Standard Glossary of Software Engineering Terminology.[1]

IEEE/EIA Std 12207.0–1996, IEEE/EIA Standard—Industry Implementation of ISO/IEC 12207: 1995, Information Technology—Software life cycle processes.

3. Definitions

For the purposes of this standard, the following terms and definitions apply. *The IEEE Standard Dictionary of Electrical and Electronics Terms* [B2],[2] IEEE Std 610.12–1990, or IEEE/EIA Std 12207.0–1996 should be referenced for terms not defined in this clause.

3.1 acquirer: An organization that procures a system, software product, or software service from a supplier. (The acquirer could be a buyer, customer, owner, user, or purchaser.)

3.2 architect: The person, team, or organization responsible for systems architecture.

3.3 architecting: The activities of defining, documenting, maintaining, improving, and certifying proper implementation of an architecture.

3.4 architectural description (AD): A collection of products to document an architecture.

3.5 architecture: The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.

3.6 life cycle model: A framework containing the processes, activities, and tasks involved in the development, operation, and maintenance of a software product, which spans the life of the system from the definition of its requirements to the termination of its use.

3.7 system: A collection of components organized to accomplish a specific function or set of functions.

3.8 system stakeholder: An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system.

3.9 view: A representation of a whole system from the perspective of a related set of concerns.

---

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

[2]The numbers in brackets correspond to those of the bibliography in Annex A.

3.10 viewpoint: A specification of the conventions for constructing and using a view. A pattern or template from which to develop individual views by establishing the purposes and audience for a view and the techniques for its creation and analysis.

4. Conceptual framework

This clause introduces a *conceptual framework,* or frame of reference, for architectural description. The framework establishes terms and concepts pertaining to the content and use of architectural descriptions. These terms and concepts will be used in subsequent clauses.

4.1 Architectural description in context

For the purposes of this recommended practice, the term *system* encompasses individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest.

A system inhabits an environment. A system's environment can influence that system. The environment, or *context*, determines the setting and circumstances of developmental, operational, political, and other influences upon that system. The environment can include other systems that interact with the system of interest, either directly via interfaces or indirectly in other ways. The environment determines the boundaries that define the scope of the system of interest relative to other systems.

A system has one or more stakeholders. Each stakeholder typically has interests in, or concerns relative to, that system. *Concerns* are those interests which pertain to the system's development, its operation or any other aspects that are critical or otherwise important to one or more stakeholders. Concerns include system considerations such as performance, reliability, security, distribution, and evolvability.

A system exists to fulfill one or more missions in its environment. A *mission* is a use or operation for which a system is intended by one or more stakeholders to meet some set of objectives.

Every system has an architecture, in the terms of this recommended practice. An architecture can be recorded by an architectural description (see Figure 1). This recommended practice distinguishes the architecture of a system, which is conceptual, from particular descriptions of that architecture, which are concrete products or artifacts. Architectural descriptions (ADs) are the subject of this recommended practice.

In the conceptual framework of this recommended practice, an architectural description is organized into one or more constituents called (architectural) *views*. Each view addresses one or more of the concerns of the system stakeholders. The term *view* is used to refer to the expression of a system's architecture with respect to a particular viewpoint.

NOTE—This recommended practice does not use terms such as *functional architecture*, *physical architecture*, and *technical architecture*, as are frequently used informally. In the conceptual framework of the recommended practice, the approximate equivalents of these informal terms would be *functional view*, *physical view*, and *technical view*, respectively.

Other information, not contained in any constituent view, may appear in an AD , as a result of an organization's documentation practices. Examples of such information are the system overview, the system context, the system stakeholders and their key concerns, and the architectural rationale.

A viewpoint establishes the conventions by which a view is created, depicted and analyzed. In this way, a view conforms to a viewpoint. The viewpoint determines the languages (including notations, model, or product types) to be used to describe the view, and any associated modeling methods or analysis techniques to be

applied to these representations of the view. These languages and techniques are used to yield results relevant to the concerns addressed by the viewpoint.

An architectural description selects one or more viewpoints for use. The selection of viewpoints typically will be based on consideration of the stakeholders to whom the AD is addressed and their concerns.

A viewpoint definition may originate with an AD, or it may have been defined elsewhere. A viewpoint that is defined elsewhere is referred to in this recommended practice as a *library viewpoint*.

A view may consist of one or more *architectural models*. Each such architectural model is developed using the methods established by its associated architectural viewpoint. An architectural model may participate in more than one view.

NOTE—In a complex system, ADs may be developed for components of the system, as well as for the system as a whole. In this case, it may be that one AD will have a view corresponding to a particular viewpoint and another AD will have a view corresponding to the same viewpoint. Although the system being described by these two views has the whole-part relationship, this is not an instance of multiple views corresponding to one viewpoint. The ADs are considered separate even though they are related by the systems they describe.

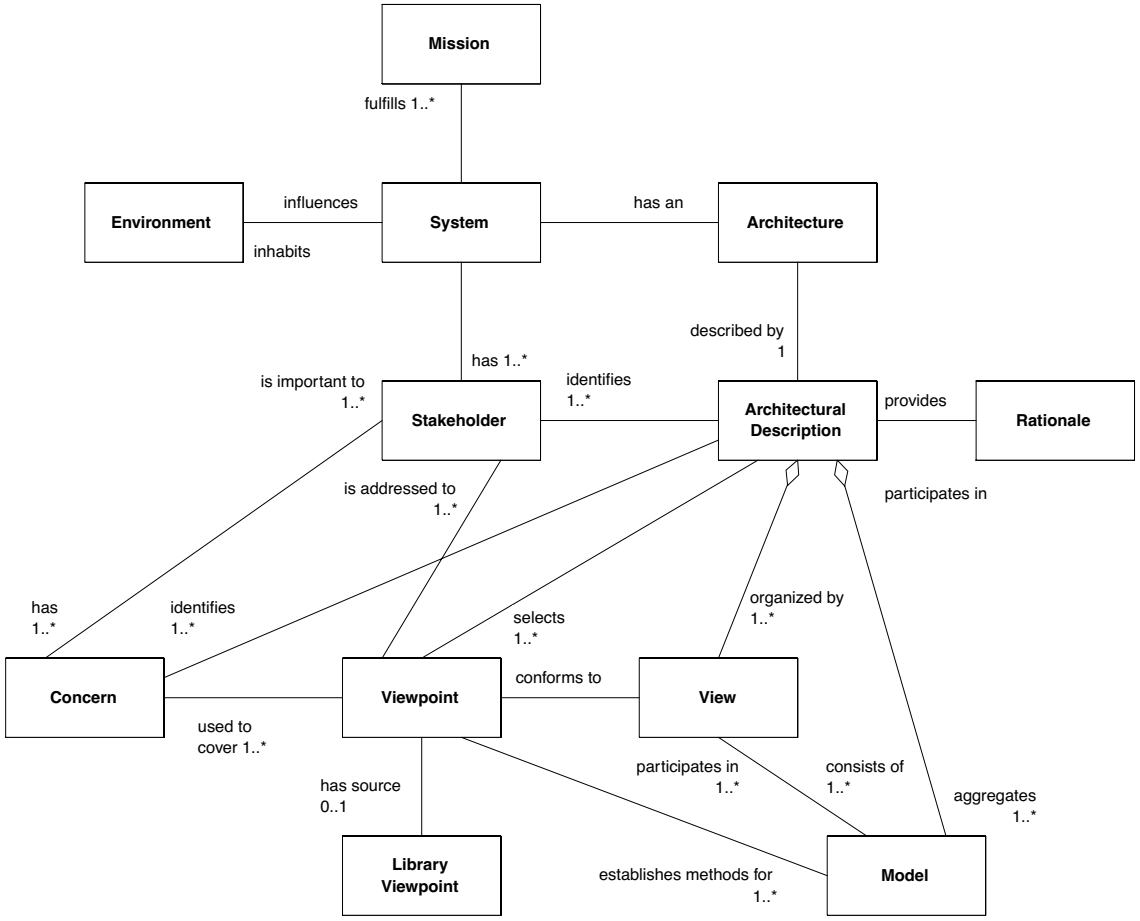


Figure 1—Conceptual model of architectural description

NOTE—Figure 1 provides an informative summary of the key concepts introduced by this recommended practice and their inter-relationships. The figure presents these concepts in the context of an architecture for a particular system and an associated architectural description. This is not to assume or require that a system has only one architecture or that there is only one architectural description depicting that architecture. In the figure, boxes represent classes of things. Lines connecting boxes represent associations between things. An association has two roles (one in each direction). A

role can optionally be named with a label. The role from A to B is closest to B, and vice versa. For example, the roles between SYSTEM and ENVIRONMENT can be read: A SYSTEM inhabits an ENVIRONMENT, and an ENVIRONMENT influences a SYSTEM. In the figure, roles are one-to-one unless otherwise noted. A role can have a multiplicity, e.g., a role marked with "1..*" is used to denote *many*, as in a one-to-many or many-to-many association. A diamond (at the end of an association line) denotes a *part-of* relationship. For example, VIEWS are a part of an ARCHITECTURAL DESCRIPTION. This notation is from the *Unified Modeling Language Specification* [B5].

4.2 Stakeholders and their roles

Stakeholders have various roles with regard to the creation and use of architectural descriptions. Stakeholders include clients, users, the architect, developers, and evaluators. Two key roles among stakeholders are the acquirer (or client) and the architect.

The architect develops and maintains an architecture for a system to satisfy the acquirer. The architect may work from requirements provided by an acquirer or may be responsible for eliciting and developing requirements as part of the architecture development. The role of the acquirer can be filled by a buyer, customer, owner, user, or purchaser (see IEEE/EIA Std 12207.0–1996[3]). The role of the architect can be filled by different persons, teams, or organizations throughout the life cycle of the architecture.

The architect records the architecture for the system's stakeholders in the form of an architectural description. In this form, the architecture serves to guide the system's developers.

4.3 Architectural activities in the life cycle

Architecting contributes to the development, operation, and maintenance of a system from its initial concept until its retirement from use. As such, architecting is best understood in a life cycle context, not simply as a single activity at one point in that life cycle.

Architecting is concerned with developing satisfactory and feasible system concepts, maintaining the integrity of those system concepts through development, certifying built systems for use, and assuring those system concepts through operational and evolutionary phases.

Detailed systems engineering activities, such as detailed requirements definition and interface specification and the architecting of major subsystems are tasks that typically follow development of the system architecture.

This recommended practice neither assumes nor prescribes a specific life cycle model—a life cycle model is to be separately chosen by users of the recommended practice. The scenarios in 4.3.1–4.3.4 are intended to suggest the range of uses of the recommended practice within a system life cycle.

4.3.1 Scenario: architecture of single systems

For new system construction, in cases where the user and the acquirer are identical, architecting is carried out in response to the user-acquirer's vision for the system, the system goals, and stakeholders' needs and constraints. The primary stakeholders are the user-acquirer and the system developers.

The architectural description can be used throughout the life cycle to predict the fitness for use of a system whose architecture conforms to the architectural description. In addition, the AD will typically evolve throughout the life cycle, and can thus act as a means for assessing changes to the system.

[3]Information on references can be found in Clause 2.

4.3.2 Scenario: iterative architecture for evolutionary systems

Under this scenario, the architecture, the delivered systems, and the stakeholders coevolve. An initial architecture is prepared in response to current and expected user needs using current and estimated technological capabilities. Initial systems are delivered using this architecture. As the architecture evolves in response to new stakeholders' needs, systems are delivered based on the architecture current at the time of development. Likewise, the set of stakeholders in the systems and architecture will change as the architecture evolves.

For evolutionary systems, the architectural description is used for system development and evaluation, but its uses and development are concurrent. Stakeholder needs are reassessed with each iteration of the architecture; the AD is used to guide each system as it moves through development, and appropriate AD versions are used to evaluate each system on delivery. Architectures developed under this scenario will naturally emphasize flexibility and adaptability more strongly than single system architectures. Architectural descriptions for evolving systems will typically be developed in an iterative pattern, or rhythm, of version changes.

The architecture development will normally be carried out by the developer as part of the overall activity of developing a sequence of systems. The architecture can also be established by the acquirer as a way of organizing the acquisition and evolution of a sequence of systems. This approach is one way to evolve a product line architecture. Evolution of the architecture will normally be sponsored by the acquirer as a part of the overall activity of sequential development or, in the case of commercial-off-the-shelf software, deployment of systems.

4.3.3 Scenario: architecture of existing systems

A third scenario occurs when system development has taken place without an architectural description, or when the AD, and perhaps the architect, are no longer available. The built system will have an architecture (since every system has an architecture, whether known or not) but it will lack an architectural description. In this case, an AD can be created through a reverse-engineering effort.

Using this recommended practice, an architectural description of the existing system is first constructed. This AD is then used to develop a new system, to guide maintenance or evolution activities, or to establish an evolutionary architecture as in the scenarios above (see 4.3.1 and 4.3.2).

In some cases, a new AD is needed because the original description does not provide views that are necessary later in a system's life cycle.

4.3.4 Scenario: architectural evaluation

The purpose of evaluation is to determine the quality of an architectural description and to predict the quality of systems whose architectures conform to the architectural description.

Quality of an architectural description refers to its capability to meet the needs and concerns of the stakeholders for whom it was constructed. Such concerns typically include understandability, consistency, completeness, and analyzability of the description.

Prediction of the quality of systems resulting from the architectural description includes such qualities as feasibility, efficiency, and reliability.

In order to evaluate conformance of an architecture to an architectural description, a process for reverse engineering an architectural description from an implementation is needed (see 4.3.3).

4.4 Uses of architectural descriptions

Architectural descriptions are applicable to a variety of uses, by a variety of stakeholders, throughout the life cycle. These uses include, but are not limited to the following:

a) Analysis of alternative architectures
b) Business planning for transition from a legacy architecture to a new architecture
c) Communications among organizations involved in the development, production, fielding, operation, and maintenance of a system
d) Communications between acquirers and developers as a part of contract negotiations
e) Criteria for certifying conformance of implementations to the architecture
f) Development and maintenance documentation, including material for reuse repositories and training materials
g) Input to subsequent system design and development activities
h) Input to system generation and analysis tools
i) Operational and infrastructure support; configuration management and repair; redesign and maintenance of systems, subsystems, and components
j) Planning and budget support
k) Preparation of acquisition documents (e.g., requests for proposal and statements of work)
l) Review, analysis, and evaluation of the system across the life cycle
m) Specification for a group of systems sharing a common set of features, (e.g., product lines)

5. Architectural description practices

This clause defines practices of architectural description that enable the uses outlined in 4.4. An architectural description conforming to this recommended practice meets the requirements set forth in this clause. Conforming architectural descriptions include the following elements (as delineated in the remainder of this clause):

a) AD identification, version, and overview information (see 5.1)
b) Identification of the system stakeholders and their concerns judged to be relevant to the architecture (see 5.2)
c) Specifications of each viewpoint that has been selected to organize the representation of the architecture and the rationale for those selections (see 5.3)
d) One or more architectural views (see 5.4)
e) A record of all known inconsistencies among the architectural description's required constituents (see 5.5)
f) A rationale for selection of the architecture (see 5.6)

NOTE—This recommended practice does not specify a delivery format for an architectural description.

5.1 Architectural documentation

An AD shall contain the following information pertaining to the AD as a whole:

a) Date of issue and status
b) Issuing organization

c) Change history
d) Summary
e) Scope
f) Context
g) Glossary
h) References

The detailed form and content of the corresponding information items shall be determined by the using organization. These items were selected to allow users to satisfy the requirements of IEEE/EIA Std 12207.0–1996.

5.2 Identification of stakeholders and concerns

An AD shall identify the stakeholders considered by the architect in formulating the architectural concept for the system.

At a minimum, the stakeholders identified shall include the following:

a) Users of the system
b) Acquirers of the system
c) Developers of the system
d) Maintainers of the system

An AD shall identify the concerns considered by the architect in formulating the architectural concept for the system.

At a minimum, the concerns identified should include the following:

— The purpose or missions of the system
— The appropriateness of the system for use in fulfilling its missions
— The feasibility of constructing the system
— The risks of system development and operation to users, acquirers, and developers of the system
— Maintainability, deployability, and evolvability of the system

The specific intent of the terms *mission*, *appropriateness*, and *feasibility* will vary from system to system. An AD shall further specify the concerns identified in the context of the system of interest.

5.3 Selection of architectural viewpoints

An AD shall identify the viewpoints selected for use therein.

Each viewpoint shall be specified by

a) A viewpoint name,
b) The stakeholders to be addressed by the viewpoint,
c) The concerns to be addressed by the viewpoint,
d) The language, modeling techniques, or analytical methods to be used in constructing a view based upon the viewpoint,

e) The source, for a library viewpoint (the source could include author, date, or reference to other documents, as determined by the using organization).

A viewpoint specification may include additional information on architectural practices associated with using the viewpoint, as follows:

— Formal or informal consistency and completeness tests to be applied to the models making up an associated view
— Evaluation or analysis techniques to be applied to the models
— Heuristics, patterns, or other guidelines to assist in synthesis of an associated view

Viewpoint specifications may be incorporated by reference (such as to a suitable recommended practice or previously defined practice).

An AD shall include a rationale for the selection of each viewpoint.

The rationale shall address the extent to which the stakeholders and concerns required in 5.2 are covered by the viewpoints selected under this clause.

NOTE—The recommended practice does not require that any particular viewpoints be selected by an architectural description. However, the rationale for selection of the viewpoints shows that the selected viewpoints cover the minimum set of stakeholders and concerns required in 5.2.

Each stakeholder and each concern identified in an AD in accordance with 5.2 shall be addressed by at least one viewpoint selected in accordance with 5.3. A stakeholder or concern may be addressed by more than one viewpoint in an architectural description.

NOTES:

1—It is envisioned that through the selection of suitable viewpoints, an AD can achieve conformance to other development or architecture standards.

2—Annex C and Annex D contain examples of viewpoints and viewpoint specifications.

5.4 Architectural views

An AD shall contain one or more architectural views.

Each view in an AD shall correspond to exactly one viewpoint, as selected in accordance with 5.3.

Each view in an AD shall conform to the specification of its corresponding viewpoint.

Each view shall include:

a) An identifier and other introductory information, as defined by the using organization
b) Representation of the system constructed with the languages, methods, and modeling or analytical techniques of the associated viewpoint
c) Configuration information, as defined by the using organization

An architectural view may consist of one or more architectural models.

5.5 Consistency among architectural views

An AD shall record any known inconsistencies among its architectural views.

An AD should contain an analysis of consistency across all of its architectural views.

5.6 Architectural rationale

An AD shall include the rationale for the architectural concepts selected.

An AD should provide evidence of the consideration of alternative architectural concepts and the rationale for the choices made.

When the AD describes a system that pre-exists the development of the AD, the rationale for the legacy system architecture shall be presented, if known.

5.7 Example use

(informative)

A brief example shows how the normative requirements of Clause 5 interconnect. To claim conformance with this recommended practice, the architect must produce an architectural description meeting the requirements in 5.1 through 5.6.

Subclause 5.1 requires that the AD include basic documentary information.

By 5.2, the AD must identify architecturally relevant stakeholders and concerns. For example, one stakeholder must be the system builder, and one of that stakeholder's concerns might be the input/output behavior of the system. This recommended practice places only minimal requirements on the concerns selected and thus deemed architectural. The using organization and the architect must make the central choice of what concerns to address.

Subclause 5.3 requires that a set of viewpoints be selected, defined, and analyzed for coverage and that the rationale be given for their selection. The architect might therefore select a behavioral viewpoint that would describe the system's input/output behavior. The architect would also be obligated to define what languages or models will be employed to describe system behavior under that viewpoint. No restriction is placed on the language or modeling technique used, but it must be explicitly specified in accordance with 5.3.

Subclause 5.4 requires the representation of the resulting architecture through one or more views. The architect will construct a behavioral view of the system, and that behavioral view must conform to the behavioral viewpoint defined previously.

Subclause 5.5 requires that any known inconsistencies among or between views (e.g., between the behavioral view provided and a structural view) be listed in the architectural description.

Lastly, in accordance with 5.6, the AD must contain a rationale for the architectural decisions and choices made by the architect.

NOTES:

1—The materials pertaining to 5.2 and 5.3, the stakeholders, concerns, and viewpoints, should be readily reusable elements for active architecting organizations. It is likely that such organizations will adopt common sets of viewpoints and

common specifications, that is, agreement and specification on concerns addressed and methods used. Indeed, viewpoint standardization could be a fruitful activity within domains where similar systems are repeatedly built. Some existing architectural standards can be recast as standards on selection and specification of viewpoints, with this recommended practice becoming a common reference point for reconciling architecting in various domains.

2—Annex D provides a description of how the recommended practice may be applied within the context of use of other architectural standards.

Annex A

(informative)

Bibliography

[B1] Bass, L., Clements, P., and Kazman, R., *Software Architecture in Practice*, Reading, MA: Addison-Wesley, 1998.

[B2] IEEE Std 100, *The IEEE Standard Dictionary of Electrical and Electronics Terms*, Sixth Edition.

[B3] ISO/IEC 10746–2: 1996, *Information technology—Open distributed processing—Reference model: Foundations*.

[B4] ISO/IEC 10746–3: 1996, *Information technology—Open distributed processing—Reference model: Architecture*.

[B5] OMG AD/97–08–05, 1997, *Unified Modeling Language Specification*, version 1.1, Object Management Group.

[B6] Perry, D. E. and Wolf, A. L., Foundations for the Study of Software Architecture. *ACM SIGSOFT Software Engineering Notes*, Vol. 17, No. 4, 1992.

[B7] Proakis, J. G., *Digital Communications*. New York: McGraw-Hill, 1995.

[B8] Shaw, M. and Garlan, D., An Introduction to Software Architecture. In *Advances in Software Engineering and Knowledge Engineering*, V. Ambriola and G. Tortora (eds.), River Edge, NJ: World Scientific Publishing Company, 1993.

[B9] Zachman, J. A., A Framework for Information Systems Architecture. *IBM Systems Journal*, Vol. 26, No. 3, 1987.

Annex B

(informative)

Notes on terminology

This recommended practice makes use of three terms, *architecture*, *view*, and *viewpoint,* which reflect community consensus, but are nevertheless employed in ways differing from some current uses. This annex discusses the terms to make clear the fundamental concepts of the recommended practice.

Architecture is defined by the recommended practice as *the fundamental organization of a system, embodied in its components, their relationships to each other and the environment, and the principles governing its design and evolution*. This definition is intended to encompass a variety of uses of the term *architecture* by recognizing their underlying common elements. Principal among these is the need to understand and control those elements of system design that capture the system's utility, cost, and risk. In some cases, these elements are the physical components of the system and their relationships. In other cases, these elements are not physical, but instead, logical components. In still other cases, these elements are enduring principles or patterns that create enduring organizational structures. The definition is intended to encompass these distinct, but related uses, while encouraging more rigorous definition of what constitutes the fundamental organization of a system within particular domains.

The terms *view* and *viewpoint* are central to the recommended practice. Within the conceptual framework of the recommended practice they refer to distinct things, and their definitions vary from some current uses of those terms. Their definitions do, however, reflect established usage in standards and engineering research. These notes discuss the two concepts to motivate the distinction and to justify the need for both terms.

It is a basic goal of the recommended practice to encompass existing architectural description practices by providing common terminology and concepts. A fundamental observation is that many existing practices represent architectures through collections of models. Typically, these models are further organized into cohesive groups. The cohesion of a group of models is determined by the observation that all of the included models address related concerns. What has been missing is some mechanism for formalizing these groupings and the models used to make the representations. In this recommended practice, *viewpoint* refers to a pattern or template for representing one set of concerns relative to an architecture, while a *view* is the actual representation of a particular system. A viewpoint provides the formalization of the groupings of models.

A *view* is a representation or description of the entire system from a single perspective. In contrast to a viewpoint, a view refers to a particular architecture of a system (i.e., an individual system, a product line, a system-of-systems, etc.). A view is primarily composed of models, although it also has additional attributes. The models provide the specific description, or content, of an architecture. For example, a structural view might consist of a set of models of the system structure. The elements of such models might include identifiable system components and their interfaces, and interconnections among those components.

The need for multiple views in architectural descriptions is widely recognized in the literature. To argue otherwise, in the terms of this recommended practice, is to claim that an architecture can be adequately described by a set of models corresponding to a single conceptual perspective. Alternatively, it would be to argue that all stakeholders and concerns can be addressed by a single set of modeling methods.

The use of multiple views to describe an architecture is therefore a fundamental element of this recommended practice. However, while the use of multiple views is widespread, authors differ on what views are needed and on appropriate methods for expressing each view.

Because of the wide range of opinion on selecting appropriate views, this recommended practice does not prescribe a fixed set of views, rather, it introduces the concept of a *viewpoint* to designate the means used to construct individual views.

In the recommended practice, the term *viewpoint* is used to designate a means for constructing a view that is independent of a particular system. The term was chosen to align with that of the ISO Reference Model of Open Distributed Processing (RM-ODP) [B3], which defines *viewpoint* as follows (but has no separate term for *view*):

> **viewpoint (on a system)**: a form of abstraction achieved using a selected set of architectural constructs and structuring rules, in order to focus on particular concerns within a system.

The relationship between viewpoint and view is analogous to that of a template and an instance of that template, or to use an object-oriented programming metaphor:

> viewpoint : view :: class : object[4]

In accordance with the recommended practice, a viewpoint must be specified by the stakeholders and concerns it addresses and the languages, models, and techniques it employs. This provides a means to capture the considerable commonality found among existing techniques.

By defining viewpoints, by requiring that viewpoints be specified in an AD before their use in views, and by not requiring specific viewpoints, the recommended practice provides a strategy for the customization and evolution of architectural practices. In this way, it is intended to satisfy the needs of *domain-specific* uses—where the domain may be, for example, application-specific, method-specific, or organization-specific.

An organization desiring to produce an architecture framework for a particular domain can do so by specifying a set of viewpoints and making the selection of those viewpoints normative for any AD claiming conformance to the domain-specific architectural framework. It is hoped that existing architectural frameworks—such as the ISO Reference Model for Open Distributed Processing (RM-ODP) [B4], the Enterprise Architecture Framework of Zachman [B9]), and the approach of Bass, Clements, and Kazman [B1] can be aligned with the standard in this manner.

In constructing an architectural description, an architect first selects the viewpoints, then constructs a set of corresponding views. Viewpoints can be thought of as being drawn from a library of viewpoints, whether or not such a viewpoint library actually exists. The architect selects the viewpoints from the library of viewpoints to be used in a particular architecture description. We imagine that actual libraries will come to exist as the recommended practice is put into use.

When a viewpoint is selected for a particular AD, it must be partially customized. The customization is in the choosing of what stakeholders and concerns it will be used to cover in this architectural description. The exact stakeholders and concerns needing to be covered is specific to a given system and AD and so cannot be fully specified in the viewpoint library. This suggests further that a viewpoint is basically a template.

Within a particular architectural description, every view is associated with exactly one viewpoint, which must have been selected for that architectural description. The required relationship between that viewpoint and its view is that the view conforms to the viewpoint. A view conforms to a viewpoint if it consists only of models that conform to the modeling methods called for in the viewpoint specification.

In principle, a given view could conform to multiple viewpoints, but should conform to only one viewpoint within a specific architectural description. However, a viewpoint can, and should, be reused across many

[4]This may be read, "a viewpoint is to a view as a class is to an object."

architectural descriptions. For example, a structural viewpoint requiring the use of a particular modeling method and technique of analysis might be selected by many different architectural descriptions.

While the relationship between views and viewpoints (within a given architectural description) is one-to-one, it should be understood that this relationship is more general outside a given architectural description. If, within one architectural description, a view corresponds to two or more viewpoints, it means the viewpoints are redundant. The stakeholders and concerns of interest to the set of viewpoints are all being addressed by one collection of models using one modeling method, so the stakeholders and concerns might as well be aggregated into a single viewpoint. If an AD is written for a system, and then another AD is written for a component of the system, as might happen in a complex system, then it is quite likely that the same viewpoints will be chosen for both ADs. Thus a given viewpoint (say a structural viewpoint) will have a corresponding view of the whole system and a corresponding view of the subsystem. The relationship is still one-to-one—within each AD. From the perspective of this recommended practice, the situation is no different than if the same viewpoint had been taken from a library and used on two completely different systems.

The relationship between views and viewpoints can be envisioned by imagining a library of viewpoints maintained by an architecting organization and a set of ADs for a variety of systems produced by that same organization. Each architectural description selects a set of viewpoints from the library of viewpoints. Each AD contains views that describe a particular system.

In addition to the required conformance of a view to its viewpoint, there is another form of checking required by the recommended practice. A viewpoint covers particular stakeholders and concerns. Every stakeholder and concern must be covered by at least one viewpoint, and may be addressed by several viewpoints.

Annex C offers examples of viewpoints.

Annex C

(informative)

Examples of viewpoints

This annex provides some examples of the central concept of viewpoint. Additional examples can be found in Annex D. The first two viewpoints (structural viewpoint and behavioral viewpoint) are frequently used in software architecture, but rarely defined explicitly. The next two viewpoints (physical interconnect viewpoint and link bit error rate viewpoint) are informal constructions intended to illustrate application of the viewpoint concept outside purely software architecture.

C.1 The structural viewpoint in software architecture

This viewpoint is motivated by current work in structure-oriented architecture description languages. The structural viewpoint has developed in the field of software architecture since 1994 and is in widespread use. This viewpoint is often *implicit* in contemporary architecture description languages.

In perhaps the earliest paper on software architecture, a structural viewpoint is implied by Perry and Wolf's [B6] definition of *elements*, as follows.

> By analogy to building architecture, we propose the following model of software architecture:
> Software Architecture = {Elements, Form, Rationale}

Perry and Wolf [B6] distinguish three classes of elements as follows:

a) Processing elements: "those components that supply the transformation of data elements"
b) Data elements: "those that contain the information that is used and transformed"
c) Connecting elements: those "elements (which at times may be either processing or data elements, or both) are the *glue* that holds the different pieces of the architecture together"

Shaw and Garlan [B8] adopt a similar approach :

> The framework we will adopt is to treat an architecture of a specific system as a collection of computational components—or simply *components*—together with a description of the interactions between these components—the *connectors*. … An architectural style, then, defines a family of such systems in terms of a pattern of *structural organization*. More specifically, an architectural style determines the vocabulary of components and connectors that can be used in instances of that style, together with a set of *constraints* on how they can be combined.

The structural viewpoint may be specified as follows:

Concerns

— What are the computational elements of a system and the organization of those elements?
— What software elements compose the system?
— What are their interfaces?
— How do they interconnect?
— What are the mechanisms for interconnection?

Viewpoint language

The structural viewpoint depends on the following entities:

— *Components* (individual units of computation)
— Components have *ports* (interfaces)
— *Connectors* (represent interconnections between components for communication and coordination)
— Connectors have *roles* (where they attach to components)

Components and connectors may be typed. All of the previously listed entities may have *attributes* of varying kinds.

Analytic methods

The structural viewpoint supports the following kinds of checking:

— Attachment (are connectors properly connecting components?)
— Type consistency (are the types of components and connectors used consistent with their attachments and any style or other constraints?)

Most architectural description languages that support the structural viewpoint provide some kind of syntax checking of the entities above.

C.2 Behavioral viewpoint

The behavioral viewpoint has a long history, prior to its use in architecture in the systems engineering, simulation, and software engineering/design communities. Examples include Petri nets, parallel path expressions, and constrained expressions.

The behavioral viewpoint may be specified as follows:

Concerns

— What are the dynamic actions of and within a system?
— What are the kinds of actions the system produces and participates in?
— How do those actions relate (ordering, synchronization, etc.)?
— What are the behaviors of system components? How do they interact?

Modeling methods

Events, processes, states, and operations on those entities.

Analytic methods

Some examples are: communicating sequential processes, the pi-calculus, and partially ordered sets of events.

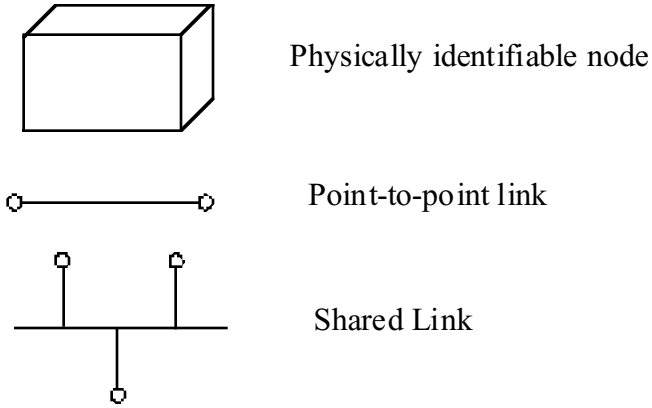
C.3 Physical interconnect viewpoint

Concerns

— What are the physical communications interconnects and their layering among system components?
— What is the feasibility of construction, compliance with standards, and evolvability?

Viewpoint language

Define the system with a set of one or more block diagrams using the following visual glossary:



One diagram is provided for each identified communication layer. Each diagram is annotated to show the addressing, link access, and routing/switching strategies at each link.

C.4 Link bit error rate viewpoint

Concerns

— What is the bit error rate on a communications link?
— What is the feasibility of building and maintaining consistency with operational information flows?

Viewpoint language

Compute bit error rate from

$$SNR = \frac{P_T L_T L_R}{N_F kTB} \quad (1)$$

Where:

P_T = is the transmit power
L_T = is the transmit losses, either cable or free space propagation
L_R = is the receive losses (or gain), either receiver only or antenna determined
N_F = is the noise figure, computed over whole receiver chain
kT = is the standard noise density
B = is the receiver detection bandwidth

Then BER = Error(SNR, modulation), where the error function is taken from a standard reference, such as Proakis [B7].

Annex D

(informative)

Relationship to other standards

Architectural descriptions can be used in a variety of settings and life cycle models. This annex demonstrates how architectural descriptions conforming to this recommended practice can be used to meet the requirements of other standards.

D.1 Use with IEEE/EIA Std 12207.0-1996

IEEE/EIA Std 12207.0–1996 defines two activities pertaining to architecture: system architectural design and software architectural design. The concept of architecture in this recommended practice is consistent with the system architectural design activity of IEEE/EIA Std 12207.0-1996. However, IEEE/EIA Std 12207.0-1996 places requirements on an architectural description in addition to those of this recommended practice. Specifically, a system architectural design must include an identification of the hardware items, software items, and manual operation items included in the system. System architectural designs must also be evaluated according to criteria including traceability to and consistency with the system requirements, appropriateness of design standards and methods, and feasibility of software and manual operations. An approach for meeting these requirements using this recommended practice is to define one or more viewpoints that meet the IEEE/EIA Std 12207.0-1996 requirements. An example of a viewpoint to satisfy IEEE/EIA Std 12207.0-1996 is defined in D.1.1.

The expected use of an architectural description may include elements of other IEEE/EIA Std 12207.0-1996 activities. In particular, an architectural description may be used in activities other than the system architectural design activity to facilitate the communications between the acquirer and the developer roles.

The software architectural design activity of IEEE/EIA Std 12207.0-1996 exemplifies a decompositional approach to architecture. Its primary goal is to decompose the software items of the system into components and then to allocate requirements to those components. Both the system AD and the products of other views in the architectural description would contribute to this activity and its products.

NOTE—The provisions of IEEE/EIA Std 12207.0-1996 relevant to architecture are identical to those found in ISO/IEC 12207: 1995.

An architectural description can conform to this recommended practice, to IEEE/EIA Std 12207.0-1996 and IEEE/EIA Std 12207.1-1997. One approach to *joint conformance* is to have a viewpoint that is specifically focused on producing the IEEE/EIA Std 12207.0-1996 architectural products. An example of a viewpoint defined for this purpose is shown in D.1.1.

D.1.1 Decomposition and allocation viewpoint

Concerns

- Identify the system requirements
- Decompose the system into items
- Allocate requirements to the items
- Ensure that all requirements are allocated to items

Modeling techniques and analytic methods

Each requirement is uniquely identified. If necessary, requirements are decomposed and extended into derived requirements to provide a full set of requirements for the system.

The system is decomposed into a set of items, where each item is a *hardware item*, a *software item* or a *manual operations item*. Interfaces between items are also identified.

System requirements are allocated to the items, such that each item satisfies one or more requirements, and each requirement is allocated to at least one item.

The initial decomposition and allocation produces a set of items with allocated requirements. This is described in terms of a System Architectural Description (see IEEE/EIA Std 12207.0–1996).

The initial software items are decomposed into subordinate components. Requirements allocated to each software item are further allocated to one or more components. Interface descriptions are provided between software components, and between software components and hardware items and manual operation items. This is described in terms of a Software Architectural Description (see IEEE/EIA Std 12207.0–1996).

D.2 Use with reference model of open distributed processing

The Reference Model of Open Distributed Processing (RM-ODP) [B3] defines an architectural framework for distributed processing systems; systems “in which discrete components may be located in different places, or where communication between components may suffer delay or may fail.”

The RM-ODP framework defines five viewpoints for specifying ODP systems.

For each viewpoint, there is an associated viewpoint language that defines “the concepts and rules for specifying ODP systems from the corresponding viewpoint.”

An architectural description conforming to this recommended practice and using 10746-3: 1996 [B4] would include the viewpoints defined by ISO/IEC 10746-3: 1996 and views to implement these viewpoints. A conforming architectural description need not be limited to the five predefined viewpoints of ISO/IEC 10746-3: 1996; the architectural description may include additional viewpoints and views, as needed.

The five viewpoints defined by ISO/IEC 10746-3: 1996 are specified here in accordance with the requirements of subclause 5.3 (D.2.1–D.2.5). Elements of that specification specific to architectural descriptions (such as stakeholders) are omitted here since they are particular to individual systems. Unless noted, all contents are direct quotes or close paraphrases from [B4].

D.2.1 Enterprise viewpoint

Concerns

— The purpose, scope, and policies for an ODP system
— Roles played by the system
— Activities undertaken by the system
— Policy statements about the system

Viewpoint language

From the enterprise viewpoint, an ODP system and its environment are represented as a community of objects. The community is defined in terms of:

— Enterprise objects composing the community
— Roles fulfilled by each of those objects
— Policies governing interactions between enterprise objects fulfilling roles
— Policies governing the creation, usage, and deletion of resources by enterprise objects fulfilling roles
— Policies governing the configuration of enterprise objects and assignment of roles to enterprise objects
— Policies relating to environment contracts governing the system. With objects defined in terms of permissions, obligations, prohibitions, and behavior

NOTE—An Enterprise Language is being developed by ISO/IEC JTC 1/SC 7.

Source: ISO/IEC 10746–3: 1996 [B4]

D.2.2 Information viewpoint

Concerns

— The semantics of information and information processing in an ODP system

Viewpoint language

The information language is defined in terms of three schemas:

— Invariant schema: Predicates on objects that must always be true
— Static schema: State of one or more objects at some point in time
— Dynamic schema: Allowable state changes of one or more objects

Source: ISO/IEC 10746–3: 1996 [B4]

D.2.3 Computational viewpoint

Concerns

— A functional decomposition of the system into objects that interact at interfaces

Viewpoint language

The computational language covers concepts for:

— Computational objects
— Binding objects
— Interactions
— Interfaces

Source: ISO/IEC 10746–3: 1996 [B4]

D.2.4 Engineering viewpoint

Concerns

— The mechanisms and functions required to support distributed interaction between objects in the system.

Viewpoint language

The engineering language includes concepts for:

— Node structures
— Channels that connect objects
— Interfaces
— Bindings
— Relocation/migration
— Clustering
— Nodes
— Failure types

Source: ISO/IEC 10746–3: 1996 [B4]

D.2.5 Technology viewpoint

Concerns

— Captures the choice of technology in the system
— How specifications are implemented
— Specification of relevant technologies
— Support for testing

Viewpoint language

The technology language addressed implementable standards and implementations.

Source: ISO/IEC 10746–3: 1996 [B4]