INTERNATIONAL STANDARD

ISO/IEC/ IEEE 42010

First edition
2011-12-01

Systems and software engineering — Architecture description

Ingénierie des systèmes et des logiciels — Description de l'architecture

Reference number
ISO/IEC/IEEE 42010:2011(E)

COPYRIGHT PROTECTED DOCUMENT

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 749 09 47
E-mail copyright@iso.org
Web www.iso.org

Institute of Electrical and Electronics Engineers, Inc.
3 Park Avenue, New York • NY 10016-5997, USA
E-mail stds.ipr@ieee.org
Web www.ieee.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

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 contained in its standards.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of ISO/IEC JTC 1 is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is called to the possibility that implementation of this standard may require the 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. ISO/IEEE is not responsible for identifying essential patents or patent claims for which a license may be required, for conducting inquiries into the legal validity or scope of patents or patent claims or determining whether any licensing terms or conditions provided in connection with submission of a Letter of Assurance or a Patent Statement and Licensing Declaration Form, 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 ISO or the IEEE Standards Association.

ISO/IEC/IEEE 42010 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*, in cooperation with the Software and Systems Engineering Standards Committee of the Computer Society of the IEEE, under the Partner Standards Development Organization cooperation agreement between ISO and IEEE.

This first edition of ISO/IEC/IEEE 42010 cancels and replaces ISO/IEC 42010:2007, which has been technically revised.

Introduction

The complexity of man-made systems has grown to an unprecedented level. This has led to new opportunities, but also to increased challenges for the organizations that create and utilize systems. Concepts, principles and procedures of architecting are increasingly applied to help manage the complexity faced by stakeholders of systems.

Conceptualization of a system's architecture, as expressed in an architecture description, assists the understanding of the system's essence and key properties pertaining to its behaviour, composition and evolution, which in turn affect concerns such as the feasibility, utility and maintainability of the system.

Architecture descriptions are used by the parties that create, utilize and manage modern systems to improve communication and co-operation, enabling them to work in an integrated, coherent fashion. Architecture frameworks and architecture description languages are being created as assets that codify the conventions and common practices of architecting and the description of architectures within different communities and domains of application.

This International Standard addresses the creation, analysis and sustainment of architectures of systems through the use of architecture descriptions.

This International Standard provides a core ontology for the description of architectures. The provisions of this International Standard serve to enforce desired properties of architecture descriptions. This International Standard also specifies provisions that enforce desired properties of architecture frameworks and architecture description languages (ADLs), in order to usefully support the development and use of architecture descriptions. This International Standard provides a basis on which to compare and integrate architecture frameworks and ADLs by providing a common ontology for specifying their contents.

This International Standard can be used to establish a coherent practice for developing architecture descriptions, architecture frameworks and architecture description languages within the context of a life cycle and its processes (not defined by this International Standard). This International Standard can further be used to assess conformance of an architecture description, of an architecture framework, of an architecture description language, or of an architecture viewpoint to its provisions.

Users of this International Standard are advised to consult Clause 4 to gain appreciation of the provided ontology, its concepts and principles.

Systems and software engineering — Architecture description

1 Scope

This International Standard specifies the manner in which architecture descriptions of systems are organized and expressed.

This International Standard specifies architecture viewpoints, architecture frameworks and architecture description languages for use in architecture descriptions.

This International Standard also provides motivations for terms and concepts used; presents guidance on specifying architecture viewpoints; and demonstrates the use of this International Standard with other standards.

2 Conformance

The requirements in this International Standard are contained in Clauses 5, 6 and 7. There are four situations in which claims of conformance with the provisions of this International Standard can be made.

— When conformance is claimed for an architecture description, the claim shall demonstrate that the architecture description meets the requirements listed in Clause 5.

— When conformance is claimed for an architecture viewpoint, the claim shall demonstrate that the architecture viewpoint meets the requirements listed in Clause 7.

— When conformance is claimed for an architecture framework, the claim shall demonstrate that the architecture framework meets the requirements listed in 6.1.

— When conformance is claimed for an architecture description language, the claim shall demonstrate that the architecture description language meets the requirements listed in 6.3.

Requirements of this International Standard are marked by the use of the verb “shall”. Recommendations are marked by the use of the verb “should”. Permissions are marked by the use of the verb “may”. In the event of a conflict between normative figures and text, the text takes precedence. Please report any apparent conflicts.

NOTE This International Standard is designed such that “tailoring” is neither required nor permitted for its use when claims of conformance are made.

3 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

3.1
architecting
process of conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture throughout a system’s life cycle

NOTE Architecting takes place in the context of an organization ("person or a group of people and facilities with an arrangement of responsibilities, authorities and relationships") and/or a project ("endeavour with defined start and finish criteria undertaken to create a product or service in accordance with specified resources and requirements") [ISO/IEC 12207, ISO/IEC 15288].

3.2
architecture
⟨system⟩ fundamental concepts or properties of a system in its environment embodied in its elements, relationships, and in the principles of its design and evolution

3.3
architecture description
AD
work product used to express an architecture

3.4
architecture framework
conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders

EXAMPLE 1 Generalised Enterprise Reference Architecture and Methodologies (GERAM) [ISO 15704] is an architecture framework.

EXAMPLE 2 Reference Model of Open Distributed Processing (RM-ODP) [ISO/IEC 10746] is an architecture framework.

3.5
architecture view
work product expressing the architecture of a system from the perspective of specific system concerns

3.6
architecture viewpoint
work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns

3.7
concern
⟨system⟩ interest in a system relevant to one or more of its stakeholders

NOTE A concern pertains to any influence on a system in its environment, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.

3.8
environment
⟨system⟩ context determining the setting and circumstances of all influences upon a system

NOTE The environment of a system includes developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences.

3.9
model kind
conventions for a type of modelling

NOTE Examples of model kinds include data flow diagrams, class diagrams, Petri nets, balance sheets, organization charts and state transition models.

3.10
stakeholder
⟨system⟩ individual, team, organization, or classes thereof, having an interest in a system

4 Conceptual foundations

4.1 Introduction

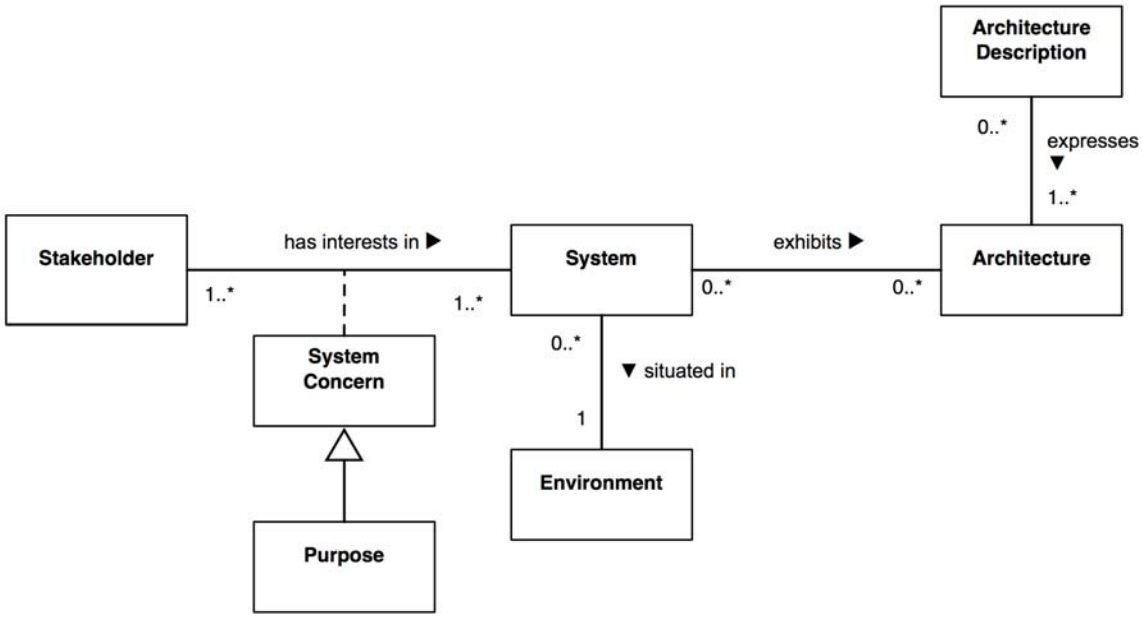
This clause introduces the conceptual foundations of architecture description comprising a conceptual model of architecture description (see 4.2); the role of architecting in the life cycle (see 4.3); uses of architecture descriptions (see 4.4); and architecture frameworks and architecture description languages (see 4.5). The concepts introduced in this clause are used in Clauses 5 through 7 to express requirements.

NOTE Annex A provides further discussion of the terms and concepts used in this International Standard and presents examples of their use.

4.2 Conceptual model of architecture description

4.2.1 Context of architecture description

Figure 1 depicts key concepts pertaining to systems and their architectures as a context for understanding the practice of architecture description.



NOTE The figure uses the conventions for class diagrams defined in [ISO/IEC 19501].

Figure 1 — Context of architecture description

The term *system* is used in this International Standard to refer to entities whose architectures are of interest. The term is intended to encompass, but is not limited to, entities within the following domains:

— **systems** as described in [ISO/IEC 15288]: “systems that are man-made and may be configured with one or more of the following: hardware, software, data, humans, processes (e.g., processes for providing service to users), procedures (e.g. operator instructions), facilities, materials and naturally occurring entities”;

— **software products and services** as described in [ISO/IEC 12207];

— **software-intensive systems** as described in [IEEE Std 1471:2000]: “any system where software contributes essential influences to the design, construction, deployment, and evolution of the system as a whole” to encompass “individual applications, systems in the traditional sense, subsystems, systems of systems, product lines, product families, whole enterprises, and other aggregations of interest”.

This International Standard takes no position on what constitutes a system within those domains—or elsewhere. The nature of systems is not defined by this International Standard.

This International Standard is intended for use in the domains of systems listed above; however, nothing herein precludes its use for architecture descriptions of entities of interest outside of those domains (for example, natural systems and conceptual systems).

The stakeholders of a system are parties with interests in that system. Stakeholders’ interests are expressed as concerns (see 4.2.3). Stakeholders ascribe various *purposes* to a system. Purposes are one kind of concern.

NOTE 1 The term *purpose* as used in this International Standard derives from its use in ISO/IEC 15288:2008, 4.31: a system is a combination of interacting elements organized to achieve one or more stated purposes.

A system is situated in an environment. The environment determines the totality of influences upon the system throughout its life cycle, including its interactions with that environment. The environment of a system can contain other systems.

NOTE 2 In this International Standard, the environment of a system is bounded by and understood through the identification and analysis of the system’s stakeholders and their concerns (see 4.2.3).

The architecture of a system constitutes what is essential about that system considered in relation to its environment. There is no single characterization of what is essential or fundamental to a system; that characterization could pertain to any or all of:

— system constituents or elements;

— how system elements are arranged or interrelated;

— principles of the system’s organization or design; and

— principles governing the evolution of the system over its life cycle.

Architecture descriptions are used to express architectures for systems of interest (see 4.2.2).

NOTE 3 The same system could be understood through several distinct architectures (for example, when considered in different environments). An architecture could be expressed through several distinct architecture descriptions (for example when different architecture frameworks are employed). The same architecture could characterise more than one system (for example a family of systems sharing a common architecture)

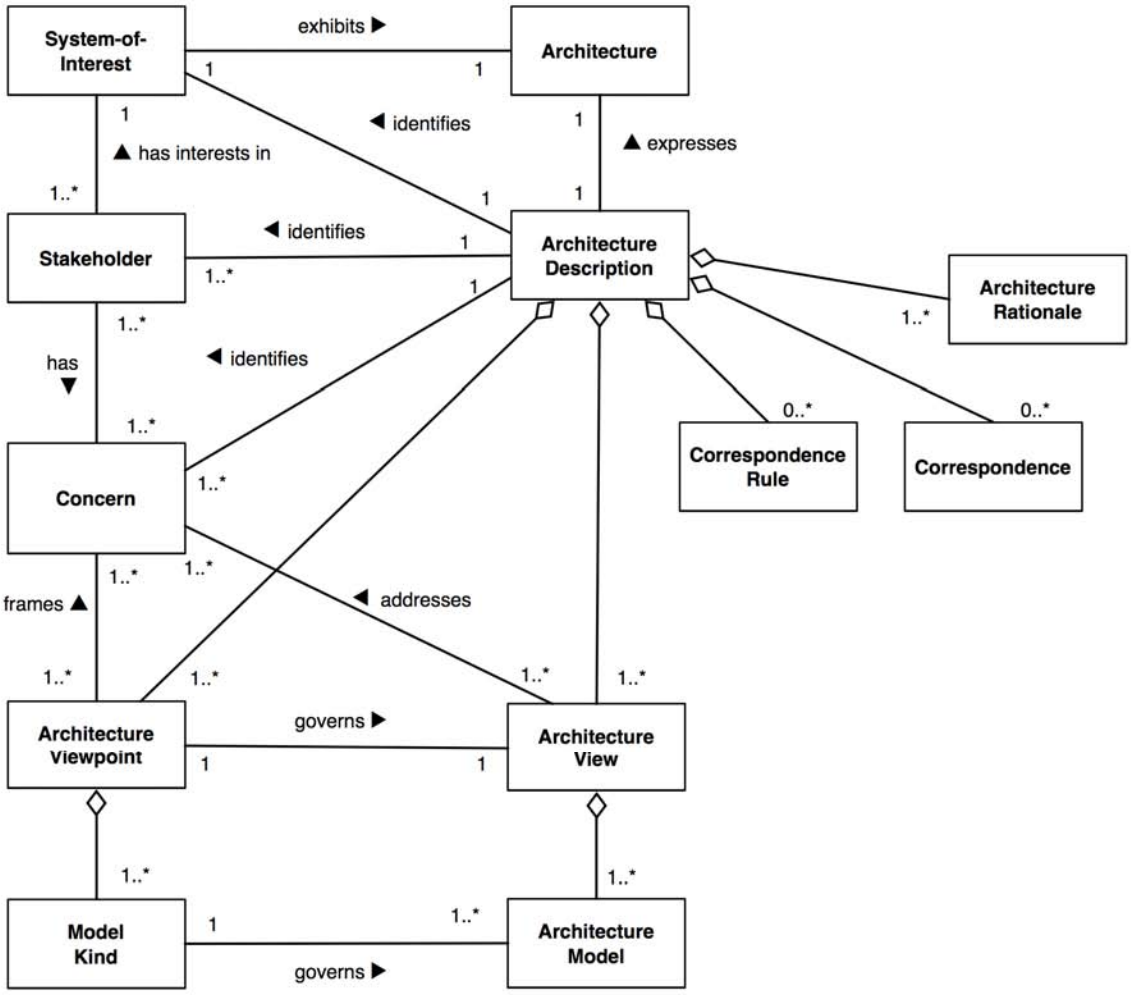
4.2.2 Architectures and architecture descriptions

Architecture descriptions are work products of systems and software architecting.

Figure 2 depicts concepts pertaining to the practice of architecture description *when applying this International Standard to produce one architecture description expressing one architecture for one system-of-interest*.

In this International Standard, the term *system-of-interest* (or simply, *system*) refers to the system whose architecture is under consideration in the preparation of an architecture description.

The figures and text in the remainder of 4.2 constitute a *conceptual model* of architecture description.



NOTE 1 The figure uses the conventions for class diagrams defined in [ISO/IEC 19501].

NOTE 2 Figure 3 provides additional details on correspondences and correspondence rules. Figure 4 provides additional details on architecture rationale.

Figure 2 — Conceptual model of an architecture description

An architecture description expresses an architecture of a system-of-interest.

This International Standard distinguishes an *architecture of a system* from an *architecture description*. Architecture descriptions, not architectures, are the subject of this International Standard. Whereas an architecture description is a work product, an architecture is abstract, consisting of concepts and properties. This International Standard specifies requirements on architecture descriptions. There are no requirements in this International Standard pertaining to architectures, or to systems or to their environments.

This International Standard does not specify any format or media for recording architecture descriptions. It is intended to be usable for a range of approaches to architecture description including document-centric, model-based, and repository-based techniques.

This International Standard does not prescribe the process or method used to produce architecture descriptions. This International Standard does not assume or prescribe specific architecting methods, models, notations or techniques used to produce architecture descriptions.

4.2.3 Stakeholders and concerns

Stakeholders of a system have concerns with respect to the system-of-interest considered in relation to its environment. A concern could be held by one or more stakeholders. Concerns arise throughout the life cycle from system needs and requirements, from design choices and from implementation and operating considerations. A concern could be manifest in many forms, such as in relation to one or more stakeholder needs, goals, expectations, responsibilities, requirements, design constraints, assumptions, dependencies, quality attributes, architecture decisions, risks or other issues pertaining to the system.

EXAMPLES The following are concerns in the terms of this International Standard: functionality, feasibility, usage, system purposes, system features, system properties, known limitations, structure, behavior, performance, resource utilization, reliability, security, information assurance, complexity, evolvability, openness, concurrency, autonomy, cost, schedule, quality of service, flexibility, agility, modifiability, modularity, control, inter-process communication, deadlock, state change, subsystem integration, data accessibility, privacy, compliance to regulation, assurance, business goals and strategies, customer experience, maintainability, affordability and disposability. The *distribution transparencies* described in the Reference Model of Open Distributed Processing [ISO/IEC 10746-1] are concerns in the terms of this International Standard. *Software properties* as described in SQUARE [ISO/IEC 25010:2011, 4.2] name concerns in the terms of this International Standard.

4.2.4 Architecture views and viewpoints

An architecture description includes one or more architecture views. An architecture view (or simply, *view*) addresses one or more of the concerns held by the system's stakeholders.

An architecture view expresses the architecture of the system-of-interest in accordance with an architecture viewpoint (or simply, *viewpoint*). There are two aspects to a viewpoint: the concerns it frames for stakeholders and the conventions it establishes on views.

An architecture viewpoint **frames**[1] one or more concerns. A concern can be framed by more than one viewpoint.

A view is **governed** by its viewpoint: the viewpoint establishes the conventions for constructing, interpreting and analyzing the view to address concerns framed by that viewpoint. Viewpoint conventions can include languages, notations, model kinds, design rules, and/or modelling methods, analysis techniques and other operations on views.

Figure 2 depicts the relations between views and viewpoints within an architecture description.

NOTE 1 This International Standard does not use phrases such as "business architecture", "physical architecture", and "technical architecture". In the terms of this International Standard, the architecture of a system is a holistic conception of that system's fundamental properties, best understood via multiple views of that architecture. Therefore, approximate equivalents of the above phrases are "business view", "physical view", and "technical view", respectively.

NOTE 2 Clause 7 specifies requirements on architecture viewpoints. Annex B provides guidance on specifying viewpoints.

4.2.5 Architecture models

An architecture view is composed of one or more architecture models. An architecture model uses modelling conventions appropriate to the concerns to be addressed. These conventions are specified by the *model kind* governing that model. Within an architecture description, an architecture model can be a part of more than one architecture view.

Figure 2 depicts the use of architecture models and model kinds within an architecture description.

NOTE This International Standard uses the term *model kind* rather than "architecture model kind" to emphasize that model kinds need not be useful exclusively in architecture descriptions.

---

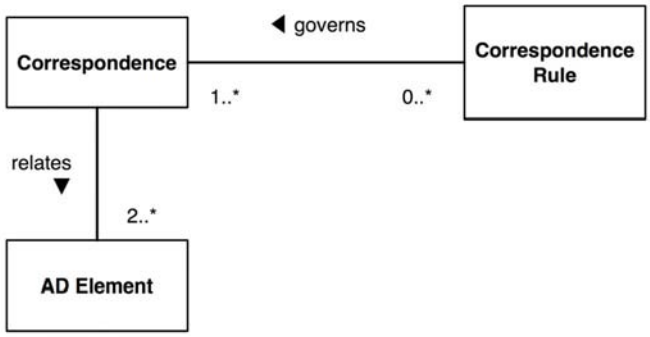
[1] In this International Standard, the verb *frame* is used in its ordinary language sense: to formulate or construct in a particular style or language; to enclose in or as if in a frame; to surround so as to create a sharp or attractive image.

4.2.6 AD elements and correspondences

An *AD element* is any construct in an architecture description. AD elements are the most primitive constructs discussed in this International Standard. Every stakeholder, concern, architecture viewpoint, architecture view, model kind, architecture model, architecture decision and rationale (see 4.2.7) is considered an AD element. When viewpoints and model kinds are defined and their models are populated, additional AD elements are introduced.

A *correspondence* defines a relation between AD elements. Correspondences are used to express architecture relations of interest within an architecture description (or between architecture descriptions). Correspondences can be governed by *correspondence rules*. Correspondence rules are used to enforce relations within an architecture description (or between architecture descriptions).

Figure 3 depicts the concepts of AD elements and correspondences.



NOTE The figure uses the conventions for class diagrams defined in [ISO/IEC 19501].

Figure 3 — Conceptual model of AD elements and correspondences

EXAMPLES Correspondences and correspondence rules are used to express and enforce architecture relations such as composition, refinement, consistency, traceability, dependency, constraint and obligation.

NOTE Requirements on using correspondences and correspondence rules are specified in 5.7. Examples of their use are given in A.6.

4.2.7 Architecture decisions and rationale

Architecture rationale records explanation, justification or reasoning about architecture decisions that have been made. The rationale for a decision can include the basis for a decision, alternatives and trade-offs considered, potential consequences of the decision and citations to sources of additional information.

Decisions pertain to system concerns; however, there is often no simple mapping between the two. A decision can affect the architecture in several ways. These can be reflected in the architecture description as follows:

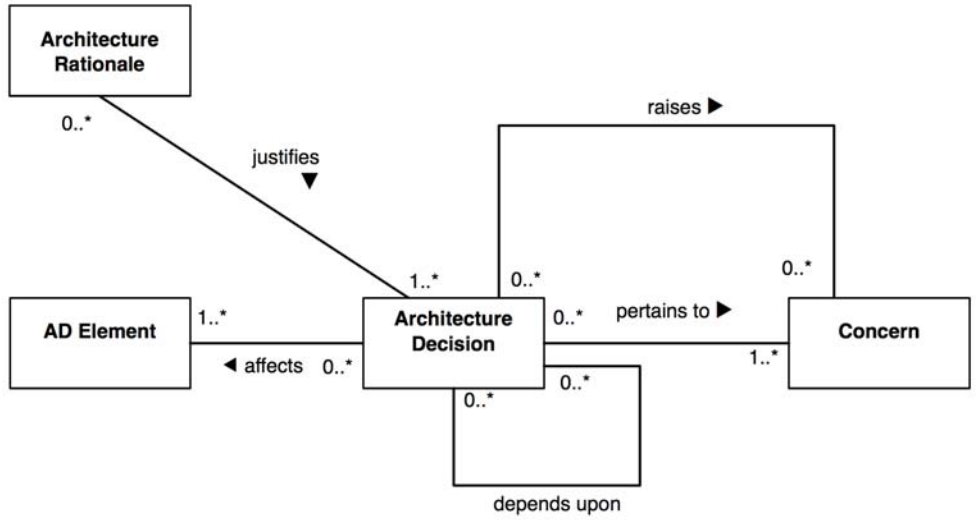
— requiring the existence of AD elements;

— changing the properties of AD elements;

— triggering trade-off analyses in which some AD elements, including other decisions and concerns, are revised;

— raising new concerns.

Figure 4 depicts concepts pertaining to architecture decisions and rationale.



NOTE The figure uses the conventions for class diagrams defined in [ISO/IEC 19501].

Figure 4 — Conceptual model of architecture decisions and rationale

NOTE Requirements for capturing decisions and rationale within an architecture description are specified in 5.8.

4.3 Architecting in the life cycle

Architecting contributes to the development, operation and maintenance of a system from its initial conception through its retirement from use and disposal. Architecting takes place within the context of a project and/or organization. Architecting is performed throughout the system life cycle, not simply within one stage of the life cycle. Therefore, a system's architecture potentially influences processes throughout the system's life cycle.

An architecture description is a work product resulting from the execution of architecting activities within the life cycle of the system-of-interest. A life cycle prescribes the stages and manner in which the contents of a conforming architecture description are to be produced. Even when an architecture description results from a single life cycle activity, its contents are likely to be the result of multiple activities. Alternatively, an architecture description can be produced by aggregation of information from other work products developed by life cycle activities.

This International Standard does not depend upon, assume or prescribe any particular life cycle.

NOTE Annex C demonstrates how this International Standard can be used when applying the life cycle processes of ISO/IEC 12207 and ISO/IEC 15288. ISO/IEC 12207 and ISO/IEC 15288 provide distinct life cycle processes for architectural design. This does not contradict the concept that architecting is performed throughout the life cycle for two reasons: (1) Any process of ISO/IEC 12207 or ISO/IEC 15288 can be regarded as executing throughout the life cycle; (2) the use of "architectural design" in ISO/IEC 12207 and ISO/IEC 15288 is more narrow than the concept of architecting in this International Standard.

4.4 Uses of architecture descriptions

Architecture descriptions have many uses by a variety of stakeholders throughout the system life cycle. Uses for architecture descriptions include, but are not limited to:

— as basis for system design and development activities;

— as basis to analyze and evaluate alternative implementations of an architecture;

— as development and maintenance documentation;

— documenting essential aspects of a system, such as:

  - intended use and environment;
  - principles, assumptions and constraints to guide future change;
  - points of flexibility or limitations of the system with respect to future changes;
  - architecture decisions, their rationales and implications;

— as input to automated tools for simulation, system generation and analysis;

— specifying a group of systems sharing common features (such as architectural styles, reference architectures and product line architectures);

— communicating among parties involved in the development, production, deployment, operation and maintenance of a system;

— as basis for preparation of acquisition documents (such as requests for proposal and statements of work);

— communicating among clients, acquirers, suppliers and developers as a part of contract negotiations;

— documenting the characteristics, features and design of a system for potential clients, acquirers, owners, operators and integrators;

— planning for transition from a legacy architecture to a new architecture;

— as guide to operational and infrastructure support and configuration management;

— as support to system planning, scheduling and budgeting activities;

— establishing criteria for certifying implementations for compliance with an architecture;

— as compliance mechanism to external and project and/or organization-internal policies (for example, legislation, overarching architectural principles)

— as basis for review, analysis, and evaluation of the system across its life cycle;

— as basis to analyze and evaluate alternative architectures;

— sharing lessons learned and reusing architectural knowledge through viewpoints, patterns and styles;

— training and education of stakeholders and other parties on best practices in architecting and system evolution.

NOTE Annex C discusses the use of architecture descriptions in the context of other standards.

4.5 Architecture frameworks and architecture description languages

Architecture frameworks and architecture description languages (ADLs) are two mechanisms widely used in architecting. Architecture frameworks and architecture description languages are specified by building on the concepts of architecture description presented in this International Standard.

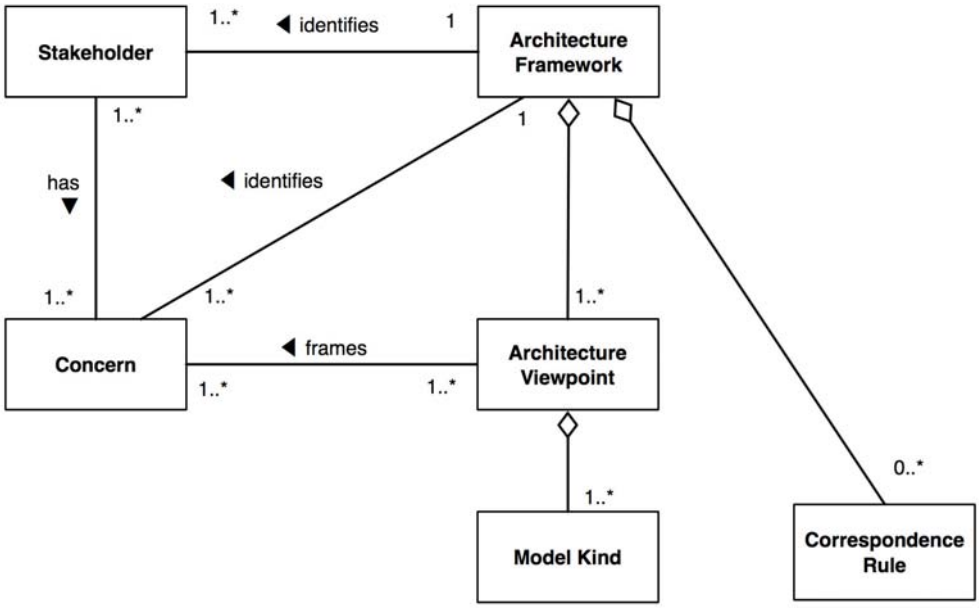
An architecture framework establishes a common practice for creating, interpreting, analyzing and using architecture descriptions within a particular domain of application or stakeholder community.

Uses of architecture frameworks include, but are not limited to: creating architecture descriptions; developing architecture modelling tools and architecting methods; and establishing processes to facilitate communication, commitments and interoperation across multiple projects and/or organizations.

NOTE 1 Architecture frameworks frequently encompass both provisions for architecture description and additional architecting practices.

EXAMPLES The following are architecture frameworks in the terms of this International Standard: Zachman's information systems architecture framework [44], UK Ministry of Defence Architecture Framework [27], The Open Group's Architecture Framework (TOGAF) [41], Kruchten's "4+1" view model [23], Siemens' 4 views method [10], Reference Model for Open Distributed Processing (RM-ODP), [ISO/IEC 10746] and Generalized Enterprise Reference Architecture (GERA) [ISO 15704].

Figure 5 depicts the contents of an architecture framework.



NOTE The figure uses the notation for class diagrams defined in [ISO/IEC 19501].

Figure 5 — Conceptual model of an architecture framework

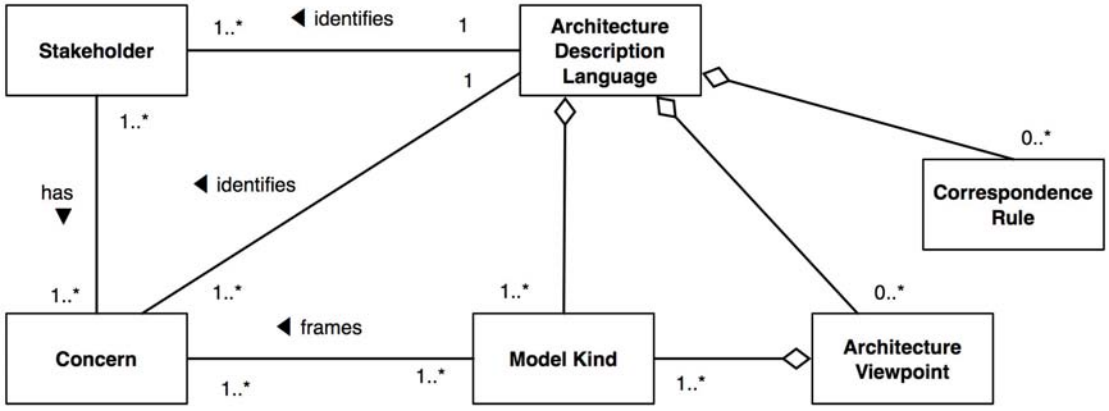
NOTE 2 Requirements on architecture frameworks are specified in 6.1.

An architecture description language (ADL) is any form of expression for use in architecture descriptions.

An ADL provides one or more model kinds as a means to frame some concerns for its audience of stakeholders. An ADL can be narrowly focused, defining a single model kind, or widely focused to provide several model kinds, optionally organized into viewpoints. Often an ADL is supported by automated tools to aid the creation, use and analysis of its models.

EXAMPLES Rapide [25], Wright [43], SysML [31], ArchiMate [40] and the viewpoint languages of RM-ODP [ISO 10746] are ADLs in the terms of this International Standard.

Figure 6 depicts the contents of an architecture description language.



NOTE The figure uses the notation for class diagrams defined in [ISO/IEC 19501].

Figure 6 — Conceptual model of an architecture description language

NOTE 3 Requirements on an architecture description language are specified in 6.3.

5 Architecture descriptions

5.1 Introduction

This clause specifies the characteristics of architecture descriptions that enable the uses listed in 4.4. Architecture descriptions include the following contents, as specified in the remainder of this clause:

— architecture description identification and overview information (see 5.2);

— identification of the system stakeholders and their concerns (see 5.3);

— a definition for each architecture viewpoint used in the architecture description (see 5.4);

— an architecture view and architecture models for each architecture viewpoint used (see 5.5 and 5.6);

— applicable AD correspondence rules, AD correspondences and a record of known inconsistencies among the architecture description's required contents (see 5.7);

— rationales for architecture decisions made (see 5.8);

The verb *include* when used in Clause 5 indicates that either the information is present in the architecture description or reference to that information is provided therein.

NOTE 1 This International Standard does not specify a format for architecture descriptions.

NOTE 2 In order to produce multiple architecture descriptions of different architectures or alternative expressions of the same architecture, the user would apply the provisions of this Clause for each architecture description. The results can be combined or separately presented, in a manner not defined by this International Standard.

5.2 Architecture description identification and overview

An architecture description shall identify the system-of-interest and include supplementary information as determined by the project and/or organization.

The detailed content of identifying and supplementary information items shall be as specified by the organization and/or project.

NOTE Examples of identifying and supplementary information in an architecture description are: date of issue and status; authors, reviewers, approving authority, issuing organization; change history; summary; scope; context; glossary; version control information; configuration management information and references. See [ISO/IEC 15289] or [ISO/IEC TR 15504-6:2008, B.1] for examples.

Results from any evaluations of the architecture or its architecture description shall be included.

5.3 Identification of stakeholders and concerns

An architecture description shall identify the system stakeholders having concerns considered fundamental to the architecture of the system-of-interest.

The following stakeholders shall be considered and when applicable, identified in the architecture description:

— users of the system;

— operators of the system;

— acquirers of the system;

— owners of the system;

— suppliers of the system;

— developers of the system;

— builders of the system;

— maintainers of the system.

An architecture description shall identify the concerns considered fundamental to the architecture of the system-of-interest.

The following concerns shall be considered and when applicable, identified in the architecture description:

— the purposes of the system;

— the suitability of the architecture for achieving the system's purposes;

— the feasibility of constructing and deploying the system;

— the potential risks and impacts of the system to its stakeholders throughout its life cycle;

— maintainability and evolvability of the system.

An architecture description shall associate each identified concern with the identified stakeholders having that concern.

NOTE 1 In general, the association of concerns with stakeholders is many-to-many.

NOTE 2 This International Standard does not prescribe: the granularity of concerns; how concerns interrelate with other concerns; or how concerns relate to other statements about a system such as stakeholder needs, system goals or requirements. These issues are subjects for specific architecture frameworks, architecting methods or other practices.

5.4 Architecture viewpoints

An architecture description shall include each architecture viewpoint used therein.

Each included architecture viewpoint shall be specified in accordance with the provisions of Clause 7.

Each concern identified in accordance with 5.3 shall be framed by at least one viewpoint.

NOTE 1 This International Standard does not require any particular viewpoints to be used.

NOTE 2 Annexes B and C provide additional information pertaining to architecture viewpoints.

5.5 Architecture views

An architecture description shall include exactly one architecture view for each architecture viewpoint used.

Each architecture view shall adhere to the conventions of its governing architecture viewpoint.

Each architecture view shall include:

a) identifying and supplementary information as specified by the organization and/or project;

b) identification of its governing viewpoint;

c) architecture models that address all of the concerns framed by its governing viewpoint and cover the whole system from that viewpoint;

d) recording of any known issues within a view with respect to its governing viewpoint.

NOTE 1 See 5.2 NOTE for examples of identifying and supplementary information per a).

NOTE 2 The requirement per c) that each architecture view covers the whole system with respect to the concerns framed by its governing viewpoint is essential to the complete allocation of concerns within an architecture description. Within a view, one or more architecture models can be used to selectively present portions of the system to highlight points of interest, without violating this requirement (see 5.6).

NOTE 3 “Known issues” per d) include unresolved issues, exceptions and deviations from the conventions. Open issues can lead to decisions to be made. Exceptions and deviations can be documented as decision outcomes and rationale (per 5.8).

An architecture description may include information not part of any architecture view.

EXAMPLES Instances of information not part of any view are system overview, model correspondences and architecture rationale.

5.6 Architecture models

An architecture view shall be composed of one or more architecture models.

Each architecture model shall include version identification as specified by the organization and/or project.

Each architecture model shall identify its governing model kind and adhere to the conventions of that model kind (see 5.4).

An architecture model may be a part of more than one architecture view.

NOTE 1 Sharing architecture models between architecture views permits an architecture description to frame distinct but related concerns without redundancy or repetition of the same information in multiple views, and reduces possibilities for inconsistency. Sharing of architecture models also permits an aspect-oriented style of architecture description: architecture models *shared across architecture views* can be used to express architectural perspectives (see [36]); architecture models shared *within an architecture view* can be used to express architectural textures (see [34]). Architecture models can be used as "containers" for applying architecture patterns [4] or architecture styles to express fundamental schemes (such as layers, three-tier, peer-to-peer, model-view-controller) within architecture views.

NOTE 2 This International Standard does not prescribe how architecture models are created. They can be individually constructed, derived from or based upon other models.

5.7 Architecture relations

5.7.1 Consistency within an architecture description

An architecture description shall record any known inconsistencies across its architecture models and its views.

NOTE While consistent architecture descriptions are to be preferred, it is sometimes infeasible or impractical to resolve all inconsistencies for reasons of time, effort, or insufficient information. In such situations, known inconsistencies are to be recorded.

An architecture description should include an analysis of consistency of its architecture models and its views.

Correspondences and correspondence rules, as specified in 5.7.2 and 5.7.3, may be used to express, record, enforce and analyze consistency between models, views and other AD elements within an architecture description.

5.7.2 Correspondences

Each correspondence in an architecture description shall be identified and identify its participating AD elements.

AD elements may be instances of any construct introduced in 4.2 (stakeholders, system concerns, architecture viewpoints, architecture views, model kinds, architecture models, architecture decisions and rationale). When viewpoints and model kinds are defined, additional kinds of AD elements may be introduced.

Each correspondence in an architecture description shall identify any correspondence rules governing it (see 5.7.3).

NOTE AD elements in a correspondence need not be distinct. A correspondence can be defined between an AD element and itself.

5.7.3 Correspondence rules

An architecture description shall include each correspondence rule applying to it.

NOTE 1 A correspondence rule applying to an architecture description could originate in the architecture description, in a viewpoint (see Clause 7) or in an architecture framework or architecture description language (see Clause 6).

For each identified correspondence rule, an architecture description shall record whether the rule holds or otherwise record all known violations.

A correspondence rule **holds** if an associated correspondence can be shown to satisfy the rule. A correspondence rule is **violated** if an associated correspondence can be shown not to satisfy the rule or when no associated correspondence exists.

NOTE 2 Correspondences in this International Standard are designed to be compatible with view correspondences in RM-ODP [ISO/IEC 10746 and 19793] (see A.6).

NOTE 3 Correspondences and correspondence rules can be applied to multiple architecture descriptions to express relations pertaining to multiple architectures or systems. By generalizing AD element to other information items, a project and/or organization could apply correspondences as defined herein between architecture descriptions and other work products (such as requirements specifications) to express other relations of architectural interest (such as traceability of AD elements to requirements).

5.8 Architecture rationale

5.8.1 Rationale recording

An architecture description shall include a rationale for each architecture viewpoint included for use per 5.4 in terms of its stakeholders, concerns, model kinds, notations and methods.

An architecture description shall include rationale for each decision considered to be a key architecture decision (per 5.8.2).

An architecture description should provide evidence of the consideration of alternatives and the rationale for the choices made.

5.8.2 Decision recording

An architecture description should record architecture decisions considered to be key to the architecture of the system-of-interest.

It is not practical to record every architecture decision about a system. A decision recording and sharing strategy should be applied by the organization and/or project to establish criteria for selecting key decisions to be recorded and supported with rationales in the architecture description. Criteria to consider are:

— decisions regarding architecturally significant requirements;

— decisions needing a major investment of effort or time to make, implement or enforce;

— decisions affecting key stakeholders or a number of stakeholders;

— decisions necessitating intricate or non-obvious reasoning;

— decisions that are highly sensitive to changes;

— decisions that could be costly to change;

— decisions that form a base for project planning and management (for example, work breakdown structure creation, quality gate tracking);

— decisions that result in capital expenditures or indirect costs.

When recording decisions, the following should be considered:

— the decision is uniquely identified;

— the decision is stated;

— the decision is linked to the system concerns to which it pertains;

— the owner of the decision is identified;

— the decision is linked to AD elements affected by the decision;

— there is rationale linked to the decision in accordance with 5.8.1;

— constraints and assumptions that influence the decision are identified;

— alternatives that have been considered, and their potential consequences, are recorded;

— consequences of the decision (relating to other decisions) are recorded;

— timestamps record when the decision was made, when it was approved and when it was changed;

— citations to sources of additional information are provided.

NOTE 1 It can be useful to record rejected alternatives and the rationale for those rejections. It can be the case in the future that these reasons no longer apply and the decision needs to be reconsidered.

NOTE 2 It can be useful to record relationships between architecture decisions. Examples of types of relationships are: constrains, influences, enables, triggers, forces, subsumes, refines, conflicts-with, and is-compatible-with (see [23, 44]).

6 Architecture frameworks and architecture description languages

6.1 Architecture frameworks

An architecture framework shall include:

a) information identifying the architecture framework;

b) the identification of one or more concerns (per 5.3);

c) the identification of one or more stakeholders having those concerns (per 5.3);

d) one or more architecture viewpoints that frame those concerns (per 7);

e) any correspondence rules (per 5.7).

The verb *include* when used in Clause 6 indicates that either the information is present in the architecture framework or reference to that information is provided therein.

An architecture framework should include conditions of applicability.

EXAMPLES The following are conditions of applicability:

— An architecture description using architecture framework *AF1* needs to identify stakeholders *A*, *M*, and *P* when the system-of-interest operates within jurisdiction *J*.

— An architecture description using architecture framework *AF2* is permitted to omit viewpoint *V1* when no real-time system concerns have been identified.

— When using architecture framework *AF3*, model kind *MK* can be omitted in viewpoint *V2* unless *S* is an identified stakeholder.

An architecture framework shall establish its consistency with the provisions of the conceptual model in 4.2.

NOTE The above requirement can be met through a metamodel, a mapping of framework constructs to the model in 4.2, a text narrative, or in some other manner.

6.2 Adherence of an architecture description to an architecture framework

An architecture description *adheres to* an architecture framework when:

— each applicable stakeholder identified in the architecture framework has been considered and identified in the architecture description (per 5.3);

— each applicable concern identified in the architecture framework has been considered and identified in the architecture description (per 5.3);

— each applicable viewpoint specified by the architecture framework (per 6.1) is included (per 5.4) in the architecture description;

— each applicable correspondence rule specified by the architecture framework is included in the architecture description (per 5.7.3); and

— the architecture description conforms to the requirements of Clause 5.

Applicable means when conditions of applicability (see 6.1) are met.

An architecture framework may establish additional rules for adherence.

NOTE An architecture description could adhere to one or more architecture frameworks, or to no frameworks. For an architecture description to adhere to more than one framework would entail a reconciliation between each framework's identified stakeholders, concerns, viewpoints, model kinds and correspondence rules within the architecture description.

6.3 Architecture description languages

An architecture description language shall specify:

a) the identification of one or more concerns to be expressed by the ADL (per 5.3);

b) the identification of one or more stakeholders having those concerns (per 5.3);

c) the model kinds implemented by the ADL which frame those concerns (per Clause 7, d));

d) any architecture viewpoints (per Clause 7);

NOTE An ADL need not provide any architecture viewpoints; it can define one or more model kinds for use in architecture viewpoints defined elsewhere.

e) correspondence rules (per 5.7) relating its model kinds per c).

7 Architecture viewpoints

An architecture viewpoint shall specify:

a) one or more concerns framed by this viewpoint (per 5.3);

b) typical stakeholders for concerns framed by this viewpoint (per 5.3);

c) one or more model kinds used in this viewpoint;

d) for each model kind identified in c), the languages, notations, conventions, modelling techniques, analytical methods and/or other operations to be used on models of this kind;

e) references to its sources.

NOTE 1 Item d) can be met with a metamodel for the model kind that defines the structure and conventions of its models. Item e) can include author, date, URLs, and/or citations to other documents.

An architecture viewpoint should include information on architecting techniques used to create, interpret or analyze a view governed by this viewpoint, such as:

— correspondence rules, criteria and methods for checking consistency (see 5.7.1) and completeness (see 5.5 d);

— evaluation or analysis methods;

— methods, heuristics, metrics, patterns, design rules or guidelines, best practices and examples to aid in view creation and synthesis.

An architecture viewpoint could be defined as a part of an architecture description (Clause 5), as a part of an architecture framework (Clause 6) or individually using the requirements of this Clause. A *library viewpoint* is an architecture viewpoint produced outside of the context of a single architecture description such that it can be used in many architecture descriptions.

NOTE 2 This International Standard does not require any particular viewpoints to be used.

NOTE 3 Annex B provides guidance on specifying viewpoints. Annex C provides examples of architecture viewpoints.

Annex A
(informative)

Notes on terms and concepts

A.1 Introduction

This Annex discusses the design principles, concepts and terms on which this International Standard is based.

This International Standard defines minimal requirements on architecture descriptions to support the scope established in Clause 1. The approach is to allow using organizations maximum flexibility in applying the standard while demonstrating conformance with the requirements in Clauses 5, 6 and 7. Given the multi-disciplinary nature of architecting, the intent is to meet the needs of multiple stakeholders and allow different ways to describe a system. The organization of architecture descriptions into views using viewpoints provides a mechanism for the separation of concerns among the stakeholders, while providing the view of the whole system that is fundamental to the notion of architecture.

Establishing the quality of an architecture being described by a conforming architecture description (*Is this a good architecture?*) or the quality of an architecture description itself (*Is this architecture description complete?*) are factors for the *evaluation* of the architecture description. This International Standard does not presume to impose conditions that are required for quality considerations. It does require that results of such evaluations be recorded, per 5.2. Quality evaluations of architectures and of architecture descriptions are subjects for future standardization efforts.

This International Standard makes use of several terms: *architecture*, *concern*, *model*, *view*, and *viewpoint*, which are in wide usage with several different meanings. This Annex discusses these terms, the motivations for their definitions in this International Standard, and contrasts these definitions with other usages.

A.2 Systems and architectures

In this International Standard, the term *architecture* is intended to convey the essence or fundamentals of a system. There are several key aspects to the definition of architecture in this International Standard (3.2). This definition was chosen to encompass a variety of previous uses of the term “architecture” by recognizing their underlying common themes. Principal among these is the need to understand and control those elements of a system-of-interest that contribute to its utility, cost, time and risk within its environment. In some cases, the fundamental elements are physical or structural components of the system and their relationships. Sometimes, the fundamental elements are functional or logical elements. In other cases, what is fundamental or essential to the understanding of a system-of-interest are its overarching principles or patterns. The definition of architecture in this International Standard is intended to encompass these distinct, but related uses, while encouraging a more rigorous delineation of what constitutes the architecture of a system.

The phrase “concepts or properties” is used in the definition (3.2) to allow two differing philosophies to use this International Standard without prejudice. These two philosophies are: *Architecture as Concept*: architecture (of a system) is a conception of a system in one’s mind; and *Architecture as Property*: architecture (of a system) is a property of that system.

Empirical studies have discovered four metaphors for architecture found in organizations [39]:

— architecture as blueprint;

— architecture as literature;

— architecture as language;

— architecture as decision.

The conceptual foundation of this International Standard does not presume any one of these metaphors; rather it works equally well with any of them. The existence of these multiple metaphors supports a central design tenet of this International Standard: that architecture is inherently based upon multiple stakeholders with multiple system concerns.

A.3 Concerns

The International Standard uses the term *concern* to mean any topic of interest pertaining to the system. The stakeholders of a system hold these concerns. Some concerns drive the architecture and therefore this International Standard requires their identification as a part of the description of that architecture.

The motivation for this term comes from the phrase “separation of concerns” in software and systems engineering, coined by Edsger W. Dijkstra in 1974:

> Let me try to explain to you, what to my taste is characteristic for all intelligent thinking. It is, that one is willing to study in depth an aspect of one’s subject matter in isolation for the sake of its own consistency, all the time knowing that one is occupying oneself only with one of the aspects. We know that a program must be correct and we can study it from that viewpoint only; we also know that it should be efficient and we can study its efficiency on another day, so to speak. In another mood we may ask ourselves whether, and if so: why, the program is desirable. But nothing is gained—on the contrary!—by tackling these various aspects simultaneously. It is what I sometimes have called “the separation of concerns”, which, even if not perfectly possible, is yet the only available technique for effective ordering of one's thoughts, that I know of. This is what I mean by “focussing one’s attention upon some aspect”: it does not mean ignoring the other aspects, it is just doing justice to the fact that from this aspect’s point of view, the other is irrelevant. It is being one- and multiple-track minded simultaneously. [7]

As specified in this International Standard, each architecture viewpoint frames one or more concerns (see 5.4) so that a view corresponding to the viewpoint addresses specific known concerns for the system-of-interest. Separating the treatment of concerns by views allows interested stakeholders to focus on a few things at a time and offers a means of managing complexity (see 5.5). The literature of systems and software engineering records a large inventory of such concerns. Examples are given in 4.2.3.

Although concerns include risks and hazards (see 5.3), the term should not be understood to be synonymous with “risks” or “worries”, but as referring to *any* topic of interest.

A.4 Architecture views and viewpoints

The terms *architecture view* and *architecture viewpoint* are central to this International Standard. Although sometimes used synonymously, in this International Standard they refer to distinct kinds of things.

It is a goal of the International Standard to encompass existing architecture description practices by providing common terminology and concepts. Many existing practices express architectures through collections of models. Typically, these models are further organized into cohesive groups, called *views*. The cohesion of a group of models is determined by the concerns addressed by that group of models. What has been missing in recent practice is a distinct term for the mechanism for formalizing these groupings and referring to the conventions by which the models are made. In this International Standard, *viewpoint* refers to the conventions for expressing an architecture with respect to a set of concerns:

> A viewpoint is a way of looking at systems; a view is the result of applying a viewpoint to a particular system-of-interest.

The use of multiple views to express an architecture is a fundamental premise of the International Standard. The need for multiple views in architecture descriptions is widely recognized. 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, this International Standard does not require a predefined set of viewpoints; it encourages the practice of defining or selecting viewpoints appropriate to the system-of-interest, and treating viewpoints as first-class elements of architecture descriptions.

The earliest work on first-class viewpoints appears in Ross' Structured Analysis (SADT) in 1977 [35]. In requirements engineering, Nuseibeh, Kramer and Finkelstein treat viewpoints as first-class entities, with associated attributes and operations [29]. These works inspired the formulation of architecture viewpoints as specified in Clause 7. The term was also chosen to align with the ISO Reference Model of Open Distributed Processing (RM-ODP), which uses the term in these ways:

> A *viewpoint* (on a system) is an abstraction that yields a specification of the whole system related to a particular set of concerns. [ISO/IEC 10746-1:1998, 6.2.2]
>
> *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. [ISO/IEC 10746-2:2009, 3.2.7]

However, where this International Standard uses *architecture view* to refer to the application of a viewpoint to a particular system, RM-ODP uses the term *viewpoint specification*.

The relationship between viewpoint and view suggests this metaphor:

> view : viewpoint :: program : programming language[2]

A viewpoint specifies the conventions (such as notations, languages and types of models) for constructing a certain kind of view. That viewpoint can be applied to many systems. Each view is one such application. Similarly, a program is one instance of applying a programming language to a specific situation or design problem.

Another metaphor to understand the difference between view and viewpoint is:

> view : viewpoint :: map : legend

A legend defines the conventions used in preparing a map (such as its scale, colors and other symbology) to aid readers in interpreting that map as intended. Just as every map should have a legend, every architecture view should have an architecture viewpoint specifying the conventions for interpreting the contents of the view.

Another term, *viewtype*, introduced by Clements *et al*. [5], establishes a categorization of viewpoints in the terms of this International Standard. Three categories of viewpoints are described in their work: module, component and connector, and allocation viewtypes.

Within an individual architecture description, this International Standard requires that each view needs to be governed by exactly one viewpoint. This means each view conforms to one set of conventions (possibly multiple model kinds). This requirement does not preclude users of this International Standard combining or composing architecture viewpoints for specific purposes (in a manner not defined by this International Standard) as long as the requirement is met *within* an individual architecture description.

In this International Standard, each architecture view needs to represent the *whole system* from the perspective of the system concerns framed by its governing viewpoint. This reflects the holistic nature of architecture. For example, a performance view of a networked system should consider both network

---

[2] This is to be read, "a view is to a viewpoint as a program is to a programming language."

transmission delays (in one model) and processing times (in another model) to produce a holistic end-to-end view of the performance of the entire system.

An architecture description can focus on the system-of-interest at a specific point of time (for example, when it is delivered to a customer), or consider the evolution of the system over many time scales. Any view can be composed of a series of models, each representing the system-of-interest at a given point of time. The composition of such models within a view would describe how that system evolves over time, while still meeting the requirement that the view deals with the whole system.

There are two common approaches to the construction of views: the synthetic approach and the projective approach. In the synthetic approach, an architect constructs views of the system-of-interest and integrates these views within an architecture description using model correspondences. In the projective approach, an architect derives each view through some routine, possibly mechanical, procedure of extraction from an underlying repository. This International Standard is usable with either of these approaches to views.

Mary Shaw writes [38]:

> Routine design involves solving familiar problems, reusing large portions of prior solutions. … Most engineering disciplines capture, organize, and share design knowledge to make routine design simpler. Handbooks and manuals are often the carriers of this organized information.

The "reusable" nature of architecture viewpoints (and architecture frameworks, as coordinated viewpoint sets) highlights their utility as mechanisms for capturing strategic architectural knowledge, within an organization or within the larger architecting community. Viewpoints codify application-specific, method-specific, or organization-specific approaches, and thereby support the growth and evolution of architecture practices.

Annexes B and C provide further information and references pertaining to architecture viewpoints.

A.5 Models, work products and architecture models

Models and modelling underlie much of systems and software architecture. The notion of *model* is central to understanding this International Standard. Different communities use *model* in different ways. Therefore, it is important to understand the term as used in this International Standard:

> *M* is a *model* of *S* if *M* can be used to answer questions about *S*.[3]

This statement has two important consequences:

(1) Every model has a subject.

(2) A model can be *anything*: (i) a model can be a concept (a "mental model"); or (ii) a model can be a *work product*.

In this International Standard, the term *model* is used in two ways. First, it is used in its ordinary language sense, as explicated above. Second, it is used in a specialized sense to define a key part of architecting, embodied in the term *architecture model* (see 5.6).

---

[3] This definition originated at the MIT Research Lab for Electronics (RLE) during the 1960s. It appears in the work of D.T. Ross and M. Minsky, who were both at RLE during that time period:

"To an observer B, an object A* is a model of an object A to the extent that B can use A* to answer questions that interest him about A." M. Minsky, *Matter, Mind and Models*, 1968.

"M is a model of A with respect to question set Q if and only if M may be used to answer questions about A in Q within tolerance T." D.T. Ross, *Technical Foundations for Characterizations*, 1977.

In the first sense of the term, there are several kinds of models related to architecting that are described in this International Standard. The difference between (2)(i) and (2)(ii) is crucial to understanding the distinction in the International Standard between an architecture and an architecture description. In the sense of (2)(i), an architecture is a conception (i.e., a mental model) of a system—useful for answering some questions about that system. In the sense of (2)(ii), there are three kinds of models defined by the International Standard realized as work products:

— an architecture description is a work product which models the architecture of a system-of-interest; its subject is the identified stakeholders' questions about all identified system concerns for that system;

— an architecture view is a work product; its subject is a specific set of stakeholder concerns framed by its governing viewpoint;

— an architecture model is a work product; its subject is determined by its model kind.

Where a *work product* is understood in this International Standard as an "artefact associated with the execution of a process" [ISO/IEC 15504-1:2004, 3.55].

A.6 Correspondences

Whenever multiple models are developed of a subject, these models can be inconsistent. In architecture descriptions, one consequence of employing multiple views is the need to express and maintain consistency between those views.

In the 2000 edition of the standard, IEEE Std 1471, this need was recognized in terms of requirements to analyze and record known inconsistencies across an architecture description's views (see 5.7.1). At that time, there was no well-established practice to be codified by the standard for expressing or enforcing such consistency.

This edition of the International Standard introduces *correspondences* to express relations between architecture description elements (AD elements). Correspondences have a number of uses. They can be used to express consistency, traceability, composition, refinement and model transformation, or dependences of any type spanning more than a single model kind. A survey of uses of model relations together with a taxonomy and classification of relation mechanisms is found in [2]. Correspondences can be used to meet the requirements of 5.7.1 for recording view consistency and inconsistencies.

The remainder of this sub-clause presents examples of correspondences and correspondence rules. The features of the correspondence mechanism, in relation to similar mechanisms in the literature, are discussed. EXAMPLE 1 presents a simple model correspondence.

EXAMPLE 1 Consider two views of a system *S*: a hardware view, *HW(S)*, and a software component view, *SC(S)*. Given that *SC(S)* includes software elements, *e1*, … *e6*, and *HW(S)* includes hardware platforms, *p1*, … *p4*, a correspondence expressing which software elements execute on which platforms is shown in Figure A.1.

(Element) ExecutesOn (Platforms) See rule: **R1**	
e1	p1, p4
e2	p2, p3
e3	p3
e4	p4

Figure A.1 — Example of a correspondence

The example meets the requirement of 5.7.2: it has a unique name (*ExecutesOn*), identifies participating elements (the *ei*s and *pj*s), and identifies an optional correspondence rule (**R1**).

A correspondence rule expresses a constraint to be enforced on a correspondence. EXAMPLE 2 presents a simple correspondence rule.

EXAMPLE 2 Consider two viewpoints, Hardware and Software Components. A correspondence rule relating the two is:

> **R1**: Every software element, *ei*, as defined by Software Components needs to execute on one or more platforms, *pj*, as defined by Hardware.

The correspondence *ExecutesOn* from EXAMPLE 1 violates rule **R1** of EXAMPLE 2 because some software elements of *SC(S)* (*e5* and *e6*) have not been assigned to execute on any platform.

Most correspondences will be expressed in terms of elements of the participating models, but this is not required. EXAMPLES 3 and 4 show other forms of correspondences.

EXAMPLE 3 Consider the following correspondence rule:

> **Tasks-Interactions**: Every instance of model kind *Tasks* needs to have a refinement to an instance of model kind *Interactions*.

This model correspondence rule could be satisfied by the correspondence shown in Figure A.2 where there are Users, Operators and Auditors. Each *Task* model (illustrated as a triangle) is refined into an *Interaction* model (illustrated as a pentagon).

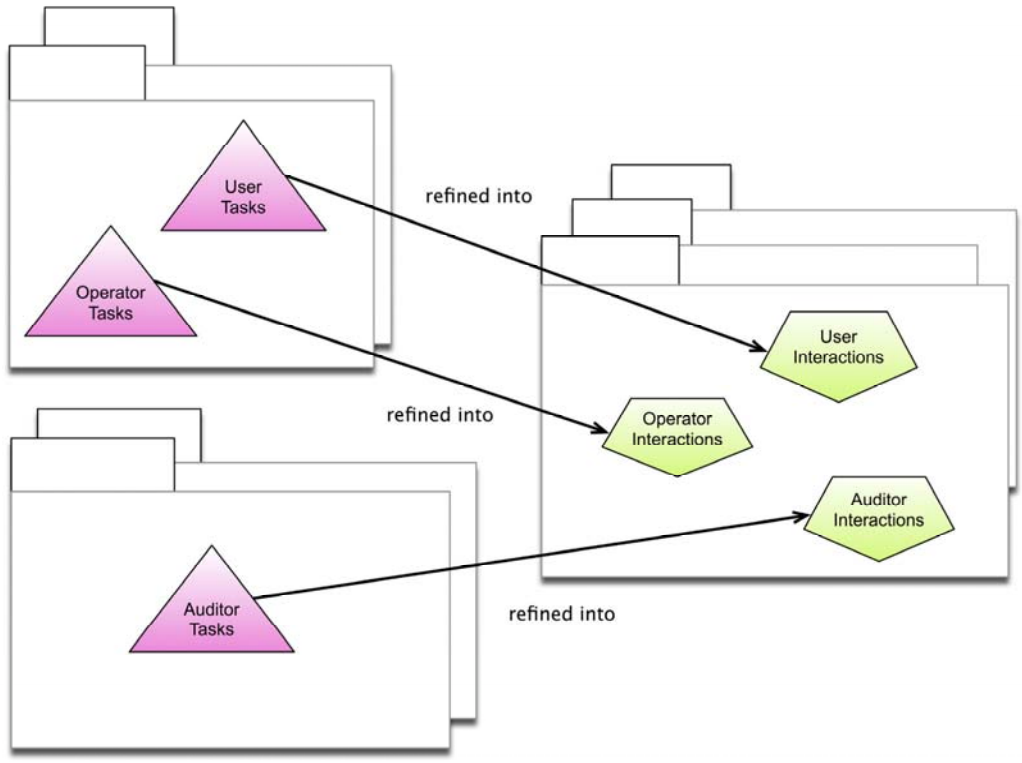


Figure A.2 — Example of a correspondence satisfying the Task-Interactions rule

In EXAMPLE 3 the participants in the correspondence are not elements of the models, but the models themselves. A correspondence can relate any AD elements (see 4.2.5 and 5.7.2); users of the International Standard are free to introduce other types of AD elements suited to their purposes.

Many correspondences will be binary, but this is not required. A correspondence can relate an arbitrary number of AD elements. EXAMPLE 4 illustrates an *n*-ary correspondence rule.

EXAMPLE 4 Consider the following model correspondence rule:

View-Versioning: The version identifier of each view needs to be greater than 1.5 prior to publication.

The term “correspondence” was chosen to align with RM-ODP. The correspondence mechanism is designed to be compatible with view correspondences in RM-ODP [ISO/IEC 19793]; however there are some differences. Notable differences are:

(1) The term “correspondence” is used in this International Standard rather than “view correspondence”. In RM-ODP, each view is homogeneous—a single viewpoint language is used per viewpoint specification. This International Standard permits heterogeneous views: each view is composed of one or more architecture models wherein each model utilizes a different modelling language (see 5.6). It is useful to be able to state a correspondence between models in different modelling languages, not just between views. Therefore, “view correspondence” is a special case of what is needed in this International Standard, and that term is somewhat misleading in this more general case;

(2) RM-ODP view correspondences are binary relations whereas model correspondences in this International Standard are *n*-ary relations; and

(3) RM-ODP view correspondences are defined on elements of viewpoint specifications whereas model correspondences in this International Standard need not refer to individual elements of models, but arbitrary AD elements.

(4) Correspondences and correspondence rules can be used to express relations across architecture descriptions.

Mathematically, a correspondence is an n-ary relation. A correspondence rule is an intensional definition of an *n*-ary relation. Relations include 1-1 mappings (isomorphisms) and functions as special cases, both of which are too restrictive for many applications of correspondences. Relations have useful properties which permit composition and reasoning, and allow efficient representation and manipulation (see [28] and references therein). EXAMPLE 5 shows some of the above examples expressed as relations in set notation.

EXAMPLE 5

ExecutesOn (R1) = { (e1, p1), (e1, p4), (e2, p2), (e2, p3), (e3, p3), (e4, p4) }

Users (Tasks-Interactions) =
{ (OperatorTasks, OperatorInteractions), (CustomerTasks, CustomerInteractions), (AuditorTasks, AuditorInteractions) }

LatestVersion (View-Versioning) = { (view1, v2.0), (view2,v2.0), (view3,v2.0), (view4,v2.0), (view5,v2.0) }

A.7 Architecture frameworks and architecture description languages

In systems and software engineering, the notion of *architecture framework* dates back to the 1970s [6, 44]. The motivation for the definition of the term (3.6) and its specification (in 6.1) in this International Standard is to provide a means of defining existing and future architecture frameworks in a uniform manner to promote sharing of information about systems, architectures and techniques for architecture description, inter-working to enable improved understanding, and interoperability between architecture communities who are using different conceptual foundations. The uniform definition of architecture viewpoints and coordinated collections of such viewpoints can promote reuse of tools and techniques to the communities using these frameworks.

The specification of architecture framework is intended to establish the relationships between an architecture framework and other concepts in this International Standard (illustrated in figures 2 and 4). Architecture frameworks often include additional content, prescriptions and relationships, such as process requirements, life cycle connections, and documentation formats, not defined by this International Standard, but potential future areas of standardization.

The term *architecture description language* (ADL) has been in use since the 1990s in the software, systems and enterprise architecture communities. Within the conceptual model of this International Standard, an architecture description language is any language for use in an architecture description. Therefore an ADL can be used by one or more viewpoints to frame identified system concerns within an architecture description.

Early ADLs included Rapide (Stanford) [25], Wright (CMU) [43], and Darwin (Imperial College). ADLs focused on structural concerns: large-scale system organization expressed in terms of components, connectors and configurations and varying support for framing behavioral concerns. More recently, “wide-spectrum” ADLs have been developed which support a wider range of concerns. These include Architecture Analysis & Description Language (AADL) [37], SysML [31], and ArchiMate [40]. EXAMPLES 1 and 2 describe two contemporary ADLs with reference to their relationship to the conceptual model defined in this International Standard.

EXAMPLE 1 ArchiMate organizes ADs into several *layers of concerns*: Business, Application and Technology (or Infrastructure); several *aspects of concerns* within each of those layers: Structural, Behavioral and Informational aspects, and defines eighteen basic viewpoints for these. Each viewpoint is defined via its own metamodel, relating that viewpoint to others, and specifying, the stakeholders, concerns, purpose, layers and aspects.

EXAMPLE 2 The Systems Modeling Language (SysML) is built upon UML. SysML defines several types of *diagrams*: Activity, Sequence, State Machine, Use Case, Block Definition, Internal Block, Package, Parametric, and Requirement diagrams. In the terms of this International Standard, each SysML diagram type is a model kind. SysML provides first-class constructs for Stakeholders, Concerns, Views and Viewpoints so that users can create new viewpoints in accordance with this International Standard.

Like an architecture framework, an ADL frames a specific set of concerns for an audience of stakeholders, by defining one or more model kinds together with any associated analysis methods or tools. Similar to an architecture framework or architecture viewpoint, an ADL is a reusable resource—it is not limited in use to an individual system or architecture description.

Annex B
(informative)

Guide to architecture viewpoints

B.1 Introduction

This Annex provides a template for documenting architecture viewpoints and an annotated guide to a sample of currently available viewpoints.

B.2 A template for documenting architecture viewpoints

B.2.1 Template overview

A template for architecture viewpoints is presented. An architecture viewpoint that is documented in this form will meet the requirements of Clause 7.

The template consists of a set of *slots* or information items (B.2.2 through B.2.11). Each slot is identified by a name (B.2.X **Slot name**) followed by a brief description of its intended content, guidance for developing that content, and in some cases "sub slots". Not every slot is needed for documenting every viewpoint. This template is based on one proposed in [9].

B.2.2 Viewpoint name

The name for the viewpoint. If there are synonyms or other common names by which the viewpoint is known, record them here.

B.2.3 Viewpoint overview

An abstract or brief overview of the viewpoint and its key features.

B.2.4 Concerns and "anti-concerns"

A listing of the architecture-related concerns to be framed by this viewpoint per 7 a). This is critical information for an architect, because it helps her decide whether this viewpoint will be useful for a particular system-of-interest.

It can be useful to document the kinds of issues a viewpoint is *not appropriate for*. Articulating anti-concerns can be a good antidote for certain overly used models and notations.

B.2.5 Typical stakeholders

A listing of the system stakeholders expected to be users or audiences for views prepared using this viewpoint per 7 b).

NOTE When a viewpoint is selected for use and applied in an architecture description, the AD needs to document the association of actual system stakeholders with concerns framed by each viewpoint (per 5.3)

B.2.6 Model kinds

B.2.6.1 Introduction

Identify each model kind specified by the viewpoint per 7 c).

For each model kind used, describe its conventions, language or modelling techniques. These are key modelling resources that the viewpoint makes available and determine the vocabularies for constructing the view. These include operations on models of the model kind (B.2.6.5)

The International Standard does not specify one style for documenting model kinds. A model kind may be documented in a number of ways, including:

(1) by specifying a metamodel that defines its core constructs;

(2) by providing a model template to be filled in by users;

(3) via a language definition or by reference to an existing modelling language; or

(4) by some combination of these methods.

Guidance on methods (1) through (3) is provided below.

B.2.6.2 Model kind: metamodel

A metamodel presents the AD elements that comprise the vocabulary of a model kind. There are different ways of representing metamodels. The metamodel should present:

— **entities**: What are the major sorts of elements that are present in models of this kind?

— **attributes**: What properties do entities possess in models of this kind?

— **relationships**: What relations are defined among entities in models of this kind?

— **constraints**: What kinds of constraints are there on entities, attributes and/or relationships in models of this kind?

Entities, attributes, relationships and constraints are all AD elements in the sense of 3.4 (also see 4.2.5 and 5.7).

NOTE When a viewpoint specifies multiple model kinds it is often useful to specify a single viewpoint metamodel unifying the definition of the model kinds. Furthermore, it is often helpful to use a single metamodel to express multiple, related viewpoints (such as when defining an architecture framework).

B.2.6.3 Model kind: templates

Provide a template or form specifying the format and/or content of models of this model kind.

B.2.6.4 Model kind: languages

Identify an existing notation or model language or define one that can be used for models of this model kind. Describe its syntax, semantics, tool support, as needed.

B.2.6.5 Model kind: operations

Define operations available on models of the kind. See the discussion of operations on views in B.2.8.

B.2.7 Correspondence rules

Document any correspondence rules defined by this viewpoint or its model kinds. Usually, these rules will be "cross model" or "cross view" since constraints *within* a model kind will have been specified as part of the conventions of that model kind.

B.2.8 Operations on views

Operations define the methods to be applied to views or to their models. Operations can be divided into categories:

— *Creation methods* are the means by which views are prepared using this viewpoint. These could be in the form of process guidance (how to start, what to do next); or work product guidance (templates for views of this type); heuristics, styles, patterns, or other idioms.

— *Interpretive methods* are the means by which views are to be understood by the reader and system stakeholders.

— *Analysis methods* are used to check, reason about, transform, predict, apply and evaluate architectural results from this view.

— *Design* or *implementation methods* are used to realize or construct systems using information from this view.

B.2.9 Examples

This section provides examples for the reader.

B.2.10 Notes

Any additional information that users of this viewpoint might need or find helpful.

B.2.11 Sources

Identify the sources for this viewpoint, if any, including author, history, literature references, prior art, per 7 e).

B.3 Annotated guide to architecture viewpoints

The following represent some resources for well-documented architectural viewpoints. Not all of these are documented in accordance with the requirements of this International Standard, but could be used in an architecture description or included in an architecture framework in a conforming manner.

— Callo-Arias, America, and Avgeriou, "Defining execution viewpoints for a large and complex software-intensive system" [4]

  Documents an "execution viewpoint catalog" for understanding the execution of complex software-intensive systems. The four viewpoints are: Execution Profile, Execution Deployment, Resource Usage and Execution Concurrency. Correspondence rules between the viewpoints are also included.

— Clements, et al., *Documenting Software Architectures: views and beyond* [5]

  Provides extensive resources for defining 3 categories of viewpoints. These categories, called *viewtypes* (see A.4), are Module, Component and Connector and Allocation viewtypes. Within each viewtype, a number of styles are defined.

— Eeles and Cripps, *The Process of Software Architecting* [8]

Defines a process for software architects, using the IEEE 1471:2000 model as a foundation. Provides a viewpoint template and viewpoint catalog including: Requirements, Functional, Deployment, Validation, Application, Infrastructure, Systems Management, Availability, Performance, Security; and the "work products" (i.e., model kinds) for each.

— ISO/IEC 42010 Viewpoints Repository [42]

The website is a repository for architecture viewpoints submitted by the community.

— Kruchten, "The '4+1' view model of architecture" [23]

Defines viewpoints for Logical, Development, Process and Physical views. The resulting views are integrated via Scenarios.

— Rozansky and Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives* [36]

Defines a catalog of viewpoints: Functional, Information, Concurrency, Development, Deployment and Operational viewpoints and *perspectives* (see 5.6, NOTE 1): Security, Performance and Scalability, Availability and Resilience, and Evolution perspectives for software-intensive systems.

Annex C
(informative)

Relationship to other standards

C.1 Introduction

Architecture descriptions can be used in a variety of settings and life cycle models. This annex illustrates how architecture descriptions created in accordance with this International Standard can meet the requirements of other standards. A general approach for meeting the requirements of other standards while using this International Standard is to define one or more viewpoints that frame the system concerns pertaining to those requirements and then produce views satisfying those requirements as a part of an architecture description. The viewpoints defined in this Annex are specified here in accordance with the requirements of Clause 7.

C.2 Use with ISO/IEC 12207:2008

C.2.1 General

ISO/IEC 12207:2008 defines two processes specifically pertaining to architecture: system architectural design (see ISO/IEC 12207:2008, 6.4.3) and software architectural design (see ISO/IEC 12207:2008, 7.1.3). The concept of architecture in this International Standard is consistent with the architectural design processes of ISO/IEC 12207. However, ISO/IEC 12207 places requirements on an architecture description in addition to those of this International Standard. Specifically, a system architectural design needs to include an identification of the hardware items, software items and manual operation items included in the system and an allocation of system requirements to those items. System architectural designs need to be evaluated against criteria of traceability to, and consistency with, system requirements, appropriateness of design standards and methods, and feasibility of software and manual operations.

The expected use of an architecture description may include other ISO/IEC 12207 processes. 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.

The Software Architectural Design Process of ISO/IEC 12207 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 architecture description and the products of other views in the architecture description would contribute to this activity and its products.

An architecture description can conform to this International Standard and to ISO/IEC 12207. One approach to “joint conformance” is to have a viewpoint that is specifically focused on producing the ISO/IEC 12207 architectural products. An example of a viewpoint for this purpose is defined in C.2.2.

C.2.2 Decomposition and allocation viewpoint

The Decomposition and Allocation viewpoint frames these concerns:

— identification of the system requirements;

— decomposition of the system into items;

— allocation of the requirements to the items;

— verification that all requirements are allocated to items.

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 ISO/IEC 12207:2008, 6.4.3.3.1).

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 ISO/IEC 12207:2008, 7.1.3.3.1).

C.3 Use with ISO/IEC 15288:2008

C.3.1 General

ISO/IEC 15288:2008 defines one process specifically pertaining to architecture: architectural design. The concept of architecture in this International Standard is consistent with the architectural design process of ISO/IEC 15288. However, ISO/IEC 15288 places requirements on an architectural description in addition to those herein. Specifically, an architectural design needs to include an identification of the system elements included in the system and an allocation of system requirements to those items.

The expected use of an architectural description may include other ISO/IEC 15288 processes. 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.

An architecture description can conform to this International Standard and to ISO/IEC 15288. One approach to ‘joint conformance’ is to have a viewpoint that is specifically focused on producing the ISO/IEC 15288 architectural products. An example of a viewpoint defined for this purpose is defined in C.3.2.

C.3.2 Decomposition and allocation viewpoint

The Decomposition and Allocation viewpoint frames these concerns:

— identification of the system requirements;

— decomposition of the system into items;

— allocation of the requirements to the items;

— verification that all requirements are allocated to items.

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 elements. Interfaces between elements are also identified.

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

The initial decomposition and allocation produces a set of elements with allocated requirements. This is described in terms of a Software Architectural Description (see ISO/IEC 15288:2008, 6.4.3.3 (c)).

C.4 Use with Open Distributed Processing standards

C.4.1 General

The Reference Model of Open Distributed Processing (RM-ODP) defines an architecture 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.” (See ISO/IEC 10746-2.)

The RM-ODP framework defines five viewpoints for specifying ODP systems and a set of correspondences between them.

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

An architecture description conforming to this International Standard and using ISO/IEC 10746-3 would include the viewpoints defined by ISO/IEC 10746-3 and views to implement these viewpoints. A conforming architecture description need not be limited to the five predefined viewpoints of ISO/IEC 10746-3; the architecture description may include additional viewpoints and views, as needed.

Elements of that specification specific to architecture 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 ISO/IEC 10746-3:1996.

NOTE ISO/IEC 19793 defines a UML profile for the specification of open distributed processing systems using these viewpoints.

C.4.2 Enterprise viewpoint

The Enterprise viewpoint frames these 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.

In the Enterprise Language, an ODP system and its environment are represented as a community of objects. The community is defined in terms of:

— enterprise objects comprising 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.

NOTE 1 Roles constrain the behavior of the objects that fulfil them.

NOTE 2 Policies are defined in terms of permissions, obligations, and prohibitions.

NOTE 3 The Enterprise Language is defined in ISO/IEC 15414.

C.4.3 Information viewpoint

The Information viewpoint frames these concerns: the semantics of information and information processing in an ODP system.

The Information Language is defined in terms of three schemata:

— invariant schema: predicates on objects which always need to 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.

C.4.4 Computational viewpoint

The Computational viewpoint frames these concerns: a functional decomposition of the system into objects which interact at interfaces.

The Computational Language covers concepts for specifying:

— computational objects;

— interfaces to objects and interface definitions;

— interactions at interfaces, as either operations or continuous streams;

— implicit and explicit bindings and compound binding objects.

C.4.5 Engineering viewpoint

The Engineering viewpoint frames these concerns: the mechanisms and functions required to support distributed interaction between objects in the system.

The Engineering Language includes concepts for specifying:

— configurations of engineering objects for management purposes, including nodes (for resources), capsules (for protection) and clusters (for activation);

— the structure of communication channels that connect engineering objects, in terms of stubs, binders, protocols and interceptors;

— templates for providing required transparencies, such as migration, relocation, replication and failure transparencies.

C.4.6 Technology viewpoint

The Technology viewpoint frames these concerns: the selection of implementable standards for the system, their implementation and testing.

The Technology Language includes concepts to:

— capture the choice of technology to be used, in terms of the selection of existing standards or domain-specific specifications for these technologies;

— express how the specifications for an ODP system are implemented;

— provide support for testing.

Bibliography

[1] ANSI/IEEE Std 1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*

[2] Boucké, N., *Composition and relations of architectural models supported by an architectural description language*. Doctoral dissertation, Katholieke Universiteit Leuven, October, 2009

[3] Buschmann F., R. Meunier, H. Rohnert, P. Sommerlad and M. Stal, *Pattern-Oriented Software Architecture: A System of Patterns*, John Wiley & Sons, 1996

[4] Callo-Arias, T. B., P. America, and P. Avgeriou, "Defining execution viewpoints for a large and complex software-intensive system", *Proceedings of WICSA/ECSA 2009*

[5] Clements P., F. Bachmann, L. Bass, D. Garlan, J. Ivers, R. Little, R. Nord, and J. Stafford, *Documenting Software Architectures: Views and Beyond*, Boston: Addison-Wesley, 2002

[6] Darnton, G. and S. Giacoletto, *Information in the Enterprise*, Burlington, MA: Digital Press, 1992

[7] Dijkstra, E. W., *On the role of scientific thought*. 1974. http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html

[8] Eeles P. and P. Cripps, *The Process of Software Architecting*. Addison Wesley, 2010

[9] Hilliard, R. "Viewpoint modelling", *First ICSE Workshop on Describing Software Architecture with UML*, May 2001

[10] Hofmeister, C., R. Nord, and D. Soni, *Applied Software Architecture*, Reading, MA: Addison-Wesley, 1999

[11] ISO/IEC 10746-1, *Information technology — Open Distributed Processing — Reference model: Overview*

[12] ISO/IEC 10746-2, *Information technology — Open distributed processing — Reference model: Foundations*

[13] ISO/IEC 10746-3, *Information technology — Open distributed processing — Reference model: Architecture*

[14] ISO/IEC 12207, *Systems and software engineering — Software life cycle processes*

[15] ISO/IEC 15288, *Systems and software engineering — System life cycle processes*

[16] ISO/IEC 15289, *Systems and software engineering — Content of systems and software life cycle process information products (Documentation)*

[17] ISO/IEC 15414:2006, *Information technology — Open distributed processing — Reference model — Enterprise language*

[18] ISO/IEC 15504-1:2004, *Information technology — Process assessment — Part 1: Concepts and vocabulary*

[19] ISO 15704, *Industrial automation systems — Requirements for enterprise-reference architectures and methodologies*

[20] ISO/IEC 19501:2005, *Information technology — Open Distributed Processing — Unified Modeling Language (UML) Version 1.4.2*

[21] ISO/IEC 19793:2008, *Information technology — Open Distributed Processing — Use of UML for ODP system specifications*

[22] ISO/IEC 25010, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — System and software quality models*

[23] Kruchten, P.B., "The '4+1' View Model of Architecture", *IEEE Software*, 12(6), 45–50, 1995

[24] Kruchten, P.B., "An Ontology of Architectural Design Decisions in Software-Intensive Systems", *Proceedings of the 2nd Groningen Workshop on Software Variability*, 54–61, 2004

[25] Luckham, D.C., J.J. Kenney, L.M. Augustin, J. Vera, D. Bryan and W. Mann, "Specification and analysis of system architecture using RAPIDE", *IEEE Transactions on Software Engineering*, 21(4), 336–355, April 1995

[26] Maier, M.W. and E. Rechtin, *The art of systems architecting*, CRC Press, 2nd edition, 2000

[27] *Ministry of Defence Architecture Framework (MODAF)*, http://www.modaf.org.uk/

[28] Muskens, J., R.J. Bril and M.R.V. Chaudron, "Generalizing consistency checking between software views", *Proceedings of the 5th Working IEEE/IFIP Conference on Software Architecture (WICSA'05)*, 169–180, Washington, DC: IEEE Computer Society, 2005

[29] Nuseibeh, B., J. Kramer and A. Finkelstein, "A framework for expressing the relationships between multiple views in requirements specification", *IEEE Transactions on Software Engineering*, 20(10), 760–773, 1994

[30] Obbink, H., P.B. Kruchten, W. Kozaczynski, R. Hilliard, A. Ran, H. Postema, D. Lutz, R. Kazman, W. Tracz, and E. Kahane. *Report on Software Architecture Review and Assessment (SARA)*, 2002. http://philippe.kruchten.com/architecture/SARAv1.pdf

[31] OMG formal/2008-11-01, *Systems Modeling Language, version 1.1*, November 2008

[32] Perry, D.E. and A.L. Wolf, "Foundations for the Study of Software Architecture", *ACM SIGSOFT Software Engineering Notes*, 17(4), 1992

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

[34] Ran, A. "ARES Conceptual Framework for Software Architecture", M. Jazayeri, A. Ran, and F. van der Linden (eds.), *Software Architecture for Product Families Principles and Practice*. Boston: Addison-Wesley, 1–29, 2000

[35] Ross, D.T., "Structured Analysis (SA): a language for communicating ideas", *IEEE Transactions on Software Engineering*, SE-3(1), 16–34, 1977

[36] Rozansky, N. and E. Woods, *Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives*, Addison-Wesley, 2005

[37] Society of Automotive Engineers, *Architecture Analysis & Design Language*, http://www.aadl.info/

[38] Shaw, M. "Prospects for an engineering discipline of software", *IEEE Software*, November 1990

[39] Smolander, K., "Four Metaphors of Architecture in Software Organizations: Finding out The Meaning of Architecture in Practice", *Proceedings of the 2002 International Symposium on Empirical Software Engineering (ISESE'02)*

[40] The Open Group, *ArchiMate 1.0 Specification*, February 2009, http://www.archimate.org/

[41] *The Open Group Architecture Framework (TOGAF)*, http://www.opengroup.org/togaf/

[42] Viewpoints Repository for ISO/IEC 42010 http://www.iso-architecture.org/viewpoints/

[43] Wright ADL website, http://www.cs.cmu.edu/~able/wright/

[44] Zachman, J.A., "A Framework for Information Systems Architecture", *IBM Systems Journal*, 26(3), 1987

[45] Zimmermann O., Koehler J., Leymann F., Polley R., Schuster N., "Managing Architectural Decision Models with Dependency Relations, Integrity Constraints, and Production Rules", *The Journal of Systems and Software and Services, Special Issue on Design Decisions and Rationale in Software Architecture Special Edition on Architectural Decisions*, Elsevier, 2009

IEEE Notice to Users

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 the advice of a competent professional in determining the exercise of reasonable care in any given circumstances.

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. 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 addressed to: Secretary, IEEE-SA Standards Board, 445 Hoes Lane, Piscataway, NJ 08854, USA.

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 IEEESA 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.

IMPORTANT NOTICE: This standard is not intended to assure 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.

Abstract: ISO/IEC/IEEE 42010:2011 addresses the creation, analysis and sustainment of architectures of systems through the use of architecture descriptions. A conceptual model of architecture description is established. The required contents of an architecture description are specified. Architecture viewpoints, architecture frameworks and architecture description languages are introduced for codifying conventions and common practices of architecture description. The required content of architecture viewpoints, architecture frameworks and architecture description languages is specified. Annexes provide the motivation and background for key concepts and terminology and examples of applying ISO/IEC/IEEE 42010:2011.

Keywords: architecting, architecture description, architecture view, architecture viewpoint, architecture model, architecture framework, architecture description language, architecture decision, architecture rationale, system concern, correspondence, correspondence rule

ICS 35.080
ISBN 978-0-7381-7142-5 STD97174 (PDF); 978-0-7381-7167-8 STDPD97174 (Print)
Price based on 37 pages