DRAFT INTERNATIONAL STANDARD

ISO/IEC/IEEE/DIS 42010

ISO/IEC JTC **1**/SC **7** Secretariat: **BIS**

Voting begins on: **2020-12-16** Voting terminates on: **2021-03-10**

Software, systems and enterprise — Architecture description

ICS: 35.080

Reference number
ISO/IEC/IEEE/DIS 42010:2020(E)

Contents

Foreword

ISO (International Organization for Standardization) and the 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 (JTC), ISO/IEC JTC 1.

IEEE Standards documents are developed within IEEE Societies and Standards Coordinating Committees of the IEEE Standards Association (IEEE-SA) Standards Board. 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 IEEE and serve without compensation. While IEEE administers the process and establishes rules to promote fairness in the consensus development process, IEEE does not independently evaluate, test, or verify the accuracy of any of the information contained in its standards. Use of IEEE Standards documents is wholly voluntary. IEEE documents are made available for use subject to important notices and legal disclaimers (see https://standards.ieee.org/IPR/disclaimers.html for more information).

IEC collaborates closely with IEEE in accordance with conditions determined by agreement between the two organizations. This Dual Logo International Standard was jointly developed by the IEC and IEEE under the terms of that agreement.

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 a minimum of 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 second edition of ISO/IEC IEEE 42010 cancels and replaces ISO/IEC IEEE 42010:2011, which has been technically revised.

The main changes compared to the previous edition are as follows:

— The term used to refer to the subject of an architecture description is changed from "system" to "entity" to be compatible with ISO/IEC IEEE 42020 and ISO/IEC IEEE 42030 standards and to allow for its application in non-system architecture situations.

— Architecture description element, introduced in the 2011 edition (see 4.2.6, 5.7 and A.6 of that edition) is now defined in Clause 3 as a generic description concept allowing representing at least stakeholders, concerns, perspectives, and aspects identified in an AD, and views, view components, viewpoints, and model kinds included in an AD.

— Architecture aspect and stakeholder perspective concepts are added to accommodate current practice where these ideas are prevalent.

— Correspondences between architecture descriptions is distinguished from correspondence between architecture description elements.

— The term architecture view component is introduced as a separable portion of one or more architecture views. This change is to account for the fact that some parts of a view are model based while others might not be. View component can be derived from an information source, which can sometimes be a model.

— Model based view components are governed by model kinds. Non-model-based view components are governed by legends.

— The concept of architecture viewpoint is updated to accommodate current practice where a viewpoint governs one or more architecture views.

— The figures of this document use an informal entity-relationship diagram notation to facilitate comprehension by readers of this document. The multiplicities of the relationships are explained in the text when necessary.

— Annex E illustrates a few concepts pertaining to architecture life cycles and architecture description life cycles.

— Annex F show an interpretation of how some Architecture Description Frameworks could comply with requirements of this document.

Introduction

The complexity of human-made entities has grown to an unprecedented level. This has led to new opportunities, and also increased challenges for organizations that create and use these entities. Concepts, principles and procedures of architecting are increasingly applied by organizations, teams and individuals, to help manage the complexity faced by stakeholders of these entities.

Examples of entities include the following: enterprise, organization, solution, system (including software systems), subsystem, processes, business, data (as a data item or data structure), application, information technology (as a collection), mission, product, service, software item, hardware item, product line, family of systems, system of systems, collection of systems, collection of applications, etc.

An architecture of an entity, expressed in an architecture description (AD), assists in understanding of the fundamental properties of the entity, pertaining to its structure, behaviour, design and evolution, such as feasibility, utility and maintainability and fundamental concepts for its development, operation, employment and uses.

Architecture descriptions (ADs) are used by the parties that create, use and manage human-made entities to improve communication and cooperation, enabling all parties, organizations teams and individuals to work together in an integrated, coherent fashion. Architecture description frameworks (ADFs) and architecture description languages (ADLs) are used to codify the conventions and common practices of architecting and the description of architectures within different communities and domains of application.

ADs have many uses, such as design, development, documentation, analysis, evaluation, maintenance, risk mitigation, down-stream user specifications, tool specification, communication, planning, guidance, life cycle support, decision support, review, training, design validation, solution trade studies, cost comparison and analysis, by a variety of stakeholders throughout the life cycles of their entities of interest. Annex D describes a variety of uses of an AD.

This document provides core terms, definitions and relationships for the ADs. The provisions of this document serve to specify desired properties of ADs. This document also gives provisions that specify desired properties of ADFs and ADLs in order to usefully support the development and use of ADs. This document provides a basis on which to consider and compare ADFs and ADLs by providing a common ontology for specifying their content.

This document can be used to establish a coherent practice for developing ADs, ADFs and ADLs within an organization the context of a life cycle of an entity of interest or its architecture. This document can further be used to assess conformance of specifications of ADs, ADFs, ADLs, viewpoints and model kinds with the provisions of this document.

Users of this document are advised to consult Clause 5 to gain appreciation of the conceptual foundations, along with the concepts and principles associated with an AD work product.

Software, systems and enterprise — Architecture description

1 Scope

This document specifies requirements on the structure and expression of architecture descriptions (ADs) for various entities, including software, systems, enterprises, systems of systems, families of systems, products (goods or services), product lines, service lines, technology, and business domains. In this document, the term entity of interest refers to the entity whose architecture is under consideration in the preparation of an architecture description (AD).

This document distinguishes the architecture of an entity of interest from an AD expressing that architecture. ADs, not architectures, are the subject of this document. Whereas an AD is a tangible work product, an architecture is intangible and abstract that can be understood through its concepts, properties and principles.

This document specifies requirements on use of the architectural concepts and their relationships captured in an AD and does not specify requirements for any entity of interest or its environment.

This document specifies requirements on architecture description frameworks (ADFs), architecture description languages (ADLs), architecture viewpoints and model kinds in order to usefully support the development and use of ADs. This document also provides motivations for use of architecture-related terms and concepts in other documents such as guides and standards.

This document specifies conformance to the requirements on ADs, ADFs, ADLs, viewpoints, and model kinds.

This document does not explicitly address completeness or correctness of an AD. Nevertheless, completeness and correctness of an AD can be partially checked, for example, through the consistency of the AD elements established, whether relationships are transitive, and whether AD elements are shown in the respective views. Consistency rules can also be defined with respect to showing the same particular AD element has correspondences with an AD.

This document does not specify the processes, architecting methods, models, notations, techniques or tools by which an AD is created, utilized or managed.

NOTE ISO/IEC IEEE 42020 [17] specifies a set of processes for architecting which can be employed in support of creating one or more ADs. The architecture elaboration process in that standard is especially relevant for creation of ADs.

This document does not specify any format or media for recording ADs. The intent of this document is to enable a range of consistent and coherent approaches to AD including document-centric and model-based techniques.

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

ISO and IEC maintain terminological databases for use in standardization at the following addresses:

— IEC Electropedia: available at http://www.electropedia.org/

— ISO Online browsing platform: available at https://www.iso.org/obp/ui

— IEEE Standards Dictionary Online: available at https://ieeexplore.ieee.org/xpls/dictionary.jsp

Note 1 to entry Access to the IEEE Standards Dictionary Online requires a free IEEE account. It does not require IEEE membership or any subscription fee.

Definitions for other related terms typically can be found in ISO/IEC IEEE 24765, *System and software engineering – Vocabulary*, available at http://www.computer.org/sevocab.

3.1
architecting
conceiving, defining, expressing, documenting, communicating, certifying proper implementation of, maintaining and improving an architecture (3.2) throughout the life cycle of an entity of interest (3.12)

[SOURCE: ISO/IEC IEEE 42020, modified – The term "entity of interest" replaced "architecture entity".]

3.2
architecture
fundamental concepts or properties related to an entity in its environment (3.13) and governing principles for the realization and evolution of this entity and its related life cycle processes

[SOURCE: ISO/IEC IEEE 42020, modified – The words "related to" replaced "of".]

3.3
architecture aspect
unit of modularization of concerns within an architecture description (3.4), capturing characteristics or features of the entity of interest (3.12)

Note 1 to entry: Aspects enable the architect to analyse, address and structure architecture concerns. In general, there is a many-to-many relation between aspects and concerns. An aspect can pertain either to an entity of interest, to an architecture, or to an environment (such as to a situation or action).

EXAMPLE Functional, structural and informational aspects of an architecture.

Note 2 to entry: See 5.2.5 for more discussion and examples.

Note 3 to entry: The identification of an aspect is often the result of prior knowledge, experience and praxis in the domain to which the aspect applies.

Note 4 to entry: Cross-cutting concerns can be shared among perspectives of stakeholders.

3.4
architecture description
AD
work product used to express an architecture (3.2)

Note 1 to entry: An AD is a tangible representation of information provided to the stakeholders. In other words, it can also be considered as an information item.

Note 2 to entry: A work product is an artefact produced by a process [SOURCE: ISO/IEC 20246:2017, Software and systems engineering — Work product reviews, 3.19]

Note 3 to entry: The subject of an architecture description is the architecture of an entity of interest

3.5
architecture description element
AD element
part of an architecture description (3.4) that expresses the architecture

Note 1 to entry: AD elements include stakeholders, concerns, perspectives, and aspects identified in an AD, and views, view components, viewpoints, and model kinds included in an AD.

Note 2 to entry: For the purpose of correspondences, an architecture description can be considered as an AD element in another Architecture description.

3.6
architecture description framework
ADF
conventions, principles and practices for the description of architectures (3.2) established within a specific domain of application or community of stakeholders (3.18)

EXAMPLE Annex B of Generalized Enterprise-Referencing Architectures Modelling Framework [10], Reference Model of Open Distributed Processing (RM-ODP) [1], Unified Architecture Framework (UAF) [39], and NATO Architecture Framework (NAF) [34].

Note 1 to entry: Architecture description frameworks promote structured organization, consistency of description, greater potential for reuse, and completeness of architecture views and models.

3.7
architecture description language
ADL
means of expression, with syntax and semantics, consisting of a set of representations, conventions, and associated rules intended to be used to describe an architecture

EXAMPLE Architecture Analysis and Design Language (AADL), ArchiMate, UML, SysML, UAF Profile.

3.8
architecture view
information item, governed by an architecture viewpoint (3.9), comprising part of an architecture description (3.4)

Note 1 to entry: A viewpoint is a frame of reference for the concerns determined by the architect as relevant to the purpose of the architecture description.

EXAMPLE A data view in an AD, as described in Clause 4 of ISO/IEC 25024:2015, concerns data for the entity of interest. A data view could include contextual schema, conceptual, logical and physical data models, data dictionary and documents as view components, and entities, relations and attributes as AD elements.

3.9
architecture viewpoint
conventions for the creation, interpretation and use of an architecture view (3.8) to frame one or more concerns (3.10)

Note 1 to entry: A viewpoint is a frame of reference for the concerns determined by the architect as relevant to the purpose of the architecture description.

Note 2 to entry: The conventions of an architecture viewpoint are included in a specification of this viewpoint. In some communities and architecture frameworks, "view specification" is used to mean the same thing.

3.10
concern
matter of relevance or importance to a stakeholder (3.18) regarding an entity of interest (3.12)

Note 1 to entry: Stated concerns are useful when relevant to the purpose of the architecting effort and refer to specific rather than categorical difficulties, problems, or requirements.

Note 2 to entry: Concern expression takes many forms, including among others: as questions about entity features or characteristics, as a keyword label for many related matters, and as expected quality attributes of the entity.

Note 3 to entry: See 5.2.3 for more discussion and examples.

3.11
correspondence
expression of a relationship among architecture description elements (3.5) or among architecture descriptions (3.4)

EXAMPLE Correspondences are used to express a wide range of relationships, such as equivalence, composition, refinement, consistency, traceability, dependency, constraint, satisfaction, and obligation.

3.12
entity of interest
subject of an architecture description (3.4)

EXAMPLE Enterprise, organization, solution, system (including software systems), subsystem, processes, business, data (as a data item or data structure), application, information technology (as a collection), mission, product, service, process, software item, hardware item, product line, family of systems, system of systems, collection of systems, collection of applications, etc.

3.13
environment
aggregate of surrounding things, conditions, contexts of, or influences upon an entity of interest

Note 1 to entry: The environment of an entity of interest includes external entities that can have various influences upon the entity of interest, such as developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological and social influences as well as external physical effects such as electromagnetic radiation, charged particles, gravitational effects, and electric and magnetic fields.

Note 2 to entry: A label attached as a qualifier to the word *environment* identifies a particular context within that environment, such as development environment, test environment, and operational environment. It would be more correct to refer to these as development context, test context, operational context, etc. A context can be to help understand an entity or its architecture, including the derivation of an architecture.

3.14
frame
to formulate or construct in a particular style or language

3.15
information item
separately identifiable body of information that is produced, stored, and delivered for human and machine use

[SOURCE: ISO/IEC IEEE 15289:2019, modified – The words "human and machine use" replaced "human use"]

3.16
model kind
category of model distinguished by its key characteristics and modelling conventions

EXAMPLE Functional models, activity models, structural models, use case diagrams, geopolitical models and economic models.

3.17
specification
information item that identifies, in a complete, precise and verifiable manner, the requirements, design, behaviour, or other expected characteristics of a system, service, or process

[SOURCE: ISO/IEC IEEE 15289:2019, 3.1.26]

3.18
stakeholder
role, position, individual, organization or classes thereof, having an interest, right, share, or claim, in an entity or its architecture (3.2)

EXAMPLE End users, operators, acquirers, owners, suppliers, architects, developers, builders, maintainers, regulators, taxpayers, certifying agencies, and markets.

3.19
stakeholder perspective
way of thinking about an entity, especially as it relates to concerns (3.10)

EXAMPLE The labels given to the middle three rows (i.e. owner, designer and builder) of the Zachman framework [56] correspond to stakeholder perspectives. The rows in the Unified Architecture Framework [39] and NATO Architecture Framework [34] grids correspond to stakeholder perspectives (although they are called ”domains” and “subjects of concerns,” respectively in those frameworks). See 5.2.4 for more examples.

Note 1 to entry: The way one thinks about an entity can be influenced by one’s beliefs, training, experience, knowledge, personality, character traits, culture, peer pressure, role or stance etc.

3.20
view component
architecture view component
separable portion of one or more architecture views (3.8) that is governed by the applicable model kind (3.16) or legend

EXAMPLE An architecture view component describing a firewall can be used in several views of an architecture description to explain functional flows, behaviour and security features of an entity.

4 Conformance

The requirements in this document are contained in Clauses 6, 7 and 8. There are five situations in which claims of conformance with the provisions of this document can be made.

1) When conformance is claimed for an architecture description, the claim shall demonstrate that the specification of the architecture description meets the requirements listed in Clause 6.

2) When conformance is claimed for an architecture description framework, the claim shall demonstrate that the specification of the architecture description framework meets the requirements listed in 7.1.

3) When conformance is claimed for an architecture description language, the claim shall demonstrate that the specification of the architecture description language meets the requirements listed in 7.2.

4) When conformance is claimed for an architecture viewpoint, the claim shall demonstrate that the specification of the architecture viewpoint meets the requirements listed in 8.1.

5) When conformance is claimed for a model kind, the claim shall demonstrate that the specification of the model kind meets the requirements listed in 8.2.

Requirements of this document 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 document is designed such that “tailoring” is neither required nor permitted for its use when claims of conformance are made.

5 Conceptual foundations

5.1 Introduction

This clause introduces the conceptual foundations of architecture description expressed in a set of conceptual models (see 5.2) and its application to ADs, ADFs (see 5.4.2) and ADLs (see 5.4.3). The concepts introduced in this clause are used in Clauses 6 through 8 to express requirements.

NOTE Annex A provides further discussion of the terms and concepts used in this document and presents examples of their use in an historical context.

5.2 Conceptual models of an architecture description

5.2.1 Context of an architecture description

The term *entity of interest* is used in this document to refer to the subject of an architecture description. The term is intended to encompass, but is not limited to, entities within the following *fields of application*, reflecting the intended scope of this document as specified in Clause 1.

— software, including software products and services per ISO/IEC IEEE 12207 [4];

— systems as discussed in ISO/IEC IEEE 15288 [5];

— enterprises as described in ISO 15704 [10], i.e. human undertakings or ventures that have mission, goals and objectives to offer products or services, or to achieve a desired project outcome or business outcome.

This document takes no position on what constitutes an entity within those fields of application or elsewhere. An entity can be a concrete entity or an abstract entity. An AD as specified in this document is suitable for not only entities in the fields of applications listed above, but also for entities in fields such as natural systems or conceptual systems.

Each entity of interest is situated in an environment. The environment determines the totality of influences upon the entity of interest and the totality of influences of the entity of interest upon that environment, including its interactions with the environment and other entities, throughout the life cycle of that entity of interest. The environment of an entity of interest can influence other entities or can be influenced by other entities.

Figure 1 depicts key concepts pertaining to entity of interest and their architectures as a context for understanding ADs.

NOTE The figures and text in the remainder of 5.2 constitute a set of conceptual models of an AD. These figures use an informal entity-relationship diagram notation to facilitate comprehension by readers of this document. In the figures rounded rectangles represent information objects, and arrows represent relationships between objects with the annotation read in the arrow direction. The figures illustrate the key concepts described throughout Clause 5.

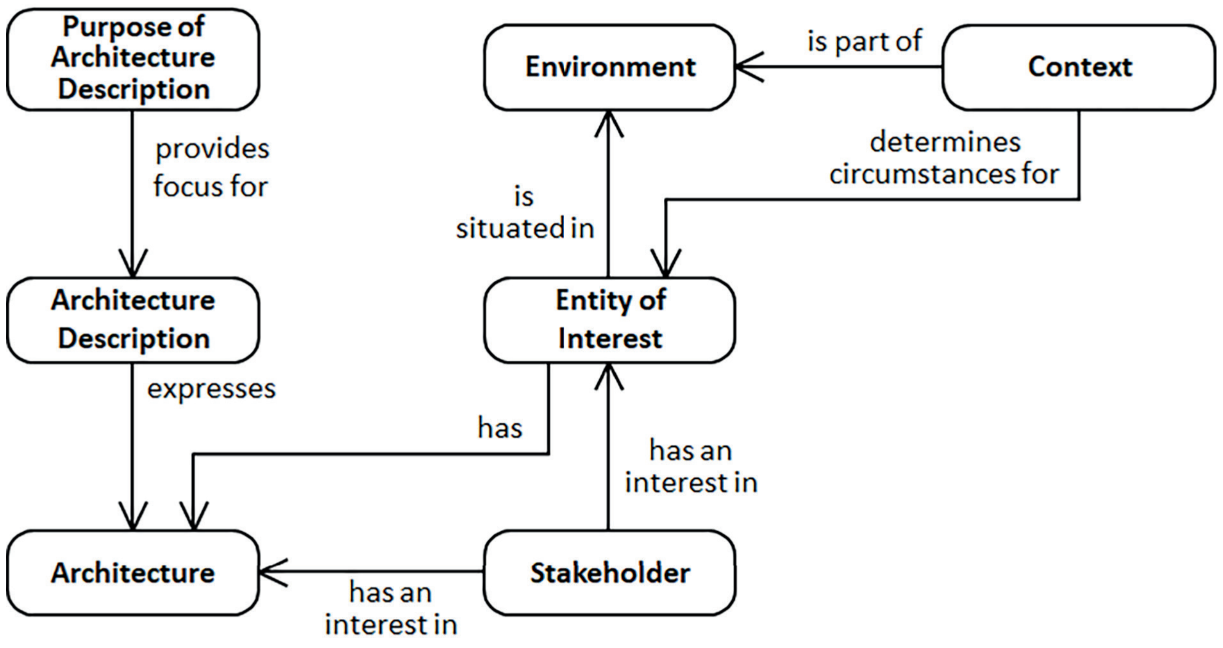

Figure 1 — Context of architecture description

5.2.2 Architectures and architecture descriptions

The architecture of an entity of interest comprises the fundamental concepts or properties of that entity considered in its environment. The architecture of an entity of interest can pertain to any or all of the entity's:

— constituent elements;

— interactions or interrelationships among its elements;

— interactions or interrelationships with other entities in its environment or the environment itself;

— behaviour and structure; and

— principles governing its design, use, operation and evolution.

An AD is an expression of an architecture. An AD comprises AD elements (see 5.2.10).

ADs are work products resulting from architecting efforts. As a work product, an AD is devised for the specific purpose for which the architecting effort is undertaken, which is distinct from the purpose of the entity of interest.

The architecture of an entity of interest can be understood through one or more distinct ADs, each created for a purpose relative to the architecture and stakeholder needs. Different ADs can, for example, be based on different stakeholders (or stakeholder perspectives), time periods (sometimes termed *epochs*), or specific contexts or usage within the environment.

NOTE ISO/IEC IEEE 42020 [16] specifies a set of processes for architecting which can be employed in support of creating one or more ADs.

5.2.3 Stakeholders and concerns

Stakeholders are parties with interests in an entity, its architecture, or its architecture description. Among the stakeholders are those parties that have influence or control over or are impacted by an entity or its architecture. A stakeholder's interests are typically expressed as concerns about an entity of interest or the architecture of which they are aware. This awareness occurs as a result of the stakeholder's perspective gained from domain knowledge, experience, training, responsibility and authority.

Concerns are matters of interest or importance to one or more stakeholders regarding an entity of interest. Stakeholders often form distinct groupings, or stakeholder perspectives, determined by subject of concerns, their role, experience, beliefs or other characteristics.

A concern can be shared by one or more stakeholders and a stakeholder can hold more than one concern. The legitimacy and importance of a concern held by a stakeholder can be a consequence of the role of the stakeholder (e.g. owner, end user or participant, developer, architect, maintainer, disposer) or financial or social rights, shares, impact or claims (e.g. funding organization, party receiving environmental impact from entity, stakeholder in entity impacted by entity of interest). Some stakeholders' interests can be contrary to the success of the entity of interest. These stakeholders can have disagreements on the grounds of political or environmental considerations, can seek active disruption of the entity's operations, or even outright destruction of the entity. Adversarial concerns could be taken into account when developing the architecture of the entity. For example, political objections could be resolved by incorporating a negotiated solution in the architecture of the entity, or threats could be mitigated by taking preventative measures.

During the entity's life cycle, concerns can arise at any time including (but not limited to) during conceptualization, when design choices are made, from construction or implementation, through deployment, operation, transfer of ownership, retirement and disposal.

Concerns can manifest in various ways in relation to stakeholder's needs, architecture goals, expectations, responsibilities, requirements, design constraints and assumptions. Concerns can also manifest in recognition of dependencies, quality attributes, architecture decisions, risks or other issues.

Concerns can pertain to influences exerted upon or by an entity of interest, including developmental, technological, business, operational, organizational, political, economic, legal, regulatory, ecological, social and physical influences. Concerns can also pertain to design influences such as internal structural features and component interoperability, particularly when architecting a system of systems or enterprises.

The expression of a concern occurs in several ways including: as questions about entity features or characteristics; as a keyword label for many related matters; and as expected quality attributes of the entity.

EXAMPLE 1 How is the system maintained? What system behaviours are safety-critical? Can the entity of interest attain compliance with national regulations? What is the cost to operate?

EXAMPLE 2 Risk, opportunity, satisfaction, affordability, complexity and trust.

EXAMPLE 3 The distribution transparencies described in the Reference Model of Open Distributed Processing [1].

EXAMPLE 4 In the case of a flight navigation system of a commercial aircraft, GPS signal availability, tracking, degree of accuracy, line of sight reception, altitude or elevations are features. These features can be seen as operational concerns.

EXAMPLE 5 Data qualities as described in Clause 4 of ISO/IEC 25012:2008 [14] or traditional system quality attributes as described in ISO/IEC IEEE 25010 [13].

EXAMPLE 6 A system's ability to maintain confidentiality, integrity, and availability for protecting operations.

EXAMPLE 7 An enterprise's ability to support a seamless transition from a legacy capability to a modernized operational capability.

5.2.4 Stakeholder perspectives

Stakeholder perspectives are ways of thinking about an entity in a context, especially as they relate to concerns. Concerns serve as a primary input to stakeholder perspectives, including through categorization by domains, stages of architecting, and levels of abstraction.

Typically, there are several ways of thinking about the architecture of the entity and therefore differing concerns result. There are likely to be multiple relevant stakeholder perspectives for any architecture or entity.

EXAMPLE 1 Operational and financial stakeholder perspectives of an industrial production system.

EXAMPLE 2 Business, management, acquisition and supply stakeholder perspectives of a banking system.

EXAMPLE 3 Development, deployment and customization perspectives of a mobile app.

EXAMPLE 4 Provider and consumer perspectives of a hospitality service.

The purpose of the AD guides the identification of concerns which can affect the perspectives of some stakeholders. A perspective can be the result of domain knowledge, professional experience, and training. Importantly, the stakeholder's perspective can also be influenced by their personality, character traits, culture, peer pressure, etc.

Because concerns arise from stakeholder perspectives, architecture views related to the concerns are often arranged in various ways into nominal stakeholder perspectives for the purpose of categorization. Often concerns are based on current interests and influences of the stakeholders and are often subjective in nature because they arise from stakeholder perspective thinking. Aspects are based on experience in characterization of architectures and are more objective in nature as they arise from agreements among experts about practice in a domain and therefore are presumed to be best practice.

Figure 2 depicts the relationships between concerns, architecture aspects, and stakeholder perspectives as utilized in an AD. It is developed from and consistent with Figure 1.

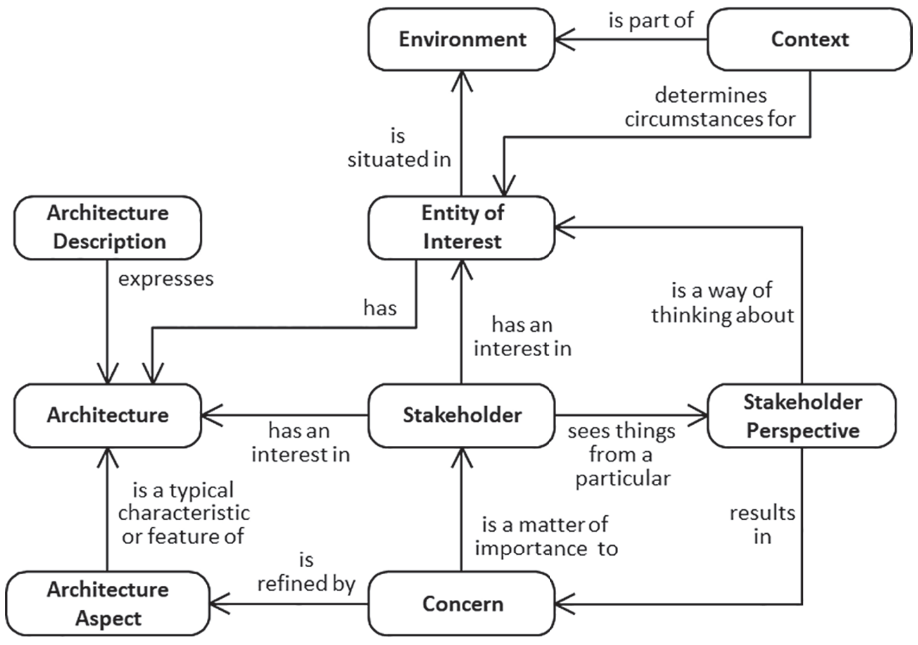

Figure 2 — Concerns, architecture aspects, and stakeholder perspectives

5.2.5 Architecture aspects

Architecture aspects are units of modularization that enable the addressing and structuring of cross-cutting concerns within an AD and associated analyses, capturing a select set of characteristics or features of the entity of interest.

Each architecture aspect can relate to concerns of stakeholders based upon prior experience within a field of application. Usage of known architecture aspects can enable a more systematic coverage of the range of established concerns and also the identification of new concerns.

By examining architecture aspects, relevant features or properties of the entity can be discerned or predicted. Each architecture aspect can lead to determination of one or more concerns or an examination of several architecture aspects can lead to determination of one concern.

While architecting, the architect(s) will identify architecture aspects to show how the architecture views address the concerns. The definition of the relationships between architecture aspects and the stakeholder concerns are based on the experience of the architects and are assessed by the stakeholders with their understanding and knowledge.

NOTE A.4.2 contains more information about the utility of architecture aspects.

EXAMPLE 1 Spatial, structural, functional, connectivity, taxonomic, information and roadmap aspects in an aircraft AD.

EXAMPLE 2 Behavioural, information and structural aspects in a computer AD.

EXAMPLE 3 Logical connectivity and physical connectivity aspects in a network AD (corresponding to the so-called logical network and physical network depictions of a configuration of links and nodes in the network).

5.2.6 Architecture considerations

Architecture considerations are factors taken into account when architecting. Concerns (5.2.3), stakeholder perspectives (5.2.4), and architecture aspects (5.2.5) are different considerations that can be used in this manner. There are other considerations that can arise due to the architecture practices followed by an organization, individuals or teams .

Architecture considerations can be used when specifying architecture viewpoints and when constructing, interpreting, organizing or using architecture views (5.2.7). Architecture considerations can be used to group specifications of architecture viewpoints such as with respect to stakeholder perspectives.

5.2.7 Architecture views and architecture viewpoints

An AD contains one or more architecture views. An architecture viewpoint governs one or more of these architecture views. A specification of an architecture viewpoint is an information item that establishes the conventions for creating, interpreting, presenting and analysing a view to address the concerns framed by that viewpoint.

Architecture concerns, perspectives and aspects can serve as an organizing basis for an AD. Architecture aspects are refinements of concerns and these aspects can also be used to establish the viewpoint conventions. These aspects are reflected in the resulting architecture view(s).

EXAMPLE 1 A telecommunications network (entity) is represented by a “network model” that can be used to express the “network connectivity deployment diagram” (architecture view) contained in a telecommunications architecture description (AD) document that addresses the communications parameters such as throughput and uptime (concerns) of operators and users (stakeholders).

EXAMPLE 2 View describing the content of a product line (entity) including common parts and individual products with their variants and options.

Figure 3 depicts the relationship between architecture views and architecture viewpoints in an AD.

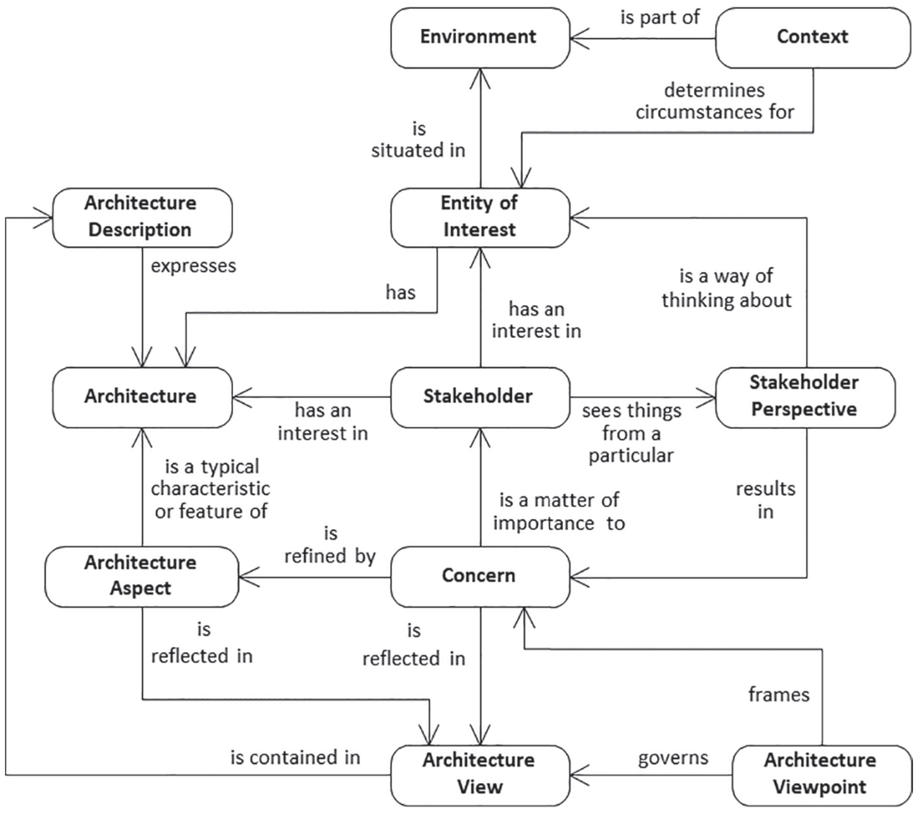

Figure 3 — Architecture views and architecture viewpoints

An architecture viewpoint frames one or more concerns (5.2.3). A concern can be framed by more than one viewpoint. The architecture viewpoint identifies the specific architecture aspects and concerns to be reflected in one or more architecture views. The specification of an architecture viewpoint provides guidance and direction to those who are creating, interpreting or using the architecture views. There are two facets to a specification of an architecture viewpoint: the architecture aspects and concerns which are identified, and the conventions it establishes for the creation, interpretation, analysis and other uses of the architecture views.

Distinct from requirements for product acceptance, architecture views can define requirements for the entity of interest that satisfies concerns and addresses architecture aspects.

Using a metamodel or other conventions, the viewpoint specification establishes the manner in which AD elements (e.g. entities, relationships, attributes and constraints) are used when creating a view by applying that viewpoint specification. (see 5.2.9).

NOTE Clause 8 specifies requirements on specification of architecture viewpoints. Annex B provides guidance on specifying architecture viewpoints.

5.2.8 Model kinds, legends and architecture view components

An architecture view is composed of one or more architecture view components. Each view component is governed by a model kind or legend defined in the architecture viewpoint. A specification of a model kind or legend establishes the conventions used within the view component. These conventions typically cover the intended uses, and the specification of notations, their syntax and semantics of its governed models. A model kind determines the conventions for model-based view components. A legend documents the conventions for non-model view components. A model kind or legend can be used by more than one viewpoint in an AD. A legend denotes the category of explanations or interpretations.

Within an AD, an architecture view component can be part of more than one architecture view to enable sharing information across views.

EXAMPLE 1 Model kinds include use cases, activity models (such as SADT and IDEF0), threat models, component models and connectivity models.

EXAMPLE 2 A data flow diagram can be a view component of a functional view. A separate control flow diagram can be a second view component in the same functional view. The functional view can also contain a narrative that explains how to interpret the flow diagrams in the view. The flow diagrams are model based while the narrative is not.

EXAMPLE 3 A symbology table can be a legend of an operational view.

Figure 4 depicts the composition of views from view components and the kinds of view components.

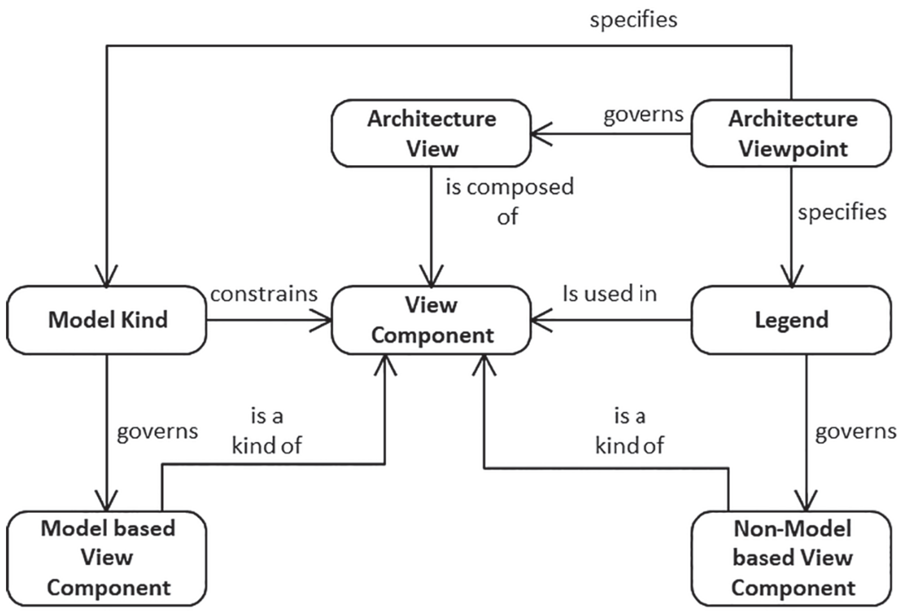

Figure 4 — Conceptual model for views and view components

5.2.9 View methods

Among the viewpoint conventions can be view methods that specify expression rules, modelling methods, analysis techniques and other operations on views. These methods specify the AD elements used when creating the view, methods to interrogate or query the views to assess how well, for example, the architecture can satisfy stakeholder concerns or address decision maker questions. Requirements on view methods are specified in 8.3.

A specification of an architecture viewpoint can include one or more view methods. View methods provide guidance, heuristics, metrics, patterns, design rules or guidelines, best practices and examples to aid in view construction and use of associated views.

View methods can be divided into categories, such as the following:

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

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

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

NOTE 1 This is not an exhaustive list of view method categories.

NOTE 2 View methods can be defined by a viewpoint, model kind, architecture description framework, or architecture description language.

EXAMPLE View methods pertaining to: chaining dependencies to assess the impact of a change; workshops to trade-off qualities or other concerns; boundary analyses to determine whether context and entity of interest are well defined; guidance on partitioning; analysis of architectural complexity; creation and enforcement of architecture styles (such as layered, aspect-oriented); pattern families to promote intended properties of the entity of interest; analyses against requirements for completeness and coverage; interpretation and integration of external models as information sources.

5.2.10 Architecture description (AD) elements

An AD element is an instance of any architectural construct in an AD. The AD elements include instances of the following items: stakeholder, concern, architecture viewpoint, architecture view, model kind, legend, architecture view component, architecture decision, architecture rationale and the relationships specified between those constructs.

As viewpoints (see 5.2.7), model kinds (see 5.2.8) and legends (see 5.2.8) are specified and applied, additional AD elements are introduced. The governing viewpoint or model kind or legend determines the syntax and semantic conventions for these introduced AD elements.

EXAMPLE AD elements introduced by viewpoints or model kinds include use case constructs such as preconditions, actors, boundaries, systems; activity model constructs such as activities, inputs, outputs, controls, and mechanisms; architecture or design patterns to be employed.

5.2.11 AD correspondence

An *architecture description* correspondence identifies an architectural relation between two or more ADs. An AD correspondence can be governed by a *correspondence method* which expresses rules, practices or models to specify the particular relation among ADs.

EXAMPLE A correspondence of a protocol dependency between the ADs of two subsystems; a correspondence between the AD of a system and the AD of a system-of-systems of which it is a part; the correspondence between an architecture “plug” and a compatible architecture “socket”; between a product line structure and a product architecture.

NOTE 1 Requirements on using correspondences and correspondence methods are specified in 6.9. Additional examples of their use are given in A.7.

NOTE 2 Correspondences in this document are compatible with view correspondences in ISO/IEC 10746 (RM-ODP) [1] and ISO/IEC 19793 [11] (see A.7).

NOTE 3 The ADs in an AD correspondence need not be distinct. A correspondence can be defined between an AD and itself.

NOTE 4 Usually correspondence methods are “cross model” or “cross view” or “cross AD” since correspondences *within* a model kind are part of the conventions of the specification of the model kind.

NOTE 5 Correspondences and correspondence methods can be applied to multiple ADs to express relations pertaining to multiple architectures or entities of interest.

NOTE 6 Correspondence methods are not limited in their occurrence to ADs. In particular, they can also be defined as part of specifications of viewpoint, model kind specifications, ADF specifications and ADL (see 6.9.3 and 8).

Figure 5 depicts the nature of AD correspondences.

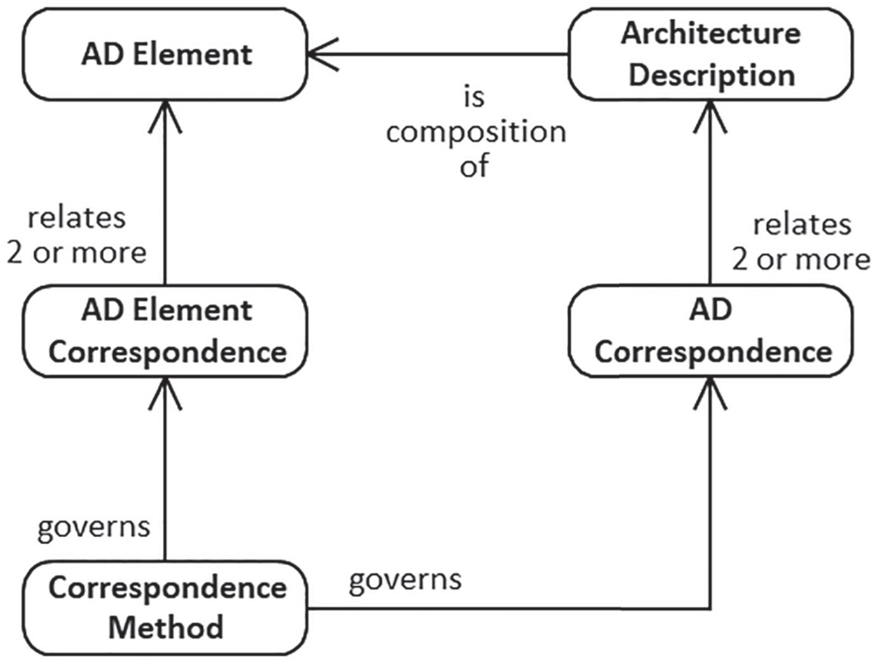

Figure 5 — Conceptual model of AD and AD element correspondences

5.2.12 AD element correspondence

An *AD element* correspondence identifies an architectural relation between two or more AD elements.

An AD element correspondence can relate:

— one or more AD elements with one or more AD elements within an AD;

— one or more AD elements with one or more AD elements occurring in multiple ADs;

— one or more AD elements with one or more AD elements within an ADF or across several ADFs; and,

— one or more AD elements with one or more AD elements within an ADL or across several ADLs.

For the purposes of correspondences, an AD itself may be considered an AD element of a different AD. An AD element correspondence may be governed by a *correspondence method* which expresses rules, practices or models to specify the particular relation between the AD elements.

AD element correspondences and correspondence methods can be used to express and enforce architecture relations such as composition, refinement, consistency, traceability, dependency, constraint, satisfaction and obligation [19] of AD elements.

EXAMPLE A correspondence between an AD element within a view and the concern that it addresses; between an architecture view and the aspect that it implements; between an AD element and the function that it implements; between an interface on a component and the stack of standards to which the interface conforms; between a data object on a functional flow and the full data structure definition; between a technical system and the organizational structure that implements it.

A correspondence method can specify that all AD elements trace to a concern or requirement. Compliance to that method can be recorded in the form of a traceability matrix, with a rationale provided for each AD element that does not track to a concern or requirement.

NOTE 1 Requirements on using correspondences and correspondence methods are specified in 6.9. Additional examples of their use are given in A.7.

NOTE 2 The AD elements in a correspondence need not be distinct. A correspondence can be defined between an AD element and itself. Examples of “self-referential” correspondence are an activity that is refined into two or more activities of the same kind, or an activity that can recursively invoke itself.

NOTE 3 Correspondences and correspondence methods can be applied to multiple AD elements to express architecture relations pertaining to multiple AD elements.

Figure 5 depicts the nature of AD element correspondences.

5.2.13 Architecture decisions and rationale

An architecture decision is a collection of choices made in the overall context of an architecture. Such decisions pertain to concerns about, perspectives on, aspects of, requirements on an entity, or influences of the environment.

EXAMPLE 1 Selection of architecture concepts, choice of AD elements, selection of ADFs, choice of architecture layering scheme, choice of underlying technology, choice of business components, choice of tactics to use for achieving system qualities, choice of business processes, choice of applicable patterns, choice of style(s) to be applied, choice of range of implementation technologies or other realizations to be considered, and choice of option sets to be considered.

Architecture rationale records explanation, justification or reasoning about architecture decisions that have been made. The rationale for a decision may include the following items: the basis for making a decision, impact on quality attributes, alternatives and trade-offs considered, potential consequences of the decision, and citations to sources of additional information.

EXAMPLE 2 Meeting cost commitments, meeting time commitments, using proven technologies, minimizing rework, reducing capital investments, achieving interface compatibility, satisfying constraints imposed by the operational environment.

EXAMPLE 3 Modelling tool selections to align with related architectures (e.g. customer’s enterprise architecture) to ensure interoperability and traceability.

NOTE Requirements for capturing decisions and rationale within an AD are specified in 6.10.

5.3 Architecture description in the life cycle

ADs are produced throughout the life of the entity of interest, from initial conception through the operation, refurbishment or final retirement from use, and eventual disposal of this entity.

ADs are the work products resulting from the execution of architecting, which takes place within the context of a project and/or organization (company, network of companies, consortium and standardization body).

ADs are used for creating, updating or changing the architecture of an entity of interest.

NOTE See Annex E for more details of the role of architecting in the life cycle.

5.4 Architecture description frameworks and languages

5.4.1 Introduction

ADFs and ADLs are now widely used in architecting to facilitate architecture information access for those constructing and using ADs. ADFs and ADLs built on the concepts of AD presented in this document, can be utilized effectively for (a) generic reference frameworks intended to guide more specific ADFs, (b) frameworks for special purposes intended to enable better analytical understanding and situational awareness, and (c) entity implementation frameworks intended to facilitate entity realization and operation.

5.4.2 Architecture description frameworks

The architecture viewpoints identified in an ADF result from architecture viewpoints specified or used by prior architecting efforts to determine satisfaction of concerns about an entity of interest to the extent of detail consistent with the purpose of the ADF.

A reference ADF establishes a common practice for creating, interpreting, analysing and using ADs within a particular domain of interest, e.g. a domain of application or a stakeholder community. For a generalized entity of interest within the context of a particular domain of practice, a reference ADF typically identifies architecture viewpoints for expected or known architecture considerations, often as stakeholder perspectives and architecture aspects related to structure, function (both behaviour and fitness) and life cycle. For a reference framework, the many different stakeholder perspectives can be generalized as generic perspectives. Utilizing an architecture viewpoint, users of a reference ADF have access to a generalized view that that can satisfy the architecture considerations framed by that viewpoint.

An architecture viewpoint identified in a reference ADF can identify the concerns, model kinds, legends, and view methods that constitute the conventions that govern views associated with that viewpoint. Users of a reference ADF can specialize the architecture considerations, the specifications of architecture viewpoints, and thus the resulting architecture view, as an implementation ADF for constructing and operating a particular entity of interest.

NOTE 1 Particular architecture decisions are made in ADFs: selection of stakeholders and related concerns, specific aspects and stakeholder perspectives. An ADF will structure ADs according to these decisions.

Figure 6 depicts the conceptual model for an ADF.

An ADF can provide a structuring formalism, i.e. a set of rules for using architecture considerations and correspondences between them, to organize the architecture viewpoints used to generate associated views, e.g. a grid framework formalism. In addition to architecture viewpoints forming parts of an ADF, correspondence methods (see 5.2.9) can also be contained in an ADF. The purpose of the structuring formalism is to provide ways of representing relationships among various elements of the architecture and enhancing opportunities for analysis of interactions among those elements.

EXAMPLE 1 Well known structuring formalisms include: RM-ODP, GERAM cube, Reference Architectural Model Industrie 4.0 (RAMI 4.0), TOGAF stack (business, information systems, technology), NAF grid, UAF grid, and Zachman Framework matrix.

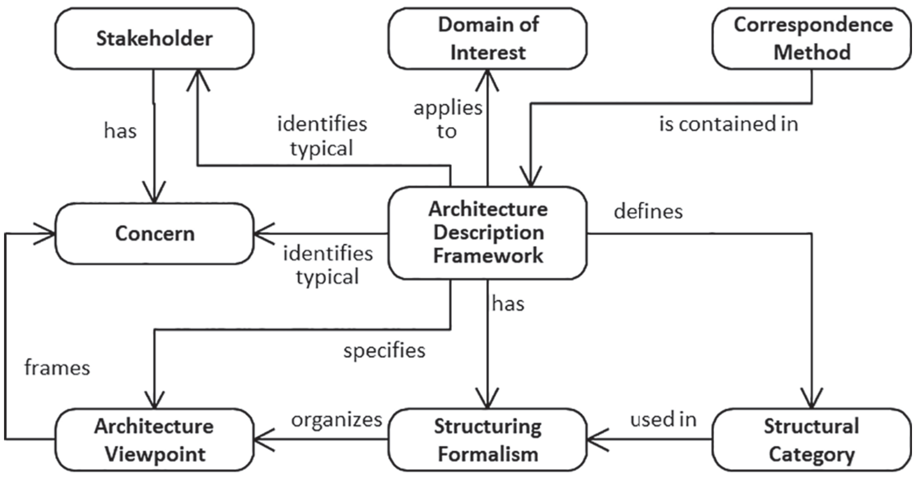

Figure 6 — Conceptual model of an architecture description framework

An ADF can define structural categories used in the structuring formalism. Sometimes these categories are represented by “dimensions” in a graphic portrayal, such as the rows and columns used in several frameworks. A structuring formalism in grid form usually has two framework dimensions but formalisms can have a single framework dimension, often segmented in a multi-layer hierarchy, or many framework dimensions where visual representation of framework dimensions above three is difficult.

EXAMPLE 2 A two-dimensional grid is used in some ADFs where stakeholder perspectives correspond to rows and architecture aspects correspond to columns.

While the reference ADF can successfully rely on generic architecture considerations and an agreed upon common vocabulary for the domain it encompasses, specializations for implementation often necessitate careful customization to meet stakeholder expectations, particularly when adapting the reference ADF for a particular purpose or for better comprehension by those stakeholders sponsoring the architecting effort. Therefore, in addition to the introduced elements of specialization, a change in vocabulary and specifications of architecture viewpoints could be necessary.

Within an ADF, architecture viewpoints are important analytical resources indicating the architecting purpose, typical stakeholders and their perspectives, identified concerns, defined architecture aspects, and particular architecture elements, which can be used for development of architecture views.

Depending upon the intended application for a framework, the extent of detail resulting from a viewpoint varies widely. A framework for reference can be expected to have more generic stakeholder perspectives and generic architecture considerations, often partitioned into multiple functional clusters, e.g. product and life cycle.

Areas of application and user communities can share typical architecture considerations and viewpoints. In that situation they can develop and maintain a less generic domain specific reference ADF with architecture considerations and viewpoints with appropriate model kinds and legends. Some reference ADFs include explicit definitions of the symbols associated with model kinds and legends to be used by each view of the entity. Nevertheless, additional model kinds and legends can be used to address needed architecture considerations not covered by the framework used.

A specialized framework or one intended for implementation rather than reference can be expected to have more specific and possibly more detailed stakeholder concerns, specific architecture aspect properties and often a narrow portion of or no consideration of life cycle.

NOTE 2 Particular practice communities establish norms in areas where similar architecting is recurring: stakeholders with recurring concerns, conventions for addressing specific aspects, and architecting practices for establishing customary perspectives. An ADF will structure ADs according to these norms.

Usage of ADFs in different situations is likely to identify new combinations of architecture considerations and useful viewpoints, model kinds, views and correspondences. In different situations the following can occur:

— life cycle is omitted;

— only some stakeholders or architecture aspects are included;

— several different sets of architecture aspects that overlap are included;

— only architecture aspects divided into sub-domains or sub-groups are included;

— new concerns emerge and thus new or adapted viewpoints; or

— previously unrealized correspondences are identified.

Stakeholder concerns are often better understood when examined from different stakeholder perspectives across different architecture aspects, such as structure, behaviour and connectivity.

EXAMPLE 3 Some stakeholders look at an architecture from a business perspective and can be interested in the functionality that is required or provided (what capability of the entity is being created or changed, or what new processes are necessary?) and some stakeholders look at an architecture from an economic perspective and can be interested in the financial consequences of the same functionality (what are the investment implications and what is the expected impact on the bottom line?).

NOTE 3 ADFs frequently encompass both provisions for AD and additional architecting practices.

NOTE 4 Requirements on ADF are specified in 7.1.

Annex G gives more information about ADFs and how they can be related to the concepts and requirements of the document.

NOTE 5 ADFs identify one or more AD elements which are instances of the architectural constructs: stakeholder, concern, stakeholder perspective, architecture aspect, architecture viewpoint, model kind, legend, etc.

5.4.3 Architecture description languages

An ADL is a specified syntax and semantics intended for use in describing the architecture of an entity of interest. An ADL is a language for stakeholders, including those involved in the architecting effort, allowing the expression of architecture considerations by means of AD elements pertaining to the entity of interest, and the architecting environment. An AD can use more than one ADL, even a different ADL for each viewpoint.

NOTE 1 Informal or natural language architecture descriptions are possible but lack the semantic intent to describe architecture and are, therefore, not considered as cases of ADL use.

Often general-purpose modelling languages, e.g. UML, SysML, OPM, are embellished with profiles specifically intended for use in ADs.

EXAMPLE 1 UML profiles for architecture description are defined using stereotypes, tag definitions, and constraints applied to specific model elements (Classes, Attributes, Operations, and Activities) with a profile collection of such extensions collectively customized for a particular domain (e.g. aerospace, healthcare, financial) or platform (J2EE, .NET), and a modelling profile can define a particular model kind as part of its specification.

NOTE 2 ADLs can be employed (or devised) to express specific architectural considerations which an architect needs to address.

NOTE 3 The use of more than one ADL in an AD requires great care to avoid confusion and misunderstanding.

An ADL provides a way to create and understand the view components that compose into architecture views. Selection of a suitable ADL is made by considering view methods that specify how information is selected, transformed and presented in an architecture view. View methods determine the information that is captured when constructing ADs, used for the analysis of captured descriptions, and needed for the portrayal of architecture concepts and features.

ADLs can provide necessary rigor for development of an AD. The semantics of an ADL can be specified in an ontology, which can be refined further, in increasing levels of formality, as a vocabulary of terms using natural language, as a formal terminology (expressed as a metamodel), as an ontological theory (in form of a formal logical theory using axioms expressed in a suitably selected logic), or as an analytical theory (such as using differential equations, tensor calculus, etc.).

As transition occurs from a very generic reference ADF through a domain specific ADF and practice specific ADF to an implementation ADF, different ADL are likely to be employed to meet more refined specifications of architecture viewpoints for architecture views.

EXAMPLE 2 The transition from a generic Zachman Framework to a UAF framework involves a transition from general information and computing technology (ICT) ADL terminology to the UAF Profile using SysML notation and semantics, which in turn is transitioned to implementation-specific ADL for a particular project modelling profile extension of UML.

As multiple ADs are created, consistency or at least traceability amongst them is important to maintain; therefore, it is necessary to capture consistency conditions using correspondences (see 6.9.2) or a unified underlying ontology (see Annex A.7 on projective and synthetic view creation approaches). The constructs of the actual ADLs used in practice are then subsets of this integrated ontology.

EXAMPLE 3 AADL [48], ArchiMate [51], Systems Modelling Language (SysML) [38], ISO 19440 [10], Business Process Model and Notation (BPMN) [37], Unified Modelling Language (UML) [41], Unified Architecture Framework (UAF) Profile [40], and the viewpoint languages of RM-ODP [2][3].

NOTE 4 ADLs identify one or more AD elements which are instances of the architectural constructs: stakeholder, concern, architecture viewpoint, model kind, legend, etc.

Figure 7 provides the conceptual model for an ADL.

NOTE 5 Requirements on ADLs are specified in 7.2.

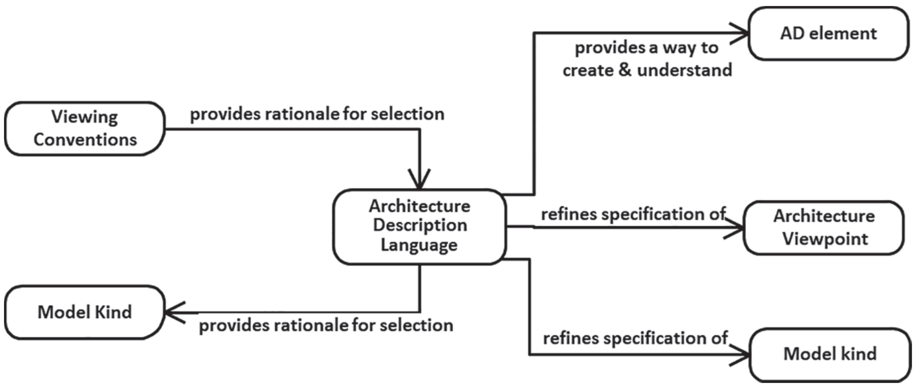
AD element
provides a way to create & understand
Viewing Conventions
provides rationale for selection
Architecture Description Language
refines specification of
Architecture Viewpoint
Model Kind
provides rationale for selection
refines specification of
Model kind

Figure 7 — Conceptual model of an architecture description language

6 Specification of an architecture description

6.1 Introduction

In this document, the specification of an AD includes or references the following items:

— AD identification, overview and supplementary information (see 6.2);

— identification of the entity of interest's stakeholders (see 6.2)

— identification of the relevant concerns (see 6.3);

— identification of the relevant stakeholder perspectives (see 6.4);

— identification of the relevant architecture aspects (see 6.5);

— a definition or a reference for each architecture viewpoint used in the AD (see 6.6);

— architecture view(s) for each architecture viewpoint used (see 6.7);

— architecture view component(s) that are shared across the architecture views (see 6.8)

— a record of known inconsistencies among the AD's required contents (see 6.9.1);

— applicable AD correspondences and AD correspondence methods (see 6.9.2 and 6.9.3);

— applicable architecting stages, domains, key concepts and features, applicable principles;

— applicable AD element correspondences and AD elements correspondence methods (see 6.9.2 and 6.9.3); and

— rationales for architecture decisions made (see 6.10).

NOTE This document does not specify a format for ADs.

6.2 Architecture description identification and overview

An AD shall identify the entity of interest and shall include identifying and supplementary information as determined by the project and/or organization.

The detail of the identifying and supplementary information items to be included shall be as specified by the organization and/or project.

EXAMPLE 1 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.

NOTE 1 See ISO/IEC 15289 [6] or ISO/IEC 15504-6:2013, B.1 [9] for additional examples.

An AD shall include a statement of its intended purpose.

NOTE 2 For a reference architecture, the entity of interest is abstract, and the purpose of the AD is to provide a specification for further ADs.

An AD shall consider the recommendations made in previous evaluations of the architecture or other ADs of related architecture. Non-conformances shall be explained with rationales.

An AD shall identify the stakeholders considered relevant to the AD and stakeholders of the architecture of the entity of interest.

EXAMPLE 2 Stakeholders include users, operators, acquirers, owners, suppliers and vendors, architects, designers and developers, implementers, maintainers, regulators (including government), testers, public-at-large, adversaries and competitors.

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

Consideration shall be given to identifying present or future stakeholders who may be impacted by the entity of interest (e.g. by its realization, operation, decommissioning, or merely because of any resulting social, environmental, political and financial impact).

6.3 Identification of concerns

An AD shall identify the concerns considered relevant to the architecture of the entity of interest.

EXAMPLE Concerns include the following kinds of considerations: suitability of the architecture for achieving the objectives for the entity of interest, enterprise capability and capacity to implement the entity of interest, the feasibility of realizing and operating the entity of interest, the potential risks and impacts of the entity of interest to its stakeholders throughout its life cycle, added value to the stakeholder(s), reuse of known architectures, resilience, extensibility, adaptability, evolvability, latency, resource utilization, effectiveness, operability, usefulness, usability, interoperability, complexity, sustainability and evolvability of the entity of interest, environmental impacts of the development, use, and disposal of the entity of interest.

An AD shall associate each identified concern with any identified stakeholders holding that concern.

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

An AD should consider present or future concerns which may be relevant to the entity of interest and shall identify each such concern.

NOTE 2 Concerns expressed as interrogative questions and with appropriate detail to the purpose of the AD enable more efficient and effective communication.

NOTE 3 Resource limitations or other constraints can prevent an AD from addressing all concerns expressed by all identified stakeholders. Concerns that are considered fundamental to the architecture should be prioritized for addressing within the available resources. Concerns that are not addressed should be listed to record that the concern exists but has not been addressed. Nevertheless, concerns not addressed by AD are areas of risk regarding stakeholder satisfaction/agreement, decision-making, feasibility of solution, performance of the project and benefits for the involved organization(s).

6.4 Identification of stakeholder perspectives

An AD shall identify stakeholder perspectives considered relevant to the architecture of the entity of interest.

EXAMPLE 1 Stakeholder perspectives include the following kinds of perspective: strategic, capability, operations, services, personnel, resources, security, systems, projects, standards, business, application, technology, logical, physical, implementation, migration, and organization. The stakeholder perspectives in this example are those identified by some of the current architecture frameworks. See Annex F for information about published ADFs.

EXAMPLE 2 Other stakeholder perspectives include the following kinds of perspective: capacity, interfaces, evolvability, transition, organization, viability, cost, risk, usability or operability, legal compliance, safety, ease of maintenance, market acceptability and share, and ease of implementation.

An AD shall identify the relationship of each identified concern with relevant stakeholder perspectives.

NOTE 1 This document does not prescribe: the granularity of concerns; the granularity and dependencies of stakeholder perspectives; how stakeholder perspectives relate to each other; or how stakeholder perspectives relate to other statements about an entity such as stakeholder needs, entity goals, or entity requirements. These issues are subjects for specific AD, ADFs, architecting methods, or other practices. See Annex F for examples of architecture frameworks that use particular stakeholder perspectives as one of the framework dimensions. An AD shall associate each identified perspective with the identified stakeholders holding that perspective.

For each identified perspective, an AD shall enumerate its resulting concerns.

NOTE 2 This document does not prescribe: the granularity and dependencies of stakeholder perspectives; how stakeholder perspectives relate to each other; or how stakeholder perspectives relate to other statements about an entity such as stakeholder needs, entity goals, or entity requirements. These issues are subjects for specific AD, ADFs, architecting methods, or other practices. See Annex F for examples of architecture frameworks that use particular stakeholder perspectives as framework dimension.

NOTE 3 Resource limitations or other constraints can prevent the AD from addressing all identified stakeholder perspectives.

The architecting effort should identify present or future stakeholder perspectives which may be relevant to the entity of interest.

6.5 Identification of architecture aspects

An AD shall identify architecture aspects considered relevant to the architecture of the entity of interest.

EXAMPLE 1 Architecture aspects include considerations such as: data, activity, function, location, people, time, motivation, taxonomy, structure, connectivity, behaviour, information, parameters, constraints, roadmaps, traceability, and requirements. The architecture aspects in this example are those identified by some of the current ADFs. See Annex F for information about ADFs.

EXAMPLE 2 Other architecture aspects include the following kinds of aspect: spatial, structural, functionality, networking, computing, dynamicity, performance, accessibility, quality of service, security, organization, statutory, regulatory, cost, harmony, comfort, disturbance, behavioural, operability, logical, and physical aspects.

NOTE 1 This document does not prescribe: the granularity and dependencies of architecture aspects; how architecture aspects relate to each other; or how architecture aspects relate to other statements about an entity such as stakeholder needs, entity goals, or entity requirements. These issues are subjects for specific ADFs, architecting methods, or other practices. See Annex F for examples of ADFs that use particular architecture aspects and stakeholder perspectives.

NOTE 2 Resource limitations or other constraints can prevent the AD from addressing all identified stakeholder perspectives associated with particular architecture aspects. Likewise, these constraints can prevent the AD from addressing all architecture aspects associated with the stakeholder perspectives of interest. Some architecture aspects might not be relevant for all stakeholder perspectives.

Architecture aspects that are considered necessary for proper description of the architecture should be prioritized so that they are addressed within the available resources and constraints.

Each identified aspect shall be associated with the concerns it refines.

The architecting effort should identify present or future architecture aspects which may be relevant to the entity of interest.

6.6 Inclusion of architecture viewpoints

An AD shall include or reference each architecture viewpoint used therein.

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

The specification associated with each included architecture viewpoint shall be in accordance with the provisions of Clause 8.

Each concern identified in accordance with 6.3 shall be framed by at least one architecture viewpoint.

Each perspective identified in accordance with 6.4 shall be mapped to the viewpoints which cover that perspective.

NOTE 1 This document does not require any particular specifications of architecture viewpoints to be used.

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

NOTE 3 A viewpoint can serve as a contract between architect and stakeholders. For the concerns framed by the viewpoint, architect and stakeholders can agree on what notations and representational conventions will be used to address those concerns. This contract agreement can be made before any detailed architecting is undertaken to reduce or avoid surprises.

6.7 Inclusion of architecture views

An AD shall include one or more architecture views for each architecture viewpoint used. Where a possible ambiguity exists, the AD shall clarify the applicability of each view to the entity of interest.

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

EXAMPLE 1 Several kinds of views can be described for a functional viewpoint: for example, functional chains express the behaviour and function trees express decomposition.

Each concern identified by the AD in accordance with 6.3 shall be addressed by at least one view in accordance with the view's governing viewpoint. Nevertheless, resource limitation or other constraints can lead to focus on a subset.

Each stakeholder perspective identified by the AD in accordance with 6.4 shall be addressed by at least one view in accordance with the view's governing viewpoint. Nevertheless, resource limitation or other constraints can lead to focus on a subset.

Each architecture aspect identified by the AD in accordance with 6.5 shall be addressed by at least one view in accordance with the view's governing viewpoint. Nevertheless, resource limitation or other constraints can lead to focus on a subset.

Each architecture view shall adhere to the conventions of its governing architecture viewpoint. Each architecture view may address more than one concern.

Each architecture view shall include or reference:

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

b) identification of its governing architecture viewpoint;

c) one or more view components that address all of the concerns (per 6.3) framed by its governing architecture viewpoint (Per 6.6) and that cover some or all of the entity of interest with respect to that viewpoint; and

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

NOTE 1 "Known issues" per d) include unresolved issues, risks, exceptions and deviations from the governing model kinds or legends. Open issues can lead to decisions to be made. Exceptions and deviations can be documented as decision outcomes and rationale (per 6.10).

NOTE 2 It is not necessary to require that each view covers the entire entity of interest with regard to the purpose and scope of the AD. It might for example be scoped to purposely be limited to one particular portion of the entity, sometimes by direction, sometimes by limited time or resources, or sometimes based on the narrow scope of the architecting effort

An AD may include other information which is not part of any architecture view.

EXAMPLE 2 Information items not part of any view could include overviews of the entity of interest, architecture principles, architecture patterns and architecture styles whose application spans more than one view; referenced bases for the architecture, such as domain or reference architectures; correspondences between views; and architecture rationale. This information can assist stakeholders and other users of the AD responsible for its maintenance and development.

6.8 Inclusion of view components

An architecture view shall be composed of one or more view components in accordance with its governing architecture viewpoint.

Each view component shall include version identification as specified by the organization and/or project.

Each view component shall identify its governing model kind, if any, and adhere to the conventions of its governing architecture viewpoint (see 6.6).

When a view component does not have a governing model kind, i.e. for an information item not described with a model, a *view component legend* shall be included to specify the conventions used in that view component.

EXAMPLE An information item may be a record of expert opinion, rather than a formal model that one can analyse using calculations, simulation, or any other suitable analysis method.

NOTE 1 Within a view, one or more view components can be used to selectively present some or all of the informational content required by the architecture viewpoint to highlight points of interest.

A view component may be a part of more than one architecture view. The relationships among components shared across such views can be expressed as correspondences.

NOTE 2 Sharing view components between architecture views permits an AD to capture distinct but related concerns without redundancy or repetition of the same information in multiple views and reduces possibilities for inconsistency. Sharing of view components also permits an aspect-oriented style of AD: view components shared across architecture views can be used to express architectural perspectives (see [45]); view components shared within an architecture view can be used to express architectural textures (see [42]). View components can be used as "containers" for applying architecture patterns (see [21]) or architecture styles to express fundamental schemes (such as layers, three-tier, peer-to-peer, model-view-controller) within architecture views.

NOTE 3 This document does not prescribe how view components are created. For example, they can be individually constructed, generated by automated tools, derived from or based upon other information sources and models.

NOTE 4 This document does not prescribe the level of formality of view components to be used in an AD. While model-based view components that have a formal specification of semantics and syntax may be less ambiguous, non-model-based view components can also be used effectively.

6.9 Recording of architecture relations

6.9.1 Consistency within an architecture description

An AD shall record any known inconsistencies.

An AD should include or reference an analysis of consistency of its architecture views, its view components and other AD elements.

Correspondences and correspondence methods, as specified in 6.9.2 and 6.9.3, may be used to express, record, enforce and analyse consistency between views, their view components and other AD elements within and among ADs.

NOTE While consistent ADs are 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.

6.9.2 Correspondences

An AD shall include or reference a list of AD element correspondences.

An AD element correspondence shall identify its participating AD elements.

An AD element correspondence may involve elements within an AD or across several ADs.

An AD element correspondence shall identify any governing correspondence methods (see 6.9.3).

Each AD correspondence shall identify the participating ADs.

Each AD correspondence shall identify any governing correspondence methods (see 6.9.3).

NOTE 1 The AD elements in a correspondence need not be distinct. A correspondence can be defined between an AD element and itself. Examples of "self-referential" correspondence are an activity is refined into two or more activities of the same kind, or an activity that can recursively invoke itself. The ADs in an AD correspondence need not be distinct. A correspondence can be defined between an AD and itself.

NOTE 2 AD Correspondences can be used to express relations among ADs, ADFs, and ADLs. See Zachman Framework [56] example in Annex F.

EXAMPLE A correspondence can have more than two participating AD elements, such as: An AD element satisfies a Requirement as demonstrated by an Evaluation Method: TracesToDemo (an AD Element, a Requirement, an Evaluation Method).

6.9.3 Correspondence methods

An AD shall include or reference a list of correspondence methods applying to itself or its AD elements.

For each applied correspondence method, an AD shall record whether the method *holds* (is satisfied) or otherwise record all known violations.

NOTE 1 A correspondence method applying to one or more AD elements could originate in the AD elements, in the AD itself; in the specification of a model kind or an architecture viewpoint used for the AD (see 8); or in the specification of an ADF or ADL used therein (see 7).

An AD shall include or reference each correspondence method applying to it.

NOTE 2 A correspondence method applying to an AD could originate in the AD; in the specification of a viewpoint or a model kind (See 8); or in the specification of an ADF or ADL selected for use in that AD (See 7).

NOTE 3 A correspondence method holds if an associated correspondence can be shown to be satisfied. A correspondence method is violated if an associated correspondence cannot be shown to be satisfied or when no associated correspondence exists.

6.10 Recording of architecture decisions and rationale

6.10.1 Decision recording

An AD should record architecture decisions considered to be key to the architecture of the entity of interest.

Since recording every architecture decision about an entity of interest is not practical, 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 AD. Criteria to consider are:

a) decisions regarding architecturally significant requirements;

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

c) decisions affecting key stakeholders or a number of stakeholders;

d) decisions necessitating intricate or non-obvious reasoning;

e) decisions that are highly sensitive to changes;

f) decisions that could be costly to change;

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

h) decisions affecting identification of fundamental concerns;

i) decisions that result in the replacement of assumptions with known information;

j) decisions that result in significant capital expenditures or indirect costs;

k) decisions affecting performance /evolvability;

l) decisions linked to requirement compliance;

m) decisions linked to technical standard selection;

n) decisions linked to system vulnerability mitigation. When recording decisions, the following should be considered:

o) decision is uniquely identified;

p) decision is stated;

q) decision is linked to the concerns about or aspects of an entity to which it pertains;

r) decision authority or owner is identified;

s) decision is linked to AD elements affected by the decision;

t) rationale linked to the decision;

u) constraints and assumptions that influence the decision are identified;

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

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

x) citations to sources of additional information are provided.

NOTE 1 The lists of decisions are not intended to be exhaustive.

NOTE 2 It can be useful to record rejected alternatives and the rationale for those rejections. It can be the case that in the future these reasons will no longer apply, and the decision might need to be reconsidered.

NOTE 3 It can be useful to record relationships between architecture decisions.

EXAMPLE Examples of types of relationships are: constrains, influences, enables, triggers, forces, subsumes, refines, conflicts-with, and is-compatible-with (see [30] and [56]). Relations among decisions may be captured via correspondences or by applying correspondence methods.

6.10.2 Rationale recording

An AD should include a rationale for each architecture viewpoint included for use (per 6.6) in terms of its stakeholders and architecture considerations.

An AD should include a rationale for each ADF and each ADL selected for use in terms of its stakeholders and architecture considerations.

An AD shall include rationale for each decision considered to be a key architecture decision (per 6.10.1).

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

An AD should include a rationale for AD limitation (e.g. resource problem, timing problem, and effort avoided for well-known description already covered by other ADs).

7 Architecture description frameworks and architecture description languages

7.1 Specification of an architecture description framework

The specification of an ADF shall include or reference:

a) information identifying the ADF and its intended scope of applicability;

b) the identification of one or more typical stakeholders (per 6.2);

c) the identification of one or more typical concerns held by typical stakeholders (per 6.3);

d) the specification of one or more architecture viewpoints that frame those typical concerns (per 8.1);

e) the identification of one or more stakeholder perspectives, if applicable (per 6.4);

f) the identification of one or more architecture aspects, if applicable (per 6.5);

g) the definition of one or more structuring formalism to organize viewpoints, if applicable (per 5.4.2);

h) the specification of one or more model kinds that apply to these architecture viewpoints, if applicable (per 8.2);

i) the definition of one or more legends that apply to these architecture viewpoints, if applicable;

j) identification of ADLs that can be used to create views (per 7.2);

k) any applicable correspondence methods (per 6.9.3);

l) any applicable view methods (per 8.3);

m) any relevant domain(s) of applicability; and,

n) any version identification as specified by the organization and/or project.

NOTE 1 The intended scope of use may range on a scale between the very generic (all industries and application domains and all kinds of entities of interest (EoI), and the use of the ADF for multiple purposes) through to the very special or particular (such as intended to cover a given industry, application domain, or kind of EoI, a particular EoI, or a particular EoI in a given stage of its life, or for a particular purpose). This classification is similar to the genericity dimension defined as a structuring formalism in ISO 15704 [10].

NOTE 2 The word *typical* above is intended to mean 'typical in the intended scope of applicability.'

The specification of an ADF should include conditions of applicability.

EXAMPLE 1 The following are conditions of applicability: An AD using a chosen architecture framework will identify certain stakeholders when the entity of interest operates within a certain jurisdiction. An AD using a chosen architecture framework is permitted to omit a certain viewpoint when no real-time system concerns have been identified. When using a chosen architecture framework, a particular model kind can be omitted unless a specific stakeholder is among those identified.

The specification of an ADF shall establish its consistency with the concepts in 5.4.2.

NOTE 3 The above requirement can be met through a metamodel, a mapping of framework constructs to the requirements in Clause 5, a text narrative, or in some other manner.

An AD *adheres to* a specification of an ADF when:

— each applicable stakeholder identified in the ADF has been considered and identified in the AD (per 6.2);

— each applicable concern (per 6.3) identified in the ADF has been considered and identified in the AD;

— each applicable architecture aspect (per 6.5) and stakeholder perspective (per 6.4) in the ADF has been considered and identified in the AD;

— each applicable viewpoint specified by the ADF (per 8.1) is included in the AD;

— each applicable correspondence method specified by the ADF is included in the AD (per 6.9.3); and

— the AD conforms to the requirements of Clause 6.

NOTE 4 *Applicable* means when conditions of applicability are met.

A specification of an ADF may establish additional rules for adherence.

NOTE 5 An AD could adhere to one or more specifications of ADFs, or to no framework specifications. For an AD to adhere to more than one framework specification would entail a reconciliation between each framework specification's identified stakeholders, concerns, architecture aspects, stakeholder perspectives, architecture viewpoints, model kinds, and correspondence methods within the AD.

A structuring formalism utilizes one or more structural categories.

EXAMPLE 2 Structural categories include things such as: domains, model kinds, perspectives, aspects, interrogatives, levels of abstraction, subjects of concern, aspects of concern, phases, layers and architectural tiers.

NOTE 6 See Annex F for examples of architecture frameworks using architecture aspects, stakeholder perspectives and other structural categories.

7.2 Specification of an architecture description language

The specification of an ADL shall include or reference:

a) the identification of one or more view methods (per 8.3) to be selected by the ADL;

b) one or more model kinds (per 8.2) implemented by the ADL which frame the relevant concerns or reflect the relevant architecture aspects;

c) architecture viewpoints (per 8.1) implemented by the ADL, if applicable;

d) correspondence methods (per 6.9.3); and,

e) any version identification as specified by the organization and/or project.

8 Architecture viewpoints and model kinds

8.1 Specification of an architecture viewpoint

The specification of an architecture viewpoint shall include or reference:

a) one or more concerns (per 6.3) framed by this architecture viewpoint;

b) relations among view components (per 6.8);

c) any stakeholder perspectives associated with this viewpoint (per 6.4);

d) one or more architecture aspects refining those concerns (per 6.5);

e) known typical stakeholders (per 6.2) holding those concerns which are framed by this architecture viewpoint (per item a));

f) a specification of which model kinds and legends are to be used when constructing views (per 8.2);

g) references to any sources.

A specification of an architecture viewpoint should identify view methods used to create, interpret or analyse views governed by the associated architecture viewpoint (per 8.3); and one or more model kind (per 8.2).

Each legend (as per item f)) shall provide guidance to readers in interpreting the (non-model) view components which it governs.

The specification of an architecture viewpoint should identify any correspondence methods.

The specification of an architecture viewpoint should identify any view methods used to create, interpret and analyse views governed by this viewpoint (per 8.3).

NOTE 1 The specification of an architecture viewpoint can be included as part of an AD (Clause 6), as a part of the specification of an ADF or ADL (Clause 7) or individually using the requirements of this Clause.

NOTE 2 When a specification of an architecture viewpoint is included and applied in an AD, the typical stakeholders of item e) are replaced by the *known* stakeholders identified in the AD.

NOTE 3 This document does not require any particular specifications of architecture viewpoints to be used.

NOTE 4 Annex B provides guidance to specification of architecture viewpoints. Annex C provides examples of specifications of architecture viewpoints.

8.2 Specification of a model kind

The specification of a model kind shall include or reference:

a) the identification of one or more concerns, perspectives and aspects (per 6.3, 6.4 and 6.5);

b) definition of a language, notation, convention, or modelling technique comprising the model kind;

c) the contents of view components of that kind (per 6.8);

d) any view methods to be used on view components of this kind (per 8.3);

e) any version identification as specified by the organization and/or project;

f) any sources of this model kind information.

NOTE Item b) can be met in a number of ways such as with a metamodel, grammar or template for the specification of a model kind that defines the structure and interpretation of its models (see B.2.9).

8.3 View methods

View methods define operations that apply to views. A specification of an architecture viewpoint may include one or more view methods. View methods provide guidance, heuristics, metrics, patterns, design rules or guidelines, and best practices to aid in view construction and use of views governed by the associated architecture viewpoint.

View methods shall be defined in the specifications of model kinds (8.2), viewpoints (8.1), ADLs (7.2), and ADFs (7.1) when it is necessary to provide guidance on how views are constructed or used.

EXAMPLE View methods pertaining to: chaining dependencies to assess the impact of a change; workshops to trade-off qualities or other concerns; boundary analyses to determine whether context and entity of interest are well defined; guidance on partitioning; analysis of architectural complexity; creation and enforcement of architecture styles (such as layered, aspect-oriented); pattern families to promote intended properties of the entity of interest; analyses against requirements for completeness and coverage; interpretation and integration of external models as information sources.

If a model is used as an information source when creating a view, a view method should define how AD elements will be portrayed in the view component, how model data are transformed or translated for use in the view component and how relationships between information from different sources are portrayed in the view component.

If a “non-model” is used as an information source when creating a view, a view method should define how its contents are portrayed in the view component as AD elements, how the non-model-related data are transformed or translated for use in the view component, and how relationships between information from different sources are portrayed in the view component.

Annex A
(informative)

Notes on terms and concepts

A.1 Introduction

This Annex discusses the principles, concepts and terms on which this document is based. This document makes use of several terms (*architecture*, *concern*, *architecture aspect*, *stakeholder perspective, architecture view, architecture viewpoint, view component, model kind)* which are in wide usage with several different meanings across the community. This Annex discusses these terms, the motivations for their definitions in this document, and contrasts these definitions with other usages.

This document defines minimal requirements on ADs to support the scope established in Clause 1. The approach is to allow organizations maximum flexibility in applying the standard while demonstrating conformance with the requirements in Clauses 6, 7 and 8. Given the multi-disciplinary nature of architecting, the intent is to meet the needs of multiple stakeholders and allow different ways to describe the architecture of an entity of interest. The organization of ADs into architecture views governed by architecture viewpoints provides a mechanism for the separation of concerns among the stakeholders, while providing an integrated view of the whole entity that is fundamental to the notion of architecture.

Establishing the quality of an architecture being described by a conforming AD (*Is this a good architecture?*) or the quality of an AD itself (*Is this AD complete and consistent?*) are factors for the *evaluation* of the AD. This document does not presume to impose conditions that are required for quality considerations. It does recommend that results of such evaluations be recorded (per 6.2).

NOTE Evaluation of architectures is the subject of ISO/IEC IEEE 42030 [18].

A.2 Entities and their architectures

In this document, the term *architecture* is intended to convey the essence or fundamentals of an entity of interest. There are several key aspects to the definition of architecture (3.2) in this document. 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 an entity 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 entity and their relationships. Sometimes, the fundamental elements are functional or logical elements. In other cases, what is fundamental or essential to the understanding of an entity are its overarching principles or patterns. Properties can also be the fundamental characteristics of an entity, its elements and their relationships. The definition of architecture in this document is intended to encompass these distinct, but related uses, while encouraging a more rigorous delineation of what constitutes the architecture of an entity.

The phrase “concepts or properties” is used in the definition of architecture (3.2) to allow two differing philosophies to use this document without prejudice. These two philosophies are:

— *Architecture as Concept*: wherein architecture is a conception of an entity in one’s mind; and

— *Architecture as Property*: wherein architecture is a property or attribute of an entity of interest.

Empirical studies have discovered four metaphors for architecture found in organizations [50]:

— architecture as blueprint;

— architecture as literature;

— architecture as language;

— architecture as decision.

The conceptual foundation of this document 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 document: that architecture is inherently based upon multiple stakeholders with multiple concerns and multiple architecture aspects.

A.3 Concerns

This document uses the term *concern* to mean any topic of interest pertaining to the entity being architected or to the architecture itself. Stakeholders, including the architect, of that subject *hold* these concerns. Some concerns drive or relate to the architecture and therefore this document requires their identification as a part of the AD. Concerns range over a wide spectrum of technical and personal interests and are often narrowly focused or vaguely stated, necessitating further elaboration before the architecting effort can address them successfully.

The motivation for using this term comes from the phrase “separation of concerns” in software and systems engineering, coined by Edsger W. Dijkstra [24].

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 “focusing 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.

As specified in this document, each architecture viewpoint frames one or more concerns (see 6.6) so that a view resulting from the application of that architecture viewpoint addresses those specific known concerns for the entity of interest. Separating the treatment of concerns by views allows stakeholders to focus on what is of special interest to them and offers a means of organizing and managing complexity of the AD (see 6.7). The literature of systems and software engineering records a large inventory of such concerns. Examples are given in 5.2.3.

A.4 Aspects and perspectives

A.4.1 Introduction

Historically architecting efforts were driven by the concerns of stakeholders. However, the advent of ADFs (and to a much lesser extent ADLs) established practices that drew upon prior architecting experience which resulted in the organizing of architecting efforts which are not necessarily driven by concerns specific to the architecting effort in question but significantly driven by this prior experience. This approach to architecting enabled specific concerns to be identified at a later point in time after first building architecture views using the ADF-driven approach.

This prior experience is typically encapsulated in grid-based ADFs, typically of two dimensions, but sometimes of three or more dimensions. While there is no uniformity in the established practice as to what the respective rows and columns comprise, it appears there are at least two orthogonal sets which are prevalent.

One such dimension is the "architecture aspect." Architects do not necessarily address all aspects simultaneously nor even all possible aspects but tend to address them in a specific order (with iteration as appropriate) dependent upon the architecting objectives, prior experience and any method being applied. Some aspects might not be important for particular architecting efforts (e.g. because they are not relevant to the architecture of interest, or its kind of architecture, or because they are not a driver for that specific architecture).

The other dimension "stakeholder perspective," is also fundamental and captures what an architect does, namely employing different ways of thinking about an entity or its architecture driven e.g. by concerns, prior experience or method. Perspective is driven more by architecting thinking and approach. Aspect is driven more by what is or might be needed to be covered in an AD. Concerns form (and are amenable to being addressed through and mapped onto) some combination of one or more architecture aspects and one or more stakeholder perspectives.

The use of aspects and perspectives is compatible with a more organized and disciplined (and standardized) approach to both architecting and AD. It is noted that some architectural issues can be direct stakeholder concerns in particular domains or circumstances. In such cases, they are usually addressed through the mechanism of stakeholder perspectives. In other cases, they are not, but instead can be addressed through the mechanism of architecture aspects. This gives rise to variation in the organizing of material employed in commonly used ADFs.

In this document, this concept of framework dimensions is called "structural categories" since the way these categories are used in various ADFs is not always aligned with the concept of "dimensions." So, the structural categories term is used herein since it is a broader and more inclusive concept.

A.4.2 Architecture aspects

This document uses the term *architecture aspect* as an organizing basis for views in an architecture description. Aspects can be used to focus architecture considerations such as structural or behavioural characteristics into cohesive subsets of an architecture description. Collectively the architecture aspects provide the basis for capturing all of the relevant architecture considerations with respect to the architecture.

The aspects align with technical architecting specializms which access relevant architectural information to be able to analyse, synthesize, assess, elaborate their particular aspect(s) and in turn add to (i.e. develop, embellish, elaborate on, increase confidence in, etc.) the architectural information. Examples of such specializms include information architects and analysts, security architects, ARM (RAM-D) engineers, human factors specialists, human organization experts, and cost forecasters.

Architecture aspects provide a way to partition the architecture to enable a more systematic examination of the architecture's fundamental concepts such as structure and properties, and the evaluation of architecture alternatives. Aspects are neutral with respect to any particular concern although these concerns can be mapped to many relevant aspects. By examining the aspects of an architecture, certain relevant features or properties of the entity can be discerned or predicted. Aspects such as structure and behaviour can be characterized as abstractions. These abstractions can be defined as ontologies and are typically shared (for example in reusable modelling patterns) within and across professional communities and application domains.

Architecture aspects often come from domain knowledge and from the experience of engineers and architects. They also come from the ADFs that have found certain aspects to be useful for architecting for the scoped domain for that framework. These aspects are also embedded in methods that are taught to architects and encapsulated in some modelling tools.

The nature of architecture aspects and concerns are different. Concerns can apply to an entity of interest or an architecture whereas an aspect is a characteristic or a feature of the architecture itself.

Treating a "stakeholder concern" as the starting point for creating architecture views assumes that one already knows a priori who the stakeholders are and what their concerns might be. In many complex entities, this is not feasible. Some stakeholders are often not known until creation of some architecture views for the aspects considered to be relevant and customary for the domain. When these views are shown to potential stakeholders, then they can likely reveal to us their concerns they might have (if any) for the implied architecture solution.

Some architecture aspects can be considered to be primary (e.g. functional and structural aspects). The primary aspects can be subdivided on the basis of what additional attributes need to be considered. For example, the functional aspect can be described by a functional decomposition diagram; the process or behavioural aspect that helps one understand the process that performs this function can be described by an activity diagram or a sequence diagram or a coloured Petri net; the state transitions can be described using state machines; the statistical properties of the process can be described by a simulation model, and so on.

Concerns of the architect as a stakeholder will typically differ from aspects in being not as extensive in scope and less structured in form. Over time and following review and consolidation these concerns can become codified into aspects, usually as they apply to a specific domain of application.

Architecture aspects can be used to examine the entity to get a more complete understanding of how the architecture addresses that concern. Architecture aspects can likewise aid understanding to what extent the architecture is not addressing that concern. The relationship between concerns and architecture aspects is complex in that (1) one can analyse, for example, the entity's behaviour (as an aspect) against several concerns, like: portability, performance, etc.; and (2) one can assess a concern such as security through analysis of, for example, behavioural, structural and organizational aspects. Architecture aspects are also useful during evaluation of alternative architectures. The concept of aspects has been used in software development to deal with "cross-cutting concerns" (See [30]).

A.4.3 Stakeholder perspectives

This document uses the term *stakeholder perspective* to mean a particular way of thinking about an entity, especially one that is influenced by one's beliefs or experiences. The way one thinks about an entity (i.e. one's perspective) can be influenced by organizational role, training, experience, knowledge, personality, character traits, culture, peer pressure, etc. Different ways of thinking about an architecture are often employed when architecting.

EXAMPLE From UAF[39]: Strategic, operational, services, personnel, resources, security, projects, standards, actual resources. From ArchiMate: Strategy, business, application, technology, physical, implementation, migration. From NAF[34]: Concepts, service specifications, logical specifications, resource specifications, architecture metadata. From DODAF[25]: Capability, operational, services, systems, standards, data, and information, projects. From ISO15704/GERAM-ISO [10]: Identity, concept, requirements, preliminary design, detailed design, implementation, operation, decommissioning.

Perspective leads to an understanding of how focused aspects of a subject relate to each other and to the whole, such as, by stakeholder (stakeholder type), domain, phase, "level of abstraction", etc. A perspective might lead to organized sets of architecture viewpoints such as by stakeholder (stakeholder type), domain, phase, level of abstraction, etc. For a given perspective there are usually multiple aspects to be considered. The determination of relevant perspectives is dependent upon the interests of, and stances adopted by, the various stakeholders that are relevant to the architecture.

A.4.4 Structuring formalisms and structural categories

An ADF can provide a structuring formalism, i.e. a set of rules for using architecture considerations and correspondences between them, to organize the architecture viewpoints used to generate associated views, e.g. a grid framework formalism. The purpose of the structuring formalism is to provide ways of representing relationships among various elements of the architecture and enhancing opportunities for analysis of interactions among those elements.

EXAMPLE 1 Well known structuring formalisms include: RM-ODP [1], GERAM cube [10], Reference Architectural Model Industrie 4.0 (RAMI 4.0) [27], TOGAF stack (business, information systems, technology) [52], NAF grid [34], UAF grid [39], and Zachman Framework matrix [56].

An ADF can define structural categories used in the structuring formalism. Sometimes these categories are represented by "dimensions" in a graphic portrayal, such as the rows and columns used in several frameworks. A structuring formalism in grid form usually has two framework dimensions but formalisms can have a single framework dimension, often segmented in a multi-layer hierarchy, or many framework dimensions where visual representation of framework dimensions above three is difficult.

EXAMPLE 2 A two-dimensional grid is used in some ADFs where stakeholder perspectives correspond to rows and architecture aspects correspond to columns.

The structural categories (i.e. dimensions) of the formalism can include things such as domains, model kinds, perspectives, aspects, interrogatives, levels of abstraction, subjects of concern, aspects of concern, phases, layers, architectural tiers.

Some commonly used ADFs have a two-dimensional grid or matrix to organize their specifications of architecture viewpoints. The two-dimensional grid originated with Zachman [56]. (Today, many other forms can be found such as cubes, stars, pentagons and ellipses.) These are examples of structuring formalisms used on those frameworks.

The rows in these grids are the perspectives referred to in this standard, although the names of these rows in the frameworks will vary: UAF [40] calls them domains, ArchiMate [51] calls them layers, NAF [34] calls them “subjects of concern”, GERAM [10] has two of its three primary dimensions representing extent of abstraction (genericity) and life-cycle modelling phase. The underlying idea is the same across these frameworks and this concept has been adopted by this standard. The concept also makes explicit that a viewpoint provides a perspective.

A.4.5 Relationship between architecture aspect and stakeholder perspective

Perspective and aspect are closely related and often confused. While perspective is the way one thinks about something, aspect is the way in which one views something. The properties or concepts associated with the entity are those perceived when viewing it from a particular aspect and thinking about it from a particular perspective. When the perspective is shifted, often the properties or concepts are different. Likewise, when the viewing aspect is changed then different properties or concepts about the entity can be discerned.

Aspects and perspectives are commonly used in the ADF as a way to organize architecture viewpoints as illustrated by the examples described in Annex F (see Figure F.1 and Figure F.2). An architecture view can be constructed for a particular aspect and a particular perspective. The perspective can represent an aggregation of concerns held by one or more stakeholders. When an architecture framework matrix or grid uses these concepts, the columns usually represent aspect-related items and the rows usually represent perspective-related items.

A.4.6 Architecture Considerations

The combination of stakeholder perspectives and architecture aspects to conduct architecting in a manner informed by prior experience can serve as a different and complementary approach to architecting driven by identified stakeholders and their specific concerns (see 5.2.3). For example, using prior experience in conducting the architecting of a similar entity situated in a similar environment, potential issues can be identified which when raised with stakeholders give rise to real concerns.

Using established architecture aspects from known stakeholder perspectives can be enormously helpful in reducing architecting effort to arrive at suitable architecture viewpoints. However, architects need to exercise care not to let these architecture aspects and stakeholder perspectives of prior architecting effort overshadow concerns not previously expressed nor to discount emerging concerns as the architecting project unfolds.

The roles fulfilled by architecture aspects and stakeholder perspectives are different. For a given perspective there are usually multiple architecture aspects to be considered. When the stakeholder perspective is changed, then the properties or concepts that are perceived are different.

A.5 Non-functional properties

Non-functional properties such as performance, cost, and quality factors (like reliability, confidentiality and resilience) are stakeholder concerns that are often structured using the notion of aspects. These non-functional properties are often termed “ilities.”

The -ilities morph from a problem focus (e.g. effectiveness), to a focus on solution (e.g. performance, availability) and implementation (e.g. capacity, response time, dependability, mean time between failure). The notion of aspects can be used to cover the -ilities and allow the architects to address all necessary considerations proficiently. Stakeholders could be concerned with specific -ilities as applicable to problem, solution or even implementation but their concerns are unlikely to cover (in a topological sense) all considerations.

The -ilities need to be structured and some will manifest in specific views (e.g. as parametric performance diagrams), as aspects (groupings of views e.g. state-based behaviour, non-state-based behaviour), as multiple aspects, overlays on views (further views), etc.

A.6 Architecture views and viewpoints

The terms *architecture view* and *architecture viewpoint* are central to this document. Although sometimes used synonymously, in this document they refer to separate and distinct concepts.

It is a goal of this document to encompass existing AD 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 or other information is determined by the perspective taken and the concerns and aspects addressed by that group of models and other information sources. In this document, a *specification of an architecture viewpoint* refers to the conventions for expressing an architecture with respect to a given perspective and to a set of concerns and aspects:

A view is a way of expressing an entity of interest from a particular viewpoint.

The use of multiple views to express an architecture is a fundamental premise of this document. The need for multiple views in ADs is widely recognized. While the use of multiple views is widespread, authors differ on what views are needed, based on audience, and on appropriate methods for expressing each view. Because of the wide range of opinion, this document does not require a predefined set of architecture viewpoints and specifications of architecture viewpoints; it encourages the practice of defining or selecting architecture viewpoints appropriate to the entity of interest.

A consequence of making architecture viewpoints first-class entities is that they are among the constructs considered as AD elements.

The next few paragraphs briefly relate the historical use and evolution of the term viewpoint in systems and software

The earliest work on first-class viewpoints appears in Ross' Structured Analysis (SADT) in 1977 [44]. In requirements engineering, Nuseibeh, Kramer and Finkelstein treat viewpoints as first-class entities, with associated attributes and operations [35]. These works inspired the formulation of architecture viewpoints as specified in Clause 8. The term was also chosen to align with the ISO Reference Model of Open Distributed Processing (RM-ODP) [1], 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. See ISO/IEC 10746-1:1998, 6.2.2 [1].

A viewpoint (on a system) is 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. See ISO/IEC 10746-2:2009, 3.2.7[1] [2].

However, where this document uses *architecture view* to refer to the application of a viewpoint to a particular entity, RM-ODP [1] used the term *viewpoint specification*.

The relationship between specification of an architecture viewpoint and view suggests this metaphor for the construction of views:

architecture viewpoint: architecture view:: programming language: program[1)]

A specification of an architecture viewpoint contains the conventions (such as notations, languages and types of models) for constructing a certain kind of view; just as a programming language specification (e.g. a reference manual) provides the, as such it offers a way of looking at an entity and provides the architect with resources for modelling an entity with respect to the concerns framed by that viewpoint. That viewpoint can be applied to many entities. Each view is one such application. Similarly, a program is one application of an algorithm in an executable form.

Another metaphor to understand the difference between view and specification of an architecture viewpoint for the usage of view is:

view: viewpoint specification:: map: legend

A legend offers readers the conventions used in preparing a map (such as its scale, colour scheme and other symbology) thereby aiding readers in interpreting that map as intended. Just as every map should have a legend, every architecture view must have a specification of an architecture viewpoint specifying the conventions for interpreting the contents of the view.

Another term, *viewtype*, introduced by Clements et al. [22], establishes a categorization of viewpoints in the terms of this document. Three categories of viewpoints are described in their work: module, component and connector, and allocation viewtypes.

Within an individual AD, this document requires that each view needs to be governed by exactly one architecture viewpoint. This means each view conforms to one set of conventions (possibly including one or more specifications of model kinds, possibly presented in more than one way for diverse stakeholders). Note that this requirement does not preclude users of this document combining or composing specifications of architecture viewpoints for specific purposes (in a manner not defined by this document) as long as the requirement is met *within* an individual AD.

It is common to have each architecture view represent the *whole entity of interest* from the perspective of the concerns framed and aspects revealed by its governing architecture viewpoint. This reflects the holistic nature of architecture and architecting. For example, a performance view of a networked entity should consider both network transmission delays (in one view component) and processing times (in another view component) to produce a holistic end-to-end view of the performance of the entire entity of interest.

An AD can focus on the entity of interest at a specific point of time (for example, when it is delivered to a customer), or consider the evolution of the architecture over several time scales. Any view can be constructed from a series of view components, each representing the entity of interest at a given point of time, stage or phase. The composition of such view components within a view would describe how that entity evolves over time, while still allowing the view to deal with the whole entity of interest.

There are two common approaches to the construction of views: the synthetic approach [46] and the projective approach [19]. In the synthetic approach, the architect constructs views of the entity of interest and integrates these views within an AD, using correspondences. In the projective approach, an architect derives each view through some routine, possibly mechanical, procedure of extraction from an underlying information source or set of models. This document is designed to be usable with either of these approaches to views.

Mary Shaw writes:

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. [49].

1) To be read as: "an architecture viewpoint is to an architecture view as a programming language is to a program."

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

A.7 Correspondences

Correspondences are used to express relations within and between ADs. In particular, correspondences express relations between AD elements. Correspondences are also used to express relations between ADs.

Although many model kinds and ADLs include constructs to capture relations (such as relationships in Entity-Relation-Attribute diagrams, associations in UML), when an AD utilizes multiple viewpoints and model kinds, there might not be any available way to express relations between these diverse representations. In such cases, correspondences can be used to express these relations. The 2011 edition of this document introduced correspondences and correspondence rules to express and enforce relations between AD elements. AD elements are instances of the constructs which occur in an AD including stakeholders, concerns, perspectives, aspects, model kinds, legends, viewpoints, views and view components. In addition, AD elements include ADs themselves and instances of the constructs introduced by viewpoints and model kinds. Any of these AD elements can participate in relations expressed by correspondences. Correspondences have a number of uses. They can be used to express consistency, traceability, composition, refinement and model transformation, and dependences of any type within and between ADs. A survey of uses of model relations together with a taxonomy and classification of relation mechanisms is found in [19]. Correspondences can be used to meet the requirements of 6.9.2 for recording view consistency and inconsistencies.

In this edition, correspondence rules are generalized to *correspondence methods*. Correspondence methods capture intended relationships that are to be enforced on correspondences within and between AD elements.

The remainder of this sub-clause presents examples of correspondences and correspondence methods. The features of the correspondence mechanism, in relation to similar mechanisms in the literature, are discussed.

EXAMPLE 1 Consider two view components in an automotive system's AD: a software application view component and an electronic control unit (ECU) view component. The software application view component includes these elements: Autopilot, Dashboard, Braking, GPS, LIDAR and Sensor Fusion. The ECU view component identifies a number of ECUs, numbered 1 through 4. A correspondence, depicting the assignment of applications to ECUs, is shown in Figure A.8 as a matrix. The form of a correspondence is not specified by this document.

Applications bundledOn ECUs See: M1	
Dashboard, GPS,	ECU 1
Autopilot,	ECU 2
LIDAR, Sensor Fusion	ECU 3
Braking	ECU 4

Figure A.8 — Example of a correspondence

The example meets the requirement of 6.8.2: it has a unique name (bundledOn), identifies participating elements (the applications and ECU), and identifies an optional correspondence method (**M1**).

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

EXAMPLE 2 **M1**: every application must be bundled onto at least one ECU.

The correspondence named bundledOn in EXAMPLE 1 satisfies **M1** because all applications are assigned to at least one ECU.

Most correspondences will be expressed in terms of elements of views or view components, but this is not required. EXAMPLES 3 and 4 show other forms of correspondences.

EXAMPLE 3 **Tasks-Interactions**: For every instance of the model kind, Tasks needs to have a refinement to an instance of model kind Interactions.

This correspondence method could be satisfied by the correspondence shown in Figure A.2 where there are Users, Operators and Auditors. Each *Task* instance (a view component depicted as a triangle) is refined into an *Interaction* instance (a view component depicted with a pentagon). Alternatively, this could be recorded as a matrix, as in EXAMPLE 1.

In EXAMPLE 3 the participants in the correspondence are not elements within view components, but view components themselves. A correspondence can relate any AD elements (see 5.2.11 and 6.9.2); users of this document 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 method.

EXAMPLE 4 **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 [1]; however, there are some differences. Notable differences are:

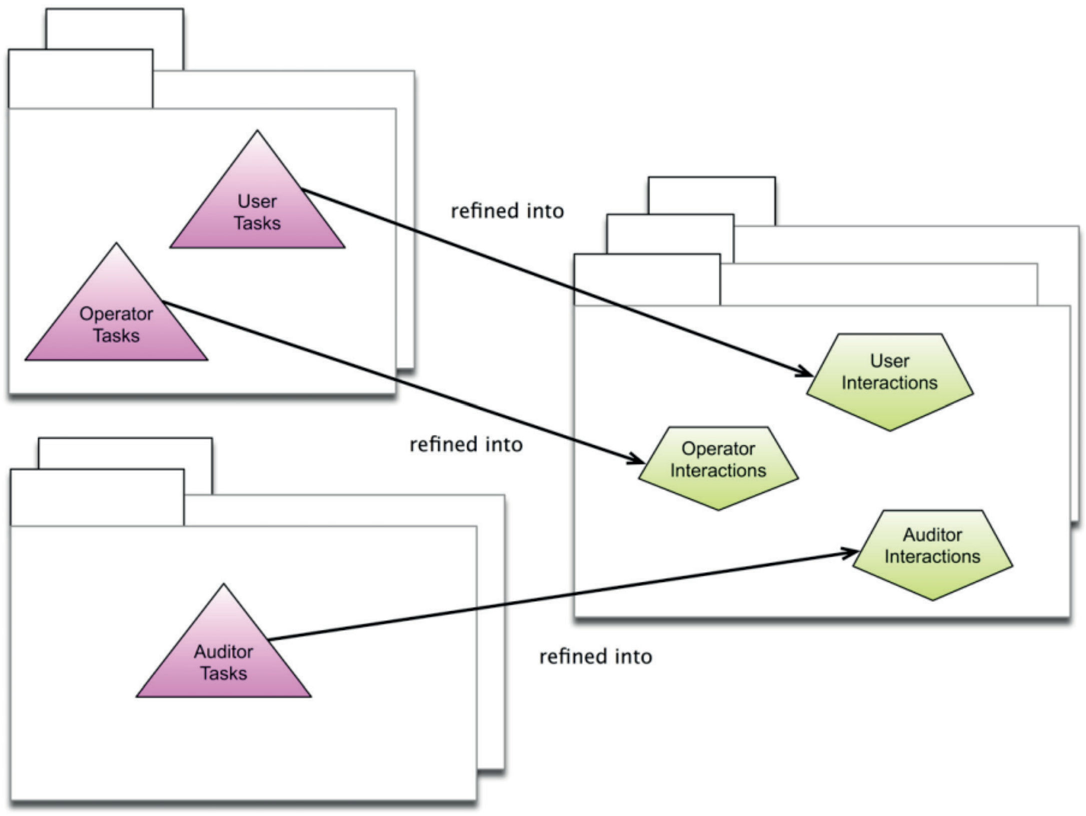

Figure A.2 — Example of a correspondence satisfying the Task-Interactions rule

(1) The term “correspondence” is used in this document rather than “view correspondence”. In RM-ODP, each view is homogeneous—a single viewpoint language is used per viewpoint specification.

In RM-ODP a viewpoint specification is what is referred to in this document as an architecture view (see A.6).This document permits heterogeneous views: each view consists of one or more view components wherein each can utilize a different modelling convention (see 6.8). It is useful to be able to state a correspondence between view components in different modelling languages, not just between views. Therefore, "view correspondence" is a special case of what is needed in this document, and that term is somewhat misleading in this more general case;

(2) RM-ODP view correspondences express binary relations where correspondences in this document express *n*-ary relations;

(3) RM-ODP view correspondences are defined on elements of view specifications whereas correspondences in this document need not refer to individual elements of models, but arbitrary AD elements; and

(4) Correspondences and correspondence methods can be used to express relations across ADs.

Mathematically, a correspondence is an n-ary relation. A correspondence method is an intentional 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, reasoning and allow efficient representation and manipulation (see [33] and references therein).

A.8 Perspectives on architecture description languages

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

Early ADLs included Rapide (Stanford) [32], Wright (CMU) [55], and Darwin (Imperial College). ADLs focused on structural concerns: large-scale system organization expressed in terms of components, connectors and configurations with varying support for framing behavioural concerns. More recently, "wide-spectrum" ADLs have been developed which support a wider range of concerns. These include Architecture Analysis and Description Language (AADL) [48], SysML [38], and ArchiMate [51]. Examples below describe some contemporary ADLs with reference to their relationship to the conceptual model defined in this document.

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, Behavioural and Informational aspects, and defines a number of basic architecture viewpoints for these. Each architecture viewpoint is defined via its own metamodel, relating that architecture viewpoint to others, and specifying the stakeholders, concerns, purpose, layers and aspects.

EXAMPLE 2 The Systems Modelling 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 document, each SysML diagram type provides a different type of view. SysML provides first-class constructs for Stakeholders, Concerns, Views and Architecture Viewpoint so that users can create new model kind in accordance with this document.

EXAMPLE 3 The Unified Architecture Framework Profile (UAFP) is a modelling language focused on representation of the enterprise and includes language extensions to UML/SysML that allow the extraction of specified and custom models from an integrated AD. The models describe the enterprise (and associated "resources") from a set of stakeholders' concerns through a set of predefined architecture viewpoints, specifications of views and associated views [51]. Given that in architecture practice multiple models would be created separately from a number of specifications of architecture viewpoints and in multiple versions, to be able to maintain cross-model consistency the UAFP allows the definition of constraints to express this. UAFP is defined as a UML 2 / SysML v1.4 profile, corresponding to the actual domain concepts (UAF concepts and relationships) separately defined in the UAF Domain Meta Model (DMM). This separation allows UAF to be used in conjunction with more than one ADF (such as DoDAF or NAF, for example).

An ADL frames a particular set of concerns for an audience of stakeholders, by defining one or more model kinds together with any other methods or tools. Similar to an ADF or an architecture viewpoint, an ADL is a reusable resource—it is not limited in use to an individual entity or AD.

Annex B
(informative)

Guide to specification of architecture viewpoints

B.1 Introduction

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

B.2 A template for documenting specification of architecture viewpoints

B.2.1 Template overview

A template for the specification of architecture viewpoints is presented. An architecture viewpoint that is documented in this form will meet the requirements of Clause 8.1.

The template comprises of a set of information items (B.2.2 through B.2.13). Each information item is identified by a name (B.2.X **Item name**) followed by a brief description of its intended content, and guidance for developing that content. In some cases, additional information items are nested within top-level information items.

B.2.2 Architecture viewpoint name

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

B.2.3 Architecture viewpoint overview

An abstract or brief overview of the architecture viewpoint and the related architecture viewpoint features.

B.2.4 Concerns

A listing of the architecture-related concerns to be framed by this architecture viewpoint per 8.1 item a). This helps decide whether the related architecture viewpoint will be useful for modelling a particular entity of interest.

It can be useful to document the kinds of issues an architecture viewpoint is *not appropriate for* to avoid misuse. Articulating these issues can be a good antidote for certain overly used viewpoints and model kinds.

B.2.5 Stakeholder perspectives

A listing of any stakeholder perspectives associated with this architecture viewpoint (per 8.1 item a)).

B.2.6 Architecture aspects

A listing of the architecture aspects refining the above concerns (per 8.1 item d)).

NOTE The identification of concerns, stakeholder perspectives and aspects are intended to assist architects and other stakeholders in determining the utility of this viewpoint for their entity of interest.

B.2.7 Typical stakeholders

A listing of the stakeholders expected to be users or audiences for views prepared using this architecture viewpoint (per 8.1 item c)).

NOTE When an architecture viewpoint is selected for use and applied in an AD, the AD needs to document the association of actual stakeholders with concerns framed by this viewpoint and related specification (per 6.3)

B.2.8 Correspondence methods

A listing of any correspondence methods defined by this viewpoint or its model kinds (per 8.1, 8.2 and 6.9.3).

These methods can be applied across view components, across views within an AD or across ADs.

B.2.9 Specification of model kinds

B.2.9.1 Introduction

Identify each model kind as a part of the architecture viewpoint (per 8.1 item f)).

For each model kind used, describe its conventions, language or modelling techniques. These are key modelling resources which the specification of the architecture viewpoint makes available that establish the vocabularies for constructing the architecture views.

This document does not require one style for documenting specifications of model kinds. A specification of a model kind can be documented in various ways, including:

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

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

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

(4) by some combination of these, or other means.

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

B.2.9.2 Metamodel related to the specification of a model kind

A metamodel presents one or more constructs which are the AD elements that comprise the vocabulary of the model kind and its specification. There are various ways of representing metamodels. The metamodel will 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?

Within an AD, instances of entities, attributes, relationships and constraints are AD elements in the sense of 6.1.

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

B.2.9.3 Templates of specifications of model kinds

Provide a template or form specifying the format or expected content of view components governed by this model kind specification.

Each such template, form, or their parts, can have a legend to be used when this model kind is used within an AD.

B.2.9.4 Language related to the specification of a model kind

Identify an existing notation or modelling language or define one that can be used when applying this model kind in an AD. Describe its syntax, semantics, and tool support, as needed.

B.2.10 View methods

Define methods available on views. (see 5.2.9).

B.2.11 Examples

This section provides examples for the reader.

B.2.12 Notes

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

B.2.13 Sources

Identify the sources for this specification, if any, including author, history, literature references and prior art (per 8.1 item f)).

B.3 Annotated guide related to the specification of an architecture viewpoint

The following represent some resources for well-documented specifications of architecture viewpoints. Not all of these are documented in accordance with the requirements of this document but could be used in an AD or included in an ADF specification in a conforming manner.

— Callo-Arias, America, and Avgeriou, "Defining execution viewpoints for a large and complex software-intensive system" [21].

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

— Clements, et al., Documenting Software Architectures: views and beyond [22].

  The source above provides extensive resources for defining 3 categories of specifications of architecture 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 [26].

  The source above defines a process for software architects, using the IEEE 1471:2000 model as a foundation. Provides a template for specifications of architecture viewpoints and architecture viewpoint catalogues including: Requirements, Functional, Deployment, Validation, Application, Infrastructure, Systems Management, Availability, Performance, Security; and the "work products" (i.e. model kind specifications) for each.

— Architecture viewpoint Repository [54]

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

— Kruchten, "The '4+1' view model of software architecture" [30]

The source above specifies architecture viewpoints for Logical, Development, Process and Physical views. The resulting views are integrated via Scenarios.

— Rozanski and Woods, Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives [45]

The source above defines a catalogue of architecture viewpoints: Functional, Information, Concurrency, Development, Deployment and Operational viewpoints and *perspectives* (See 5.2.4): Security, Performance and Scalability, Availability and Resilience, and Evolution perspectives for software-intensive systems.

NOTE Rozanski and Woods' perspectives do not fit the definition in this document.

Annex C
(informative)

Relationship to other standards

C.1 Introduction

This annex illustrates how ADs created in accordance with this document can meet the requirements of other standards. This document specifies the core terminology and the concepts usable for AD. Other standards can use this document as a normative reference in order to define one or more architecture viewpoints, concerns, aspects and perspectives to be used in ADs within projects and enterprises.

C.2 Use with ISO/IEC IEEE 42020:2019

ISO/IEC IEEE 42020:2019 [17] specifies requirements, recommendations and permissions for architecture processes suitable for the enterprise, organization or project. In this reference document, AD is mainly addressed by a process called "architecture elaboration," explaining how to describe or document an architecture in a sufficiently complete and correct manner for the intended uses of the architecture. Nevertheless, another ISO/IEC IEEE 42020 process, called "architecture conceptualization," could be undertaken to serve as the basis for an AD in order to characterize the problem space and determine suitable solutions that address stakeholder concerns, achieve architecture objectives and meet relevant requirements. Both architecture conceptualization and architecture elaboration are specified in ISO/IEC IEEE 42020 through an extensive set of activities, tasks, outcomes and work products. Recognizing that particular projects or organizations need not use all of the processes specified by this document, full conformance and tailored conformance are defined to accommodate a flexible implementation approach to claim conformance to ISO/IEC IEEE 42020.

C.3 Use with ISO/IEC IEEE 12207:2017

ISO/IEC IEEE 12207:2017 [4] defines one process specifically pertaining to architecture: architecture definition (see ISO/IEC IEEE 12207:2017, 6.4.4). The concept of architecture in this document is consistent with the architectural design processes of ISO/IEC IEEE 12207. However, ISO/IEC IEEE 12207 places requirements on an AD in addition to those of this document. Specifically, an architecture definition needs to include an identification of the architecture entities included in the system and an allocation of key stakeholder concerns and critical system requirements to those items.

As observed in the Note accompanying ISO/IEC IEEE 12207:2017 Clause 6.4.4.3c2, architecture is not necessarily concerned with all requirements, but rather only with those that drive the architecture, hence the focus of the architecture definition process on the critical system requirements.

The expected use of an AD can include other ISO/IEC IEEE 12207 processes. In particular, an AD is used to communicate the architecture to the Design Definition process and can be used to facilitate the communications between the acquirer and the developer roles in other system life cycle processes.

An AD can conform to this document and to ISO/IEC IEEE 12207.

C.4 Use with ISO/IEC IEEE 15288:2015

ISO/IEC IEEE 15288:2015 [5] defines one process specifically pertaining to architecture: architecture definition. The concept of architecture in this document is consistent with the architecture definition process of ISO/IEC IEEE 15288. However, ISO/IEC IEEE 15288 places requirements on an AD in addition to those herein. Specifically, an architecture definition needs to include an identification of the architecture entities included in the system and an allocation of key stakeholder concerns and critical system requirements to those items. These can be achieved in various ways, such as by creating decomposition and allocation architecture viewpoints, or through use of correspondences.

As observed in the Note accompanying ISO/IEC IEEE 15288:2015 Clause 6.4.4.3c2, architecture is not necessarily concerned with all requirements, but rather only with those that drive the architecture, hence the focus of the architecture definition process on the critical system requirements.

The expected uses of an AD can include other ISO/IEC IEEE 15288 processes. In particular, an AD is used to communicate the system architecture to the design definition process and can be used to facilitate the communications between the acquirer and the developer roles in other system life cycle processes.

C.5 Use with Open Distributed Processing standards

C.5.1 General

The Reference Model of Open Distributed Processing (RM-ODP) [1] defines an ADF 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 [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 AD conforming to this document and using ISO/IEC 10746-3 [3] would include the viewpoints defined by ISO/IEC 10746-3 [3] and views to implement these viewpoints. A conforming AD need not be limited to the five predefined viewpoints of ISO/IEC 10746-3 [3]; the AD can include additional viewpoints and views, as needed.

Elements of that specification specific to ADs (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 [3].

NOTE ISO/IEC 19793 [11] defines a UML profile for the specification of open distributed processing systems using these viewpoints.

C.5.2 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 behaviour 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 [7].

C.5.3 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.5.4 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.5.5 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.5.6 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.

Annex D
(informative)

Uses of architecture descriptions

D.1 Introduction

ADs can be used in a variety of settings and life cycle models. This annex illustrates a few uses of ADs throughout the life cycle of their entities of interest.

D.2 Uses of architecture descriptions

Uses for ADs include, but are not limited to:

a) as basis for entity design and development activities;

b) as basis to analyse and evaluate alternative implementations of an architecture;

c) as development and maintenance documentation;

d) to support informed technical, investment or other strategic decisions, to reduce or mitigate attendant risk;

e) documenting essential features of an entity of interest, such as:

  1) intended use and environment;

  2) principles, assumptions and constraints to guide future change;

  3) points of flexibility or limitations of the entity with respect to future changes;

  4) architecture decisions, their rationales and implications;

  5) recording its architecture styles;

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

g) specifying a group or family of entities sharing common features (such as could be codified as a reference architecture, reference model or product line architecture);

h) communicating among parties involved in the development, production, deployment, operation and maintenance of an entity of interest;

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

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

k) documenting the characteristics, features and design of an entity for potential clients, acquirers, owners, operators and integrators;

l) planning for transition from a legacy architecture to a new architecture;

m) as guide to operational and infrastructure support and configuration management;

n) as support to life cycle planning, scheduling and budgeting activities;

o) establishing criteria for certifying implementations for compliance with an architecture;

p) as compliance mechanism to external, program-, project- or organization-specific policies (for example, legislation, or overarching architecture principles adopted by governance);

q) as basis for review, analysis, and evaluation of the entity throughout its life cycle;

r) as basis to analyse and evaluate alternative architectures;

s) sharing lessons learned and reusing architectural knowledge through definition of architecture viewpoints, architecture patterns and architecture styles;

t) training and education of stakeholders and other parties on best practices in architecting and evolution.

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

Annex E
(informative)

Architecture and architecture description life cycles

E.1 Introduction

ADs can be used in a variety of settings and life cycle models. This annex illustrates a few concepts pertaining to architecture life cycles and AD life cycles.

E.2 Architecting in the life cycle

Architecting contributes to the development, operation and maintenance of an entity from its initial conception through its operation, refurbishment or final retirement from use, and eventual disposal.

Architecting can take place throughout the life cycle of the entity, not necessarily only within the early stage of its life. Therefore, an entity's architecture potentially influences processes throughout the entity's life cycle. While it is expected that the architecture of an entity remains stable over a longer period, from time to time the architecture of the entity (or a part of the entity) can incrementally evolve. Less frequently an entity can go through a significant transformation, with necessary re-architecting being performed.

ADs are the work products resulting from the execution of architecting efforts to transmit, share and record the architecture of the entity and associated decisions across the life cycle. The original and subsequent (incremental or extensive) architecting efforts communicate and document the outcome from using ADs. The status and history of solution alternatives and versions are expressed as architecture descriptions.

NOTE 1 ISO/IEC IEEE 15288 [5] describes the Architecture Definition process. ISO/IEC IEEE 42020 [17] complements this definition toward a complete set of architecture processes in support of architecting efforts.

The developed models and other information items, as well as the ADs themselves, each have their own life cycle. During architecture development the need for the architecture is identified, the architecture concepts are defined, its details are elaborated, trade-offs and possible realizations are analysed, evaluated and approved, and released for use, then utilized. Later the AD can be updated, and new versions released. Ideally the stages of utilization are long and stable, because of the efficiencies achieved by using a shared architecture in the life cycle of multiple entities. Therefore, the architecture of the entity(ies) of interest has its own life cycle.

Conversely, the life cycle of any entity is fundamentally influenced by the life cycle of its architecture. For example, investment decisions to create a project to develop a system of systems based upon the existing systems in a service ecosystem heavily depend on the existence of a stable, trusted, shared and agreed-upon architecture of its integrating infrastructure. When planning the life cycle stages of an entity, the plan will take into account the predicted life cycle stages of the chosen architecture.

The process of architecting involves thinking about the entity of interest from multiple perspectives, and these perspectives correspond to life cycle phase activities. For example, the architect will consider the entity from the business perspective, including the entity's identity, as well as the concept of the entity (such as its role in the market, the capabilities, the relationships to the business environment, and the policies and principles that need to govern the solution, etc.). The architect will also consider the high-level requirements that the entity needs to satisfy (functional and non-functional requirements), and from the business analyst's perspective translate this to a specification. The architect will also translate these requirements to an architectural solution, which is a preliminary design of the entity of interest and prepare the solution for evaluation.

The phases (identity definition, concept development, requirements definition, preliminary design, as defined in ISO 15704 [10]) are activity *types*, that are normally iterated within the architecture development stage (as defined in ISO/IEC IEEE 15288 [5]) until the relevant stakeholder questions can all be addressed in a satisfactory manner. The concept of phase is therefore referring to a type of activity, whereupon instances of that activity are repeated as necessary during a stage and typically across multiple future stages of the entity's life.

The need for iterations depends on multiple factors, such as the innovation content of the architecture development, skills and maturity of the organization, the availability of reference architectures or patterns that can be reused, and the methodology used.

It is typical for architecture development to be performed in the organizational setting of a project or program, in which case the program and its projects need to be architected as well, as entities in their own right. The same is true in the setting of systems of systems and of supporting systems (as defined in ISO/IEC IEEE 15288 [5] and ISO/IEC IEEE 24748 [12]), each of these systems are implemented as entities that would need to be architected if not pre-existing.

In the process of architecture development architects must engage in two-way communication with stakeholders – both for getting informed and to inform. Work products incorporate various information items (more and more often various models), which then form the basis of creating ADs that are in a form that stakeholders and architects mutually understand and that satisfies the goal of architecting as defined by architecture governance and management.

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

NOTE 2 Annex C demonstrates how this document can be used when applying the life cycle processes of ISO/IEC IEEE 12207 [4] and ISO/IEC IEEE 15288 [5]. ISO/IEC IEEE 42020 [17] specifies a set of processes for architecting, and for architecture governance, management, and enablement within the context of a life cycle.

Annex F
(informative)

Architecture description frameworks

F.1 Introduction

This Annex provides examples of ADFs and the way they use the concepts defined in 5.4.2 (Figure 6). Each of them complies with the requirements of 7.1 to some degree.

F.2 Evolution of ADFs

In systems and software engineering, the notion of *ADF* dates back to the 1970s [23][56]. The motivation for definition of the term (3.6) and its specification (7.1) in this document is to provide a means of evolving existing and future ADFs in a uniform manner to promote sharing of information about entities, architectures and techniques for AD, better uniformity to enable improved understanding of the architectures being described, and interoperability between architecture communities who use different conceptual foundations. The uniform definition of architecture viewpoints and coordinated collections of associated specifications can promote reuse of tools and techniques by the communities using these frameworks.

The specification of ADF is intended to establish the relationships between an ADF and other concepts in this document (illustrated in Figure 2 and Figure 6). ADFs often include additional content, prescriptions and relationships, such as process guidance, life cycle connections, and documentation formats, not defined by this document. These are potential future areas of standardization.

F.3 ADF concepts

F.3.1 ADF domains

Various architecture frameworks have been in existence over a few decades.

EXAMPLE GERA Framework provided by the Generalized Enterprise Reference Architecture and Methodology (GERAM) in ISO 15704 [10] Industrial Internet Architecture Framework [29], Kruchten's "4+1" view model [31], NATO Architecture Framework (NAF) [34], US Department of Defence Architecture Framework (DoDAF) [25], UK Ministry of Defence Architecture Framework (MODAF) [53], OASIS - Reference Architecture Foundation for Service Oriented Architecture [36], Reference Model for Open Distributed Processing (RM-ODP) [1], The Open Group's Architecture Framework (TOGAF) [52], OMG's Unified Architecture Framework [39], and Zachman's information systems architecture framework [56].

NOTE GERAM provides an ADF for an enterprise architecture modelling activity and is therefore a modelling framework covering the whole of life for an entity of interest.

Most of them are dedicated to creating ADs in different domains like:

— Zachman [56] and TOGAF [52] for information systems

— Kruchten's "4+1" view model [31] for software

— NAF [34], DoDAF [25] and UAF [39] for enterprises and systems of systems.

— GERAM [10] for enterprise modelling of socio-technical systems of systems, e.g. industrial automation, transformation, critical infrastructure, etc.

F.3.2 Identification of stakeholders and definition of their perspectives

Some ADFs like Zachman [56] clearly identify typical stakeholders like: Planner, Owner, Designer, Builder, Implementer and User.

Others identify stages in the AD related to a class of typical stakeholders like: The Open Group's Architecture Framework (TOGAF) [52], with Business Architecture, Information System Architecture and Technology Architecture.

Others identify stakeholder perspectives like:

- UAF [39] with "domains": Metadata, Strategic, Operational, Services, Personnel, Resources, Security, Projects, Standards, and Actual Resources
- ArchiMate [51] with "layers": Strategy, Business, Application, Technology, Physical, Implementation, and Migration
- NAF [34] with "subjects of concerns": concept specification, service specification, logical specification, physical resource specification, and architecture meta-data
- DoDAF [25] with "viewpoints": Capability, Operational, Services, Systems, Standards, and Data and Information
- ISO 15704/GERAM [10] with "levels of abstraction" or "life cycle phases": Identity, concept, requirements, preliminary design, detailed design, implementation, operation and decommissioning.

F.3.3 Definition of architecture aspects

ADFs organize the properties and features of architecture models and views in various ways, such as:

- Zachman [56] with "interrogatives": What, How, Where, Who, Where, Why
- NAF [34] with "aspects of concerns": Taxonomy, Structure, Connectivity, Processes, States, Sequences, Information, Constraints and Roadmap
- UAF [39] with "model kinds": Taxonomy, Structure, Connectivity, Processes, States, Interaction Scenarios, Information, Parameters, Constraints, Roadmap and Traceability

ADFs attempt to generalize or harmonize multiple particular ADs to offer guidance for creation of new particular cases by specifying generic stakeholder perspectives and generic architecture aspects (see Zachman [56] and others).

F.3.4 Specification of architecture viewpoint

An ADF generally specifies architecture viewpoints providing specifications of model kinds to frame typical concerns and to govern the architecture views and ease understanding of the legends.

EXAMPLE Reference Model for Open distributed (RM-ODP) [2] defines the following concepts: enterprise, information, computational, engineering and technology viewpoint specifications.

ADFs often utilize one or more structural categories to represent distribution of architecture viewpoints in a two-dimensional grid or matrix. See Figure F.1 and Figure F.2 for example. There are some ADFs, such as GERAM [10] and SABSA [47], which use three or more categories to convey the complexity of describing architectures.

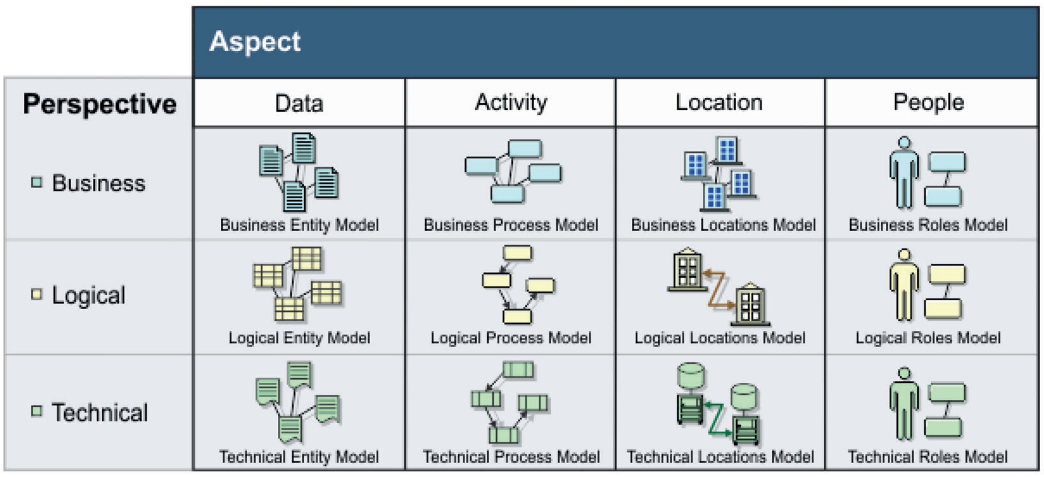

Figure F.1 — Unified Architecture Method (UAM)

Source: http://www.unified-am.com.

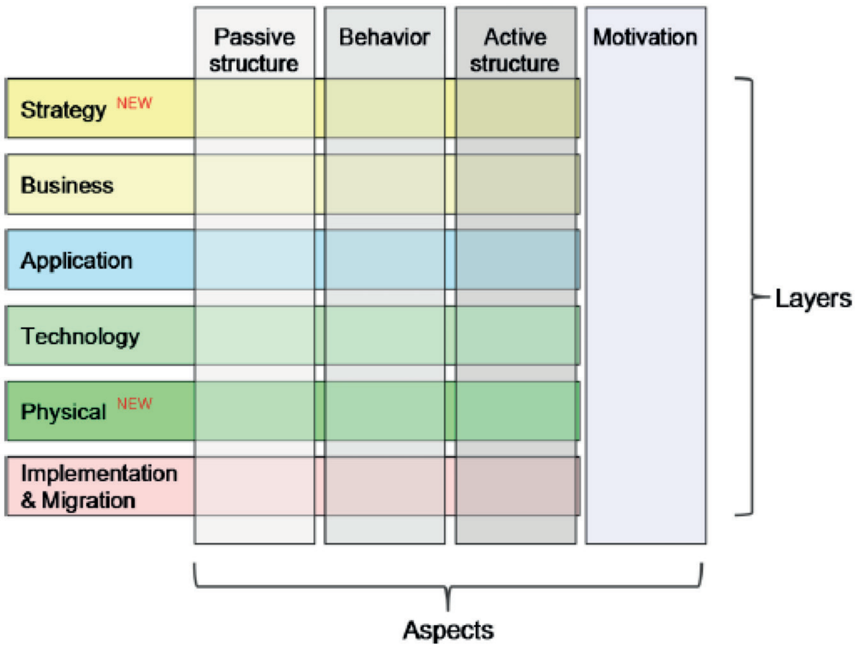

Figure F.2 — ArchiMate Framework

[Source: An Introduction to the ArchiMate® 3.0 Specification, Open Group, June 2016]

F.3.5 Specification of formalisms and languages

Generally the ADFs define their formalism for constructing models with a metamodel: like DoDAF's [25] DoDAF Metamodel (DM2), UAF's [39] Domain Metamodel (DMM), and NAF's [34] metamodel (MM).

Some of them also provide languages, which could be used to implement the formalism, like:

- UAF [39] with its UAF Profile [40]
- ArchiMate modelling language [51]

- ISO 19440 Ed2 Constructs for enterprise modelling [8]

F.4 Compliance with ADF requirements

Currently none of the ADF fully comply with all the requirements itemized in 7.1. Nevertheless, each of the items is addressed by several of the ADF and all these items are considered as necessary in an AD. Consequently, the provisions in 7.1 provide areas for improvement for these frameworks.

Table F.1 and Table F.2 show an interpretation of how some of the published ADFs comply with 7.1 requirements.

NOTE In Table F.1 and Table F.2 "Partial" means that the requirement is fulfilled to some degree but is not expressed with explicit wording. "Yes" is used when the requirement is met. "No" states that the requirement is not addressed.

Table F.1 — ADF requirements compliance (1/2)

	GERAM	RM-ODP	Zachman	TOGAF
Information identifying the ADF	Implementation dependent	Implementation dependent	Yes	Yes
Stakeholder identification	Organization viewpoint	No	Yes	No
Concern identification	Covered by identification, concept and requirements life cycle phases	Yes (in Viewpoints)	Partial	Partial
Architecture aspects	Aspect-oriented views	No	Called "interrogatives"	No
Stakeholder perspectives	Covered by identification, concept and requirements life cycle phases	No	Yes	Called "phases"
Formalism	GERAM Metamodel	UML Profile	No	No
Architecture Viewpoint	Aspect-oriented viewpoints	Viewpoints	Partial	No
Model kinds	Criteria explained	One per viewpoint	Partial	Partial
Legends and correspondence methods	Partial	Formalized	Partial	Partial
Framework methods	GERAM	No	Partial	TOGAF/ADM

Table F.2 — ADF requirements compliance (2/2)

	UAF	NAF	DoDAF	ArchiMate
Information identifying the ADF	Yes	Yes	Yes	Yes
Stakeholder identification	Yes (in view specifications)	No	No	No
Concern identification	Yes (in view specifications)	Partial	Partial	Yes (in Viewpoints)
Architecture aspects	Called “Model kinds”	Called “Aspects of concerns”	No	Called “Aspects”
Stakeholder perspectives	Called “domains”	Called “subjects of concerns”	Called “viewpoints”	Called “Layers”
Formalism	DMM metamodel and a Profile	NAF metamodels	DM2 metamodel	ArchiMate Specification
Architecture Viewpoint	Called “view specifications” within a Grid	Called “viewpoints” within a Grid	Called “Models”	Grid
Model kinds	UAF Profile’s view specifications	Partial	Partial	ArchiMate Specification
Legends and correspondence methods	Partial	Partial	Partial	Partial
Framework methods	No	NAF Chapter 2	6 Step Approach	No

Bibliography

Standards:

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

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

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

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

[5] ISO/IEC IEEE 15288:2015, *Systems and software engineering — System life cycle processes*

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

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

[8] ISO 19440:2020, *Enterprise modelling and architecture — Constructs for enterprise modelling*

[9] ISO/IEC 15504-6:2013, *Information technology — Process assessment — Part 6: An exemplar system life cycle process assessment model*

[10] ISO 15704:2019, *Enterprise modelling and architecture — Requirements for enterprise-referencing architectures and methodologies*

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

[12] ISO/IEC IEEE 24748-1:2018, *Systems and software engineering — Life cycle management — Part 1: Guidelines for life cycle management*

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

[14] ISO/IEC 25012:2008, *Software engineering — Software product Quality Requirements and Evaluation (SQuaRE) — Data quality model*

[15] ISO/IEC 25024:2015, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Measurement of data quality*

[16] ISO/IEC 33001:2015, *Information technology — Process assessment — Concepts and terminology*

[17] ISO/IEC IEEE 42020:2019, *Software, systems and enterprise — Architecture processes*

[18] ISO/IEC IEEE 42030:2019, *Software, systems and enterprise — Architecture evaluation framework*

Other references:

[19] Atkinson C., Gerbig R., Tunjic C., "A multi-level modeling environment for SUM-based software engineering", Proceedings of the 1st Workshop on View-Based, Aspect-Oriented and Orthographic Software Modeling (VAO '13), ACM Press, 2013

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

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

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

[23] Darnton G., Giacoletto S., Information in the Enterprise. Digital Press, Burlington, MA, 1992

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

[25] DoD Architecture Framework, version 2.02, https://dodcio.defense.gov/library/dod-architecture-framework/

[26] Eeles P., Cripps P., The Process of Software Architecting. Addison Wesley, 2010

[27] Heidel R., The Reference Architecture Model RAMI 4.0 and the Industrie 4.0 component, 2019

[28] Hilliard R., "Viewpoint modeling", First ICSE Workshop on Describing Software Architecture with UML, May 2001

[29] Industrial Internet Architecture Framework. https://www.iiconsortium.org/

[30] Kiczales G., Lamping J., Menhdhekar A., Maeda C., Lopes C., Loingtier J.M. et al. , Aspect-oriented programming. In Aksٖit, M., Matsuoka, S., eds.: Proceedings European Conference on Object-Oriented Programming. Volume 1241. Springer-Verlag, Berlin, Heidelberg, and New York (1997) 220–242

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

[32] Luckham D.C., Kenney J.J., Augustin L.M., Vera J., Bryan D., Mann W., Specification and analysis of system architecture using Rapide. IEEE Trans. Softw. Eng. 1995 April, **21** (4) pp. 336–355

[33] Muskens J., Bril R.J., Chaudron M.R.V. "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

[34] Architecture Framework N.A.T.O., version 4, https://www.nato.int/cps/en/natohq/topics_157575.htm

[35] Nuseibeh B., Kramer J., Finkelstein A., A framework for expressing the relationships between multiple views in requirements specification. IEEE Trans. Softw. Eng. 1994, **20** (10) pp. 760–773

[36] OASIS, Reference Architecture Foundation for Service Oriented Architecture Version 1.0

[37] OMG formal/13-11-03, Business Process Model and Notation (BPMN™), version 2.0.1, November 2013

[38] OMG formal/2008-11-01, Systems Modeling Language, version 1.5, May 2017

[39] Object Management Group. Unified Architecture Framework 1.0, formal/17-12-01, https://www.omg.org/uaf/

[40] OMG formal/2017-12-01, Unified Architecture Framework Profile (UAFP), Version 1.0, November 2017

[41] OMG formal/12-05-06 and formal/12-05-07, Unified Modeling Language (UML®), version 2.4.1, May 2012

[42] Object Management Group. Unified Profile for DoDAF/MODAF, https://www.omg.org/updm/

[43] 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

[44] Ross D.T., Structured Analysis (SA): a language for communicating ideas. IEEE Trans. Softw. Eng. 1977, **SE-3** (1) pp. 16–34

[45] Rozanski N., Woods E., Software Systems Architecture: Working with Stakeholders Using Viewpoints and Perspectives. Addison-Wesley, 2005

[46] Sabetzadeh M., Finkelstein A., Goedicke M. "Viewpoints", P. Laplante (ed.), *Encyclopedia of Software Engineering*, Taylor and Francis, 2010

[47] SABSA Institute Resource. The SABSA White Paper, W100, Published 2009, SHA256 676403bbb6d3e0336a6de898ac5ea7ad7e677caf6385934c3f6a6dd5f0dc68bf

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

[49] Shaw M., Prospects for an engineering discipline of software. IEEE Softw. 1990 November

[50] 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)

[51] The Open Group. ArchiMate 3.0.1 Specification, August 2017, https://publications.opengroup.org/archimate-library/archimate-standards

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

[53] UK Ministry of Defence Architecture Framework. https://www.gov.uk/guidance/mod-architecture-framework

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

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

[56] Zachman J.A., A Framework for Information Systems Architecture. IBM Syst. J. 1987, **26** (3)