Annex A
(normative)

Tailoring process

NOTE This annex is an adaptation of annex A in ISO/IEC/IEEE 15288.

A.1 Introduction

This Annex provides requirements for the tailoring of this document.

NOTE 1 Tailoring is not a requirement for conformance to the standard. In fact, tailoring is not permitted if a claim of "full conformance" is to be made. If a claim of "tailored conformance" is made, then this process is applied to perform the tailoring.

NOTE 2 Additional guidance for tailoring can be found in the ISO/IEC/IEEE TR 24748 guides, on the application of life cycle processes.

A.2 Process usage

The architecture processes defined in this document can be used by any organization when acquiring, using, creating, or supplying a system, as well as when operating, evolving, or transforming an enterprise. They can be applied at any level in an enterprise and at any stage in the life cycle of the architecture or associated systems.

The functions these processes perform are defined in terms of specific purposes, outcomes and the set of activities and tasks that constitute the process.

A.2.1 Process ordering

Each architecture process in Figure 1 can be invoked, as required, at any time throughout the life cycle. The order that the processes are presented in this document does not imply any prescriptive order in their use. However, sequential relationships are introduced by the definition of a life cycle model. The detailed purpose and timing of use of these processes throughout the life cycle are influenced by multiple factors, including social, trading, organizational and technical considerations, each of which can vary during the life of an architecture. An individual architecture life cycle is thus a complex aggregation of processes that will normally possess concurrent, iterative, recursive and time dependent characteristics. Concurrent use of processes can exist within an organization (e.g. when the current architecture is being implemented at the same time that a future architecture is being formulated for the same system), and between organizations (e.g. when architecture entities are developed at the same time under different project responsibility).

A.2.2 Process iteration

When the application of the same process or set of processes is repeated on the same level of an architecture structure, the application is referred to as iterative. The iterative use of processes is important for the progressive refinement of process outputs, e.g. the interaction between successive architecture evaluation efforts can incrementally build confidence in the suitability of the architecture. Iteration is not only appropriate but also expected. New information is created by the application of a process or set of processes. Typically this information takes the form of questions with respect to architecture objectives, stakeholder needs, analyzed risks or opportunities. Such questions should be resolved before completing the activities of a process or set of processes.

A.2.3 Process recursion

The recursive use of processes, i.e. the repeated application of the same process or set of processes applied to successive levels of architecture entities in an architecture's structure, is a key aspect of the application of this document. The outputs of processes at any level, whether information, artifacts or services, are inputs to the processes used at the level below (e.g. during system realization) or level above (e.g. during enterprise transformation). The outcomes from one application are used as inputs to the next lower (or higher) architecture in the architecture structure to arrive at a more detailed or mature set of outcomes. Such an approach adds value to successive architectures in the architecture structure.

A.2.4 Life cycle stages

The changing nature of the influences on the architecture (e.g. operational environment changes, new opportunities for architecture entity implementation, modified structure and responsibilities in organizations) requires continual review of the selection and timing of process use. Process use in the life cycle can be dynamic, responding to the many external influences on the architecture. The life cycle approach also allows for incorporating the changes in the next stage. The life cycle stages assist the planning, execution and management of architecture processes in the face of this complexity in life cycles by providing comprehensible and recognizable high-level purpose and structure. The set of processes within a life cycle stage are applied with the common goal of satisfying the exit criteria for that stage and/or the entry criteria of the formal progress reviews within that stage.

The discussion in this section on iterative and recursive use of the architecture processes is not meant to imply any specific hierarchical, vertical, or horizontal structure for the architecture, system-of- interest, enabling system, organization, or project.

A.2.5 Process instantiation

Where justified by product quality risks, detailed descriptions of process instances in the context of the specific product may also be created. Instantiation of processes involves identifying specific success criteria for a process instance, derived from the product requirements, and identifying the specific activities and tasks needed to achieve the success criteria, derived from the activities and tasks identified in this document. Creating detailed descriptions of process instances enables better management of product quality risks by establishing the link between the process and the specific product requirements.

Further elaboration of these concepts can be found in the ISO/IEC/IEEE TR 24748 guides, on the application of architecture processes.

A.2.6 Process reference model

Annex C in ISO/IEC/IEEE 15288 defines a Process Reference Model (PRM) at a level of abstraction higher than that of the detailed requirements contained in the main text of this document. The PRM is applicable to an organization that is assessing its processes in order to determine the capability of these processes. The purpose and outcomes are a statement of the goals of the performance of each process. This statement of goals permits assessment of the effectiveness of the processes in ways other than simple conformity evaluation.

NOTE ISO/IEC 33004 can used as guidance for specification of the PRM, consideration of process assessment and associated maturity model.

A.3 Tailoring process steps

A.3.1 Purpose

The purpose of the Tailoring process is to adapt the processes of this document to satisfy particular circumstances or factors that:

a) Surround an organization that is employing this document in an agreement;

b) Influence a project that is required to meet an agreement in which this document is referenced;

c) Reflect the needs of an organization in order to supply products or services.

A.3.2 Outcomes

As a result of the successful implementation of the Tailoring process:

a) Modified or new life cycle processes are defined to achieve the purposes and outcomes of a life cycle model.

b) Justification for tailoring is provided in support of ensuring a successful outcome to the architecture processes.

A.3.3 Activities and tasks

If this document is tailored, then the organization or project shall implement the following tasks in accordance with applicable policies and procedures with respect to the Tailoring process, as required.

a) Identify and record the circumstances that influence tailoring. These influences include, but are not limited to:

   1) instability of, and variety in, operational environments;
   2) risks, commercial or performance, to the concern of interested parties;
   3) novelty, size and complexity;
   4) starting date and duration of utilization;
   5) integrity issues such as safety, security, privacy, usability, availability;
   6) emerging technology opportunities;
   7) profile of budget and organizational resources available;
   8) non-availability of the services of enabling products, services, or other enabling item;
   9) roles, responsibilities, accountabilities and authorities in the overall life cycle of the system;

10) the need to conform to other standards.

b) In the case of properties critical to the architecture entity, take due account of the life cycle structures recommended or mandated by standards relevant to the dimension of the criticality.

c) Obtain input from parties affected by the tailoring decisions. This includes, but may not be limited to:

1) the stakeholders for the architecture entity(ies);

2) the interested parties to an agreement made by the organization;

3) the contributing organizational functions.

d) Make tailoring decisions in accordance with the decision management process to achieve the purposes and outcomes of the selected life cycle model.

NOTE 1 Organizations establish standard life cycle models as a part of the Life Cycle Model Management process. It is sometimes appropriate for an organization to tailor processes of this document in order to achieve the purposes and outcomes of the stages of a life cycle model to be established.

NOTE 2 Projects select an organizationally-established life cycle model for the project as a part of the Project Planning process. It is sometimes appropriate to tailor organizationally adopted processes to achieve the purposes and outcomes of the stages of the selected life cycle model.

NOTE 3 In cases where projects are directly applying this document, it is sometimes appropriate to tailor processes of this document in order to achieve the purposes and outcomes of the stages of a suitable life cycle model.

e) Record the rationale for the tailoring in terms of the reason for tailoring, the architecture process impacted, and the nature and extent of such impact.

NOTE 4 The nature and extent of such impact shall be captured in terms of the change, restriction or removal of architecture processes. It may be expressed in terms of processes, outcomes, activities, and work products.

f) Select the life cycle processes that require tailoring and add modify or delete relevant outcomes, activities, or tasks.

NOTE 5 Irrespective of tailoring, organizations and projects are always permitted to implement processes that achieve additional outcomes or implement additional activities and tasks beyond those required for conformance to this document.

NOTE 6 An organization or project sometimes encounter a situation where there is the desire to modify a provision of this document. Modification is to be avoided because of unanticipated consequences on other processes, outcomes, activities or tasks. If necessary, modification is performed by deleting the provision (making the appropriate claim of tailored conformance) and, with careful consideration of consequences, implementing a process that achieves additional outcomes or performs additional activities and tasks beyond those of the tailored standard.

g) Provide explanation for why a life cycle process was tailored or why an outcome, activity, or task was deleted.

Annex B
(informative)

Metrics for architecture processes

B.1 Introduction

It would be difficult to provide a definitive list of process metrics that would be applicable in all situations. Therefore, this annex provides guidance on the information that can be considered when process metrics particular to an organization are defined.

Two definitions can be considered for metrics:

- A composite of two or more measurements resulting in a value that defines a characteristic of the process. [SEI]
- A quantifiable entity that allows the measurement of the achievement of a process goal. Metrics should be SMART—specific, measurable, actionable, relevant and timely. Complete metric guidance defines the unit used, measurement frequency, ideal target value (if appropriate) and also the procedure to carry out the measurement and the procedure for the interpretation of the assessment. [COBIT]

B.2 Guidelines for developing architecture process metrics

B.2.1 Metrics for governance effort

- Business resources indicators.
- Architectures status and evolution/intentions with regards to business objectives
- Associations amongst the architectures of interest: models identifying commonalities, differences, and dependencies.

B.2.2 Metrics for management effort

- Indicators allowing monitoring of architecture tasks and enabling planning evolution:
  - Scope (breadth of coverage, level of details, partitioning characteristics).
  - Schedule (time period, delivery schedule dates, provision for risk, etc.).
  - Resource utilization (staffing availability, manpower loading limitations, facility availability dates, capacity restrictions, and use of particular materials or reusable hardware or software units).
- Indicators allowing monitoring whether governance directives and guidance are being followed.
- Status or progress of management process via critical success factors, and numeric and graphical key performance indicators.
- Decision-making aids used to help to define actions when variations and trends are pointing out some risks.

B.2.3 Metrics for conceptualization effort

- Indicators allowing monitoring whether directives and guidance are being followed:
  - Governance directives and guidance,
  - Management directives and guidance.
- Check-lists to verify if conceptualization techniques, methods and tools are available, and if personnel are trained in the use.
- Status or progress of conceptualization process via critical success factors, and numeric and graphical key performance indicators.
- Risks and opportunities:
  - Indicators of risks with regards to likelihood and severity, and opportunities to be anticipated.
  - Monitoring aids used to trigger warning and alert when variations and trends are pointing out some risks.
  - Indicators of opportunities.
  - Dependencies within the process.
  - Dependencies between this process and other processes.

B.2.4 Metrics for evaluation effort

- Indicators allowing monitoring whether directives and guidance are being followed:
  - Governance directives and guidance,
  - Management directives and guidance.
- Check-lists to verify if evaluation techniques, methods and tools are available, and if personnel are trained in the use.
- Status and progress of evaluation process via critical success factors, and numeric and graphical key performance indicators.
- Risks and opportunities:
  - Indicators of risks with regards to likelihood and severity, and opportunities to be anticipated.
  - Monitoring aids used to trigger warning and alert when variations and trends are pointing out some risks (e. g. unsatisfied stakeholders).
  - Indicators of opportunities.
  - Dependencies within the process.
  - Dependencies between this process and other processes.

B.2.5 Metrics for elaboration effort

- Indicators allowing monitoring whether are directives and guidance being followed:
  - Governance directives and guidance,

— Management directives and guidance.

- Check-lists to verify if elaboration techniques, methods and tools are available, and if personnel are trained in the use.
- Status and progress of elaboration process via critical success factors, and numeric and graphical key performance indicators.
- Risks and opportunities:

  — Indicators of risks with regards to likelihood and severity, and opportunities anticipated.

  — Dependencies within the process.

  — Dependencies between this process and other processes.
- Monitoring aids used to trigger warning and alert when variations and trends are pointing out some risks e. g. deviation with respect to architecture description rules, consistency with requirement elaboration process, etc.

  — Indicators of opportunities: bypassing techniques, customization, etc.

B.2.6 Metrics for enablement effort

- Indicators allowing monitoring whether directives and guidance are being followed:

  — Governance directives and guidance,

  — Management directives and guidance.
- Check-lists to verify if enablement information and communications technology (ICT) resources and tools (e. g. licenses, documentation) are available and used.
- Check-list to verify performance of training, coaching and mentoring.
- Status of enablement process via critical success factors, and numeric and graphical key performance indicators.
- Risks and opportunities:

  — Indicators of risks with regards to likelihood and severity, and opportunities to be anticipated.

  — Monitoring aids used to trigger warning and alert when variations and trends are pointing out some risks. For example: users administration, interface consistency with other tools (import/export format), impact of software upgrades (software bloat), storage capacity, overcritical access delay, recurrent software bugs, etc.

  — Indicators of opportunities (e. g. software bypassing, information and communications technology (ICT) infrastructure enhancement, etc.)

  — Dependencies within the process.

  — Dependencies between this process and other processes.

Annex C
(normative)

Interactions with other processes

C.1 Relationship with system life cycle processes and stages

The architecture processes will interact with the following categories of system life cycle processes (specified in ISO/IEC/IEEE 15288):

- Agreement processes,
- Organizational project-enabling processes,
- Technical management processes,
- Technical processes.

The links are identified in three cases:

- When system life cycle processes provide information to architecture-related processes. For example, the validation process in ISO/IEC/IEEE 15288 might identify interfaces required for validation and fulfillment of constraints related to the architecture.
- When architecture processes provide information to a system life-cycle process. For example, the architecture elaboration process in this document might provide a validated architecture description package to the design definition process.
- When an architecture process in this document is implemented inside a particular system life cycle process in ISO/IEC/IEEE 15288.

Table C-1 describes how architecture should be used during each stage of a system life cycle. The relationship between system life cycle stages and these architecture processes is discussed in clause A.2.4.

When the systems engineering process is applied to things like a product line or an enterprise, these system life cycle stages might not be appropriate. For this description the stages are those described in the INCOSE SE Handbook and ISO TR 24748-1, Figure 6.

Table C-1 – Architecture use along a system life cycle

Stages	Architecture usage
Concept	Selling the program, procurement of funding, discovery of needs with stakeholders, opportunity assessment, operational analysis, exploration of system concepts and concepts of operations, exploration of support concepts, problem definition, understand feasibility and alternatives
Development	Requirements definition, concept of operations development, system analysis, system design, verification planning, support system design, organizational interface design, support data mapping, etc.
Production	Problem resolution, production planning, integration planning
Utilization	Operational planning, training of users, logistics planning, defect resolution
Support	Problem resolution, anomaly investigation, evolution planning, define support workflows, support function interactions
Retirement	Reuse planning, decommissioning decision, repurposing analysis

System life cycle processes in ISO/IEC/IEEE 15288 should use architecture-related information as specified in Table C-2.

Table C-2 – Uses of architecture by ISO/IEC/IEEE 15288 processes

15288 Clause	System Life Cycle Process	Uses of Architecture by this Process
6.1	**Agreement processes**	
6.1.1	Acquisition	Basis of supplier evaluation
6.1.2	Supply	Basis of solution to be supplied
6.2	**Organizational project-enabling processes**	
6.2.1	Life cycle model management	Identification of systems and system elements, system transition points
6.2.2	Infrastructure management	Identification of infrastructure needed by systems
6.2.3	Portfolio management	Identification of systems and system elements, system transition points, system inter-dependencies
6.2.4	Human resource management	Determination of necessary knowledge, skills and expertise
6.2.5	Quality management	Identification of systems and system elements
6.2.6	Knowledge management	Identification of architecture features to be used for tagging information items in knowledge repository, management of architectures and architecture information
6.3	**Technical management processes**	
6.3.1	Project planning	Identification of systems and system elements, system transition points, system attributes and measure, system inter-dependencies
6.3.2	Project assessment and control	Identification of systems and system elements, system transition points, system attributes and measures
6.3.3	Decision	Architecture evaluation recommendations. Identification of systems

15288 Clause	System Life Cycle Process	Uses of Architecture by this Process
	management	and system elements, system transition points, system attributes and measures.
6.3.4	Risk management	Identification of systems and system elements, system transition points, system attributes and measures, system inter-dependencies
6.3.5	Configuration management	Identification of systems and system elements, system interfaces, system configurations and options
6.3.6	Information management	Identification of architecture features to be used for tagging information items to be managed
6.3.7	Measurement	Identification of systems and system elements, system transition points, system attributes and measures
6.3.8	Quality assurance	Identification of systems and system elements, system transition points, system attributes and measures
6.4	**Technical processes**	
6.4.1	Business or mission analysis	Understanding of current and planned architectures and related systems
6.4.2	Stakeholder needs and requirements definition	Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.3	System requirements definition	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.4	Architecture definition	Basis for definition of architecture, architecture conceptualization, architecture elaboration
6.4.5	Design definition	Basis for design of system and non-system solutions. Note: Design definition may employ architecture at lower levels within a system hierarchy, e.g. at the system element level.
6.4.6	System analysis	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions. Note: Problem analysis will occur as part of the Architecture Conceptualization process.
6.4.7	Implementation	Understanding of intended use of architecture-related systems.
6.4.8	Integration	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions. Note: Integration addresses the composition of systems from their constituent elements and the integration of systems into their operational context/environment. Such considerations are identified, conceptualized and elaborated by architecting. So, architecting can be applied for this purpose.
6.4.9	Verification	Identification of systems and system elements, system transition

15288 Clause	System Life Cycle Process	Uses of Architecture by this Process
		points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.10	Transition	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.11	Validation	Identification of systems and system elements, system transition points, system attributes and measures. Identification of architecture features and functions.
6.4.12	Operation	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.13	Maintenance	Identification of systems and system elements, system transition points, system attributes and measures. Understanding of current and planned architectures and related systems. Identification of architecture features and functions.
6.4.14	Disposal	Identification of systems and system elements, system transition points. Understanding of current and planned architectures and related systems. .

Architecture processes in ISO/IEC 42020 should use system life cycle-related information as specified in Table C-3.

Table C-3 – Information used by 42020 architecture processes

15288 Clause	**System Life Cycle Process**	**Information Used by Architecture Processes**
6.1	**Agreement processes**	
6.1.1	Acquisition	Acquisition plans, supplier evaluation criteria
6.1.2	Supply	Proposed solution attributes
6.2	**Organizational project-enabling processes**	
6.2.1	Life cycle model management	Life cycle models of systems and system elements
6.2.2	Infrastructure management	Infrastructure features, functions and services
6.2.3	Portfolio management	Portfolio evaluation criteria, program and project dependencies
6.2.4	Human resource management	Knowledge, skills and expertise of current or planned personnel
6.2.5	Quality management	Quality assessment criteria
6.2.6	Knowledge management	General information from knowledge repository
6.3	**Technical management processes**	
6.3.1	Project planning	Project plans
6.3.2	Project assessment and control	Project assessment data
6.3.3	Decision management	Architecture-related decisions, decision criteria
6.3.4	Risk management	Identification of system risks and risk mitigation plans
6.3.5	Configuration management	Identification of configuration items. Baselined requirements and requirements changes (proposed and actual).
6.3.6	Information management	General information from information repository
6.3.7	Measurement	Measurement parameters and values
6.3.8	Quality assurance	Quality assurance criteria
6.4	**Technical processes**	
6.4.1	Business or mission analysis	Business needs, gaps and shortfalls. Mission needs, gaps and shortfalls.
6.4.2	Stakeholder needs and requirements definition	Identification of stakeholders and their concerns. Prioritization of concerns. Definition of perceived needs, gaps and shortfalls.

15288 Clause	System Life Cycle Process	Information Used by Architecture Processes
6.4.3	System requirements definition	Proposed and approved system requirements. Identification of key requirements constraints, conditions and challenges.
6.4.4	Architecture definition	Architecture plans and roadmaps. Architecture descriptions.
6.4.5	Design definition	Design plans and roadmaps. Design descriptions. Design evaluation results.
6.4.6	System analysis	System analysis results. System analysis tools methods capabilities and limitations.
6.4.7	Implementation	Implementation plans and roadmaps. Identification of key implementation constraints, conditions and challenges.
6.4.8	Integration	Integration plans and roadmaps. Identification of key integration constraints, conditions and challenges.
6.4.9	Verification	Verification plans and roadmaps. Identification of key verification constraints, conditions and challenges. Note: Verification approaches may impose issues such as architecture controllability and observability.
6.4.10	Transition	Transition plans and roadmaps. Identification of key transition constraints, conditions and challenges.
6.4.11	Validation	Validation plans and roadmaps. Identification of key validation constraints, conditions and challenges.
6.4.12	Operation	Operations plans and roadmaps. Identification of key operations constraints, conditions and challenges.
6.4.13	Maintenance	Maintenance plans and roadmaps. Identification of key maintenance constraints, conditions and challenges.
6.4.14	Disposal	Disposal plans and roadmaps. Identification of key disposal constraints, conditions and challenges.

C.2 Relationship with enterprise processes

One or more organizations participate in an enterprise to perform architecture processes to improve the ability for achieving mission and business objectives and desired outcomes.

Table C-4 describes how architecture should be used during each stage or phase of life cycle of an enterprise or any of its entities. For this description the phases are those described in ISO 15704 (called GERA life cycle phases) for any enterprise or entity. (GERA means generalized enterprise reference architecture.)

Table C-4 – Architecture use along an enterprise life cycle

Phases	Architecture usage
Identification	Selling the venture, procurement of funding,
Concept	Discovery of needs with stakeholders, identification of the enterprise assets. NOTE Architecture may be used to structure or restructure the enterprise itself e.g. conceptualization.
Requirements	Finding and definition of the true requirements, identification of enterprise projects
Design	Enterprise analysis, preliminary and detailed design, definition of enterprise projects
Implementation	Problem resolution, production planning, integration planning, set up of enterprise projects
Operation	Operational planning, training, logistics planning, monitoring of enterprise projects, anomaly investigation, problem resolution, evolution planning
Decommissioning	Reuse planning, decommissioning decision, repurposing analysis

The architecture processes will interact with the enterprise engineering process (specified in ISO 15704 A.3.2 – Enterprise engineering methodologies), taking into account aspects such as:

- human factors,
- project management,
- economic, financial and commercial considerations,
- New and changing standards.

The links are identified in three cases:

- When enterprise life cycle processes provide information to architecture-related processes. For example, when the enterprise strategy impacts the architecture governance activities.
- When architecture processes provide information to the enterprise life cycle process. For example, the architecture definition process updates the vision of the asset management of the enterprise.
- When an architecture process is implemented inside a particular enterprise life cycle process. For example, when the architecture definition activities are perform to elaborate an enterprise architecture aiming to structure the enterprise itself and its projects.

Annex D
(informative)

Relationship with other standards

The following list provides standards related to architecture:

- ISO/IEC/IEEE 42010, *Systems and software engineering — Architecture description*
- ISO/IEC 42030 (draft), *Enterprise, systems and software — Architecture evaluation elements*
- ISO/IEC/IEEE 15288, *Systems and software engineering — System life cycle processes*
- ISO/IEC/IEEE 15289, *Systems and software engineering — Content of life-cycle information products (documentation)*
- ISO/IEC/IEEE TR 24748-1, *Systems and software engineering — Life cycle management*
- ISO/IEC/IEEE 12207, *Systems and software engineering — Software life cycle processes*
- ISO 15704, *Industrial automation systems — Requirements for enterprise-reference architectures and methodologies*
- ISO/CEN 19439, *Enterprise integration — Framework for enterprise modelling*
- ISO/IEC TR 38502, *Information technology — Governance of IT — Framework and models*
- ISO 21500, *Project, programme and portfolio management — Guidance on project management*
- ISO/DIS 21505.2, *Project, programme and portfolio management — Guidance on governance*

NOTE The JTC1/SC7 Architecting Guidance Study Report provides the main references about architecture and architecting.

The following Figure D-1 describes the main relationships between this document and other ISO standards related to architecture and related activities. This document does the following:

- Refines the Architecture Definition process of the ISO/IEC/IEEE 15288 and ISO/IEC/IEEE 12207
- Frames the processes of these two standards
- Refines the Enterprise Reference Architecture of the ISO 15704 and the process description provided in the GERAM annex.
- Is considered with the enterprise principles defined by ISO 15704
- Provides the architecture context where the ISO/IEC/IEEE 42010 and ISO/IEC 42030 serve to respectively describe Architecture Description and Architecture Evaluation.

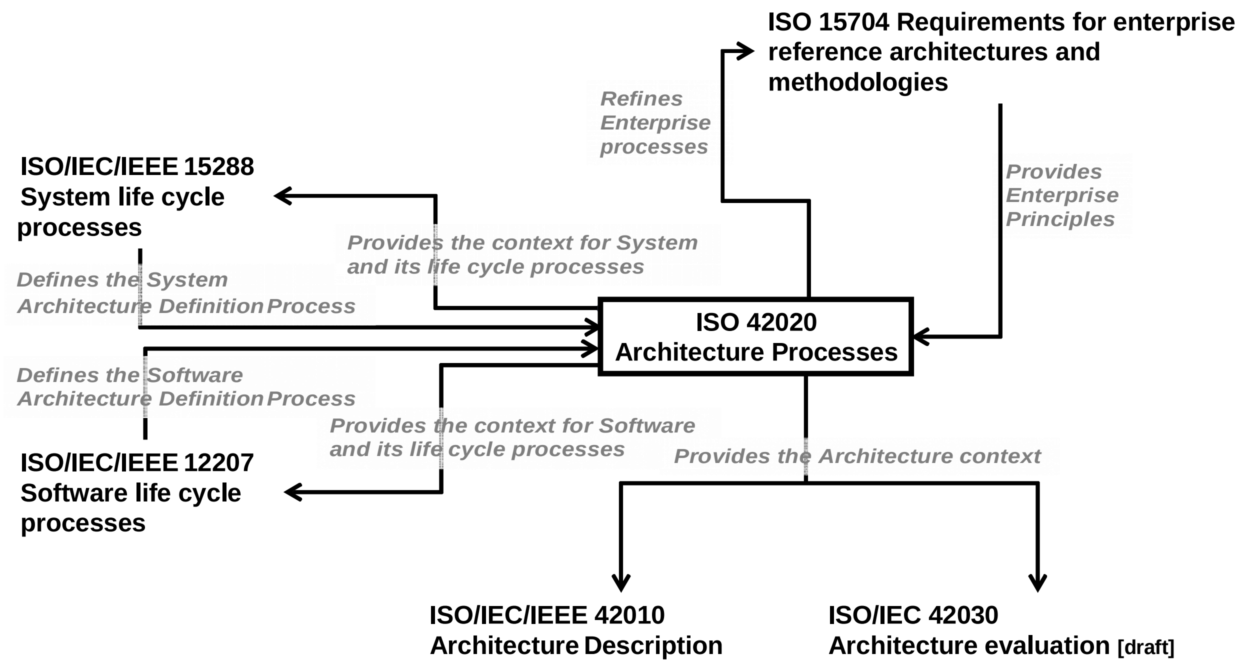

ISO 15704 Requirements for enterprise reference architectures and methodologies
Refines Enterprise processes
Provides Enterprise Principles
ISO/IEC/IEEE 15288 System life cycle processes
Provides the context for System and its life cycle processes
Defines the System Architecture Definition Process
ISO 42020 Architecture Processes
Defines the Software Architecture Definition Process
Provides the context for Software and its life cycle processes
ISO/IEC/IEEE 12207 Software life cycle processes
Provides the Architecture context
ISO/IEC/IEEE 42010 Architecture Description
ISO/IEC 42030 Architecture evaluation [draft]

Figure D-1 – Main relationships between ISO/IEC 42020 and other ISO standards

Annex E
(informative)

Notes on terms and concepts

E.1 Introduction

This annex complements Clauses 4 and 5 with additional information about key terms and concepts used in this document.

E.2 Architecture concepts

E.2.1 Architecture solution concepts

Solution is a term used very often in scientific and technical activities. Figure E-1 gives an overview of the solution concepts.

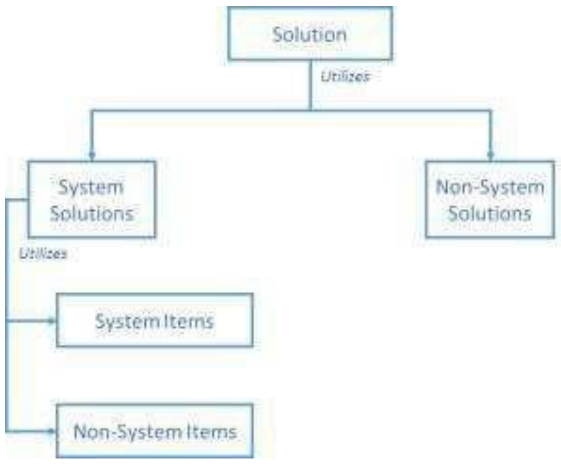


Figure E-1 – Solution concepts

A solution is an answer to a problem that addresses concerns of stakeholders.

EXAMPLE 1 Solution can be business, information technology, mission, capability, service, architecture building block (as defined by the TOGAF framework), and reference architecture (as defined by ISO 15704).

This solution may utilize system solutions and/or non-system solutions.

NOTE 1 The term *system* is used in ISO/IEC/IEEE 42010 to refer to entities whose architectures are of interest. The term is intended to encompass, but is not limited to, entities within the following domains:

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

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

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

EXAMPLE 2 System solution can be system of systems, class of systems, individual systems and solution building block (as defined by the TOGAF framework).

EXAMPLE 3 Non-system solutions can be product line, family of systems, data, technology, policy, process and mission thread.

System Solutions may utilize system items and/or non-system items.

EXAMPLE 4 System item can be class of systems, system of systems, individual systems, product system, service system and natural system.

EXAMPLE 5 Non-system item can be hardware item, firmware, software item, database, personnel, role, resource, service and natural resources (e.g. water, air, animals).

NOTE 2 Architecture can exhibit any part of the solution being considered as entity-of-interest.

E.2.2 Architecture life concepts

Figure E-2 identifies key architecture life concepts and their relationships.

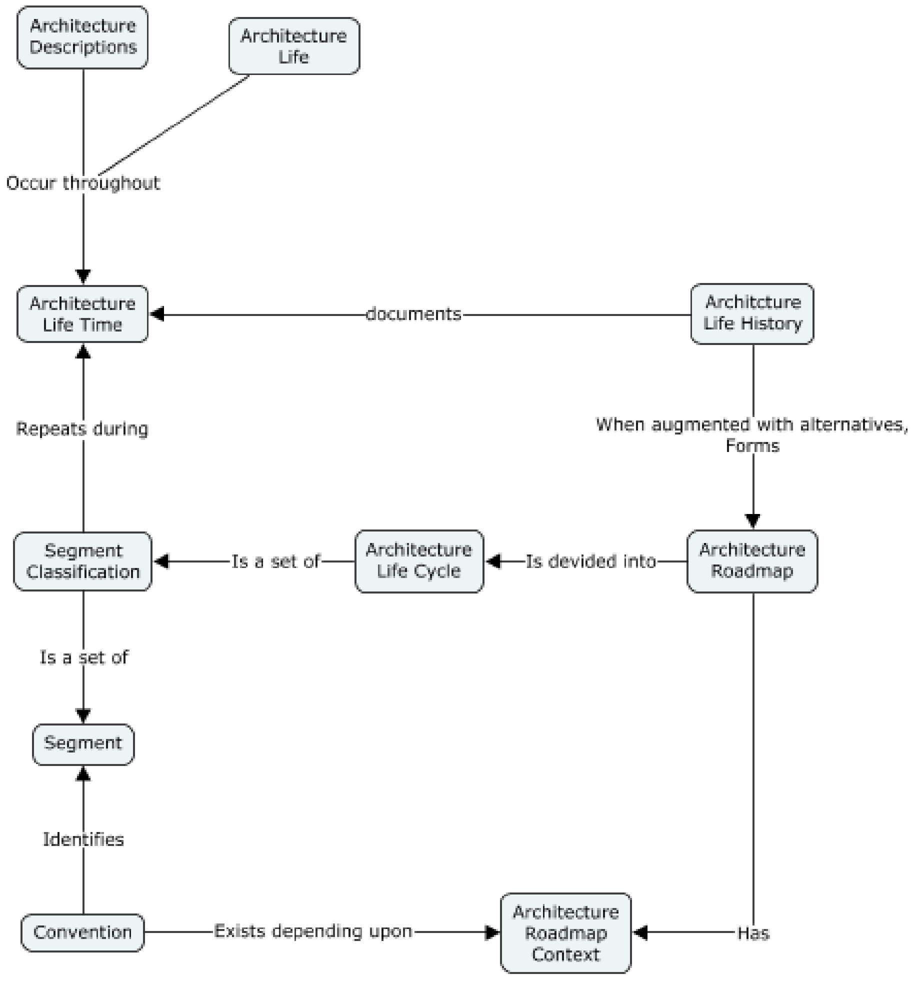


Figure E-2 – Architecture life concepts

Architecture life concepts refer to the characterization of entity architecture as documented by architecture descriptions that occur throughout the entity's life span. Each entity for which an architecture is said to exist, shall have a life time described by a life history associated with the evolution of the architecture, in whatever form that reality takes, to end of life. This architecture life history, when augmented with alternatives considered along the way, forms a roadmap for the architecture of the entity described, whether that entity is a single system-of-interest or some grouping of systems as yet lacking particular clarification.

Since many entities may utilize the same architecture, and since that architecture may change over time for particular entities or may be reused at some other time, the architecture itself shall have a life history distinct from the life history of the entities utilizing that architecture.

To classify commonly encountered segments along that roadmap several conventions exist depending upon roadmap context. Collectively, these conventions identify distinct segments found in most roadmaps as a life cycle, i.e. while a particular entity may progress from one segment to the next during its life span without repetition of a segment, all entities of that particular kind transition through the same set of segments. Life cycle shall refer to the set of distinct segment classifications that commonly occur within a context even when one or more of those segments repeats during the entity's life span. In this regard, the life cycle metaphor is conceptual since no life cycle instance in nature repeats a metamorphic segment for a particular individual.

Life cycle is set of distinguishable phases or stages that an entity goes through from its conceptualization until it ceases to be used (See 4.11). Phases are collections of logically related activities (see 4.14) while stages are periods within the life cycle that relates to the state of its description or realization (see 4.19).

Two life cycle classification schemes have found utility in International Standards associated with system and enterprise architecture. Independently developed at about the same time, ISO/IEC 15288 is the result of work associated with the standardization of engineered systems while ISO 15704 is the result of work associated with the standardization of industrial automation systems. Since initial publication, amendment and revision for both standards now position them for broader and more general application than originally published, the focus of ISO/IEC 15288 is still systems and the focus of ISO 15704 is still enterprises.

E.2.3 Life cycle models

Every entity has an associated life cycle. According to ISO/IEC/IEEE 15288 a life cycle is the "evolution of a system, product, service, project or other human-made entity from conception through retirement". A life cycle can be described using an abstract functional model that represents the conceptualization of a need for the entity, its realization, utilization, evolution and disposal. Such life cycle models may be defined as a "framework of processes and activities concerned with the life cycle that may be organized into stages, which also acts as a common reference for communication and understanding".

An entity progresses through its life cycle as the result of tasks performed and managed by people in organizations, using processes and practices for the execution of these tasks. A life cycle model is expressed in terms of processes, their outcomes, relationships and sequencing/concurrency. ISO/IEC/IEEE 15288 defines a set of processes, termed life cycle processes, which can be used to describe a system's life cycle. Further details may be found in Annex C.

An architecture can also be viewed as having a life cycle that is distinct from the associated entity life cycle.

Every architecture goes through various distinct stages of development, use and revision before it is ultimately discarded. It is therefore useful to think of an architecture as having a life cycle, which is not necessarily aligned completely with any particular entity. This perspective is important because it allows the identification of the appropriate infrastructure needed to manage architecting products. The nature of the architecture life cycle is determined by the purpose of the architecture.

Some typical architecture life cycles featuring varying degrees of use (and revision) are indicated in the figure below.

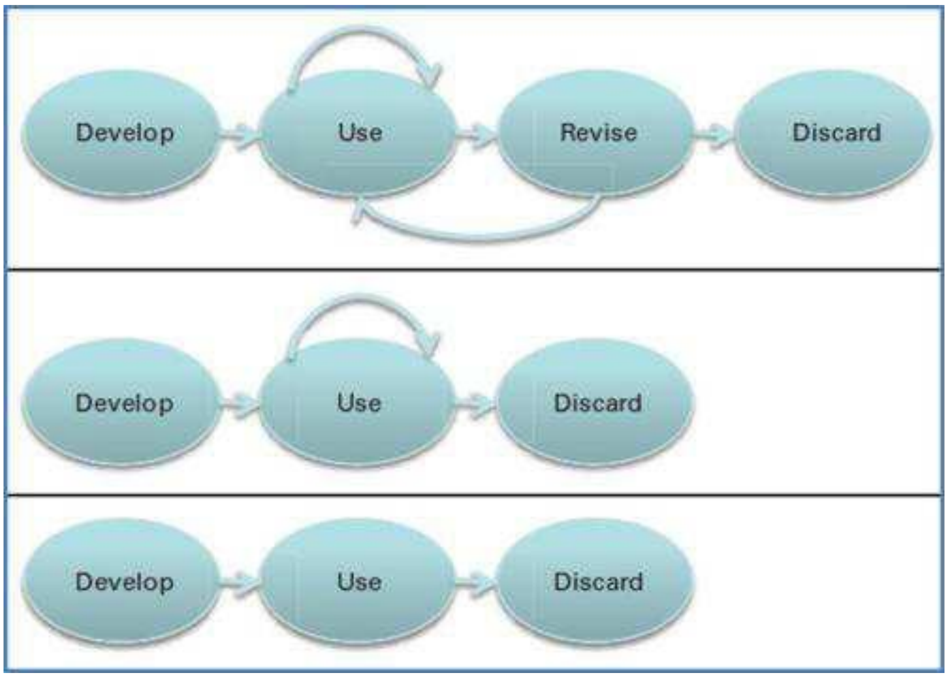


Figure E-3 – Architecture life cycle options

E.3 Architecting strategies and approaches

E.3.1 Architecting strategy

E.3.1.1 Introduction to Architecting strategy

An architecting strategy defines the starting point for conducting an architecting activity. It defines the background and reasons for trying to form a system concept. While there is no complete list, there are a number of common cases. The situation, the approach, and the other points made here are distinct but not independent. Arbitrary combinations will often not make practical sense, but neither does each solely depend on the other.

Note that these strategies are described in terms of being applied to a "system" architecture although they can be also applied to enterprise architectures, as well as to products and services that are not otherwise considered to be systems (e.g. software item).

E.3.1.2 New development, (sometimes known as a "greenfield" approach)

This is the classic case where there is no prior system, the system (or other kind of entity) being architected will be all new. In the most extreme case the sponsor will desire a system delivering an unprecedented capability implemented with a technology with limited or non-existent precedent (consider the development of nuclear submarines in the 1950's).

E.3.1.3 New Product in a Product-Line

In this case there are pre-existing systems with capabilities and technology quite similar to the desired system. The new system is intended to make a precedented extension of the pre-existing related systems that have been genericized into a product line.

E.3.1.4 Legacy evolution

Here the architect has an existing system or collection of systems (the "legacy") and is tasked to evolve or extend that legacy. The goals may be to add capability, reduce operating cost, eliminate obsolete technology, or something else. A key aspect is that the legacy exists and cannot be abandoned.

An outstanding example of legacy evolution is planned periodic technology refresh such as submarine combat system technology with alternating biennial refresh of the hardware and software baselines in even and odd years to minimize logistics costs for hardware and provide a more capable computing infrastructure to support future evolution.

E.3.1.5 Legacy revolution

In this case the legacy exists but the sponsors intent is to radically depart from that legacy. The usual case here is that while the legacy works there is a belief that the adoption of a radical departure (usually a radical departure in both technical approach and concept of operations) will enable large changes in capabilities or costs. The situation is not entirely a new development because the legacy capability exists and the "revolutionary" system will presumably have to interface to it in some fashion, but the deliberate intent is to abandon some large fraction of existing infrastructure.

E.3.1.6 Incremental start

Here the sponsor wants to build something new, but the uncertainty about what will be the best fit is great. Instead of making a large-scale and irrevocable bet on the future system the intent is to make an incremental step, to be followed by later steps that move toward a superior system. The architecting activity embraces the uncertainty and develops both initial steps and options for subsequent steps that account for known (and possibly unknown) uncertainties in user demand or technology.

E.3.1.7 Product-line start

This is a particular case of a new start where the intent is to build a product-line rather than a single system. The result should be a genericized solution used to produce an indeterminate number of future, related systems, related by reliance on a common base of design, technology, or production, and possibly variable elements.

E.3.2 Architecting approaches

Several different approaches to architecting (i.e. conceptualization, evaluation and elaboration) exist. These are categorized primarily in terms of the strategy (starting points) for forming the system architecture. The approach which should be adopted depends upon the complexity of the system of interest, its novelty, realization mechanisms, and/or the uncertainty in the stakeholder needs.

Note that a particular system architecting effort sometimes requires the use of more than one of these approaches.

Some examples of architecting approaches are given in the following sub-clauses.

E.3.2.1 Forward

In forward architecting the architecting process moves from consideration of the problem space to the solution space, initially at an architectural level of consideration. Within forward architecting several more specific approaches may be employed, either singly or in combination, for example top-down or bottom-up. Architecture evaluation is also likely to be employed to ensure that the architecture is sound in meeting its intended purpose.

E.3.2.2 Bottom-up

In bottom-up architecting the starting points are the artifacts, capabilities and/or services that are available and/or realizable which are then composed and formed into a system architecture exhibiting the desired (or desirable) emergent properties.

Note that in studying an existing system there is a clear distinction between producing an architecture description document and comprehending the pre-existing architecture of the system. One may have a clear comprehension of the architecture of the system of interest without having developed a complete document. Conversely, one may have a comprehensive document (albeit one badly written) and still have no clear comprehension of any organizing structure or principles of the system (perhaps because there are no such organizing abstractions).

E.3.2.3 Middle-out

In the middle-out approach an arbitrary level of abstraction in the system hierarchy is used as the starting point. Reasoning about, and architecting of, the system then progresses both upwards (towards the goals) and downwards (towards artifacts/capabilities/services).

E.3.2.4 Outer-in

This approach to system architecting starts at both the top (system goals) and bottom (artifacts/capabilities/services) and works towards the middle. It entails balancing and harmonizing desirable and achievable system properties.

E.3.2.5 Reverse

Reverse architecting is an aspect of reverse engineering for making the architectures of existing (or designed) systems explicit. It involves the extraction, abstraction and presentation of system information. The devising of "as is" architectures can be devised in this manner. Reverse architecting may address the goals of the existing systems, their components, or both. Architecture evaluation may form part of reverse architecting in ascertaining for example as to whether an existing architectural solution is suitable for continued usage.

E.3.2.6 Top-down

In top-down system architecting the starting points are the system goals. The approach proceeds through conceptualization to form the system architecture, stopping when appropriate levels of definitional formality and detail have been achieved. The devising of "to-be" architectures generally involves some top-down architecting.

E.3.2.7 Zigzagging

An approach to decomposition which entails moving from the functional domain to the physical domain (and the process domain) since, according to axiomatic design, decomposition of functional requirements and design parameters (and process variables) cannot be achieved by remaining in a single domain.

E.3.3 Useful architecting mechanisms

Various mechanisms can be used to devise a system architecture. Some commonly used mechanisms are:

E.3.3.1 Reference Architectures

Reference Architectures are defined in E.4.1.3. They can be instantiated and specialized to yield potential architecture solutions.

E.3.3.2 Architectural Patterns

Architectural patterns are patterns as applicable to architectures and architectural artifacts; they tend to be more specific in scope or aspect than reference architectures. They are more abstract in nature and have broader applicability than design patterns. Patterns may be implemented as a set of tactics.

They apply to functional components and modules, their structure, organization and interaction. They include rules and guidelines for organizing the relationships between the constituent elements, serving as templates for functional composition. They may be used to identify commonality and re-usability in systems and to deliver key functionality.

E.3.3.3 Tactics

Tactics are primitive techniques architects employ to achieve particular system characteristics. They are simpler and more primitive than patterns and are abstract in nature. Architectural patterns and styles incorporate and are implemented using multiple tactics. Tactics may be employed, for example, to ease modifiability of a system or to ensure performance levels are achieved.

E.3.3.4 Heuristics

Within the scope of this standard, heuristics are guidelines for architecting. They are derived from experience. Not all heuristics apply in all circumstances. Heuristics exist for different aspects of architecting such as partitioning functionality, aggregating functionality, evaluating systems architectures etc.

E.4 Architecture kinds, views and styles

E.4.1 Architecture kinds

Different kinds of architecture can be considered according to their purpose, domains of application, and roles within entity and architecture life cycles. Architecting may require the use (including development and/or application) of architectures of several kinds.

NOTE Descriptions expressing a viewpoint, an aspect, an abstraction level, or a perspective are sometimes called architecture kinds. Examples are security architecture, logical architecture and physical architecture.

In discussing architecture kinds it is important to keep in mind the distinction in ISO standards (especially ISO/IEC/IEEE 42010) between architecture descriptions (as documents) and architectures (as conceptual things).

When the entity of interest is intended as a generalized case to be used as a guide for use in the architecting effort, then this kind of architecture is sometimes called a "reference architecture". The entity itself in this case is not intended to be instantiated but is used as the basis for creating a realizable architecture of some entity that is a more concrete example of the abstract reference entity. An example of a reference architecture of this kind is the Open Systems Interconnection model (OSI model) specified in ISO/IEC 7498-1.

Some commonly used architecture kinds are given in the following sub-clauses.

E.4.1.1 Enterprise architecture

An enterprise architecture is either the architecture of an enterprise [INCOSE UK APP] or the architecture of an entity from an enterprise point of view. In both cases, it has a defined overall business objective and may include one or more participating organization.

Enterprise architecture is the organizing logic for business processes and information technology infrastructure reflecting the integration and standardization requirements of the company's operating model. The operating model is the desired state of business process integration and business process standardization for delivering goods and services to customers. [MIT Center for Information Systems Research, Peter Weill, Director, as presented at the Sixth e-Business Conference, Barcelona Spain, March 27, 2007]

E.4.1.2 Overarching or strategic architecture

An overarching architecture provides a strategic architectural context for a collection of entities and their associated architectures, including the interactions between these entities and any dependencies between the architectures. It concentrates on high-level objectives at the level of capabilities, systems of systems, or portfolios of projects and of necessity addresses such considerations at a comparatively high level of abstraction given the breadth of coverage.

NOTE Notion of overarching architecture is described in the NATO Architecture Framework. A similar architecture kind is called "strategic architecture" in The Open Group Architecture Framework (TOGAF).

EXAMPLE An overarching architecture can be done for maritime surveillance of a country. This architecture orients programs and projects which occur in this scope, focusing particular domains of activities, like maritime search and rescue.

E.4.1.3 Reference architecture

A reference architecture is used by a community of interest as a shared and agreed reference description that can be used for that community's business purposes. It is usually generic and is instantiated as architectures specific for individual business purposes. Reference architectures are used to (1) aid understanding of the forms of likely solutions to problems within a particular domain, and (2) to maximize the possible commonality in forms of solutions to similar problems within such a domain.

When the entity of interest is intended as a generalized case to be used as a guide for use in the architecting effort, then this kind of architecture is sometimes called a "reference architecture". In this case, the entity itself is not intended to be instantiated but is used as the basis for creating a realizable architecture of some entity that is a more concrete example of the abstract reference entity. An example of a reference architecture of this kind is the Open Systems Interconnection model (OSI model) specified in ISO/IEC 7498-1.

NOTE Notion of reference architecture is described in many architecture frameworks, engineering methodologies and guides.

EXAMPLE See OASIS Reference Architecture for Service Oriented Architecture

E.4.1.4 Domain architecture

A generic, organizational structure or design for systems in a domain.

[Based on IEEE Std 1517-1999 (R2004) IEEE Standard for Information Technology — Software Life Cycle Processes — Reuse Processes, and modified by generalizing in terms of systems rather than software systems]

NOTE The domain architecture contains the architectural forms that are capable of satisfying requirements within a specific domain. A domain architecture 1) can be adapted to create designs for systems within a domain, and 2) provides a framework for configuring assets within individual systems.

E.4.1.5 Baseline architecture

Baseline architecture is the definition of the architecture being defined for a given point of time. They serve as synchronization key-points for review and provision of architectures as references. Particular cases are:

- Current architecture (or "as-is" architecture) is the definition of the architecture currently in use
- Target architecture (or "to-be" architecture) gives the expected definitive definition.

NOTE A definition of target architecture is provided below.

E.4.1.6 Target architecture

Target architecture is a description of an envisioned architecture concerning the ultimate evolution of the entity of interest.

E.4.1.7 System Architecture

System architecture addresses the architecture of a system. It can be defined for one or more epochs according to a roadmap.

NOTE 1 System is used here to cover anything studied with a systemic approach.

NOTE 2 System architecture definition will apply the directives given by the relevant Overarching Architecture, if any.

NOTE 3 A system architecture can be derived totally or partially from one or several Reference Architectures.

NOTE 4 If the system architecture is defined for one or more epochs according to a roadmap, there is one (or several) system Baseline Architecture and one system Target Architecture.

E.4.1.8 Product line architecture

According to ISO/IEC/IEEE 24765, a product line is:

- From the commercial viewpoint, a group of products or services sharing a common, managed set of features that satisfy specific needs of a selected market or mission.
- From the engineering viewpoint, a collection of systems that are potentially derivable from a single domain architecture.

Architecture description of a product line formulates a set of products addressing similar problems and exhibiting an appropriate degree of architectural and solution commonality.

E.4.1.9 System of Systems Architecture

The architecture of a system that meets the criteria for a "system-of-systems". Systems of systems will be distinguished from large but monolithic systems by the operational and managerial independence of their components, their evolutionary nature, emergent behavior, and a geographic extent that limits the interaction of their components to information exchange. [Mark Maier, "Architecting Principles for Systems-of-Systems"]

E.4.1.10 Product-Service Systems Architecture

The architecture of a system that meets the criteria for a "product-service system". A product service-system will be a system of products, services, networks of "players" and supporting infrastructure that continuously strives to be competitive, satisfy customer needs and have a lower environmental impact than traditional business models. The product/service ratio in this system will vary depending on the function fulfillment or economic value. The architecture of the product-service system will be capable of jointly satisfying a user's need while taking into consideration the shift from techno-productive dimension (focus on functionality) to the social and cultural dimension (focus on value).

NOTE Service-Oriented Architecture (SOA) is sometimes identified as an architecture kind but is more accurately termed an architecture style. In fact, any of the architecture kinds considered can be defined with a service orientation.

E.4.1.11 Data Architecture

A data architecture is composed of models, policies, rules or standards that govern which data is collected, and how it is stored, arranged, integrated, and put to use in data systems and in organizations. [Wikipedia]

E.4.2 Architecture views

Architecture views express the architecture from the perspective of specific concerns about the architecture entity. Typically several such [complementary] views are formed during architecting. Some commonly used views are given in the following sub-clauses.

E.4.2.1 Contextual [view of] Architecture

The contextual view of an architecture deals with contextual factors and drivers including, for example, PESTEL (political, economic, social, technical, environmental or legal) aspects or description of DOTMLPFI (Doctrine, Organization,. Training, Material, Leadership, Personnel, Facilities, Interoperability) aspects.

E.4.2.2 Conceptual [view of] Architecture

The conceptual view of an architecture provides the main ideas or concepts from outside of the architecture. This provides the fundamental view that can be used as a starting point for further architectural efforts.

E.4.2.3 Functional [view of] architecture

ISO/IEC/IEEE 42010 conformant architecture description document will normally contain a view capturing the functions of the system of interest (it is not a 42010 normative requirement but would almost always be included). A functional architecture can be said to be the essential or organizing functional structure, the way abstracted inputs are transformed into outputs by the system to achieve its mission.

E.4.2.4 Logical [view of] architecture

The logical view of an architecture in an architecture description is typically an integrated model of both the system's functions and retained data, where the abstraction level is chosen to be directly relevant to users rather than implementation. The logical view of the architecture defines data as it makes sense in the problem domain, and defers definition of how it might be represented to other views. So, for example, the logical view would define data and functional transformations in terms of positions, currency amounts, or objects of user interest and not in terms of XML records or database fields.

NOTE What constitutes a logical or physical architecture is often somewhat subjective and a spectrum of such architectures may be so employed.

E.4.2.5 Physical [view of] architecture

A physical view of the architecture description is an arrangement of system elements and physical interfaces that provides the design solution for a product, service, or enterprise, and is intended to satisfy logical architecture elements and system requirements.

E.4.2.6 Physical [view of] Architecture

A physical view of the architecture description is defined in terms of technical recognizable objects that compose a systems implementation and the interfaces among them. It is implementable through technologies.

E.4.2.7 Organizational [view of] architecture

The Organizational View represents the responsibilities and authorities on all entities identified in the other enterprise views (processes, information, and resources). It caters for the structure of the enterprise organization by organizing the identified organizational units into larger units such as departments, divisions, sections, etc. [ISO 15704:2000 A.3.1.5.3.2]

The Organization related aspects have to do with decision level, responsibilities and authorities, the operational ones relate to the capabilities and qualities of humans as enterprise resource elements. [ISO 15704:2000 A.3.1.1]

E.4.3 Architecture styles

An architecture (or architectural) style is a set of principles and/or a generic pattern that provides a canonical guidance for architecting. It can be defined by the architecture elements, their topological layout, connectors and interaction mechanisms, and applicable constraints. Architecture styles may describe deployment patterns, structure and design issues, and communication factors. Their use improves structuring and understanding (through the use of established mechanisms and vocabularies), promotes design reuse (through the development of entities based upon proven forms of solution), and supports the consideration of pertinent technical issues.

NOTE Architecture styles are distinct from the notion of architecting styles. Architecting styles refer to ways of architecting which are codified according to the architecture's primary purpose including its extent of influence. [See INCOSE UK APP or Wilkinson/Evans papers]

Some examples of architecture styles particularly as applicable to computer-based systems are given in the following sub-clauses.

E.4.3.1 Client server

This architecture distributes data and processing physically across different types of system element. Servers provide specific services such as printing, data management, etc. Clients call on these services. Networks allow clients to access servers.

E.4.3.2 Component-based architecture

This style decomposes system functionality into reusable cohesive functional or logical components that expose well-defined communication interfaces.

E.4.3.3 Data-driven architecture

Data-driven architectures are concerned with the acquisition, manipulation and dissemination of data. They may be considered as being composed of pipelines of filters (which perform functional transformations of input data to produce data output) and pipes (which convey streams of data).

E.4.3.4 Event-driven architecture

Event-driven architectures promote the production, detection, consumption of, and reaction to events, where an event is a significant change in system state. Event-driven systems comprise event emitters (or agents), event consumers (or sinks), and event channels.

E.4.3.5 Layered architecture

Layered Architectures hierarchically structure functionality as several layers of increasing abstraction typically ranging from a problem focus at the top level to realization considerations at the lowest level. Interaction between layers is often restricted to adjacent layers.

E.4.3.6 Object-oriented architecture

An object is a collection of functions (called methods) and associated data. Object-orientation is an analysis and design paradigm based on the division of responsibilities for a system into individual reusable and self-sufficient objects, each containing the data and the behavior relevant to the object.

E.4.3.7 Publish-subscribe oriented architecture

Publish-subscribe is a style of information exchange in which certain elements offer data to other elements through published messages. Elements requiring such input data subscribe to the relevant messages.

E.4.3.8 Repository architecture

In this style of architecture the information exchange between system elements is physically realized through a central data repository which can be accessed (and where appropriate, contributed to) by all system elements.

E.4.3.9 Service-Oriented Architecture (SOA)

SOA is an architecture style that supports the service-orientation paradigm by exposing (and consuming) functionality from distributed systems as independent services in the form of stateless functions and using contracts and messages. SOA promotes reuse at the macro (service level) rather than micro (e.g. object) level.

Any of the architecture kinds described above can be defined with a service orientation. With this approach, services are preferred to expressed outcomes and interactions between entities (actors and constituents of the solution), with consideration of various operational, system, application and technical views.

NOTE Entities can support multiple architecture styles (heterogeneous architecture) for example to support different architectural concerns (information centricity, process/flow-orientation) but this increases solution complexity. The use of a single architecture style (homogeneous architecture) may ease design but may compromise certain required (or desired) system capabilities

E.5 Architecture motivation model

Enterprise activities, including architecture ones, will be driven by a set of motivation elements:

- Business aspiration including vision, goal, objectives and mission,
- Business means including strategy, policies, rules and guidance,
- Business constraints including laws, regulation and influencers,
- Existing and expected business assets including products, tools and people.

EXAMPLE An example of an architecture motivation model is provided by [OMG BMM, 2008].

These motivation elements can be used to form a dashboard for monitoring of the architecture activities:

- Aspiration elements are elaborated and used by governance,
- Means are drivers for management,
- Constraints and assets have to be considered for any process.

Criteria are derived from the dashboard for analysis and assessment of architectures activities and of the architectures themselves.

E.6 Quality measures

E.6.1 Introduction

This clause provides information on quality attributes, quality measures, and quality models that could be useful in applying this standard. The following items are covered in this Annex:

- Architecture quality attributes
- Boehm's quality models
- ISO/IEC 25000, Series of standards on System and Software Quality Requirements and Evaluation (SQuaRE)

The inclusion of these items does not imply endorsement of these particular ways of addressing quality. Exclusion of other items is not intended to imply their shortfalls. The intention is to include those items that can be related to the processes in this document.

E.6.2 Architecture quality attributes

Architecture quality attributes are the extent to which the architecture is able to deliver value to its stakeholders. It is a set of essential and distinguishing attributes that have a pragmatic interpretation of the architecture's inferiority or superiority. It is a function of: 1) architecture process outcomes, 2) impact of the architecture on various stakeholders, 3) measure of extent of achievement of stakeholder concerns, 4) measure of capabilities of the architecture.

Architecture quality attributes are the overall factors that affect behavior, structure, design and experience of architectures. They represent areas of concern that potentially impact the structure and behavior exhibited by the realized system. The extent to which the architecture handles a combination of quality attributes indicates the success of the architecting effort and overall quality of the realized system. The taxonomy for each architecture quality attribute would be:

a) Concerns: The parameters by which the attributes are measured.

b) Factors: Policies and mechanisms of the system and its environment that impact the stakeholder concerns.

c) Methods: Techniques for addressing concerns and processes for realizing the quality attributes during productions.

While conceptualizing architecture to address the architecture quality attributes, it is necessary to consider potential impact of each of the quality attributes on other stakeholder concerns. While tradeoff analysis techniques aid architects in prioritizing architecture quality attributes, architectural tactics describe how a specific quality attribute can be achieved. The importance of each architecture quality attribute depends on the context and the stakeholders concerns for which the specific architecture is conceptualized. The table below provides an example list of architecture quality attributes.

Table E-1 – Architecture quality attributes

Quality Attribute	Description
Coherence	Being logical and consistent
Completeness	Ability to form a whole
Elegance	Graceful and stylish
Hierarchy	Levels of abstractions
Modularity	Separation of concerns
Homeostasis	Balancing of factors
Epigenesis	Evolving progressively
Morphogenesis	Evolution of form

E.6.3 Boehm's quality model

Barry Boehm's [Boehm, 1978] models attempts to qualitatively define quality by a set of attributes and metrics. Boehm utilizes a hierarchical quality model structured around high-level quality characteristics, intermediate level quality characteristics and primitive quality characteristics for this purpose. Each of these characteristics contributes to the overall quality level. Boehm considers high level quality characteristics to represent the basic high-level requirements, intermediate level quality characteristics to represent the quality factors (portability, reliability, efficiency, usability, testability, understandability and flexibility), and primitive level quality characteristics to represent the quality metrics (device independence, accuracy, completeness, robustness, consistency, accountability and so on) that measure a given primary characteristic.

E.6.4 25000 Series of standards on quality

The series of standards ISO/IEC 25000, also known as SQuaRE (System and Software Quality Requirements and Evaluation), has the goal of creating a framework for the evaluation of software product quality. Some of these are discussed in this Annex:

- ISO/IEC 25000, SQuaRE – Quality model framework
- ISO/IEC 25010, SQuaRE – System and Software Quality models
- ISO/IEC 25020, SQuaRE – Measurement reference model and guide
- ISO/IEC 25012, SQuaRE – Data Quality model

E.6.4.1 ISO/IEC 25000 Quality model framework

ISO/IEC 25000 quality model framework, categorizes product quality into characteristics, which in some cases are further subdivided into sub-characteristics. A sub-characteristic in some cases can be divided into sub-sub-characteristics. In essence this results in a quality breakdown structure as illustrated in Figure E-4.

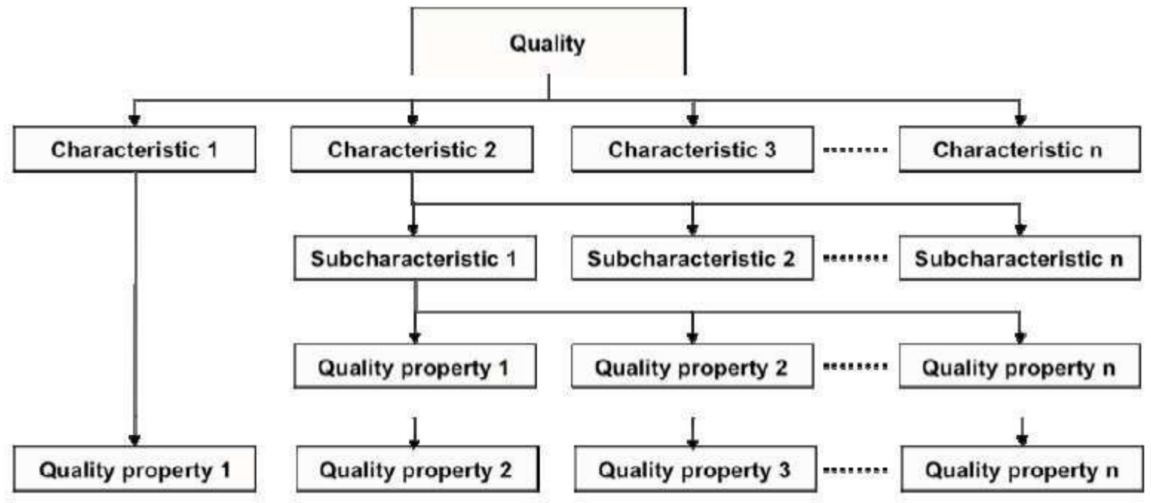


Figure E-4 – ISO 25000 Quality model framework

E.6.4.2 ISO/IEC 25010 System and software quality models

E.6.4.2.1 ISO/IEC 25010 Quality in use model

ISO/IEC 25010 quality in use model defines five characteristics related to outcomes of interaction with a system. It characterizes the impact that the product has on stakeholders. This model is presented in Figure E-5.

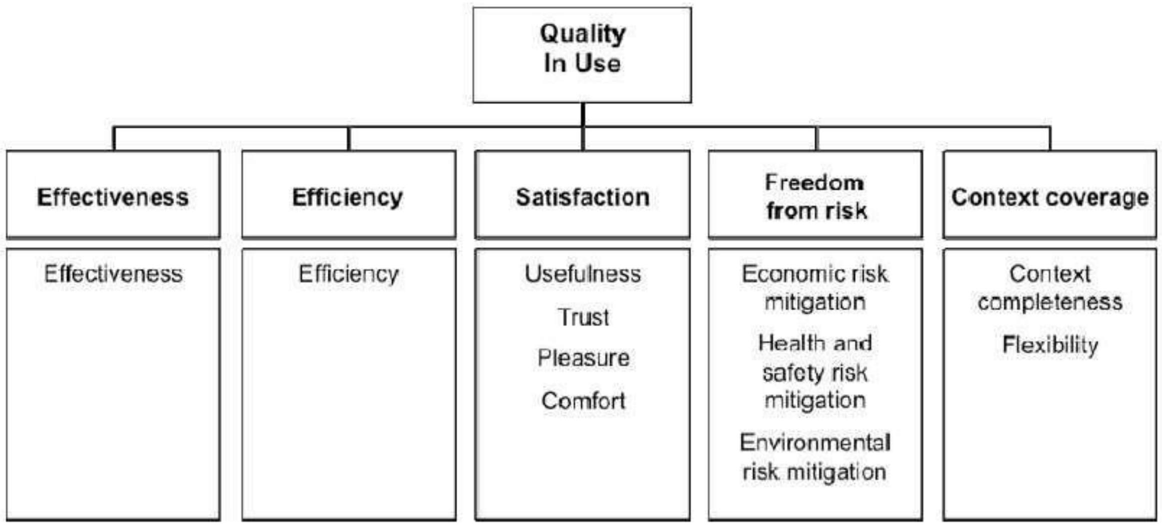


Figure E-5 – ISO 25010 Quality in use model

E.6.4.2.2 ISO/IEC 25010 System/software product quality model

The ISO/IEC 25010 system/software product quality model categorizes a system/software product quality properties into eight characteristics that focuses on the target system. This model is presented in Figure E-6.

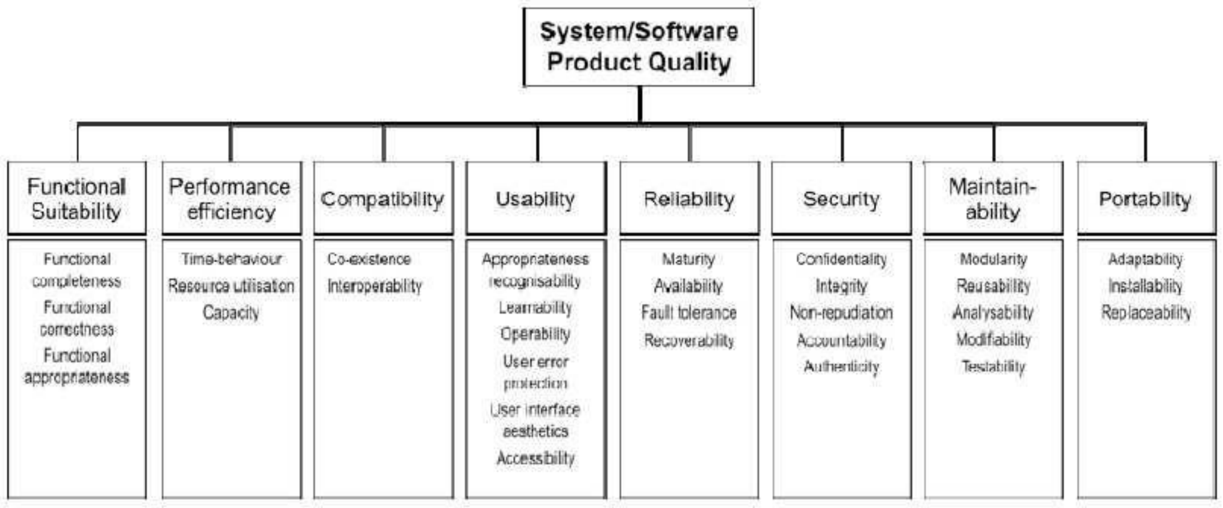


Figure E-6 – ISO/IEC 25010 System/software product quality model

E.6.4.3 ISO/IEC 25020 System and software product quality measurement reference model

ISO/IEC 25020 system and software product quality measurement reference model describes the relationship between a quality model, its associated quality characteristics (and subcharacteristics), and system and software product attributes with the corresponding software quality measures, measurement functions, quality measure elements and measurement methods. This model is presented in Figure E-7.

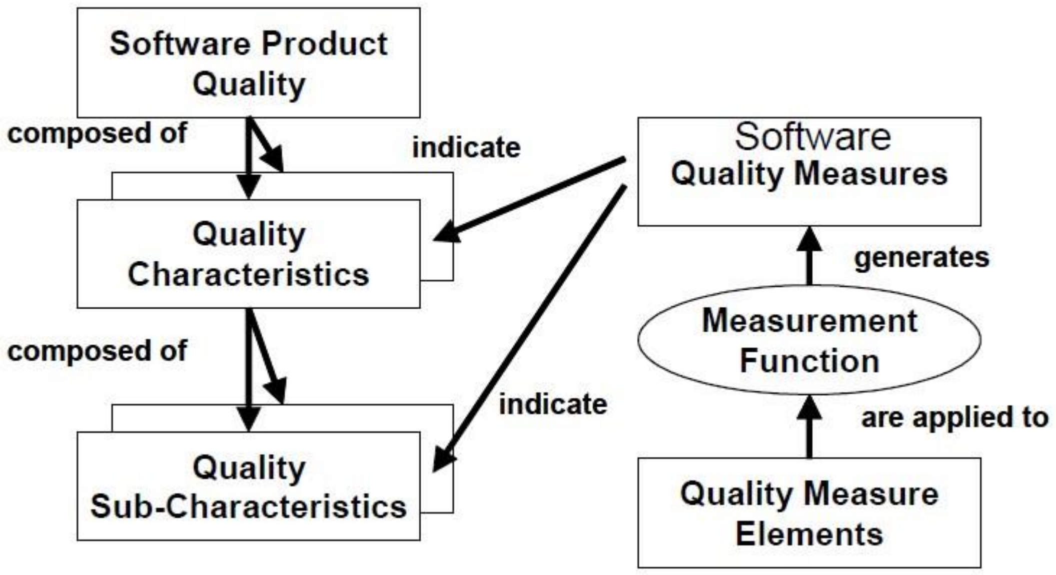


Figure E-7 – ISO/IEC 25020 System/software product quality measurement reference model

An illustration of this model as given in ISO/IEC 25021 is presented in the **Table E-2**.

Table E-2 – ISO/IEC 25021 product quality measurement reference model

SNo	Element	Particulars
1	Quality Measure Element name	Number of records
2	Objective	To determine data quality of target data
3	Property to Quantify	Record is a set of related data items treated as a unit
4	Relevant Quality Measures	Measure of Accuracy
5	Measurement method	Review and analyze data records
6	List of sub-properties	Data Item: Lowest component of a group of data File: A set of related records
7	Input for the Quality measure element	Physical files of a database
8	Numerical rules	Adding total records
9	Context of the Quality measure element	Measure the accuracy and completeness to a group of data
10	Measurement constraints	Verify the impact of technology on the number of records generated for the same information

E.6.4.4 ISO/IEC 25012 Data quality model

ISO/IEC 25012 data quality model, as illustrated in Table E-3, categorizes data quality attributes into fifteen characteristics that are considered by two points of view: inherent and system dependent.

While the inherent data quality refers to the degree to which the quality characteristics of data have the potential to satisfy needs when data is used in specified conditions, system dependent data quality refers to the degree to which data quality is reached and preserved within a system when data is used under specific conditions.

Table E-3 – ISO/IEC 25012 Data quality model

Characteristics	**Inherent**	**System dependent**	**Characteristics**	**Inherent**	**System dependent**
Accuracy	X		Efficiency	X	X
Completeness	X		Precision	X	X
Consistency	X		Traceability	X	X
Credibility	X		Understandability	X	X
Currentness	X		Availability		X
Accessibility	X	X	Portability		X
Compliance	X	X	Recoverability		X
Confidentiality	X	X			

Annex F
(informative)

Architecture enablement and process-enabling resources

F.1 Architecture enablement

Architecture enablement is needed for establishing and maintaining consistent practices, standard approaches, reusable items, and uniform ways of communication, and for proper utilization of these things. The enablers consist of a collection of tools, techniques, technologies, skills, practices, frameworks, methods, and processes used in architecture processes in support of the accomplishment of an organization's objectives. It is implemented for a given area of responsibility to guide the proper selection, development, utilization and improvement of enablers (tools, technologies, approaches, and so on). This process maintains information about the various work-products and also provides checks and balances to ensure that the information about these work-products is of high quality.

Architecture enablement has oversight over the performance of architecture process operations and decision making, as well as the efficient organization of people, capabilities, processes and other resources to achieve the architecture goals and objectives. It enables users to quickly understand available information so that they can make better and faster decisions and efficiently achieve architecture goals and objectives. Architecture enablement involves dealing with interactions, interconnections, activities, outcomes, and work-products and how they need to be structured to produce the desired governance, management and architecting functions.

Architecture enablement includes the following elements:

- Establishing an architecture repository that provides for the storage and archiving of architecture process artifacts and work products. The repository can be used to store different classes of architecture work products that facilitates coordination and cooperation between the architecture process stakeholders.

  NOTE The architecture repository referred to in this document is not necessarily isomorphic with architecture repositories in commercial use.
- Establishing an architecture library concerning enabling capabilities, services and resources that can be used by or deemed to be useful to those who perform role specific architecture activities. The library can provide guidelines, templates, patterns, and other forms of source material that can be leveraged by the architecture processes.

- Establishing an architecture registry that keeps information about the artifacts and work products contained in the repository and library as well as the changes made to them. The registry can be used to discover and use current and relevant information items for an ongoing architecture endeavor.

F.2 Architecture process enabling resources

Examples of resources that can be used in performing the architecture processes are listed below.

- Architecture patterns: a pattern addresses a specific architecture problem, in a context, to provide a solution maximizing reuse and permitting a range of tradeoffs.
- Architecture kinds: several kinds may be needed to fully express the essential properties and concepts (see E.4.1).
- Architecture styles: an idiom for organizing an architecture to achieve certain properties (see E.4.2).
- Model kinds: conventions for a type of modeling, to address specific types of concerns [ISO/IEC/IEEE 42010]
- Architecture description languages: any defined form of expression for use in architecture descriptions [ISO/IEC/IEEE 42010]
- Architecture viewpoints: work product establishing the conventions for the construction, interpretation and use of architecture views to frame specific system concerns [ISO/IEC/IEEE 42010]
- Architecture frameworks: conventions, principles and practices for the description of architectures established within a specific domain of application and/or community of stakeholders [ISO/IEC/IEEE 42010]
- Architecture methods: practice, technique, or procedure with rules to guide architecture processes
- Skills and knowledge associated with specific roles identified to perform architecture-related activities:
    - The skills required by each role
    - The depth of knowledge required to fulfill the role successfully
- Norms and standards associated with the activities and the work products.
- Tools and languages sustaining the activities and allowing the formulation of work products and their related information.

NOTE Catalogs can be used to collect homogeneous sets of metadata, resources and related information. The repositories can be implemented with catalogs used as references for governance, management and usage.

Annex G
(informative)

Architecture governance and management

G.1 Architecture governance

Architecture governance is needed for consistent management, cohesive standards and policies, proper guidance, uniform processes and appropriate decision-rights. It is implemented for a given area of responsibility to ensure proper oversight and accountability. This enables the organization to identify, manage, audit, and disseminate all information related to architecture decisions, management actions in response to these decisions, contracts affecting the architecture(s), and implementation of architecture changes.

Architecture governance has oversight of the architecture objectives for the architecture collection to ensure their consistency with organizational goals and objectives, among other things. Each set of architecture objectives will be considered with respect to factors of maintenance, servicing, and upgrade with minimal disruption of the everyday operations. In particular, consideration will be given to available internal and external resources in order to determine when general resources can be adapted for specific needs and to determine where specific solutions can be generalized to support wider re-use.

Architecture governance is the practice and orientation by which architectures are managed and controlled at an organization-wide level. It includes the following:

a) Formulating directives and guidelines of all the architectural components and processes, to ensure effective conceptualization, evaluation, elaboration, implementation and evolution of the collection of architectures within the organization.

b) Ensuring compliance with industry and governmental standards and regulatory obligations.

c) Establishing processes that support effective adherence to the directives, guidelines, policies, standards and other regulatory obligations.

d) Developing practices that ensure accountability of the architectural decisions for a collection of architectures to the governance board.

Architecture governance is typically performed at higher levels of the organization providing oversight over business units, programs, projects, etc. Architecture governance has responsibilities for legal compliance, alignment with organizational goals and objectives, optimum utilization of resources, maintaining focus on the long term vision, responding to changes in the marketplace and user community, anticipating new forces and scenarios that will likely arise, maximizing shareholder gains, etc.

Architecture governance acts on an architecture collection in order to check the alignment between them and for compliance with the organizational mandates and expectations. Usually the collection consists of several architectures that are related to each other, but the collection could consist of a single architecture if appropriate.

Each activity is governed by principles. An organizational authority should be in charge of checking that the activities are performed according to these principles. This authority is sometimes called a

“Design Authority”, identified for governance according to architecture principles with an escalation approach when necessary.

G.2 Architecture management

Architecture management is needed for centralized management of current and proposed collection of architectures. It establishes, maintains and uses a coherent set of guidelines, principles and management regimes that provides direction and instructions for the design and development of an architecture. It involves identifying potential risks and mitigating them in accordance with the governance directives. The objectives of architecture management are to determine, manage and control the risks, processes and resources necessary for architecting the collection of architectures while taking into account constraints, conflicts, strategic objectives and governance directives.

Architecture management is concerned with managing the architectures, not the activities of architecting. The main focus of architecture management is on managing the implementation and evolution of the architecture(s) to maximize alignment with strategic goals and objectives. The Architecture Management process does not manage the development of the architecture but rather its evolution and its implementation in the design, build, deployment, operations, maintenance, decommissioning, etc. of one or more systems related to the architecture. Architecture management is responsible for implementing the guidance and direction from architecture governance where this is accomplished by giving management guidance and direction to the other architecture processes. Architecture management provides plans and status to architecture governance on how well the architectures are evolving and being implemented.

Architecture management has oversight over the architectural decisions for the collection of architectures to ensure their consistency. Each set of architecture decisions will be considered with respect to factors of risks, evolution, cost, budget, and with minimal disruption of schedule and effort. In particular, consideration will be given to coordination, communication and control in order to determine when shared resources can be used efficiently and effectively.

Architecture management is the practice by which a set of architectural decisions are managed and controlled at a collection level. Architecture management includes the following elements:

a) Formulating an architecture management charter that defines the statement of work for a collection-of architectures.
b) Ensuring compliance with project, industry and governmental quality requirements.
c) Establishing management hierarchies and management plan that support architectural decision making.

Annex H
(informative)

Mapping architecture framework processes to 42020 processes

H.1 Introduction

This annex provides information on how the elements of various architecture frameworks relate to the processes in this standard. The following frameworks are covered in this annex:

- TOGAF Framework
- Pragmatic Enterprise Architecture Framework (PEAF)
- Generalized Enterprise Reference Architecture and Methodology (GERAM) framework
- RM-ODP (Reference Model – Open Distributed Processing) framework

The inclusion of these frameworks does not imply endorsement of these particular frameworks. Exclusion of other frameworks is not intended to imply shortfalls of those frameworks. The intention is to include those frameworks that contain processes similar in nature to the processes in this document so that the relationships can be better understood, or that include elements that can be related to the processes in this document.

Well-known other architecture frameworks available in 2016 like DoDAF, NAF, MODAF, AUS-DAF, UPDM, UAF and Archimate provide formalisms and sometimes notations; but do not provide description of architecture processes. These frameworks can be used during architecture elaboration in order to provide a formalized way of describing the architectures. The architecture elaboration process as described in clause 10 does make any assumption on formalisms or notations.

H.2 TOGAF framework

H.2.1 Framework overview

TOGAF, an Open Group standard, is a framework that provides a detailed method and a set of supporting tools for developing enterprise architectures. [TOGAF v9.1, 2011] It includes a process for developing the architecture description called the Architecture Development Method (ADM) (see Figure **H-1**), as well as general principles for doing architecting and for architecture governance.

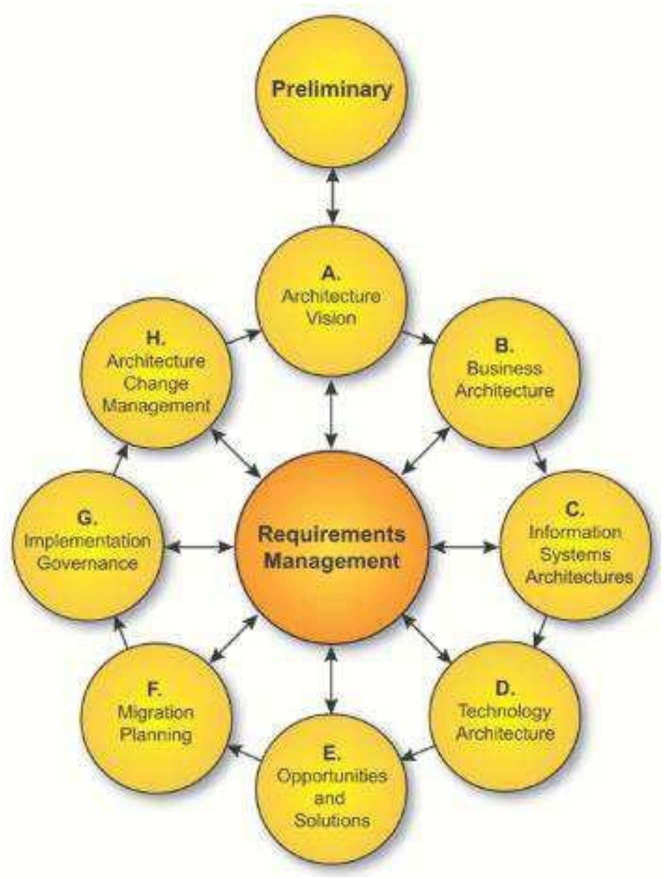

Preliminary
A. Architecture Vision
B. Business Architecture
C. Information Systems Architectures
D. Technology Architecture
E. Opportunities and Solutions
F. Migration Planning
G. Implementation Governance
H. Architecture Change Management
Requirements Management

Figure H-1 – Architecture development cycle [TOGAF v9.1, 2011]

The ADM (part II of the TOGAF framework) together with the rest of the guidance detailed in subsequent parts of the TOGAF framework cover the set of processes and guidance provided in the 42020 standard as shown in Table **H-1**.

H.2.2 Mapping to framework elements

While the coverage mapping is not one-to-one (e.g. several sections of the TOGAF framework map to more than one ISO process and vice versa), it is intended as a quick reference for users of the TOGAF framework who may wish to document how coverage of the ISO 42020 processes is achieved. An X indicates that full or partial coverage of the 42020 process is achieved via activities as described in the corresponding phase or in Parts III, V, or VII of the TOGAF Framework.

NOTE 1 42020 uses the definition for phase from [ISO/IEC/IEEE 24765:2010]: "a collection of logically related project activities, usually culminating in the completion of a major deliverable.“ In the TOGAF framework, the phases refer to iterative states used to group activities around specific content of the architecture description (e.g. business, technology, etc.)

NOTE 2 Refinement and updates to architecture governance and architecture management documents can occur during any of the TOGAF phases.

Table H-1 – Mapping of processes to the TOGAF framework

TOGAF Architecture Development Method (ADM) Phase	Architecture Governance	Architecture Management	Architecture Conceptualization	Architecture Evaluation	Architecture Elaboration	Architecture Enablement
CH 6 Preliminary Phase	X	X		X		X
Phase A: Architecture Vision	X	X	X			
Phase B: Business Architecture		X			X	
Phase C: Information Systems Architectures		X			X	
Phase D: Technology Architecture		X			X	
Phase E: Opportunities and Solutions		X		X	X	
Phase F: Migration Planning	X	X		X	X	
Phase G: Implementation Governance		X		X		
Phase H: Architecture Change Management	X	X		X		
Ch 26 Business Scenarios and Business Goals			X	X		
Ch 27 Gap Analysis				X		
Part V: Enterprise Continuum and Tools, Chs 38-32						X
Ch 46 Establish an Architecture Capability						X
Ch 47 Architecture Board	X					
Ch 48 Architecture Compliance	X	X		X		
Ch 49 Architecture Contracts		X				
Ch 50 Architecture Governance	X	X				
Ch 51 Architecture Maturity Models	X					X
Ch 52 Architecture Skills Framework				X		X

Source: [TOGAF v9.1 2011] TOGAF® Version 9.1, The Open Group, © 2009-2011

H.2.3 Items in the TOGAF framework not addressed in 42020:

ISO/IEC 42020 scope does not include roadmaps, the concept of time-specific architecture (baseline, target) and plans for transitioning (the enterprise) from baseline to target architectures.

H.2.4 Items in 42020 not addressed in the TOGAF framework:

ISO/IEC 42020 is concerned with the architecting effort, including architecture management and dealing with stakeholders for whom the architecture is being developed, evaluation of the architecture, etc.

ADM is only concerned with the steps needed to develop the Architecture Description. To augment this, the TOGAF framework includes additional information in Parts III, IV, and V, specifically, Chapter 26 covers Business Scenarios and Business Goals, and Chapter 27 covers Gap Analysis in Part III, while Part IV covers the Architecture Content Framework, and Part V covers the Enterprise Continuum and Tools.

H.3 PEAF framework

H.3.1 Framework overview

The Pragmatic Enterprise Architecture Framework (PEAF) is an element of the "Pragmatic Family of Frameworks" (PF^2) designed to help improve the maturity of how enterprises carry out their business.

PEAF instantiates the methods, artifacts, cultural and environmental sections defined in the "Pragmatic Ontology for Enterprise Transformation" (POET) framework in order to set the context for the context for strategizing and roadmapping enterprise architecture, as part of enterprise transformation.

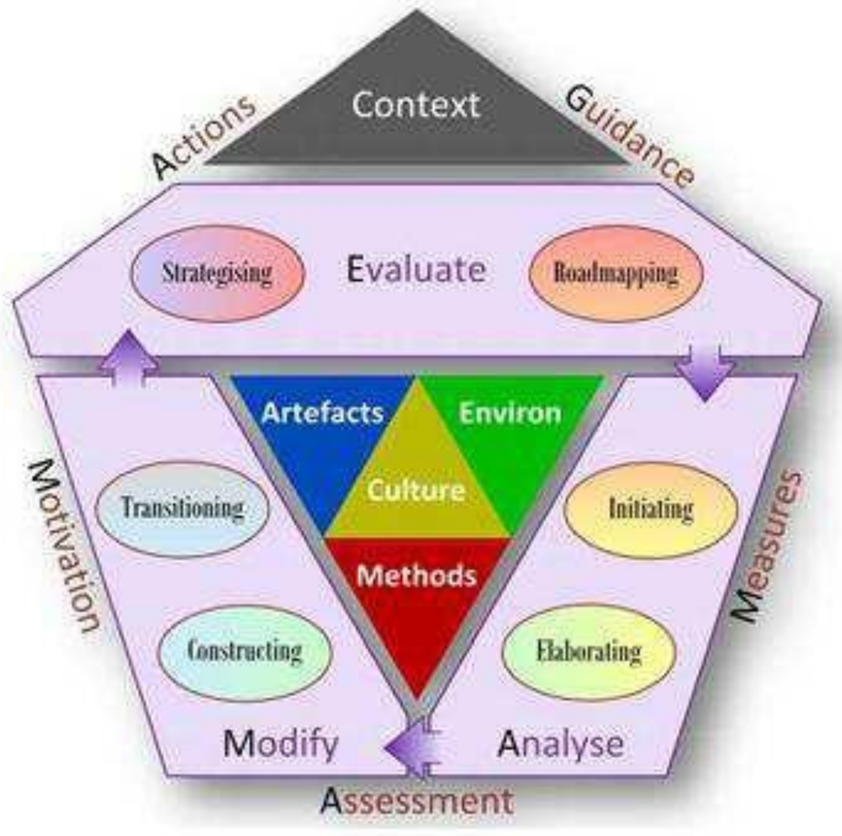


Figure H-8 – POET scope [PEAF v3, August 2014]

POET defines an ontology for enterprise transformation with information existing at different levels of Idealization/Realization: Motivation, Actions, Guidance, Measures and Assessment. Height fundamental phases of transformation are identified: Strategizing, Roadmapping, Initiating, Elaborating, Transitioning, using and Governance & Lobbying.

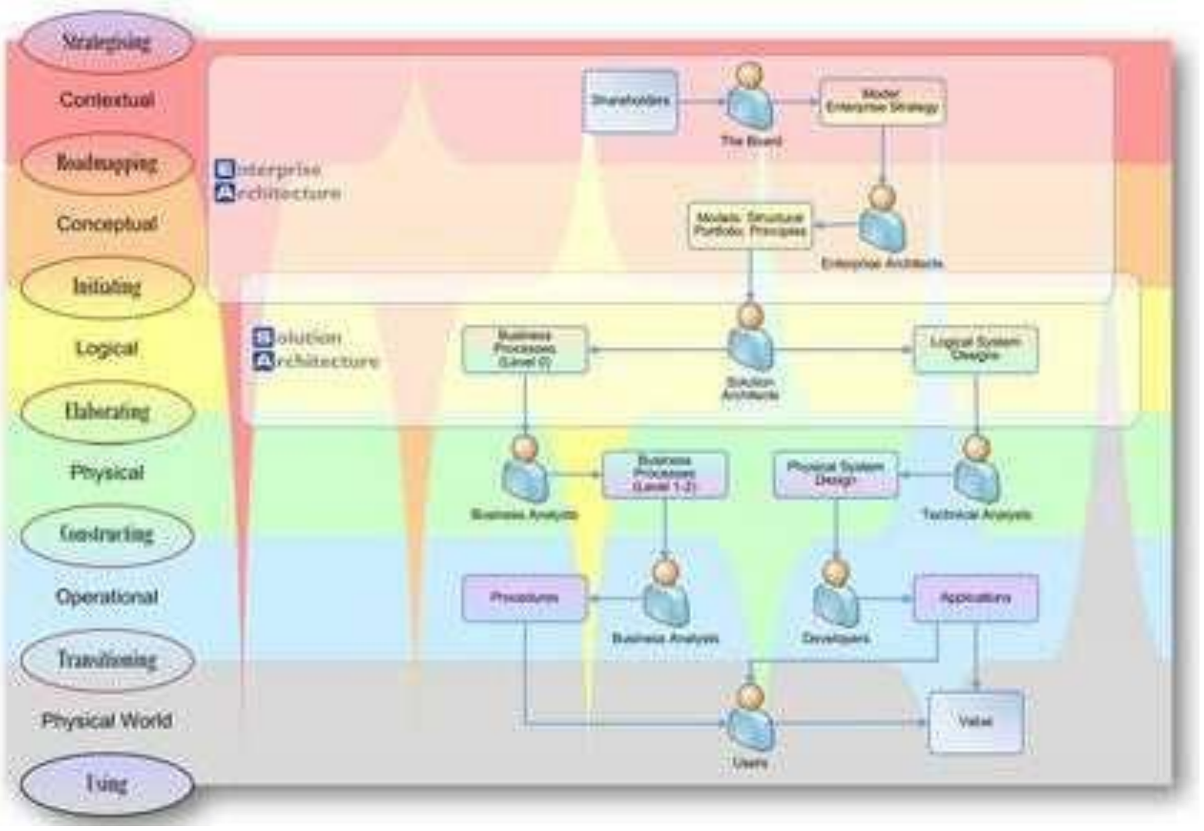


Figure H-9 – Enterprise architectural context where PEAF fits [PEAF v3, August 2014]

The enterprise architecture context described by PEAF relies on: Processes, Disciplines, Levels, Input and Output. Viewpoints to describe this context are: Contextual, Conceptual, Logical, Physical and Operational.

PEAF artifacts are metamodels used for enterprise planning and governance in order to develop: Business Model, Roadmap Model, Operating Model, Capability Model and Enterprise Context. From these basic models come two aggregate and overlapping artifacts: Enterprise Strategy and Transformation Strategy.

H.3.2 Mapping to framework elements

PEAF proposes to perform enterprise transformation based on methods, artifacts, culture and environment with actions and foundation described in the following figure.

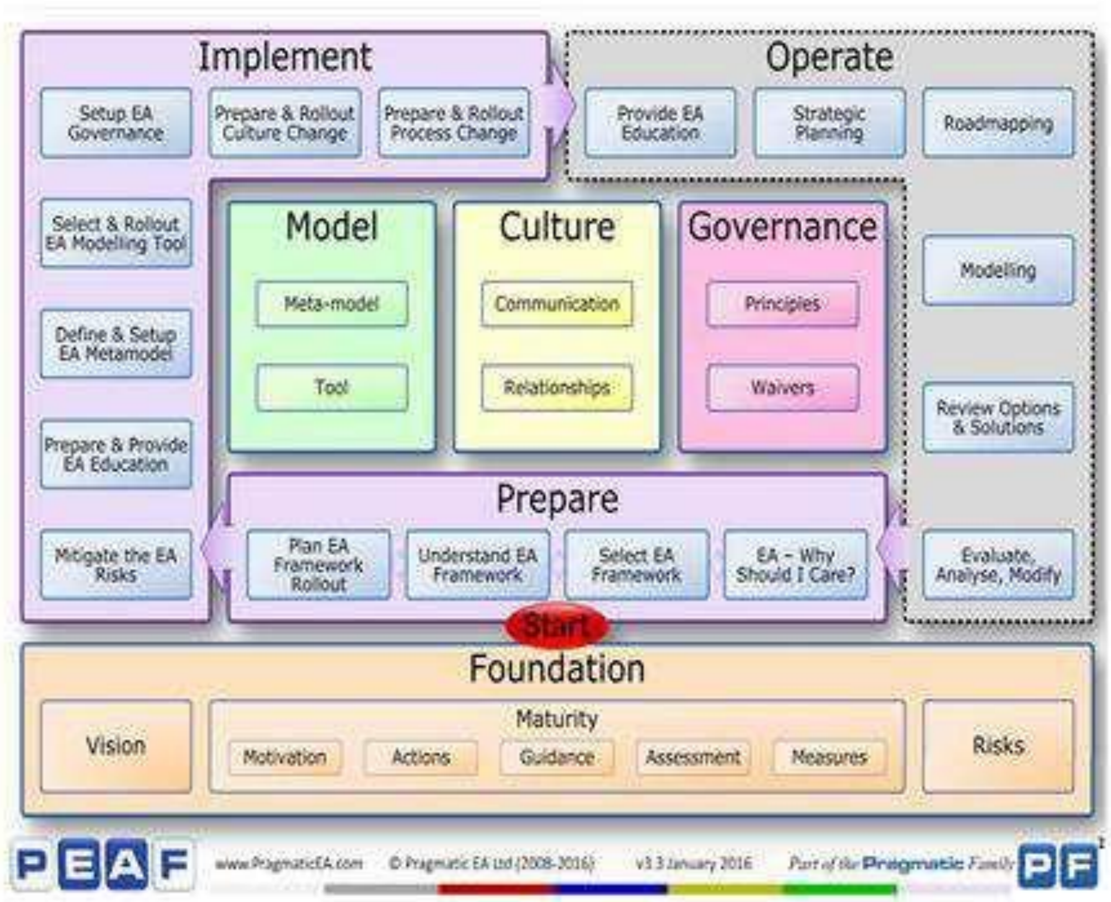


Figure H-10 –PEAF functional areas and foundation [PEAF v3, August 2014]

The following table relates the PEAF actions with the processes defined in this document.

NOTE In this table, "?" occurrences show potential extension of the 42020 scope.

Table H-2 – Mapping of processes to the PEAF framework

		42020 Processes					
PEAF functional domain	**PEAF actions**	Architecture Governance	Architecture Management	Architecture Conceptualization	Architecture Evaluation	Architecture Elaboration	Architecture Enablement
Prepare	Strategizing (Why should I care?)	?					
	Roadmapping (Select EA Framework)						?
	Initiating (Understand EA Framework)		?				
	Elaborating (Plan EA Framework Rollout)		X				
Implement	Constructing (Prepare Culture Change)	X					
	Constructing (Setup EA Governance)	X					
	Constructing (Prepare Process Change)	?					
	Constructing (Prepare Education)	?					
	Constructing (Define & Setup EA Metamodel)			X			
	Constructing (Develop EA Change)			X			
	Constructing (Select EA tools)						X
Operate	Transitioning (Rollout EA Changes: strategic planning; EA roadmapping)	X					
	Transitioning (Provide EA Education)	?					
	Transitioning (Rollout EA Modeling)			X			
	Transitioning (Manage Value and Evolution: Review Options & solutions; Evaluate, Analyze and Modify)				X		

H.3.3 Items in the PEAF framework not addressed in 42020

ISO/IEC 42020 scope does not include rationale data for enterprise architecture setup, culture and education related activities and data, roadmaps, and plans for transitioning the enterprise from baseline to target architectures.

H.3.4 Items in 42020 not addressed in the PEAF framework

PEAF addresses very lightly architecture management and does not include the elaboration activities. The scope is specifically dedicated to enterprises transformation and does not consider architecture/transformation of any king of architecture entity.

NOTE 1 Frameworks like TOGAF can complement PEAF for management of enterprise architectures.

NOTE 2 Frameworks like DoDAF, NAF and UAF can complement PEAF with formalism and architecting concepts for architecture entities.

H.4 GERAM framework

H.4.1 GERAM framework overview

The Generalized Enterprise Reference Architecture and Methodology (GERAM) framework is a generalization of frameworks and defines a number of fundamental concepts that any architecture framework needs to cover. GERAM is lightweight in the sense that it defines placeholders for necessary components, but leaves the population of these to the collective development of the enterprise architecture body of knowledge. For example, GERAM defines the concept of methodologies, but acknowledges that depending on the industry domain and a number of other factors there can be many legitimate and useful methodologies.

The requirements that frameworks need to satisfy are the normative part of ISO15704 (under review as of December 2016), and the GERAM framework is an annex that demonstrates how these requirements can be met. Historically the GERAM framework was developed first, then the normative part of ISO15704 was extracted from it. Similarly, ISO15704 defines enterprise entity life cycle (consisting of phases, each being a set of life cycle processes considering the enterprise entity on a given level of abstraction), and life history (consisting of stages in time, similar to stages in ISO15288): whereupon GERAM defines eight phases, the normative part of ISO15704 only requires that life cycle phases be defined and leaves that subdivision to individual frameworks.

According to GERAM's philosophy and Enterprise is implemented as a socio-technical system of systems, that are embodied in concrete (and sometimes also virtual) enterprise entities, such as business units, corporate headquarters, programs, projects, various entities that implement supporting systems, infrastructure service entities, virtual organizations, networks of organizations, etc.

When the architecture of an enterprise entity is devised, a fundamental (architectural) decision is made about the nature of how the entity implements the system (the way design parameters map to the functions of the system), and the nature (and timing) of this mapping decides various non- functional systemic properties of the implemented system.

GERAM also defines a modeling framework, defining a list of (open ended) 'aspects' (such as functional, information, resource, organizational, economic,…), whereupon each can be populated by various kinds of models, for the purpose of supporting various life cycle processes. We call these 'aspects' here to make this description independent from the outcome of current terminological developments both in ISO15704 and the ISO42000 series of standards.

The modeling framework defined three categories of models: Particular Models (describing an entity of interest), Partial (or Reference-) Models (describing reusable models that can be specialized and instantiated to build Particular models), and Generic Models (describing the semantics of the models populating the first two (this can be done on various levels of formalization, such as illustrated text, meta-models, or formal ontological theories expressed in a suitably selected logic).

Figure 4.1. shows version 1.6.3 (current) and version 2.0 (being part of the 15704 review, therefore it is not yet final) of the GERAM meta-model; the basic difference is the adjustment to reflect the terminological development of the ISO42000 series of standards.

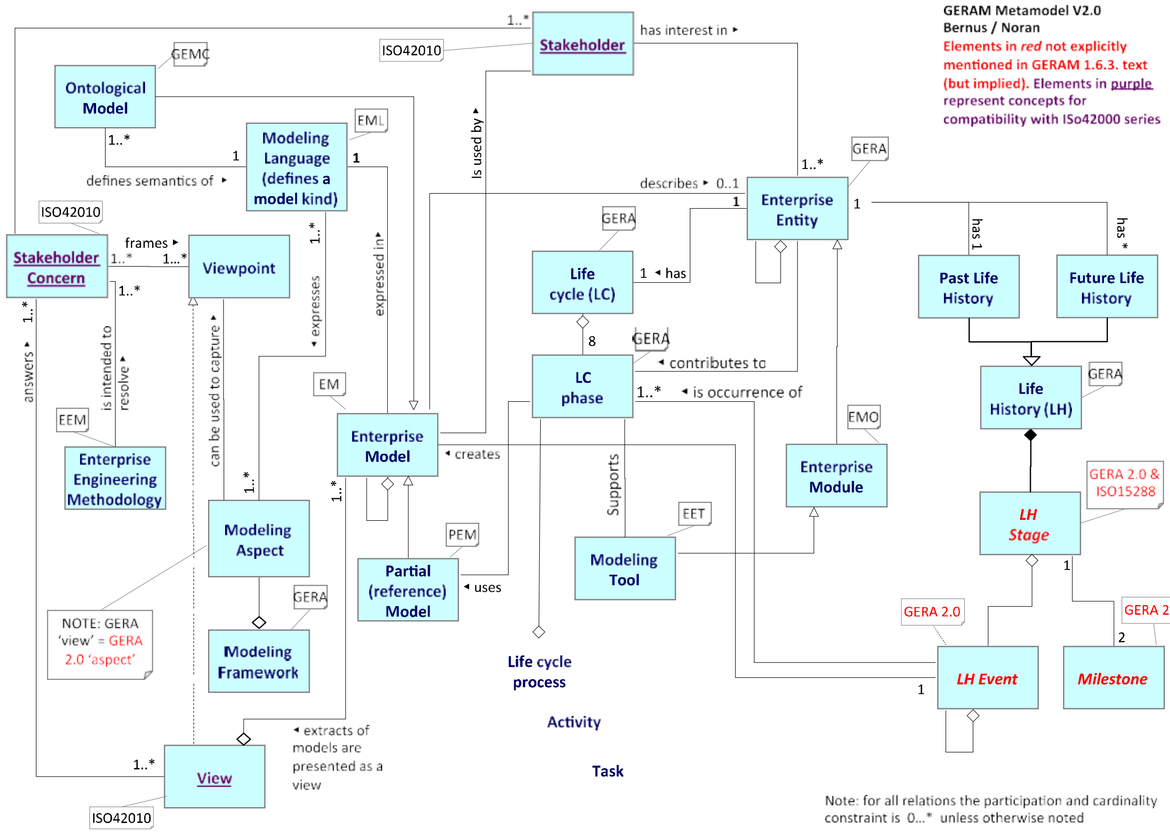
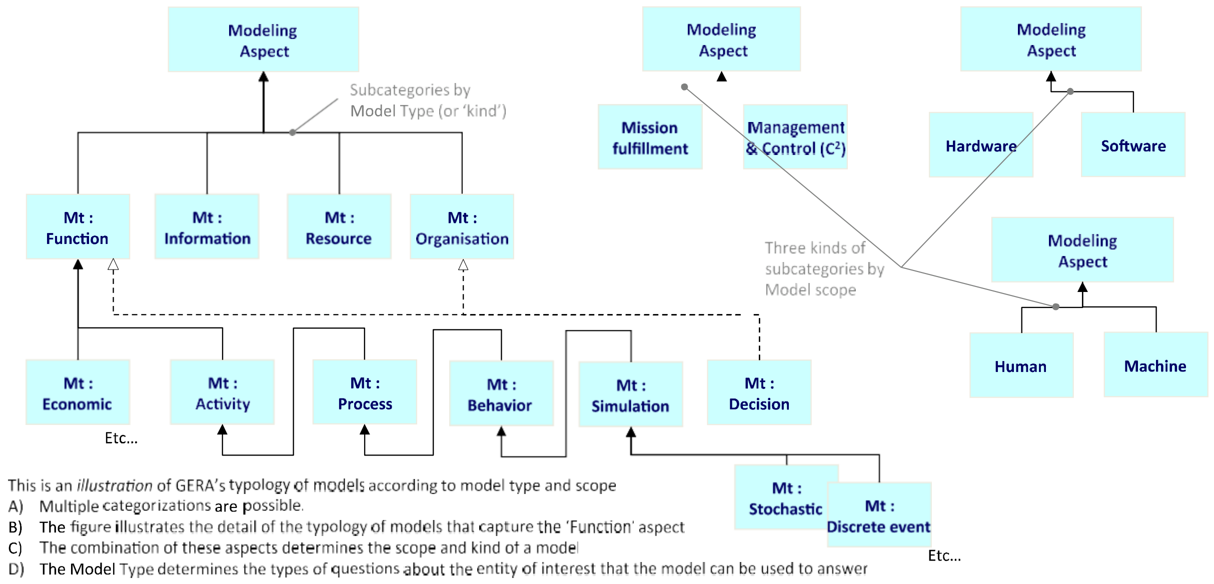


Figure 4.1a GERAM1.6.3 and GERAM2.0 meta-models



This is an *illustration* of GERA's typology of models according to model type and scope

A) Multiple categorizations are possible.
B) The figure illustrates the detail of the typology of models that capture the 'Function' aspect
C) The combination of these aspects determines the scope and kind of a model
D) The Model Type determines the types of questions about the entity of interest that the model can be used to answer

Fig 4.1b GERAM 2.0 Modeling Aspect concept (aspect was called 'view' in v1.6.3) (NB model scope may span multiple enterprise entities)

H.4.2 Mapping 42020 to GERAM framework elements

In order to illustrate the role of ISO42020 Architecture Processes standard (as a reference model) we show in Fig 4.2 a 'dynamic business model' of a typical enterprise (decomposed into its constituent entities). The figure shows enterprise entities, the relationships among their life-cycles, and the role of reference models. Fig. 4.2a shows the case of EA practice adoption, and Fig 4.2b shows EA practice (including architecture processes) in operation.

When describing or (re)designing various entities in the above, a large number of reference models are used, helping to instantiate various (e.g. process-, information-, organizational-, financial-, decisional-, structural- etc.) designs.

Architecture Processes need to be distributed across the entities in question in two senses:

- As part of the introduction of architecture practices, these process reference models need to be adopted as part of a portfolio activity (possibly through an EA maturity improvement program and its projects). The reference models would be adopted, adapted, particularized, and distributed among organizational roles. There exist multiple different ways, depending on whether EA is a centralized function or distributed across business units, supported by some central service, for example. Also, the reference models can either be adopted through policy instruments or through introducing them as standard procedures (depending on the skill levels and experience of the roles among which the processes or parts thereof are distributed). Depending on these decisions the introduction of ISO 42020 Architecture Processes may only have to be defined on the level of the requirements and preliminary design life cycle phases (without any detailed design being necessary as the rest of the details are to be filled by highly skilled personnel who take the respective roles), or additional detailed design level procedures would be defined, which are then followed by personnel filling the respective roles.
- As shown in Fig 4.2a, in this *establishment stage* of EA practice, a typical distribution of Architecture Processes includes activities and tasks being allocated to and rolled out to roles in Corporate Management, in Business Units, as well as in Program management, Project mission fulfillment, Project management, as well as possibly external entities (e.g., consulting and other service providers). As shown in Fig.4.2a, the build life cycle phase is concerned with a) establishing the required human architecting, governance and management skills and competencies (through hiring & training individuals, forming committees, and appointing personnel to roles), b) selecting, commissioning and deploying of tools (in support architecting, modeling, communication, management and governance) and of their respective repositories.

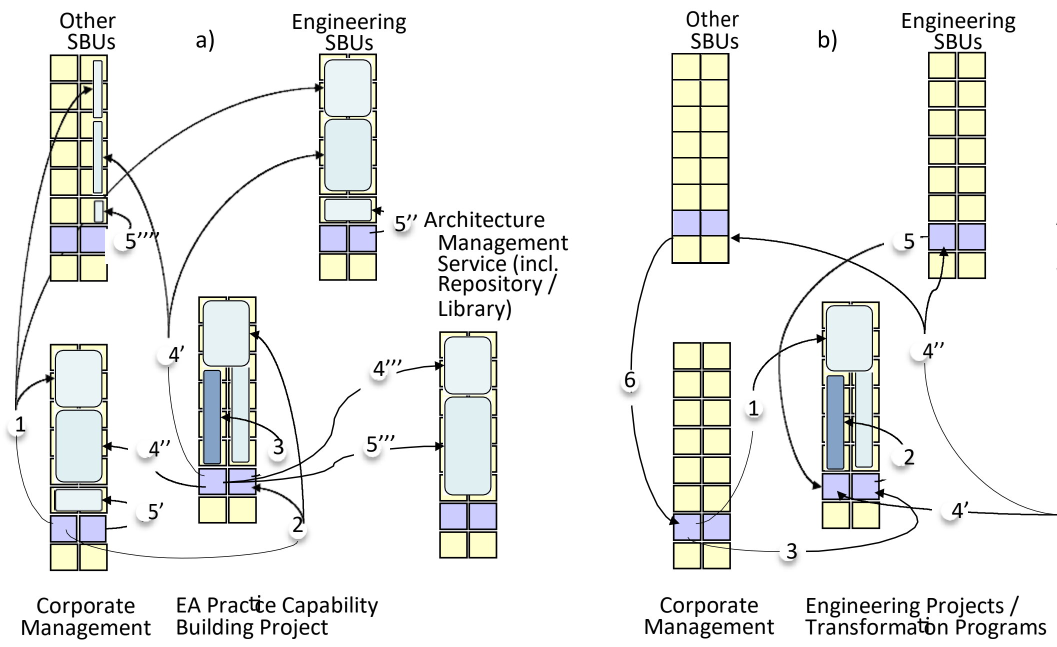

Other SBUs
a)
Engineering SBUs
5'' Architecture Management Service (incl. Repository / Library)
5''''
4'
4'''
1
4''
3
5'''
5'
2
Corporate Management
EA Practice Capability Building Project
Other SBUs
b)
Engineering SBUs
Architecture Management Service (incl. Repository / Library)
5
4''
6
1
2
4'
3
Corporate Management
Engineering Projects / Transformation Programs

Figure 4.2 a) Deploying Architecture Processes (establishment stage), and b) Applying Architecture Processes (operation stage)

The establishment stage uses ISO42020 as a reference model (see Fig 4.2a): (1) Corporate management decides on the need to establish Architecture Processes in its architecture practice, this has consequences to the policies and principles that govern how SBUs (and Corporate management) do business; (2) ~ defines the mandate of an EA Practice Capability Building Project (and in turn participates in project supervisory capacity) and appoints project management; (3) Project management works out the detail of the project; (4'…4''') the project uses 42020 as a reference model to design the changes necessary in Engineering SBUs and other SBUs, as well as specifies the need for an architecture management service entity that incorporates architecture management, library and repository services (including technology and human roles), as well as designs the necessary localized processes and organizational roles necessary to build Architecture Governance; (5'…5'''') respective entities roll out the above, including the commissioning and deployment of the tools that support architecture processes.

As part of (the *operation stage* of) EA practice, Architecture Processes are applied. Fig 4.2b illustrates the typical life cycle relationships through which corporate strategic portfolio managers and transformation programs can exercise direction (by operating governance, management, control, communication & coordination processes). The figure also illustrates the typical use of the architecting processes proper (including architecture development, elaboration and evaluation).

In the operation stage uses the established 42020 processes as follows (see Fig 4.2b):(1) any strategic engineering project, or transformation program's mandate is defined by Corporate Management (through its established architecture governance roles), including the mandate to use architecture definition, elaboration and evaluation processes; (2) Project management defines the details of the project (program), including the use of the so mandated processes.; (3) Corporate Management through its Architecture Governance processes participates in the supervision of projects / programs; (4'…4'') Architecture management services (including architecture management and supporting services for architecture work) provides operational support to all of the other entities involved; (5) Engineering SBUs participate in these projects / programs, and as part of that participation perform architecture definition, elaboration and evaluation processes; (6) interests of other SBUs not involved in architecture work are still represented by contributing to architecture governance.

H.4.3 Items in GERAM not addressed in 42020

ISO 42020 does not explicitly address the introduction of architecture processes into architecture practice (*cf*. Fig 4.2a). It is to be noted, that 42020 is not defining information, organizational, or structural models: it is up to the introduction effort to standardize these (or not, and leave the details to be decided on a case-by-case basis in various transformation projects or programs).

H.4.4 Items in 42020 not addressed in GERAM

The details of the architecture processes described in ISO 42020 (as a reference model) are out of scope of GERAM, because GERAM only stipulates the *need* for reference modes (referring the details to standards such as ISO42020, and other industry reference models); GERAM does not prescribe any preferred set. It is part of the introduction of EA practice to identify, evaluate for suitability, select, and adopt (or adopt with adaptations), and finally deploy process reference models that suit best the characteristics of the given industry and organization.

H.5 RM-ODP framework

H.5.1 Framework overview

Reference Model – Open Distributed Processing (RM-ODP) is a reference model based on precise concepts derived from current distributed processing developments and, as far as possible, on the use of formal description techniques for specification of the architecture. Many RM-ODP concepts, possibly under different names, have been around for a long time and have been rigorously described and explained in exact philosophy and in system-thinking. Some of these concepts—such as abstraction composition, and emergence—have recently been provided with a solid mathematical foundation in category theory.

RM-ODP has four fundamental elements:

- an object-modeling approach to system specification;
- the specification of a system in terms of separate but interrelated viewpoint specifications;
- the definition of a system infrastructure providing distribution transparencies for system applications; and
- a framework for assessing system conformance.

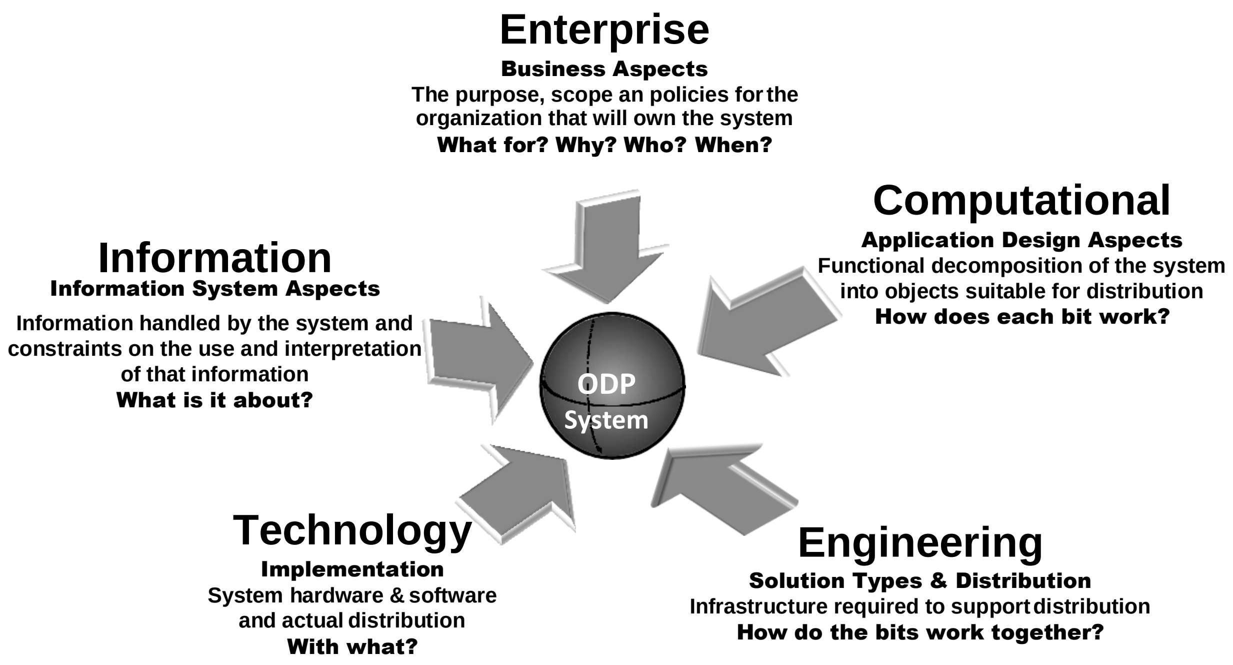

Enterprise
Business Aspects
The purpose, scope an policies for the organization that will own the system
What for? Why? Who? When?
Computational
Application Design Aspects
Functional decomposition of the system into objects suitable for distribution
How does each bit work?
Information
Information System Aspects
Information handled by the system and constraints on the use and interpretation of that information
What is it about?
ODP System
Technology
Implementation
System hardware & software and actual distribution
With what?
Engineering
Solution Types & Distribution
Infrastructure required to support distribution
How do the bits work together?

Figure H-11 – RM-ODP framework

The above figure shows the RM-ODP framework. As shown this, the RM-ODP consists of 5 viewpoints: Enterprise viewpoint, Information viewpoint, Computational viewpoint, Engineering viewpoint, Technology viewpoint. Each viewpoint prescribes its own architecture constituents.

The RM-ODP family of recommendations and international standards defines a system of interrelated essential concepts necessary to specify open distributed processing systems and provides a well-developed enterprise architecture framework for structuring the specifications for any large-scale systems including software systems.

H.5.2 Mapping to framework elements

The RM-ODP is a framework for specification of system's architecture. This doesn't correspond to architecture process perspective directly. Therefore, it is impossible to indicate mapping. However, approximated mapping is illustrated in this section. This mapping is shown in correspondence to RM-ODP viewpoints.

Table H-3 – Mapping of processes to the RM-ODP Framework

RM-ODP viewpoint	42020 Processes					
	Architecture Governance	Architecture Management	Architecture Conceptualization	Architecture Evaluation	Architecture Elaboration	Architecture Enablement
Enterprise viewpoint	X	X	X			
Information viewpoint			X		X	
Computational viewpoint					X	
Engineering viewpoint					X	X
Technology viewpoint					X	X

H.5.3 Items in the RM-ODP Framework not addressed in 42020

The RM-ODP prescribes specification notion/description for systems. Especially, in ISO/IEC 19793, concrete elements of all viewpoints are defined as UML profile. Therefore, System specifications are constructed using these elements in diagrams Furthermore, the RM-ODP represents from the high abstraction specification (Enterprise specification) to detail one (Technology/Engineering specification).

H.5.4 Items in 42020 not addressed in the RM-ODP Framework

ISO/IEC 42020 prescribes the Architecture process for system development. However, the RM-ODP doesn't include development process. Furthermore, the RM-ODP mainly focuses on Enterprise Architecture, on the contrary, ISO/IEC 42020 covers Enterprise/System Architecture.

Bibliography

ISO References:

ISO/IEC/IEEE 12207, Systems and software engineering — Software life cycle processes

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

ISO/IEC 15939, Systems and software engineering -- Measurement process

ISO 15704, Industrial automation systems — Requirements for enterprise-reference architectures and methodologies

ISO 19439, *Enterprise integration—Framework for enterprise modelling*

ISO 21500, *Project, programme and portfolio management — Guidance on project management*

ISO/DIS 21505.2, *Project, programme and portfolio management — Guidance on governance*

ISO/IEC/IEEE TR 24748, *Systems and software engineering -- Life cycle management (series of guides)*

ISO/IEC/IEEE 24765, *Systems and software engineering -- Vocabulary*

ISO/IEC TR 24774, *Systems and software engineering -- Life cycle management -- Guidelines for process description*

ISO/IEC 25010, *Systems and software engineering -- Systems and software Quality Requirements and Evaluation (SQuaRE) -- System and software quality models*

ISO 31000, *Risk management -- Principles and guidelines*

ISO/IEC 33001, *Information technology -- Process assessment -- Concepts and terminology*

ISO/IEC 33002, *Information technology -- Process assessment -- Requirements for performing process assessment*

ISO/IEC 33004, *Information technology -- Process assessment -- Requirements for process reference, process assessment and maturity models*

ISO/IEC 38500, *– Corporate governance of information technology*.

ISO/IEC TR 38502, *Information technology — Governance of IT — Framework and models*.

ISO/IEC/IEEE 42010, Systems and software engineering — Architecture description

ISO/IEC 42030, *Enterprise, systems and software — Architecture evaluation elements (Under development)*.

ISO/IEC 7498-1, *Information technology -- Open Systems Interconnection -- Basic Reference Model: The Basic Model*

Other references:

[ABUSHAREKH 2010]	Abusharekh, A, Gloss, L, Levis, A., "Evaluation of Service Oriented Architecture-Based Federated Architectures," Wiley Online Library (wileyonlinelibrary.com), DOI 10.1002/sys.20162, 26 January 2010
[Alexander, 1964]	Alexander, Christopher, *Notes on the Synthesis of Form,* Harvard University Press, 1964, ISBN 0-674-62751-2
[Ang, 2005]	Huei Wan Ang, Dave Nicholson, and Brad Mercer, "Improving the Practice of DoD Architecting with the Architecture Specification Model," The MITRE Corporation, June 2005, http://www.mitre.org/publications/technical-papers/improving-the-practice-of-dod-architecting-with-the-architecture-specification-model
[Astudillo, 2005]	"Five ontological levels to describe and evaluate software architectures". Hernan Astudillo. In: Revista Facultad de Ingenieria – Universidad de Tarapaca 13.1 (2005), pp. 69–76. url: http://redalyc.uaemex.mx/pdf/114/11413107.pdf
[ATAM]	Kazman, Rick, Mark Klein, Mario R Barbacci, Tom Longstaff, Howard Lipson, and Jeromy Carriere. July 1998. The Architecure Tradeoff Analysis Method. Software Engineering Institute, CMU/SEI-98-TR-008.
[Barbacci, 1999]	Analyzing Quality Attributes. Column in SEI newsletter, Mario R. Barbacci. The Architect. Mar. 1999. url: http://www.sei.cmu.edu/library/abstracts/news-at-sei/architectmar99.cfm [Blevins, 2010] Blevins, Terry, Fatma Dandashi, and Mary Tolbert, 2010, TOGAF ADM and DoDAF Models, The Open Group White Paper.
[Boehm, 1978]	Boehm, B. W., Brown, J. R., Kaspar, H., Lipow, M., McLeod, G., and Meritt, M., Characteristics of Software Quality, Edition 2, North-Holland Pub. Co., 1978[Broy, 2009] Automotive Architecture Framework: Towards a Holistic and Standardised System Architecture Description, An overview on description concepts, models and methods.
[Chen, 2008]	D. Chen, G. Doumeingts, F. Vernadat, Architecture for enterprise integration and interoperability: Past, present and future, appearing in Computers in Industry, 59 (2008) 647-659, Elsevier B.V.
[CJCSI 3170.01H]	Joint Capabilities Integration and Development System (JCIDS) and JCIDS Manual, 10 January 2012. http://www.dtic.mil/cjcs_directives/cdata/unlimit/3170_01.pdf

[CJCSI 6212.01F] Interoperability and Supportability of Information Technology and National Security Systems, 21 March 2012. http://www.dtic.mil/cjcs_directives/cdata/unlimit/6212_01.pdf

[Crnkovic , 2004]	"Classification of Quality Attributes for Predictability in Component-Based Systems". Ivica Crnkovic and Magnus Larsson. In: Journal of Econometrics. 2004, pp. 231–250. url: http://www.mrtc.mdh.se/publications/0710.pdf
[Dijkstra, 1974]	Dijkstra, E. W., On the role of scientific thought (1974), http://www.cs.utexas.edu/users/EWD/transcriptions/EWD04xx/EWD447.html

[DoDI 4630.08, 2004]	Procedures for Interoperability and Supportability of Information Technology (IT) and National Security Systems (NSS), June 30, 2004. http://www.dtic.mil/whs/directives/corres/pdf/463008p.pdf
[ECPD, 1947]	Engineers' Council for Professional Development. (1947). Canons of ethics for engineers
[Eeles, 2010]	The Process of Software Architecting. Peter Eeles and Peter Cripps. Addison Wesley, 2010. URL: http://processofsoftwarearchitecting.com
[Emes, 2012]	M. R. Emes, P. A. Bryant, M. K. Wilkinson, P. King, A. M. James and S. Arnold, Interpreting "systems architecting" (pages 369–395) appearing in Systems Engineering Winter 2012, Volume 15, Issue 4 Article first published online: 16 MAY 2012 | DOI: 10.1002/sys.21202
[FEAPO, 2013]	Cameron, et. al., A Common Perspective on Enterprise Architecture, Architecture & Governance Magazine, Vol 9, No. 4, 2013. http://ea.ist.psu.edu/documents/A&G_Issue9_4-FEAPOcut.pdf
[Harrison , 2011]	"Pattern-Based Architecture Reviews". Neil B Harrison and Paris Avgeriou. In: IEEE Software 28 (2011), pp. 66–71. URL: http://doi.ieeecomputersociety.org/10.1109/MS.2010.156
[INCOSE, 2014]	A world in motion, Vision 2015, International Council on Systems Engineering, 2014 http://www.incose.org/docs/default-source/aboutse/se-vision-2025.pdf
[HFM155, 2008]	The NATO Human View Handbook, NATO RTO HFM-155 Human View Workshop, January 2008
[Hoffman, 2007]	Martin Hoffmann, Analysis of the current State of Enterprise Architecture Evaluation Methods and Practices, Information Technology Research Institute, University of Jyväskylä, Finland https://jyx.jyu.fi/dspace/bitstream/handle/123456789/41367/Article_Analysis_of_the_Current_State_of_EA_Evaluation_Methods_and_Practices.pdf?sequence=4
[Hofmeister, 2007]	Christine Hofmeister et al. "A general model of software architecture design derived from five industrial approaches," The Journal of Systems and Software, 2007
[Kruchten, 1995]	Architectural Blueprints–The "4+1" View Model of Software Architecture, Philippe Kruchten, Paper published in IEEE Software 12 (6), November 1995, pp. 42-50
[Lago, 2010]	Patricia Lago, Paris Avgeriou, and Rich Hilliard. Guest editors' introduction, Software Architecture: Framing Stakeholders' Concerns, IEEE Software 27(6) (November/December 2010), pp. 20–24.
[Lankhorst, 2013]	Enterprise Architecture at work, Marc Lankhorst, Springer (Third Edition)
[Lapkin, 2005]	Gartner's Enterprise Architecture Process and Framework Help Meet 21st Century Challenges. Tech. rep. G00133132. Anne Lapkin. The Gartner Group, Nov. 2005. URL: http://www.gartner.com/resources/133100/133132/gartners_enterprise_architec_133132.pdf
[Lattanze, 2005]	Lattanze, Anthony, 2005, "The Architecture Centric Development Method," Carnegie Mellon University report CMU-ISRI-05-103.

[Li, et. al, 2011]	Performance Evaluation for Industrial Automation System Integration Based on Enterprise Architecture Standards and Application in Cotton Textile Industry, in Proceedings of 2011 International Conference on System Science, Engineering Design and Manufacturing Informatization (ICSEM 2011), IEEE pp 184 – 189.
[Maier, 2009]	Art of Systems Architecting, Mark W. Maier, CRC Press; 3 edition (January 6, 2009)
[Martin, 2004]	R. Martin, E. Robertson, J. Springer, Architecture principles for enterprise frameworks. Technical Report. Computer Science Dept., Indiana Univ., 2004. www.cs.indiana.edu/Research/techreports/TR594.
[MDA 2003]	Overview and guide to OMG's architecture, Model Driven Architecture, OMG, http://www.omg.org/cgi-bin/doc?omg/03-06-01
[Merriam-Webster, 2014]	http://www.merriam-webster.com/dictionary/principium
[Mills, 1985]	John A. Mills. "A pragmatic view of the system architect". Communications of the ACM 28(7) (1985), pp. 708–717.
[MITRE, 2008]	Thoughts on architecture and How to Improve the Practice (Presented to Systems Engineering Colloquium Naval Postgraduate School Monterey, California, Version 3.4), http://www.nps.edu/Academics/Institutes/Meyer/docs/Jan%2031%202008Thoughs_on_Architecting.pdf
[MITRE 2014a]	Systems Engineering Guide, "Approaches to Architecture Development," http://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/system-architecture/approaches-to-architecture-development, MITRE Corp. 2014
[MITRE 2014b]	Systems Engineering Guide, "Architectural Frameworks, Models, and Views," http://www.mitre.org/publications/systems-engineering-guide/se-lifecycle-building-blocks/system-architecture/architectural-frameworks-models-and-views, MITRE Corp. 2014
[Muller, 2011]	System Architecting: A Business Perspective, Gerrit Muller, CRC Press (September 8, 2011)
[NAS, 2013]	Interim Report of a Review of the Next Generation Air Transportation System Enterprise Architecture, Software, Safety, and Human Factors Copyright © National Academy of Sciences, 2013, ISBN 978-0-309-29831-5
[OMG 2014a]	The OMG Hitchhikers Guide, v7.8, omg/2008-09-02
[OMG, 2014b]	Policies and Procedures of the OMG Technical Process, Ver 3.0, pp/12-12-01`
[OMG BMM, 2008]	Business Motivation Model, Version 1.0, OMG Document Number: formal/2008-08-02, standard document URL: http://www.omg.org/spec/BMM/1.0/PDF
[OMG MDA, 2003]	MDA Guide version 1.0.1, omg/03-06-01, June 2003
[Parnell, 2017]	Trade-off Analytics: Creating and Exploring the System Tradespace (Wiley Series in Systems Engineering and Management, January 2017) by Gregory S. Parnell (Editor).

[Parnell, 2013]	Handbook of Decision Analysis (Wiley Handbooks in Operations Research and Management Science, April 2013) by Gregory S. Parnell, Terry Bresnick, Steven N. Tani, and Eric R. Johnson.
[Proper, 2011]	Greefhorst, D., Proper, E. "The Roles of Principles in Enterprise Architecture, http://archixl.nl/files/tear2010_principles.pdf
[RING, 2004]	"An Activity-Based Methodology for Development and Analysis of Integrated DoD Architectures," Steven J. Ring, Dave Nicholson, Jim Thilenius, The MITRE Corporation, Stanley Harris, Lockheed-Martin Corporation, March 2004 http://www.mitre.org/publications/technical-papers/an-activitybased-methodology-for-development-and-analysis-of-integrated-dod-architectures
[SABSA, 2011]	TOGAF® and SABSA® Integration, white paper by The Open Group TOGAF-SABSA Integration Working Group, October 2011
[SARA]	H. Obbink et al. Report on Software Architecture Review and Assessment (SARA), version 1.0. Feb. 2002. url: http://philippe.kruchten.com/architecture/SARAv1.pdf
[SEI, 2009]	U.S. Army Workshop on Exploring Enterprise, System of Systems, System, and Software Architectures, Mike Gagliardi, John Klein, Rob Wojcik, Bill Wood, Technical Report, CMU/SEI-2009-TR-008, The Software Engineering Institute, ESC-TR-2009-008, March 2009. http://www.sei.cmu.edu
[TUM-IBM, 2009]	White Paper of the IBM Corporation and TUM Technical Report, Manfred Broy, Mario Gleirscher, Peter Kluge, Wolfgang Krenzer, Stefano Merenda, and Doris Wild, TUM-I0915, July 2009
[Vitruvius, BC]	De architectura, Marcus Vitruvius Pollio (1st century BC) (Transl. Morris Hicky Morgan, 1960), The Ten Books on Architecture. Courier Dover Publications. ISBN 0-486-20645-9.