INTERNATIONAL STANDARD

ISO/IEC 26552

First edition
2019-05

Software and systems engineering — Tools and methods for product line architecture design

Ingénierie du logiciel et des systèmes — Outils et méthodes pour la conception architecturale des gammes de produits

Reference number
ISO/IEC 26552:2019(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html.

This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html.

Introduction

The main purpose of this document is to deal with the capabilities of methods and tools of architecture design for software and systems product line (SSPL). This document defines how the tools and methods can support for the software and systems product line-specific architecture processes.

Domain architecture provides structures and constraints that govern all the subsequent SSPL lifecycle processes as well as being transferred into the architecture design of a member product at the application design processes. Therefore, SSPL architecture design should be defined in detail, considering constraints, so that other processes have a consistent foundation. Supporting tools and methods of architecture design should consider those engineering processes that use and are affected by architecture design.

Product line architecture design can be differentiated from a single product development because of the following aspects:

— There are two core processes in architecture design: domain and application architecture design. The major aims of the domain architecture design processes are to design architectural structure and texture based on domain requirements which includes commonality and variability for a family of products, and to prepare necessary variability information for variability modelling. On the other hand, the major aims of the application architecture design processes are to derive application architecture through binding and add application-specific architectural structure.

— The outcomes of domain requirements engineering form the basis for product line architecture design and application-specific requirements might compel to add new components or tailor the structure unlike in the case of a single product development.

— The architectural texture, one of the major outcomes of product line architecture design defines common ways to deal with variability in domain realisation as well as in application design and application realisation. Domain realization should adhere to the rules defined in the architectural texture, and application architecture should comply with the rules defined in the architectural texture.

This document can be used in the following modes:

— by the users of this document — to benefit people who conduct domain and application architecture design for software and systems product lines;

— by a product line organization — to provide guidance in the evaluation and selection for methods and tools for domain and application architecture design;

— by providers of methods and tools — to provide guidance in implementing or developing tools and methods by providing a comprehensive set of the capabilities of tools and methods for domain and application architecture design.

The ISO/IEC 26550 family of standards addresses both engineering and management processes and capabilities of methods and tools in terms of the key characteristics of product line development. This document provides processes and capabilities of methods and tools for domain design and application design. Other standards in the ISO/IEC 26550 family are as follows:

ISO/IEC 26550, ISO/IEC 26551, ISO/IEC 26552, ISO/IEC 26554, ISO/IEC 26555, ISO/IEC 26556, ISO/IEC 26557, ISO/IEC 26558 and ISO/IEC 26559 are published. ISO/IEC 26560, ISO/IEC 26561 and ISO/IEC 26562 are to be published. ISO/IEC 26563 is a planned International Standard.

— Processes and capabilities of methods and tools for domain requirements engineering and application requirements engineering are provided in ISO/IEC 26551;

— Processes and capabilities of methods and tools for domain realization and application realization are provided in ISO/IEC 26553;

— Processes and capabilities of methods and tools for domain testing and application testing are provided in ISO/IEC 26554;

— Processes and capabilities of methods and tools for technical management are provided in ISO/IEC 26555;

— Processes and capabilities of methods and tools for organizational management are provided in ISO/IEC 26556;

— Processes and capabilities of methods and tools for variability mechanisms are provided in ISO/IEC 26557;

— Processes and capabilities of methods and tools for variability modelling are provided in ISO/IEC 26558;

— Processes and capabilities of methods and tools for variability traceability are provided in ISO/IEC 26559;

— Processes and capabilities of methods and tools for product management are provided in ISO/IEC 26560;

— Processes and capabilities of methods and tools for technical probe are provided in ISO/IEC 26561;

— Processes and capabilities of methods and tools for transition management are provided in ISO/IEC 26562;

— Processes and capabilities of methods and tools for configuration management of asset are provided in ISO/IEC 26563;

— Others (ISO/IEC 26564 to ISO/IEC 26599): To be developed.

Software and systems engineering — Tools and methods for product line architecture design

1 Scope

This document, within the context of methods and tools for architecture design for software and systems product lines:

— defines processes and their subprocesses performed during domain and application architecture design. Those processes are described in terms of purpose, inputs, tasks and outcomes;

— defines method capabilities to support the defined tasks of each process;

— defines tool capabilities to automate/semi-automate tasks or defined method capabilities.

This document does not concern processes and capabilities of tools and methods for a single system but rather deals with those for a family of products.

2 Normative references

There are no normative references in this document.

3 Terms and Definitions

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

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

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

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

3.1
application architecture
architecture concept, including the architectural structure and rules (e.g. common rules and constraints), and architecture artefacts (such as descriptions) that constrains a specific member product within a product line

3.2
architectural texture
texture
collection of common development rules, guidelines and constraints that deals with common and variable aspect of the *product line architecture* (3.8)

3.3
architecture structure
physical or logical layout of the components of a system design and their internal and external connections

Note 1 to entry: Annex C provides an example of architecture structure and *texture* (3.2).

3.4
aspect
special consideration within product line engineering process groups and tasks to which one can associate specialized methods and tools

3.5
domain architecture
common architecture for a product line that can embrace *variability* (3.10) of member products

3.6
external variability
variability (3.10) that is visible to customers

3.7
internal variability
variability (3.10) that is hidden from customers

3.8
product line architecture
architecture, including both *domain architecture* (3.5) and *application architecture* (3.1)

3.9
reference architecture
core architecture that captures the high-level architecture concept of *domain architecture* (3.5) and *application architecture* (3.1)

3.10
variability
characteristics that can differ among member products of a product line

4 Reference model for product line architecture design

4.1 Overview

Product line architecture design consists of two development life cycle processes, domain design and application design. Domain design develops the domain architecture, including architectural structure and texture that enables domain realization and all member products within a product line. The domain architecture captures the high-level design of a product line, including the external and internal variabilities defined in domain requirements and architecture design as well as commonalities. Most of internal variabilities due to the chosen technical solutions are defined during architecture design, and external variabilities are refined into internal variabilities if necessary. The domain architecture should be valid for those variabilities.

Application design derives the application architecture from the domain architecture in accordance with application requirements and decisions for technical solutions. Member-product specific adaptations shall be done to satisfy member product specific requirements. The adaptations should adhere to the texture of the domain architecture.

NOTE 1 Annex A describes the cross-reference with ISO/IEC/IEEE 42010, ISO/IEC/IEEE 42020 and ISO/IEC/IEEE 15288.

The reference model specifies the structure of supporting processes and subprocesses for product line architecture design. The reference model for product line architecture design in Figure 1 is structured into five processes, architecture management, domain design, variability management in design, asset management in design and application design.

Each process is divided into subprocesses and each subprocess is described in terms of the following attributes:

— the title of the subprocess;

— the purpose of the subprocess;

— the inputs to produce the outcomes;

— the tasks to achieve the outcomes;

— the outcomes of the subprocess;

— the capabilities of methods and tools required for performing the tasks effectively and efficiently.

NOTE 2 When the process, subprocess, outcomes and tasks are listed or described in a sentence they are italicized in order to get them noticed.

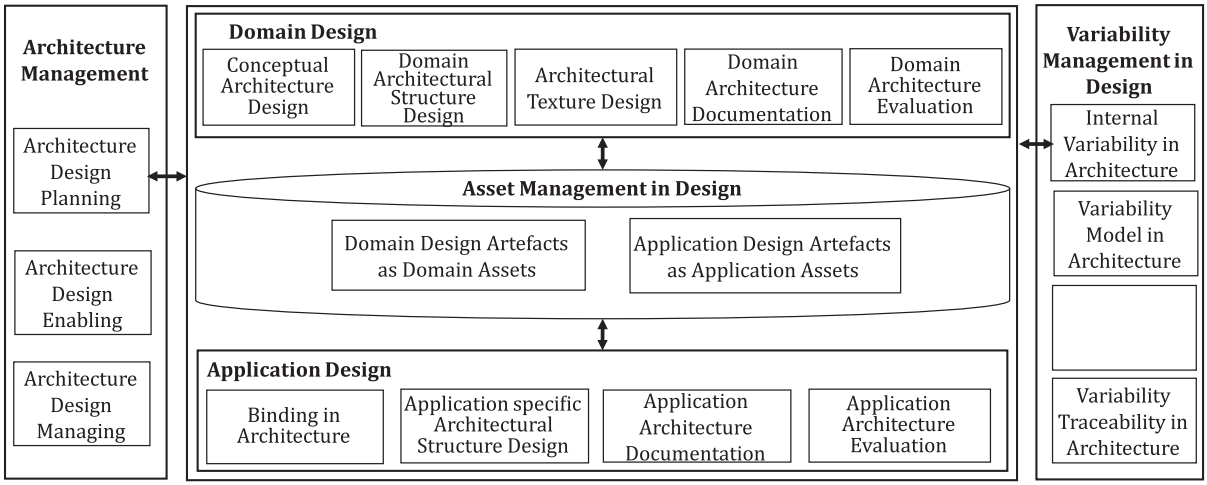

Figure 1 — Product line architecture design reference model

4.2 Architecture management

The architecture management shall serve to do the following and to define capabilities of tools and methods for supporting them:

— *Architecture design planning* makes and maintains overall plans, including guidance for making well-aligned domain and application architecture design with product line objectives. This subprocess also defines schedules and required resources for architecture design;

— *Architecture design enabling* establishes environments required for domain architecture design, for deriving architectures for member products from the domain architecture that complies with the architectural texture, and for adding new components satisfying member product specific requirements. Enabling environment includes organizational structure, technologies and tools that support product line architecture design;

— *Architecture design managing* monitors and controls the status of product line architecture design for harmonizing works conducted in other domain engineering and application engineering processes. It aims at managing domain architecture and maintenance for the relevant artefacts undergoing evolutions.

4.3 Domain design

The domain design develops a domain architecture that enables the realization of the planned commonality and variability within a product line. The aim of the domain design is to produce a domain architecture including architectural textures. The domain architecture reflects additional or refined internal variability introduced by the technical solution. The domain design is divided into five subprocesses and shall serve to do the following and to define capabilities of tools and methods for supporting them:

— *Conceptual architecture design* sets up high-level domain architecture design taking into account the commonality and variability in requirements without going into implementation details; it confirms the readiness for performing domain architecture design;

— *Domain architectural structure design* produces a domain architecture that decomposes a product line into its key components and their relationships. Analysis and modelling should be conducted to assure that the domain architecture covers all commonality and variability in requirements and that added or refined internal variability is covered;

— *Architectural texture design* defines constraints and rules called the architectural texture that can guide architecture designers and domain engineers in evolving a product line over time without destroying its key architectural concepts. The architectural texture deals with specific situations that can occur during architecture design, realization and testing;

— *Domain architecture documentation* describes domain architectural decisions, structure and textures so that domain realization, domain testing and application architecture design use the documentation as the basis for performing their tasks;

— *Domain architecture evaluation* reviews the domain architectural structure and texture for assuring that the domain architecture meets functional and non-functional (quality) requirements.

4.4 Asset management

The asset management in design shall serve to do the following and to define capabilities of methods and tools for supporting them:

— *Domain design artefacts as domain assets* supports member products of a product line to develop their architectures, including architectural specifications using the reusable architecture design artifacts by processing them as domain assets. Domain architecture design artefacts that will be reused by the product line members includes architecture specification or architecture descriptions (described using architecture description languages). Reusable domain artefacts such as components, interfaces and test cases (for integration testing) are also selected and managed as domain assets;

— *Application design artefacts as application assets* supports member products to manage their reusable application specific architecture design artefacts for further reuse. Application specific components, interfaces and test cases (for integration testing) are selected and managed along with the application specific architecture.

4.5 Variability management in design

Some external variabilities are refined into several internal variabilities, and many internal variabilities are newly introduced as well during architecture design. Thus, variability modelling, tracing and supporting mechanisms should be carefully managed in architecture design stage. The variability management in design shall serve to do the following and to define capabilities of methods and tools for supporting them:

— *Internal variability in domain architecture* refines external variability into internal variability at the architecture level and defines technical issues as internal variability. Most internal variabilities of a product line are refined and newly introduced at the architecture level;

— *Variability model in architectur*e integrates internal variability introduced in the domain architecture with the variability model in requirements and explicitly represent and document the results, including the detailed binding information;

— *Variability mechanism in architecture* deals with the capabilities to implement variants in architecture;

— *Variability traceability in architecture* establishes and maintains trace links between variability and architecture including trace links between different abstraction levels of variability.

4.6 Application design

The application design derives the application architecture from the domain architecture based on the application requirements. Product-specific adaptations are then built as far as the texture allows. The application architecture should be consistent with the domain architecture so as to enable the reuse of domain artefacts. Application design shall serve to do the following and to define capabilities of methods and tools for supporting them:

— *Binding in architecture* binds the variants for the variation points of the domain architecture according to the architectural texture and binding decisions. Through binding, variation points are substituted with the selected variants and domain components of the domain architecture will be selected or unselected; it confirms the readiness for performing the application specific architecture design;

— *Application specific architectural structure design* is conducted for covering application specific requirements to parts of the architecture. A large part of the application architecture is derived from the domain architecture, but application specific requirements have to be designed by the application architect;

— *Application architecture documentation* describes application specific architectural decisions and textures so that application realization and testing use the documentation as the basis for performing their tasks;

— *Application architecture evaluation* is conducted only for the application specific architecture. However, the pre-evaluated parts of the architecture during domain design should be assessed in order to confirm their integrity with the bound variants.

The identification and analysis of the key differentiators between single-system engineering and management and product line engineering and management can help organizations to understand the product line and to formulate a strategy for successful implementation of product line engineering and management. The key aspects have been defined in ISO/IEC 26550 and Table 1 shows the category of the key aspects.

Table 1 — Key aspects for identifying product line-specific architecture design tasks

Category	Aspects
Reuse management	application engineering, domain assets, domain engineering, product management, platform, reusability
Variability management	binding, variability
Complexity management	collaboration, configuration, enabling technology support, reference architecture, texture, traceability
Quality management	measurement and tracking, cross functional verification and validation

The following are the descriptions for each aspect of Table 1 concerning domain and application architecture design. The domain and application architecture design relevant processes and tasks shall be identified on the basis of these aspects. The concerns specific to domain and application architecture design enable an organization to understand domain and application architecture design relevant processes, subprocesses, tasks, methods and tools' capabilities:

— Application engineering: Application architecture design is the stage of application engineering in which the domain architecture and selected reusable components and interfaces are reused. Application-specific architectural element should be developed and integrated. The provision of systematic reusability is a key aspect peculiar to product line development.

— Binding: Application architects bind the variation points and variants in domain assets (i.e. domain architecture, selected reusable components and interfaces) according to the application variability models, so processes, methods and tools of architecture design for product lines should contain capabilities for binding.

— Collaboration: Domain requirements engineering and domain architecture design should be performed in a closely related and iterative way. Hence, requirements engineers and architects should collaborate for determining right technical solutions and for producing a well-structured domain architecture.

— Configuration: Since selected components and interfaces of the domain architecture should be reused in application architecture and they should be realized by the subsequent phases in accordance with the defined architectural texture, configurations of domain assets should be consistently managed.

— Domain asset: Domain design assets such as domain architecture, selected components and selected interfaces that will be reused in the application design and realization are the major domain assets of product line architecture, which distinguishes architecture design of the product line development from that of single product development.

— Domain engineering: Domain engineering processes, which include domain design, are product line-specific aspects that do not exist in single product development.

— Enabling technology support: Enabling technologies for supporting efficient reuse and management of variability and assets distinguish architecture design for a product line from single product development.

— Measurement and tracking: Reusability of components and interfaces selected during domain and application design should be properly measured and tracked.

— Platform: Domain architecture design enables the development of a platform and the development of application architecture based on the platform.

— Product management: The domain architecture should evolve in accordance with the evolution and changes of market situations and/or technology trends.

— Reference architecture: The reference architecture is the major outcome from the domain design process.

— Reusability: The domain architecture should be developed using available or generic artefacts as the basis for application architecture. Achieving desired level of reusability in a domain architecture is a key aspect peculiar to product line development.

— Texture: The architectural texture should be defined in necessary detail so that domain realization, domain testing and application design activities follow the constraints and rules defined in the texture.

— Traceability: Traceability among requirements, architecture and realization for domain and application should be maintained and managed.

— Cross functional validation and verification: Detailed processes for validation and verification for domain and application architecture as well as textures should be performed.

— Variability: Internal variability as well as external variability should be modelled and managed. Variability modelling and management are distinguished aspects of product line development.

5 Architecture management

5.1 General

Architecture management supports the following subprocesses:

— *Architecture design planning*;

— *Architecture design enabling*;

— *Architecture design managing*.

5.2 Architecture design planning

5.2.1 Principal constituents

5.2.1.1 Purpose

The purpose of this subprocess is to establish and maintain overall plans, including key procedures, schedules and required resources for making well-aligned domain and application architecture design, complying with architecture design needs and goals.

5.2.1.2 Inputs

The following inputs should be available to perform the architecture design planning process:

— Domain requirements specification.

— Product line objectives.

— Historical documents and data related to architecture design planning.

5.2.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the architecture design planning process:

— *Plans for architecture managing, enabling and designing* are established.

— *Plans* are documented.

— *Changes of plans* are traced and maintained.

5.2.1.4 Tasks

The organization shall implement the following tasks with respect to the architecture design planning process:

— *Establish architecture design goals*: Establish quality goals that the product line architecture should achieve.

— *Define key procedures for architecture design*: Define procedures uniformly used for overall product line architecture design.

— *Define schedules and required resources for architecture design*: Define schedules, milestones and resources for designing architecture in a product line.

— *Specify how to monitor, measure and control the effectiveness of architecture design*: Define procedures, measures and criteria for governing the monitoring, measuring and controlling of the effectiveness of architecture design activities.

— *Document the architecture design plan*: Document the architecture design plan and obtain approvals for it.

5.2.2 Establish architecture design goals

The goal of this task is to establish quality goals of the product line architecture achieved by the product line architecture. Architecturally significant and common requirements to all member products are identified, so that they should be addressed through the product line architecture.

The method should support establishing architecture design goals with the following capabilities:

— identifying common or variable architecturally significant requirements;

— identifying conflicts between requirements for different member products;

— defining design goals that should be achieved through the product line;

— establishing quality goals that should be achieved through the domain architecture.

A tool should support establishing architecture design goals by allowing the user to do the following:

— access the domain requirements specification;

— support variable requirements conflicts resolution among different member products;

— communicate and maintain design goals and quality goals.

5.2.3 Define key procedures for architecture design

The goal of this task is to define uniform procedures for the product line architecture design and to share the defined procedures at the application specific architecture design stage.

The method should support defining key procedures for architecture design with the following capabilities:

— defining procedures for determining the use of the platforms;

— defining procedures obeyed by member products when application architects add application-specific variants;

— defining procedures that guides the collaboration activities between the domain and application architects.

A tool should support defining key procedures for architecture design by allowing the user to do the following:

— access the existing procedure definition for architecture design;

— discriminate common and variable procedures in the defined key procedures;

— share key procedures with relevant architecture design participants.

5.2.4 Define schedules and required resources for architecture design

The goal of this task is to define schedules, milestones and resources for designing the domain architecture and for designing the application architectures of member products.

The method should support defining schedules and required resources for architecture design with the following capabilities:

— estimating efforts required for domain architecture design;

— estimating efforts required to make the domain architecture reusable;

— estimating efforts required for binding and application specific architecture design for member products;

— determining schedules and resources needed for domain and application architecture design;

— assessing the accuracy of each estimate;

— identifying risk management items in domain and application architecture design.

A tool should support defining schedules and required resources for architecture design by allowing the user to do the following:

— access history data related to efforts for architecture design;

— (semi-)automate estimation, scheduling, and resource allocation;

— integrate the estimated results of domain and application architecture design.

5.2.5 Specify how to monitor, measure and control the effectiveness of architecture design

The goal of this task is to define procedures, measures and criteria for governing the monitoring, measuring and controlling of the effectiveness of architecture design activities.

The method should support specifying how to monitor, measure and control the effectiveness of architecture design with the following capabilities:

— defining measures, procedures and criteria for monitoring, measuring and controlling the effectiveness of architecture enabling;

— defining measures, procedures and criteria for monitoring, measuring and controlling the effectiveness of architecture managing;

— defining measures, procedures and criteria for monitoring, measuring and controlling the effectiveness of domain and application architecture designing;

— defining measures, procedures and criteria for monitoring, measuring and controlling the effectiveness of asset management in architecture;

— defining measures, procedures and criteria for monitoring, measuring and controlling the effectiveness of variability management in architecture.

A tool should support specifying how to monitor, measure and control the effectiveness of architecture design by allowing the user to do the following:

— access history data related to architecture design;

— document the defined measures, procedures and criteria;

— monitor actions (What, Who, Where, When, How) and collect data;

— measure and analyse actions and data;

— control actions (What, Who, Where, When, How) and obtain feedbacks from the controls.

5.2.6 Document the architecture design plan

The goal of this task is to document the architecture design plan and obtain approvals for the architecture design plan.

The method should support documenting the architecture design plan with the following capabilities:

— providing documentation templates;

— integrating artefacts produced during previous tasks;

— formulating architecture design plans;

— sharing architecture design plans with the corresponding user.

A tool should support documenting the architecture design plan by allowing the user to do the following:

— document the architecture design plan according to templates;

— establish communication channels among domain and application architecture design participants;

— establish mechanisms necessary for plan versus actual analysis.

5.3 Architecture design enabling

5.3.1 Principal constituents

5.3.1.1 Purpose

The purpose of this subprocess is to develop, deploy and improve enabling capabilities and resources for architecture design, description, documentation and management.

5.3.1.2 Inputs

The following inputs should be available to perform the architecture design enabling process:

— Plans for architecture managing, enabling, and designing.

— Current enabling capabilities and resources.

— Current improvement endeavours in enabling capabilities and resources.

5.3.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the architecture design enabling process:

— *Architecture enablers and resources* are deployed.

— *Required improvements for architecture enablers and resources* are assessed and activated.

5.3.1.4 Tasks

The organization shall implement the following tasks with respect to the architecture design enabling process:

— *Prepare for the architecture enablement*: Get capabilities and resources ready for managing and designing domain and application architecture.

— *Develop and establish enabling capabilities and resources*: Develop capabilities and resources that leverage for achieving architecture design goals derived from product line objectives.

— *Deploy capabilities and resources for architecture enablement*: Deploy capabilities and resources for architecture enablement appropriate to domain and application architecture design.

— *Improve architecture enablement capabilities and resources*: Examine gaps between deployed and required enablers and provide the improved enablers.

5.3.2 Prepare for the architecture enablement

The goal of this task is to identify and define enabling capabilities and resources including the proper roles and responsibilities for participants involved in architecture enablement for managing and designing domain and application architecture.

NOTE ISO/IEC/IEEE 42020 defines that enabling capabilities include such things as procedures, methods, tools, frameworks, architecture viewpoints, work product templates, and reference models. Enabling resources include such things as architecture repository, architecture library, architecture registry, communication channels and mechanisms, and licenses for tools and methods.

The method should support preparing for the architecture enablement with the following capabilities:

— identifying enabling capabilities and resources suitable for designing the domain architecture being used by multiple products of a product line;

— identifying enabling capabilities and resources suitable for implementing variability in architecture;

— identifying enabling capabilities and resources suitable for reusing the domain architecture for configuring an architecture for each member product;

— identifying enabling capabilities and resources suitable for tailoring the configured architecture for the specific requirements;

— identifying enabling capabilities and resources suitable for improving and evolving the domain architecture.

A tool should support preparing for the architecture enablement by allowing the user to do the following:

— integrate required enabling capabilities and resources;

— perform mapping actual versus required enabling capabilities and resources;

— recode the current status of enabling capabilities and resources;

— identify the gaps of enabling capabilities and resources that should be prepared.

5.3.3 Develop and establish enabling capabilities and resources

The goal of this task is to develop and establish capabilities and resources that leverage for achieving architecture design goals derived from product line objectives.

The method should support developing and establishing enabling capabilities and resources with the following capabilities:

— developing capabilities and resources for monitoring, assessing and controlling whether the architecture managing activities follow architecture management and variability management directives;

— developing capabilities and resources for designing the domain architecture that supports all products in the product family and necessary variability management of the product family;

— developing capabilities and resources for configuring application architecture based on the domain architecture;

— developing capabilities and resources for managing assets and its variabilities.

A tool should support developing and establishing enabling capabilities and resources by allowing the user to do the following:

— gauge the level of enabling capabilities and resources;

— decide whether the provided enabling capabilities and resources are sufficient or not;

— implement enabling capabilities and resources that will utilize information technology services.

5.3.4 Deploy capabilities and resources for architecture enablement

The goal of this task is to deploy developed and established capabilities and resources for architecture enablement appropriate to domain and application architecture design processes for leveraging the achievement of architecture design goals.

The method should support deploying capabilities and resources for architecture enablement with the following capabilities:

— deploying capabilities and resources for enabling architecture design to the right roles and tasks;

— establishing communication channels among enablers deployed domain and application architecture design, variability management and asset management.

A tool should support deploying capabilities and resources for architecture enablement by allowing the user to do the following:

— perform correct and rapid deployment of capabilities and resources to do domain architecture design and management;

— share capability and resource among domain and application architecture-relevant roles for enhancing reusability;

— establish communication channels and mechanisms among the related domain and application participants.

5.3.5 Improve architecture enablement capabilities and resources

The goal of this task is to examine gaps between deployed and required enablers, and to provide the improved enablers.

The method should support improving architecture enablement capabilities and resources with the following capabilities:

— collecting data for evaluating the effectiveness of architecture enablement capabilities and resources from designing and managing activities;

— analysing required improvements for architecture enablement capabilities and resources;

— establishing action plans and success measures for improving architecture enablement capabilities and resources.

A tool should support improving architecture enablement capabilities and resources by allowing the user to do the following:

— accumulate data related to enabling capabilities and resources from domain and application architecture design and management activities;

— use a trace board for tracing plan versus actual status in the course of improvement initiatives.

5.4 Architecture design managing

5.4.1 Principal constituents

5.4.1.1 Purpose

The purpose of this subprocess is to monitor and control the status of domain and application architecture design for harmonizing works conducted in other domain engineering and application engineering processes for managing domain architecture and maintenance along with product line evolutions.

5.4.1.2 Inputs

The following inputs should be available to perform the architecture design managing process:

— Outcomes of architecture design planning.

— Outcomes of architecture design enabling.

— Outcomes of domain architecture design.

— Outcomes of application architecture design.

— Outcomes of asset management.

— Outcomes of variability management.

5.4.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the architecture design managing process:

— *Management directives of architecture* are documented.

— *Architecture management plans* are documented.

— *Monitoring and controlling status of architecture management* are documented.

— *Decisions of architecture management* are documented and traced for the fulfilment or execution.

5.4.1.4 Tasks

The organization shall implement the following tasks with respect to the architecture design managing process:

— *Prepare for the architecture management execution*: Get ready for managing architecture design.

— *Implement the architecture management plans*: Perform architecture management activities in accordance with the plans.

— *Close and prepare for the architecture management plan change*: Finish the currently used architecture management plan and get ready to change the plan for the next iteration.

5.4.2 Prepare for architecture management execution

The goal of this task is to develop architecture management directives for making the domain architecture used by several member products and for deriving application architecture by reusing the domain architecture, and thereafter prepare the architecture management.

The architecture management effort includes objectives, measurements, people, resources, processes, tools and technologies for management execution.

The method should support preparing for architecture management execution with the following capabilities:

— defining architecture management objectives, tools and technologies for ensuring the right development and reuse of architecture at domain and application architecture design stage;

— formalizing separated management roles and responsibilities in domain and application architecture;

— defining a way to integrating the separated domain and application architecture management execution;

— documenting architecture management plans.

A tool should support preparing for architecture management execution by allowing the user to do the following:

— integrate management efforts required for domain and application architecture design activities;

— document architecture management plans;

— disseminate architecture management plans and separated architecture management roles and responsibilities.

5.4.3 Implement the architecture management plans

The goal of this task is to perform architecture management activities according to the different roles, responsibilities and directives of the domain and application architecture management defined in plans.

The method should support implementing the architecture management plans with the following capabilities:

— Assigning roles, responsibilities and tasks as defined in the architecture management plans;

— monitoring and measuring the execution of architecture management plans as a whole;

— analysing measurement results for improving domain and application architecture management activities, artefacts and information items.

A tool should support implementing the architecture management plans by allowing the user to do the following:

— assign roles, responsibilities and tasks to stakeholders as defined in the architecture management plans;

— perform status monitoring (e.g. status chart) for architecture management;

— record management results and anomalies encountered during domain architecture and application architecture management.

5.4.4 Close and prepare for the architecture management plan change

The goal of this task is to close the currently progressing architecture management endeavour and identify changes to be made in the next iteration of architecture management plan. Lessons learned during management for the domain architecture are the major causes of changes.

The method should support closing and preparing for the architecture management plan change with the following capabilities:

— integrating lessons learned during architecture management;

— analysing the rationale for changing architecture management plan;

— preparing for architecture management plan change in accordance with architecture evolution.

A tool should support closing and preparing for the architecture management plan change by allowing the user to do the following:

— access change history of architecture management plan;

— store lessons learned into the permanent storage for the further use.

6 Domain design

6.1 General

Domain design supports the following subprocesses:

— *Conceptual architecture design*;

— *Domain architectural structure design*;

— *Architectural texture design*;

— D*omain architecture documentation*;

— *Domain architecture evaluation.*

6.2 Conceptual architecture design

6.2.1 Principal constituents

6.2.1.1 Purpose

The purpose of this subprocess is to set up high-level domain architecture design taking into account the commonality and variability in requirements without going into implementation details. It confirms the readiness for performing domain architecture design.

6.2.1.2 Inputs

The following inputs should be available to perform the conceptual architecture design process:

— Product management documents, including market definition for a product family, technology scanning for a product family, product family definition, trade-off analysis for a product family and product family evolution.

— A set of architecturally significant functional requirements that is common to all member products and variable to certain member products that possess strategic importance.

— A set of architecturally significant quality requirements that is common to all member products and variable to certain member products that possess strategic importance.

6.2.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the conceptual architecture design process:

— *Domain architecture conceptualization information, data and supporting materials* are produced.

— *Conceptual architecture for the product line* is developed.

6.2.1.4 Tasks

The organization shall implement the following tasks with respect to the conceptual architecture design process:

— *Analyse problem space of the domain architecture*: Understand the area of interest for finding solutions.

— *Synthesize potential solution alternatives*: Develop alternative solutions for a product line.

— *Formulate potential domain architecture(s)*: Develop the initial domain architecture, including principles, guidelines, protocols and standards and high-level architectural elements.

— *Capture domain architecture concepts and properties*: Describe the conceptual domain architecture with the relevant viewpoint, views, development models and variability models.

— *Hand off conceptualized domain architecture to users and other stakeholders*: Deliver the conceptual domain architecture to participants of subsequent domain and application subprocesses.

6.2.2 Analyse problem space of the domain architecture

The goal of this task is to identify and understand the context of a domain, common and variable requirements, common and variable quality attributes and their complexities in finding solutions to determine gaps or shortfalls or restrictions in addressing problems. Objectives, scopes, common and variable requirements that should be addressed by the domain architecture are carefully examined.

The method should support analysing problem space of the domain architecture with the following capabilities:

— reviewing key drivers (reusability, variability and variability mechanisms can be SSPL specific key drivers) of the domain architecture;

— classifying firm and evolving set of architecturally significant functional requirements that are common to all member products and those that are variable to certain products;

— classifying firm and evolving set of architecturally significant quality requirements that are common to all member products and those that are variable to certain products;

— identifying conflicts among architecturally significant requirements that are variable to certain products;

— defining technical difficulties expected to cover variabilities in the domain architecture.

A tool should support analysing problem space of the domain architecture by allowing the user to do the following:

— access information about markets, technologies, competitors, organizational strategies and so on;

— access the functional and quality requirements specification, including relevant detailed information for classification;

— organize common and variable requirements and their relevant information for conflict analysis;

— access technical requirements, technical readiness level and lessons learned in the single system development or previous SSPL;

— estimate technical difficulties that is possible to be brought up for implementing variability in the domain architecture.

6.2.3 Synthesize potential solution alternatives

The goal of this task is to develop potential solution alternatives for each of problems, and review relations among solution alternatives to formulate overall solutions of a product line. Solution alternatives include ways to resolving variabilities in requirements.

The method should support synthesizing potential solution alternatives with the following capabilities:

— performing solution evaluation to reflect architecturally significant requirements that are common to all member products;

— performing solution evaluation for resolving conflicts among architecturally significant requirements that are variable to certain member products;

— analysing trade-offs based on the solution evaluation results;

— defining common and variable boundaries and interfaces between relevant different solutions.

A tool should support synthesizing potential solution alternatives by allowing the user to do the following:

— arrange potential solutions according to their strategic worthiness;

— analyse trade-offs using simulation, prototyping, etc.;

— express the defined common and variable boundaries and interfaces explicitly;

— document the synthesized solutions and its rationale.

6.2.4 Formulate potential domain architecture(s)

The goal of this task is to formulate the potential architecture, including principles, guidelines, protocols and standards and high-level architectural elements identified with allocated common and variable requirements and quality attributes.

The method should support formulating potential architecture(s) with the following capabilities:

— defining required architecture models that describe the domain at the high level;

— providing a way to holding relationships (particularly, variability relevant relationships) among different domain architecture models;

— handling variability which is unique in certain products in conceptual architecture;

— identifying high-level architectural elements with allocated common and variable requirements and quality attributes;

— formulating high-level architectural principles, guidelines, protocols and standards.

A tool should support formulating potential architecture(s) by allowing the user to do the following:

— use notation for describing product line specific architecture concepts and properties defined in models;

— use template for formulating conceptual architecture;

— represent variable elements contained in conceptual architecture;

— conduct prototyping for validating whether the specific architectural solution solves the defined problems;

— document high-level architectural principles, guidelines, protocols and standards.

6.2.5 Capture domain architecture concepts and properties

The goal of this task is to develop the conceptual domain architecture description consisting of relevant viewpoint, views, development models, variability models, and model correspondences and express common and variable characteristics.

The method should support capturing domain architecture concepts and properties with the following capabilities:

— capturing all required concepts and properties of variabilities addressed by the domain architecture;

— finding conflicts among variable requirements in conceptual architecture;

— validating whether the conceptual domain architecture solves the defined problems and ensures them to be realizable.

A tool should support capturing domain architecture concepts and properties by allowing the user to do the following:

— represent all common and variable architecture concepts and properties from different conceptual architecture models and views;

— document the conceptual domain architecture with the identified conflicts, their resolutions, and potential risks.

6.2.6 Hand off conceptualized domain architecture to users and other stakeholders

The goal of this task is to deliver the conceptualized domain architecture to participants of subsequent subprocess and application architecture design.

The method should support handing off the conceptualized domain architecture to users and other stakeholders with the following capabilities:

— preparing domain architecture conceptualization information, data and supporting materials for use by subsequent processes and member products of a product line;

— deliver domain architecture conceptualization information, data and supporting materials to member products of a product line;

— ensure the delivered are correct and complete enough.

A tool should support handing off the conceptualized domain architecture to users and other stakeholders by allowing the user to do the following:

— implement handoff mechanisms for sharing the conceptualized domain architecture with users and other stakeholders;

— document feedbacks and follow-up actions from/to users and other stakeholders.

6.3 Domain architectural structure design

6.3.1 Principal constituents

6.3.1.1 Purpose

The purpose of this subprocess is to produce the domain architecture that decomposes a product line into its key components and their relationships. Analysis and modelling should be conducted in this subprocess also to assure that the domain architecture covers all commonality and variability in requirements and that added or refined internal variability are covered.

6.3.1.2 Inputs

The following inputs should be available to perform the domain architectural structure design process:

— Conceptual architecture for the product line.

— Architecturally significant common and variable requirements.

— High-level architectural principles and guidelines.

— Architectural texture.

6.3.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the domain architectural structure design process:

— *Architecture viewpoints* are defined.

— *Domain architecture models and views* are defined.

— *Domain architecture* is formulated and validated.

— *Traceability from domain architecture to requirements is established.*

6.3.1.4 Tasks

The organization shall implement the following tasks with respect to the domain architectural structure design process:

— *Develop architecture viewpoints for the product line*: Develop architecture viewpoints that address different concerns of particular stakeholders that represent multiple similar products of a product line.

— *Develop models and views of the domain architecture*: Develop the domain architectural structure by allocating common and variable concerns, behaviours, functions, quality attributes and constraints.

— *Relate the domain architecture to requirements*: Establish relations between common/variable requirements and domain architectural entities.

— *Relate the domain architecture to detailed design*: Establish relations between domain architectural entities and detailed design elements.

6.3.2 Develop architecture viewpoints for the product line

The goal of this task is to select, adapt or develop architecture viewpoints that address different concerns of particular stakeholders that represent multiple similar products of a product line. Architecture viewpoints should consider variability that need to be specified together with a specification of the view.

The method should support developing architecture viewpoints for the product line with the following capabilities:

— selecting, adapting or developing viewpoints and kinds of model used for the product line;

— developing hierarchical viewpoints for dealing with complexity of a software and systems product line;

— defining attributes, associations, constraints and semantics for variability within architecture models;

— supporting consistencies of variabilities expressed from different viewpoints;

— establishing potential domain architecture frameworks to be used for developing architecture models and views.

A tool should support developing architecture viewpoints for the product line by allowing the user to do the following:

— express different abstraction levels of viewpoints for a product line;

— describe attributes, associations, constraints, semantics and so on for variability within architecture models;

— implement the domain architecture framework that can be tailored for the specific domain;

— check consistencies of variabilities expressed from different viewpoints.

6.3.3 Develop models and views of the domain architecture

The goal of this task is to select, adapt, or develop and describe the domain architectural structure models by allocating common and variable concerns, behaviours, functions, quality attributes and constraints. Views are composed of models in accordance with identified viewpoints to express how the domain architecture addresses a particular stakeholder concern. Models and views of the domain architecture include variable architectural entities.

The method should support developing models and views of the domain architecture with the following capabilities:

— identifying entities to be modelled and relationships corresponding to variabilities;

— developing models including variabilities;

— harmonizing models and views with variability in architecture;

— realizing variation points in the domain architecture that allows variants to be bound.

A tool should support developing models and views of the domain architecture by allowing the user to do the following:

— describe technical variabilities on architecture views and models;

— implement modes for filtering different views of the model including variabilities;

— check consistency among different views of the model, including variabilities.

6.3.4 Relate the domain architecture to requirements

The goal of this task is to map common and variable requirements to domain architectural entities and/or vice versa.

The method should support relating the domain architecture to requirements with the following capabilities:

— analysing relations between domain architectural elements and corresponding requirements;

— establishing relations between variation points of the domain architecture and the corresponding variation points of the requirements specification;

— establishing relations between variants of the domain architecture and variants of the requirements specification.

A tool should support relating the domain architecture to requirements by allowing the user to do the following:

— trace references from the core elements at all viewpoints to domain requirements;

— trace references from the core elements at all views and models to domain requirements.

6.3.5 Relate the domain architecture to detailed design

The goal of this task is to map domain architectural entities to detailed design elements and/or vice versa.

The method should support relating the domain architecture to detailed design with the following capabilities:

— identifying relations between domain architectural elements and detailed design elements;

— allocating domain architectural elements to corresponding detailed design entities;

— establishing relations between variation points of the domain architecture and the corresponding variation points of the detailed design specification;

— establishing relations between variants of the domain architecture and variants of the detailed design specification.

A tool should support relating the domain architecture to detailed design by allowing the user to do the following:

— trace allocation from the core elements at all viewpoints to domain detailed design elements;

— trace allocation from the core elements at all views and models to domain detailed design elements.

6.4 Architectural texture design

6.4.1 Principal constituents

6.4.1.1 Purpose

The purpose of this subprocess is to define the architectural texture, including constraints and rules that can guide architecture designers and domain engineers in evolving a product line over time without destroying its key architectural concepts.

6.4.1.2 Inputs

The following inputs should be available to perform the architectural texture design process:

— Architecturally significant requirements.

— Rules and constraints defined at requirements stage.

— High-level architectural principles and guidelines.

— Technological rules and constraints at architecture design stage.

6.4.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the architectural texture design process:

— *Architectural textures* are documented.

— *Detail-level architectural principles and guidelines* are documented.

6.4.1.4 Tasks

The organization shall implement the following tasks with respect to the architectural texture design process:

— *Analyse common rules guiding realization*: Identify common rules that should be adhered by domain realization, including detailed design and implementation and application architecture design.

— *Define common ways to deal with variability at domain realization*: Identify the texture that restricts variability implementation in domain realization.

— *Define common ways to deal with variability at application design and realization*: Identify the texture that restricts application architecture design and realization.

— *Formulate architectural texture*: Describe the rules and constraints that should be adhered to during domain architecture design, domain realization, application architecture design and application realization.

6.4.2 Analyse common rules guiding realization

The goal of this task is to identify the texture that should be adhered to by domain realization, including detailed design and implementation and application architecture design.

The method should support analysing common rules guiding realization with the following capabilities:

— reviewing implementation conventions, patterns, styles and principles that commonly guide the domain detailed design and realization;

— defining a consistent approach to interpreting and solving common or recurring problems of a product line;

— identifying core concepts that guide consistent evolution of domain architecture.

A tool should support analysing common rules guiding realization by allowing the user to do the following:

— access common rules;

— document rationale for the identified texture.

6.4.3 Define common ways to deal with variability at domain realization

The goal of this task is to identify the texture that restricts variability implementation in domain realization.

The method should support defining common ways to deal with variability at domain realization with the following capabilities:

— defining a consistent approach to interpreting and solving recurring variability;

— defining variability implementation mechanisms that should be used to realize the variation points in the domain architecture;

— providing a consistent approach to avoiding architectural erosion of the domain architecture.

A tool should support defining common ways to deal with variability at domain realization by allowing the user to do the following:

— document the defined common ways using modelling elements such as attributes, associations and semantics;

— document rationale for the identified texture.

6.4.4 Define common ways to deal with variability at application design and realization

The goal of this task is to identify the texture that restricts application architecture design and realization.

The method should support defining common ways to deal with variability at application design and realization with the following capabilities:

— defining a consistent approach to interpreting and solving recurring variability;

— defining variability implementation mechanisms that should be used to realize the variation points in the domain architecture.

A tool should support defining common ways to deal with variability at application design and realization by allowing the user to do the following:

— document common rules and constraints to be adhered to for reuse of the domain architecture;

— document rationale for the identified texture.

6.4.5 Formulate architectural texture

The goal of this task is to combine the architectural texture related to the domain architecture design and add descriptions of rules and constraints that restrict bindings, detailed domain design and realization and application architecture design.

The method should support formulating the architectural texture with the following capabilities:

— integrating the defined rules and constraints that guides design and realization during not only the initial implementation, but also during maintenance;

— resolving conflicts among rules and constraints;

— describing the texture that should be compliant among designers, architects and developers as the part of the domain architecture.

A tool should support formulating the architectural texture by allowing the user to do the following:

— explicitly describe attributes, associations and semantics related to common and variable rules in the architectural texture;

— document the texture with detail-level architectural principles and guidelines;

— (semi-)automate checking the violation of the architectural texture for designers, architects and developers.

6.5 Domain architecture documentation

6.5.1 Principal constituents

6.5.1.1 Purpose

The purpose of this subprocess is to describe domain architectural decisions, structure and textures so that domain realization, domain testing and application architecture design use the documentation as the basis for performing their tasks.

6.5.1.2 Inputs

The following inputs should be available to perform the domain architecture documentation process:

— Conceptual architecture.

— Architecture viewpoints.

— Architecture models.

— Architecture views.

— Architectural textures.

6.5.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the domain architecture documentation process:

— *Domain architecture(s) specification* is produced.

— *Conceptual architecture* is finalized and documented.

— *Architecture viewpoints* are finalized and documented.

— *Architecture models* are finalized and documented.

— *Architecture views* are finalized and documented.

— *Architectural textures* are finalized and documented.

— *Domain variability model* is finalized and documented.

6.5.1.4 Tasks

The organization shall implement the following tasks with respect to the domain architecture documentation process:

— *Assess the domain architecture documentation for structure and texture*: Evaluate the domain architecture documentation for structure and texture.

— *Hand off architecture documentation to downstream users*: Deliver the domain architecture documentation to identified users.

6.5.2 Assess the domain architecture documentation for structure and texture

The goal of this task is to ensure the domain architecture documentation for structure and texture is traceable, maintainable and evolvable.

The method should support assessing the domain architecture documentation for structure and texture with the following capabilities:

— integrating architecture structure and texture;

— ensuring consistencies between architecture structure and texture;

— analysing architecture structure and texture whether there are no conflicts among requirements of member products in a product line;

— improving and finalizing the documents of conceptual architecture, architecture viewpoints, architecture models, architecture views, the variability model and architectural textures as a set of the domain architecture documentation.

A tool should support assessing the domain architecture documentation for structure and texture by allowing the user to do the following:

— (semi-)automate document generation;

— conduct automatic consistency checking between architecture structure and texture;

— use a common format for exporting domain artifacts that include variation points.

6.5.3 Hand off architecture documentation to downstream users

The goal of this task is to deliver the domain architecture documentation, including relevant descriptions, model and data to identified users.

The method should support handing off architecture documentation to downstream users with the following capabilities:

— preparing the domain architecture documentation for use by subsequent processes and member products of a product line;

— delivering the domain architecture documentation to member products of a product line;

— ensuring the delivered are correct and complete.

A tool should support handing off architecture documentation to downstream users by allowing the user to do the following:

— disseminate the domain architecture specification to subsequent processes and member products of a product line;

— establish communication channels and dissemination mechanisms;

— document feedbacks/follow-up actions from/to downstream users.

6.6 Domain architecture evaluation

6.6.1 Principal constituents

6.6.1.1 Purpose

The purpose of this subprocess is to assess the domain architectural structure and texture for assuring whether the domain architecture meets functional and non-functional (quality) requirements.

6.6.1.2 Inputs

The following inputs should be available to perform the domain architecture evaluation process:

— Product management documents, including market definition for a product family, technology scanning for a product family, product family definition, trade-off analysis for a product family and product family evolution.

— A set of architecturally significant functional requirements that is common to all member products and variable to certain member products that possess strategic importance.

— A set of architecturally significant quality requirements that is common to all member products and variable to certain member products that possess strategic importance.

— Architecture design goals, including quality goals.

— Key drivers of the domain architecture.

— Domain architecture specification.

— Domain architecture documentation.

— Variability mechanism pool.

— Architecture evaluation process.

6.6.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the domain architecture evaluation process:

— *Findings and recommendations for the domain architecture* are documented.

— *Trade-off analysis results* are documented.

— *Evaluation results and recommendations* are documented.

— *Action plans for improving domain architecture* are established.

6.6.1.4 Tasks

The organization shall implement the following tasks with respect to the domain architecture evaluation process:

— *Determine evaluation criteria for domain architecture*: Define process, trade-off analysis, measure and decision criteria used for domain architecture evaluation.

— *Establish measurement techniques for domain architecture*: Define techniques for measuring whether the domain architecture fulfils quality goals.

— *Review evaluation-related information for domain architecture*: Examine information that will be used to evaluate the domain architecture.

— *Analyse domain architecture and assess stakeholder satisfaction*: Ensure that the domain architecture fulfils the quality goals.

— *Formulate findings and recommendations for domain architecture*: Describe findings and recommendations for improving the domain architecture.

— *Communicate evaluation results*: Report findings and recommendations of the domain architecture.

6.6.2 Determine evaluation criteria for domain architecture

The goal of this task is to define process, trade-off analysis, measure and decision criteria used for evaluating the domain architecture.

The method should support determining evaluation criteria for the domain architecture with the following capabilities:

— analysing identified design goals, key drivers and architecturally significant quality requirements for evaluating the domain architecture;

— analysing common and variable quality attributes for evaluating the domain architecture, e.g. use case scenarios;

— defining process, trade-off analysis, measure and decision criteria for evaluating the domain architecture.

A tool should support determining evaluation criteria for the domain architecture by allowing the user to do the following:

— document the determined evaluation criteria for the domain architecture;

— discriminate common and variable evaluation criteria in the document;

— (semi-)automate process, trade-off analysis, measurement and decision procedure that are used for evaluating the domain architecture.

6.6.3 Establish measurement techniques for domain architecture

The goal of this task is to define techniques for evaluating whether the domain architecture fulfils design goals and deals with variability and uses right variation mechanism to ensure correct bindings from the domain architecture to the application architecture.

The method should support establishing measurement techniques for the domain architecture with the following capabilities:

— providing techniques for evaluating whether the domain architecture fulfils design goals and deals with variability in correct and complete way;

— supporting techniques for evaluating whether the trade-off analysis of the domain architecture ensures that necessary quality attributes can be realized.

A tool should support establishing measurement techniques for the domain architecture by allowing the user to do the following:

— perform (semi-)automatic manoeuvre of the established measurement techniques;

— access the measurement techniques pool.

6.6.4 Review evaluation-related information for domain architecture

The goal of this task is to collect and examine relevant and necessary information for domain architecture evaluation, including required domain architecture models and views.

The method should support reviewing evaluation-related information for the domain architecture with the following capabilities:

— reviewing conceptual architecture;

— reviewing architecture viewpoints;

— reviewing architecture models;

— reviewing architecture views;

— reviewing architectural textures;

— reviewing variability models in architecture;

— reviewing models for each view corresponding to variability.

A tool should support reviewing evaluation-related information for the domain architecture by allowing the user to do the following:

— access evaluation-related information for the domain architecture;

— record the review results of variability models in architecture and models for each view corresponding to variability.

6.6.5 Analyse domain architecture and assess stakeholder satisfaction

The goal of this task is to evaluate the domain architecture against the design goals and objectives. Architectural scenarios or alternatives that include variable entitles can be identified and used for evaluation.

The method should support analysing the domain architecture and assessing stakeholder satisfaction with the following capabilities:

— defining architecture scenarios, which can deal with variability for the domain architecture;

— executing a trade-off analysis for the defined architecture scenarios;

— evaluating whether the domain architecture can support the design goals and objectives of the product line;

— evaluating whether the domain architecture supports necessary variability.

A tool should support analysing the domain architecture and assessing stakeholder satisfaction by allowing the user to do the following:

— access architecture models, architecturally significant requirements and the variability model in architecture;

— document the defined architecture scenarios;

— (semi-)automate trade-off analysis, measurement and risk analysis that are used for evaluating the domain architecture;

— document evaluation results with traceability.

6.6.6 Formulate findings and recommendations for domain architecture

The goal of this task is to validate and assess implications of findings so as to develop recommendations for the domain architecture.

The method should support formulating findings and recommendations for the domain architecture with the following capabilities:

— integrating findings and recommendations derived during the process of architecture evaluation;

— documenting findings and recommendations with rationale;

— identifying follow-up actions if necessary.

A tool should support formulating findings and recommendations for the domain architecture by allowing the user to do the following:

— document the findings and recommendations of the domain architecture evaluation;

— document the trade-off analysis of the domain architecture evaluation for future reference.

6.6.7 Communicate evaluation results

The goal of this task is to report and communicate findings and recommendations of evaluation results for the domain architecture.

The method should support communicating evaluation results with the following capabilities:

— reconciling domain architecture evaluation results with key stakeholders of the domain architecture;

— defining action plans in accordance with the findings and recommendations of the domain architecture.

A tool should support communicating evaluation results by allowing the user to do the following:

— use communication channels and mechanisms;

— collect and document feedbacks and follow-up actions from/to key stakeholders of the domain architecture.

7 Variability management in design

7.1 General

Variability management in design supports the following subprocesses:

— *Internal variability in domain architecture;*

— *Variability model in architecture;*

— *Variability mechanism in architecture;*

— *Variability traceability in architecture.*

7.2 Internal variability in domain architecture

7.2.1 Principal constituents

7.2.1.1 Purpose

The purpose of this subprocess is to refine external variability into internal variability at the architecture level and to identify solution-dependent variability at the architecture level as internal variability.

7.2.1.2 Inputs

The following inputs should be available to perform the internal variability in the domain architecture process:

— Variability models in requirements.

— Requirements specification.

— Conceptual architecture and architecture viewpoints.

— Architecture models and views.

— Problems refined at the architecture stage.

— Variabilities at the architecture stage.

7.2.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the internal variability in the domain architecture process:

— *Refined variabilities at the architecture stage* are documented.

— *Identified variabilities at the architecture stage* are documented.

— *Inter and intra variability relationships at architecture design stage* are documented.

7.2.1.4 Tasks

The organization shall implement the following tasks with respect to the internal variability in the domain architecture process:

— *Identify newly added internal variability*: Capture variability that is necessary for technical reasons.

— *Refine external variability into internal variability*: Refine external variabilities at a lower abstraction level.

— *Relate internal variability with variability in requirements*: Establish relations between internal variability and its relevant external variability.

7.2.2 Identify newly added internal variability

The goal of this task is to capture technical differences among member products in a product line at the architecture stage.

The method should support identifying newly added internal variability with the following capabilities:

— analysing technical differences for solving problems of member products in a product line;

— defining variation points with variants for the identified technical differences.

A tool should support identifying newly added internal variability by allowing the user to do the following:

— generate a mapping table between potential technologies related to problem solving of member products in a product line;

— document variation points and variants with dependencies and constraints;

— perform bindings for generating member product's architecture models.

7.2.3 Refine external variability into internal variability

The goal of this task is to analyse external variabilities from technological viewpoints, to refine them into internal variability.

The method should support refining external variability into internal variability with the following capabilities:

— analysing external variability from solution viewpoints;

— defining internal variability by refining external variability;

— identifying possible variation points with their variants for external variability.

A tool should support refining external variability into internal variability by allowing the user to do the following:

— maintain refinement log and traceability information;

— document variation points and variants with dependencies and constraints;

— perform bindings for generating member product's architecture models.

7.2.4 Relate internal variability with variability in requirements

The goal of this task is to map inter and intra variability relationships at the architecture stage and to link corresponding variability in requirements.

The method should support relating internal variability with variability in requirements with the following capabilities:

— analysing intra variability relationships at the architecture stage;

— analysing inter variability relationships with requirements stage;

— establishing relations between variation points of the domain architecture and the corresponding variation points of the requirements specification;

— establishing relations between variants of the domain architecture and variants of the requirements specification.

A tool should support relating internal variability with variability in requirements by allowing the user to do the following:

— relate variabilities describes in various domain architecture models;

— relate decomposition relations from the refined variability to relevant variability in requirements;

— relate allocation relations from variability in architecture to relevant variability in requirements.

7.3 Variability model in architecture

7.3.1 Principal constituents

7.3.1.1 Purpose

The purpose of this subprocess is to integrate internal variability introduced in the domain architecture with the variability model in requirements, and to explicitly represent and document the results, including the detailed binding information. Annex B depicts core elements of variability that should be specified when variability is described by using ADL (architecture description language).

7.3.1.2 Inputs

The following inputs should be available to perform the variability model in the architecture process:

— Variability model in requirements.

— Set of internal variabilities newly introduced at architecture design stage or refined variabilities at architecture design stage, including relevant information.

7.3.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the variability model in the architecture process:

— *Variability model* is refined and extended.

— *Variability in architecture* is explicitly documented.

7.3.1.4 Tasks

The organization shall implement the following tasks with respect to the variability model in the architecture process:

— *Model variability in views of architecture(s)*: Refine and extend the variability model at the architecture level.

— *Maintain variability model in architecture*: Manage the variability model in architecture.

— *Document variability in architecture*: Express and elaborate variabilities in architecture explicitly.

7.3.2 Model variability in views of architecture(s)

The goal of this task is to refine and extend the variability model in architecture as variability is refined or newly added at the architecture design stage.

The method should support modelling variability in views of architecture(s) with the following capabilities:

— supporting the variability refinement and extension in the variability model;

— providing ways for discriminating variability in the variability model that is newly added, refined or extended in architecture from variability introduced in the previous stages;

— supporting the verification and validation of the variability model.

A tool should support modelling variability in views of architecture(s) by allowing the user to do the following:

— perform variability model refinement and extension at the architecture design stage;

— use notations for discriminating variability in the variability model that is newly introduced, refined or extended at architecture design stage with variability from the previous stages (e.g. annotation, coloured notation);

— maintain variability model configuration by each domain engineering stage;

— document variability model refinement history.

7.3.3 Maintain variability model in architecture

The goal of this task is to manage the refinement and evolution of the variability model in architecture.

The method should support maintaining the variability model in architecture with the following capabilities:

— supporting impact analysis for the proposed changes;

— allowing changes of the variability model at the architecture design stage;

— supporting consistent evolution of the variability model at the architecture design stage.

A tool should support maintaining the variability model in architecture by allowing the user to do the following:

— perform impact analysis for the proposed changes;

— conduct version management of the variability model in architecture;

— record change history of the variability model.

7.3.4 Document variability in architecture

The goal of this task is to conduct documentation for variation points, variants, and the variability model with dependencies and constraints. This can include a description of used variability mechanisms and possible reuse contexts.

The method should support documenting variability in architecture with the following capabilities:

— confirming whether variation points, variants and the variability model with dependencies and constraints at the architecture stage are consistent;

— documenting variation points, variants and the variability model with dependencies and constraints at the architecture stage;

— supporting a description of used variability mechanisms and possible reuse contexts.

A tool should support documenting variability in architecture by allowing the user to do the following:

— use notations that differentiate variable architectural entities from those in common;

— visualize relationships among variable architectural entities documented in different ways;

— differentiate architectural artefacts related variability from those in common;

— document variation points, variants, and the variability model with dependencies and constraints at the architecture stage.

7.4 Variability mechanism in architecture

7.4.1 Principal constituents

7.4.1.1 Purpose

The purpose of this subprocess is to deal with the capabilities to locate and implement variability in architecture to architecture design.

7.4.1.2 Inputs

The following inputs should be available to perform the variability mechanism in the architecture process:

— Guidance for variability mechanism selection (ISO/IEC 26557).

— Variability mechanism operationalization for design (elicited from variability mechanism pool).

7.4.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the variability mechanism in the architecture process:

— *Variability mechanisms used for implementing variability in architecture* are categorized.

— *Guidance is tailored specific to architecture design stage*.

— *Usage status of variability mechanism category in architecture design* is traced.

7.4.1.4 Tasks

The organization shall implement the following tasks with respect to the variability mechanism in the architecture process:

— *Identify variability mechanisms in architecture by category*: Find suitable variability mechanisms for the refined or newly added variabilities in architecture.

— *Guide the use of variability mechanism category in architecture*: Refine the user guide of the variability mechanism category to meet the architecture level requirements.

— *Trace the usage status of variability mechanism category in architecture*: Trace the effectiveness and efficiency of the variability mechanism category.

— *Update variability mechanism category in architecture*: Improve the variability mechanism category.

7.4.2 Identify variability mechanisms in architecture by category

The goal of this task is to find variability mechanisms suitable for the characteristics of domain and application architecture. Variability mechanisms should be reviewed depending on the decisions for architectural structure, texture and enabling environment.

The method should support identifying variability mechanisms in architecture by category with the following capabilities:

— reviewing the available variability mechanisms for checking whether they comply with the explored architectural decisions and textures;

— assuring whether a variability mechanism fully covers the defined variability including dependencies and constraints;

— assuring whether a variability mechanism fully covers the defined variability including dependencies and constraints that should be bound in the architecture design stage;

— evaluating ease of binding and configuration.

A tool should support identifying variability mechanisms in architecture by category by allowing the user to do the following:

— use the architecture level variability mechanisms pool with their usage guidance;

— refer rationales on selection of variability mechanisms.

7.4.3 Guide the use of variability mechanism category in architecture

The goal of this task is to provide guidance for selection, decisions for binding time and configuration mechanism, and performing binding in a member product. The overall guidance is provided by ISO/IEC 26557. This can include concrete guides with architecture design specific practices.

The method should support guiding the use of the variability mechanism category in architecture with the following capabilities:

— tailoring variability mechanism selection criteria specific to the architecture design stage;

— tailoring detailed procedures for variability mechanism selection and use in the architecture design stage;

— tailoring configuration guides and rules in the architecture design stage.

A tool should support guiding the use of the variability mechanism category in architecture by allowing the user to do the following:

— use guides and rules for variability mechanism operation with relevant participants;

— access guides and rules for performing variability mechanism operation.

7.4.4 Trace the usage status of variability mechanism category in architecture

The goal of this task is to trace the effectiveness and efficiency of variability mechanisms identified in category.

Measures and metrics used for tracing the usage status of variability mechanisms are defined, the usage status is measured, and the results should be properly analysed for providing valuable feedbacks.

The method should support tracing the usage status of the variability mechanism category in architecture with the following capabilities:

— confirming that variability mechanisms in architecture (e.g. variability mechanisms in codes) can correctly address the defined variability dependencies;

— confirming that variability mechanisms in architecture can correctly address the defined constraints;

— providing evaluation algorithm for verifying whether a variability mechanism confirms the defined variability dependencies and constraints;

— verifying the testability of variability mechanisms used in architecture.

A tool should support tracing the usage status of the variability mechanism category in architecture by allowing the user to do the following:

— visualize variability mechanisms included in architecture;

— verify whether the variability mechanism properly supports the defined variability including dependencies and constraints;

— refer the testability of variability mechanisms used in architecture.

7.4.5 Update variability mechanism category in architecture

The goal of this task is to improve the established variability mechanism category.

Variability mechanisms in architecture identified by category can be re-organized and improved based on analysis results of their usage status.

The method should support updating the variability mechanism category in architecture with the following capabilities:

— preparing items for variability mechanism category improvement;

— improving variability mechanism category;

— propagating improvement results to variability mechanism pool.

A tool should support updating variability mechanism category in architecture by allowing the user to do the following:

— share improvement items for the variability mechanism category among appropriate participants;

— share improved variability mechanism category;

— perform automatic propagation of improvement results to the variability mechanism pool.

7.5 Variability traceability in architecture

7.5.1 Principal constituents

7.5.1.1 Purpose

The purpose of this subprocess is to establish and maintain trace links between the variability model and architectural artefacts, including trace links between different abstraction levels of variability.

7.5.1.2 Inputs

The following inputs should be available to perform the variability traceability in the architecture process:

— Architecture specification.

— Variability models in architecture design.

7.5.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the variability traceability in the architecture process:

— *Trace links among different architecture design artefacts that include variability* are established and maintained.

— *Trace links between architecture design artefacts and variability models* are established.

7.5.1.4 Tasks

The organization shall implement the following tasks with respect to the variability traceability in the architecture process:

— *Define trace links among variability in different architectural artefacts*: Trace variability distributed over several different architectural artefacts.

— *Define trace links between architectural artefacts and variability model*: Trace variability in architectural artefacts from the variability model.

7.5.2 Define trace links among variability in different architectural artefacts

The goal of this task is to establish and maintain explicit trace links among variability that are defined in different architecture design artefacts.

The method should support defining trace links among variability in different architectural artefacts with the following capabilities:

— defining bidirectional links among variability in different architectural artefacts;

— maintaining bidirectional links among variability in different architectural artefacts.

A tool should support defining trace links among variability in different architectural artefacts by allowing the user to do the following:

— visualize links among variability in different architectural artefacts;

— maintain bidirectional links among variability in different architectural artefacts.

7.5.3 Define trace links between architectural artefacts and variability model

The goal of this task is to establish and maintain explicit trace links between domain/application architecture design artefacts and the domain/application variability model.

The method should support defining trace links between architectural artefacts and the variability model with the following capabilities:

— defining bidirectional links between architectural artefacts and the variability model;

— maintaining links of the variability model with variability in architectural artefacts.

A tool should support defining trace links between architectural artefacts and the variability model by allowing the user to do the following:

— visualize trace links between architectural artefacts and the variability model;

— maintain bidirectional links between variability in architectural artefacts and the variability model.

8 Asset management in design

8.1 General

Asset management supports the following subprocesses:

— *Managing domain design artefacts as domain assets;*

— *Managing application design artefacts as application assets.*

8.2 Managing domain design artefacts as domain assets

8.2.1 Principal constituents

8.2.1.1 Purpose

The purpose of this subprocess is to manage domain architecture design artefacts that will be reused by the product line members, which include process definitions, plans, guides, specifications, models, components, interfaces, test cases, etc.

8.2.1.2 Inputs

The following inputs should be available to perform the domain design artefacts as domain assets process:

— Domain architecture models.

— Domain architecture views.

— Architectural textures.

— Domain architecture evaluation.

— Domain architecture specification.

8.2.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the domain design artefacts as domain assets process:

— *Configurations of domain design artefacts* are established.

— *Evolutions of domain design artefacts* are performed.

— Intra and inter traceability of domain design artefacts are maintained.

8.2.1.4 Tasks

The organization shall implement the following tasks with respect to the domain design artefacts as domain assets process:

— *Identify architectural artefacts managed as domain assets*: Identify reusable domain architectural artefacts.

— *Define configuration and annotation for domain architecture assets*: Define configuration of architectural artefacts and add suitable annotation for supporting the reuse of architectural artefacts.

8.2.2 Identify architectural artefacts managed as domain assets

The goal of this task is to identify architectural artefacts that that will be reused as domain assets including both common and variable artefacts.

The method should support identifying architectural artefacts managed as domain assets with the following capabilities:

— collecting domain design artefacts that are produced throughout the domain engineering process;

— provisioning of checklists and evaluation criteria for the control of reusability;

— establishing intra trace links within domain design artefacts;

— establishing backward trace links with domain requirements.

A tool should support identifying architectural artefacts managed as domain assets with the following capabilities:

— access domain design artefacts that have reuse potential;

— manage repository of architectural artefacts as domain assets;

— establish intra and inter trace links of architectural artefacts as domain assets.

8.2.3 Define configuration and annotation for domain architecture assets

The goal of this task is to define and develop the further structure of architectural artefacts that will help with reuse at the application developments.

The method should support defining configuration and annotation for domain architecture assets with the following capabilities:

— defining configuration of domain design artefacts that help users to retrieve, update, delete or maintain traceability;

— providing annotations necessary to reuse architectural artefacts as domain assets at the application developments;

— validating configuration and annotations for architectural artefacts.

A tool should support defining configuration and annotation for domain architecture assets with the following capabilities:

— access the existing information for defining configuration and annotation (the configuration includes trace links with the relevant domain requirements artefacts);

— access architectural artefacts;

— use an editor for template definition.

8.3 Managing application design artefacts as application assets

8.3.1 Principal constituents

8.3.1.1 Purpose

The purpose of this subprocess is to manage application specific architecture design artefacts, which include process definitions, plans, guides, specifications, models, components, interfaces, test cases, etc.

8.3.1.2 Inputs

The following inputs should be available to perform the application design artefacts as application assets process:

— Application-specific architecture models.

— Architectural textures.

— Application-specific architecture evaluation.

— Application-specific architecture specification.

8.3.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the application design artefacts as application assets process:

— *Configurations of application design artefacts* are established.

— *Evolutions of application design artefacts* are performed.

— *Intra and inter traceability of application design artefacts* are maintained.

8.3.1.4 Tasks

The organization shall implement the following tasks with respect to the application design artefacts as application assets process:

— *Identify architectural artefacts managed as application assets*: Identify reusable application architectural artefacts.

— *Define configuration and annotation for application architecture assets*: Define configuration of architectural artefacts and add suitable annotation for supporting the reuse of architectural artefacts.

8.3.2 Identify architectural artefacts managed as application assets

The goal of this task is to identify application design artefacts such as architecture models, views and specifications to be maintained in each application development.

The method should support identifying architectural artefacts managed as application assets with the following capabilities:

— collecting application design artefacts that are produced throughout the application engineering process;

— provisioning of checklists and evaluation criteria for the control of suitability as architectural artefacts of application assets;

— establishing intra trace links within application design artefacts;

— establishing inter trace links with domain design artefacts;

— establishing backward trace links with application requirements.

A tool should support identifying architectural artefacts managed as application assets with the following capabilities:

— access application design artefacts that have reuse potential;

— access application design artefacts that are specific to certain member products;

— manage repository of architecture assets;

— establish intra and inter trace links of application architecture assets.

8.3.3 Define configuration and annotation for application architecture assets

The goal of this task is to define or develop the structure and information for application design assets that will be referred by the successive application development. Configuration for application design assets that will help it to be managed as assets is defined.

The method should support defining configuration and annotation for application architecture assets with the following capabilities:

— defining configuration of application design assets that help users to retrieve, update, delete, or maintain traceability;

— providing annotations necessary to use application design assets at the application developments;

— validating configuration and annotations for application design assets.

A tool should support defining configuration and annotation for application architecture assets with the following capabilities:

— access existing information for defining configuration and annotation;

— access application design assets;

— access domain design assets;

— use an editor for template definition;

— establish trace links among application design assets;

— establish trace links between assets resulting from the later application engineering processes.

9 Application design

9.1 General

Application design supports the following subprocesses:

— *Binding in architecture;*

— *Application specific architectural structure design;*

— *Application architecture documentation;*

— *Application architecture evaluation.*

9.2 Binding in architecture

9.2.1 Principal constituents

9.2.1.1 Purpose

The purpose of this subprocess is to bind the selected variant(s) for variation points of the domain architecture according to the architectural texture and binding decisions.

9.2.1.2 Inputs

The following inputs should be available to perform the binding in the architecture process:

— Variability models in architecture.

— Variability mechanisms in architecture.

— Variability traceability in architecture.

— Domain architecture models.

— Architectural texture.

— Application requirements artefacts.

9.2.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the binding in the architecture process:

— *Application architecture models for a member product* are derived.

— *Application variability model for a member product* is derived.

9.2.1.4 Tasks

The organization shall implement the following tasks with respect to the binding in the architecture process:

— *Decide values of variabilities in architecture*: Select variants of variation points in domain design models.

— *Conduct bindings in architecture*: Produce application architecture models for a member product through bindings.

— *Validate consistencies with bindings in requirements*: Evaluate whether bindings in architecture are consistent with those relevant bindings in requirements.

— *Validate whether binding decisions adhere to the architectural texture*: Evaluate whether bindings in architecture adhere to the common rules and constraints in the architectural texture.

9.2.2 Decide values of variabilities in architecture

The goal of this task is to select variants of variation points in domain design models. All variabilities of which binding times are at the architecture stage are considered.

The method should support deciding values of variabilities in architecture with the following capabilities:

— reviewing binding decisions in requirements that have impacts on bindings in architecture;

— reviewing the architectural texture;

— deciding the value of internal variability of which binding time is at the architecture stage.

A tool should support deciding values of variabilities in architecture by allowing the user to do the following:

— access binding decisions in requirements;

— access the variability model in architecture, including binding time information;

— access the architectural texture.

9.2.3 Conduct bindings in architecture

The goal of this task is to produce architecture models for a member product through bindings.

The method should support conducting bindings in architecture with the following capabilities:

— configuring architecture models according to binding decisions;

— confirming consistencies among architecture models in views after binding;

— checking all influenced variable elements of architecture models in accordance with bindings;

— updating the variability model for a member product in accordance with bindings.

A tool should support conducting bindings in architecture by allowing the user to do the following:

— perform (semi-)automatic derivation of application architecture models according to binding decisions;

— conduct (semi-)automatic update of the variability model for a member product through bindings;

— check consistencies for the derived architecture models.

9.2.4 Validate consistencies with bindings in requirements

The goal of this task is to evaluate whether bindings in architecture are consistent with those relevant bindings in requirements.

The method should support validating consistencies with bindings in requirements with the following capabilities:

— validating consistencies between external bindings and their deduced implicit internal bindings;

— validating configured architecture models and their elements in accordance with binding decisions in requirements.

A tool should support validating consistencies with bindings in requirements by allowing the user to do the following:

— access binding results in requirements;

— establish traces from bound values in the architecture model to relevant requirements.

9.2.5 Validate whether binding decisions adhere to the architectural texture

The goal of this task is to evaluate whether bindings in architecture are conducted under the adherence of the defined architectural texture during domain design.

The method should support validating whether binding decisions adhere to the architectural texture with the following capabilities:

— validating procedures related to bindings;

— validating rules and constraints related to binding decisions.

A tool should support validating whether binding decisions adhere to the architectural texture by allowing the user to do the following:

— access the architectural texture;

— document the rationale of binding decisions.

9.3 Application specific architectural structure design

9.3.1 Principal constituents

9.3.1.1 Purpose

The purpose of this subprocess is to design the architectural structure for fulfilling application specific requirements even though a large part of the application architecture is derived from the domain architecture.

9.3.1.2 Inputs

The following inputs should be available to perform the application specific architectural structure design process:

— Conceptual architecture for the product line.

— Domain architecture.

— Architecturally significant application specific requirements.

— High-level architectural principles and guidelines.

— Architectural texture.

9.3.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the application specific architectural structure design process:

— *Application specific architecture viewpoints* are defined.

— *Application specific architecture models and views* are defined.

— *Application specific architecture* is formulated and validated.

— *Traceability from application architecture to application requirements* is established.

— *Traceability from application architecture to domain architecture* is established.

9.3.1.4 Tasks

The organization shall implement the following tasks with respect to the application specific architectural structure design process:

— *Develop models of the application specific architecture*: Develop the application specific architectural structure by allocating application specific concerns, behaviours, functions, quality attributes and constraints.

— *Validate whether the application specific architecture adheres to the architectural texture.*

— *Relate the application specific architecture to requirements*: Establish relations between application requirements and application specific architectural entities.

— *Relate the application specific architecture to detailed design*: Establish relations between application specific architectural entities and detailed design elements.

9.3.2 Develop models of the application specific architecture

The goal of this task is to select, adapt, or develop and describe the application specific architectural structure models by allocating application specific concerns, behaviours, functions, quality attributes and constraints.

The method should support developing models of the application specific architecture with the following capabilities:

— binding variation points and variants in the domain architecture as needed;

— identifying application specific entities to be modelled and relationships;

— developing models including application specific variabilities;

— harmonizing models and views of the application architecture.

A tool should support developing models of the application specific architecture by allowing the user to do the following:

— perform bindings of variation points and variants in the domain architecture as needed;

— add, refine or tailor architectural elements based on the derived architecture models;

— check consistencies for the application specific architecture models.

9.3.3 Validate whether the application specific architecture adheres to the architectural texture

The goal of this task is to evaluate whether application specific architecture models are added, refined or tailored under the adherence of the defined architectural texture during domain architecture design.

The method should support validating whether the application specific architecture adheres to the architectural texture with the following capabilities:

— validating rules and constraints adhered to by added, refined or tailored architecture model elements;

— aligning the application specific architecture with the domain architecture in accordance with the architectural texture.

A tool should support validating whether the application specific architecture adheres to the architectural texture by allowing the user to do the following:

— access the architectural texture;

— align the application specific architecture in accordance with the domain architecture and the architectural texture.

9.3.4 Relate the application specific architecture to requirements

The goal of this task is to map application specific requirements to application architectural entities and/or vice versa.

The method should support relating the application specific architecture to requirements with the following capabilities:

— configuring relations established during application architecture design in accordance with binding at the application design level;

— allocating application specific requirements to corresponding application architectural elements;

— establishing relations between application specific requirements and application architectural elements.

A tool should support relating the application specific architecture to requirements by allowing the user to do the following:

— trace references from the core elements at all viewpoints to application requirements;

— trace references from the core elements at all views and models to application requirements.

9.3.5 Relate the application specific architecture to detailed design

The goal of this task is to map application architectural entities to detailed design elements and/or vice versa.

The method should support relating the application specific architecture to detailed design with the following capabilities:

— mapping application architectural elements to corresponding application specific detailed design entities;

— establishing relations between bindings of application architecture artefacts and bindings of the application detailed design specification.

A tool should support relating the application specific architecture to detailed design by allowing the user to do the following:

— trace allocation from the core elements at all viewpoints to application detailed design elements;

— trace allocation from the core elements at all views and models to application detailed design elements.

9.4 Application architecture documentation

9.4.1 Principal constituents

9.4.1.1 Purpose

The purpose of this subprocess is to document application specific architectural decisions so that application realization and testing use the documentation as the basis for performing their tasks.

9.4.1.2 Inputs

The following inputs should be available to perform the application architecture documentation process:

— Conceptual architecture.

— Architectural textures.

— Domain design artefacts.

— Application architecture viewpoints.

— Application architecture models.

— Application architecture views.

9.4.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the application architecture documentation process:

— *Application architecture(s) specification* is produced.

— *Application architecture viewpoints* are finalized and documented.

— *Application architecture models* are finalized and documented.

— *Application architecture views* are finalized and documented.

— *Application variability model* is finalized and documented.

9.4.1.4 Tasks

The organization shall implement the following tasks with respect to the application architecture documentation process:

— *Assess the application specific architecture documentation*: Evaluate the application specific architecture documentation for structure and texture.

— *Hand off application specific architecture documentation to downstream users*: Deliver the application specific architecture documentation to identified users.

9.4.2 Assess the application specific architecture documentation

The goal of this task is to ensure the application architecture documentation for structure is traceable, maintainable and evolvable.

The method should support assessing the application specific architecture documentation for structure and texture with the following capabilities:

— ensuring consistencies between domain and application architecture structure;

— analysing the application specific architecture structure to determine whether there are conflicts with the domain architecture;

— improving and finalizing the application specific architecture documents in terms of conceptual architecture, architecture viewpoints, architecture models, architecture views and the variability model as a set of application architecture documentation.

A tool should support assessing the application specific architecture documentation for structure by allowing the user to do the following:

— perform semi-automatic application architecture document generation;

— conduct automatic consistency checking between application architectural structure and texture;

— use a common format for importing domain artifacts that include variation points.

9.4.3 Hand off application specific architecture documentation to downstream users

The goal of this task is to deliver the application architecture documentation, including relevant descriptions, model and data to identified users.

The method should support handing off the application specific architecture documentation to downstream users with the following capabilities:

— preparing application architecture documentation for use by subsequent processes;

— delivering application architecture documentation to application specific stakeholders;

— ensuring the delivered are correct and complete enough.

A tool should support handing off the application specific architecture documentation to downstream users by allowing the user to do the following:

— disseminate the application architecture specification to subsequent processes and to relevant stakeholders;

— establish communication channels and dissemination mechanisms;

— document feedback and follow-up actions from/to downstream users and relevant stakeholders.

9.5 Application architecture evaluation

9.5.1 Principal constituents

9.5.1.1 Purpose

The purpose of this subprocess is to evaluate the application specific architecture. However, the pre-evaluated parts of the architecture during domain design can be re-assessed in order to confirm their integrity with the bound variants.

9.5.1.2 Inputs

The following inputs should be available to perform the application architecture evaluation process:

— Product management documents including market definition, technology scanning, trade-off analysis for the member product.

— A set of architecturally significant functional requirements for the member product.

— A set of architecturally significant quality requirements for the member product.

— Application architecture design goals, including quality goals.

— Key drivers of the application architecture.

— Application architecture and its specification.

— Application specific variability.

— Domain architecture documentation.

— Variability mechanism pool.

— Architecture evaluation process.

9.5.1.3 Outcomes

The following outcomes shall be available as a result of the successful implementation of the application architecture evaluation process:

— *Findings and recommendations for the application architecture* are documented.

— *Trade-off analysis results of the member product* are documented.

— *Evaluation results and recommendations of the member product* are documented.

— *Action plans for improving application architecture* are established.

9.5.1.4 Tasks

The organization shall implement the following tasks with respect to the application architecture evaluation process:

— *Determine application specific evaluation criteria*: Define process, trade-off analysis, measure and decision criteria used for application specific architecture evaluation.

— *Establish application specific measurement techniques*: Define techniques for measuring whether the application specific architecture fulfils quality goals, including application specific quality goals.

— *Review evaluation-related information for application architecture*: Examine information that will be used to evaluate the application architecture.

— *Analyse application architecture and assess stakeholder satisfaction*: Ensure that the application specific architecture fulfils the quality goals.

— *Formulate findings and recommendations for application architecture*: Describe findings and recommendations for improving the application architecture.

— *Communicate evaluation results*: Report findings and recommendations of the application architecture.

9.5.2 Determine application specific evaluation criteria

The goal of this task is to define process, trade-off analysis, measure, decision criteria that can be used for evaluating the application architecture.

The method should support determining application specific evaluation criteria with the following capabilities:

— analysing identified design goals, key drivers and architecturally significant quality requirements for evaluating the application architecture;

— analysing application specific quality attributes for evaluating the application architecture, e.g. use case scenarios;

— defining process, trade-off analysis, measure and decision criteria for evaluating the application architecture.

A tool should support determining application specific evaluation criteria by allowing the user to do the following:

— reuse domain architecture evaluation criteria;

— document application specific evaluation criteria;

— (semi-)automate process, trade-off analysis, measure and decision that are used for evaluating the application architecture.

9.5.3 Establish application specific measurement techniques

The goal of this task is to define techniques for evaluating whether the application architecture fulfils application design goals and deals with application specific variability and uses right variation mechanism to ensure correct bindings from the domain architecture to the application architecture.

The method should support establishing application specific measurement techniques with the following capabilities:

— providing techniques for evaluating whether the application architecture fulfils application design goals and deals with application specific variability in a correct and complete way;

— supporting techniques for evaluating whether the trade-off analysis of the application architecture ensures that necessary quality attributes can be realized.

A tool should support establishing application specific measurement techniques by allowing the user to do the following:

— perform (semi-)automatic manoeuvre of the established measurement techniques;

— access the measurement techniques pool.

9.5.4 Review evaluation-related information for application architecture

The goal of this task is to collect and examine relevant and necessary information for application architecture evaluation, including required application architecture models and views.

The method should support reviewing evaluation-related information for the application architecture with the following capabilities:

— reviewing the conceptual architecture;

— reviewing application architecture viewpoints;

— reviewing application architecture models;

— reviewing application architecture views;

— reviewing architectural textures;

— reviewing variability models in the application architecture design level;

— reviewing models for each view corresponding to application specific variability.

A tool should support reviewing evaluation-related information for application architecture by allowing the user to do the following:

— access evaluation-related information for the application specific architecture;

— record the review results of application variability models in architecture and models for each view corresponding to variability.

9.5.5 Analyse application architecture and assess stakeholder satisfaction

The goal of this task is to evaluate application architecture against the application specific design goals and objectives. Architectural scenarios or alternatives that include variable entitles can be identified and used for evaluation.

The method should support analysing application architecture and assessing stakeholder satisfaction with the following capabilities:

— defining application architecture scenarios, which can deal with variability bindings for the application architecture;

— executing a trade-off analysis for the defined application architecture scenarios;

— evaluating whether the application architecture can support the application design goals and objectives of the specific member product;

— evaluating whether the application architecture supports application specific variability.

A tool should support analysing the application architecture and assessing stakeholder satisfaction by allowing the user to do the following:

— access application architecture models, architecturally significant requirements and the application variability model in architecture;

— document the defined application architecture scenarios;

— (semi-)automate trade-off analysis, measure, risk analysis that are used for evaluating the application architecture;

— document application architecture evaluation results with traceability.

9.5.6 Formulate findings and recommendations for application architecture

The goal of this task is to validate and assess implications of findings and develop recommendations for improving the application architecture.

The method should support formulating findings and recommendations for the application architecture with the following capabilities:

— integrating findings and recommendations derived during the process of application architecture evaluations;

— documenting findings and recommendations with rationale;

— identifying follow-up actions as necessary.

A tool should support formulating findings and recommendations for the application architecture by allowing the user to do the following:

— document findings and recommendations of application architecture evaluation;

— document the trade-off analysis of application architecture evaluation for future reference;

— perform version management for document of application architecture models.

9.5.7 Communicate evaluation results with application specific stakeholders

The goal of this task is to report and communicate findings and recommendations of evaluation results for the application architecture.

The method should support communicating evaluation results with application specific stakeholders with the following capabilities:

— reconciling application architecture evaluation results with key stakeholders of the application architecture;

— resolving conflicts between domain and application architects;

— defining action plans in accordance with the findings and recommendations of the application architecture.

A tool should support communicating evaluation results with application specific stakeholders by allowing the user to do the following:

— use communication channels and mechanisms;

— collect and document feedbacks and follow-up actions from/to key stakeholders of the application architecture.

Annex A
(informative)

Cross-reference with ISO/IEC/IEEE 42020, ISO/IEC/IEEE 42010 and ISO/IEC/IEEE 15288

Table A.1 — Mapping table among ISO/IEC/IEEE 42020, ISO/IEC/IEEE 42010, ISO/IEC/IEEE 15288, and ISO/IEC 26552

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
5.2	Architecture design planning	6.4.4.3	a) Prepare for architecture definition	6	Architecture Governance process		
5.2.2	Establish architecture design goals			6.4.1	Prepare for and plan the architecture governance effort		
5.4	Architecture design managing			6.4.2	Monitor, assess and control the architecture governance activities		
5.2.2	Establish architecture design goals			6.4.3	Establish architecture collection objectives		
5.4	Architecture design managing			6.4.4	Make architecture governance decisions		
5.4	Architecture design managing			6.4.5 6.4.6	Monitor and assess compliance with governance directives and guidance Review implementation of governance directives and guidance		
5.4	Architecture design managing	6.4.4.3	f) Manage the selected architecture	7	Architecture Management process		
5.4.2	Prepare for architecture management execution			7.4.1	Prepare for and plan the architecture management effort		
5.4.3	Implement the architecture management plans			7.4.2	Monitor, assess and control the architecture management activities		
5.4.2	Prepare for architecture management execution			7.4.3	Develop architecture management approach		
5.4.3	Implement the architecture management plans			7.4.4 7.4.5	Perform management of the architecture collection Monitor architecting effectiveness		
5.4.4	Close and prepare for the architecture management plan change			7.4.6	Prepare for completion of the architecture management plan		
6.2	Conceptual architecture design			8	Architecture Conceptualization process		

Table A.1 *(continued)*

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
5.2	Architecture design planning			8.4.1 8.4.4	Prepare for and plan the architecture conceptualization effort Establish architecture objectives and critical success criteria		
5.4	Architecture design managing			8.4.2	Monitor, assess and control the architecture conceptualization activities		
6.2.2	Analyse problem space of the domain architecture			8.4.3	Characterize problem space		
6.2.3	Synthesize potential solution alternatives			8.4.5	Synthesize potential solution(s) in the solution space		
6.2.4	Formulate potential domain architecture(s)			8.4.7	Formulate candidate architecture(s)		
6.2.5	Capture domain architecture concepts and properties			8.4.8	Capture architecture concepts and properties		
6.2.6	Hand off conceptualized domain architecture to downstream users			8.4.10	Coordinate use of conceptualized architecture by intended users		
6.6 9.5	Domain architecture evaluation Application architecture evaluation			9	Architecture Evaluation process		
5.2	Architecture design planning	6.4.4.3	a) Prepare for architecture definition	9.4.1	Prepare for and plan the architecture evaluation effort		
5.4	Architecture design managing			9.4.2	Monitor, assess and control the architecture evaluation activities		
6.6.2 9.5.2	Determine evaluation criteria for domain architecture Determine application specific evaluation criteria			9.4.3 9.4.4	Determine evaluation objectives and criteria Determine evaluation methods and integrate with evaluation objectives and criteria		

Table A.1 *(continued)*

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
6.6.3 9.5.3	Establish measurement techniques for domain architecture Establish application specific measurement techniques			9.4.5	Establish measurement techniques, methods and tools		
6.6.4 9.5.4	Review evaluation-related information for domain architecture Review evaluation-related information for application architecture			9.4.6	Collect and review evaluation-related information		
6.6.5 9.5.5	Evaluate domain architecture and assess stakeholder satisfaction Evaluate application specific architecture and assess stakeholder satisfaction	6.4.4.3	e) Assess architecture candidates	9.4.7 9.4.8	Analyse architecture concepts and properties and assess stakeholder value Characterize architecture(s) based on assessment results		
6.6.6 9.5.6	Formulate findings and recommendations for domain architecture Formulate findings and recommendations for application architecture			9.4.9	Formulate findings and recommendations		
6.6.7 9.5.7	Communicate evaluation results Communicate evaluation results with application specific stakeholders			9.4.10	Capture and communicate evaluation results		
6.3 9.3	Domain architectural structure design Application specific architectural structure design			10	Architecture Elaboration process	5	Architecture descriptions
9.2	Binding in architecture						

Table A.1 *(continued)*

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
5.2	Architecture design planning			10.4.1	Prepare for and plan the architecture elaboration effort		
5.4	Architecture design managing			10.4.2	Monitor, assess and control the architecture elaboration activities		
6.3.2	Develop architecture viewpoints for the product line	6.4.4.3	b) Develop architecture viewpoints	10.4.3	Identify or develop architecture viewpoints	5.4	Architecture viewpoints
6.3.3 9.3.2	Develop models and views of the domain architecture Develop models of the application specific architecture	6.4.4.3	c) Develop models and views of candidate architectures	10.4.4	Develop models and views of the architecture(s)	5.5 5.6	Architecture views Architecture models
9.3.3	Validate whether the application specific architecture adheres to the architectural texture						
6.3.4 9.3.4	Relate the domain architecture to requirements Relate the application specific architecture to requirements						
6.3.5 9.3.5	Relate the domain architecture to detailed design Relate the application specific architecture to detailed design	6.4.4.3	d) Relate the architecture to design	10.4.5	Relate the architecture to other architectures and to relevant affected entities	5.7	Architecture relations
6.4	Architectural texture design						
6.4.2	Analyse common rules guiding realization						
6.4.3	Define common ways to deal with variability in domain realization						

Table A.1 *(continued)*

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
6.4.4	Define common ways to deal with variability in application design and realization						
6.4.5	Formulate architectural texture						
6.5 6.5.2	Domain architecture documentation Assess the domain architecture documentation for structure and texture	6.4.4.3	e) Assess architecture candidates	10.4.6	Assess the architecture elaboration		
6.5 6.5.3	Domain architecture documentation Hand off architecture documentation to downstream users			10.4.7	Coordinate use of elaborated architecture by intended users		
5.3	Architecture design enabling			11	Architecture Enablement process		
5.3.2	Prepare for the architecture enablement			11.4.1	Prepare for and plan the architecture enablement effort		
5.4	Architecture design managing			11.4.2 11.4.3	Monitor, assess and control the architecture enablement activities Manage the architecture process enablers		
5.3.3	Develop and establish enabling capabilities and resources			11.4.4	Acquire, develop and establish enabling capabilities, services and resources		
5.3.4	Deploy capabilities and resources for architecture enablement			11.4.5	Deploy enabling capabilities, services and resources		
5.3.5	Improve architecture enablement capabilities and resources			11.4.6	Improve architecture enablement capabilities, services and resources		

Table A.1 *(continued)*

ISO/IEC 26552		ISO/IEC/IEEE 15288:2015		ISO/IEC/IEEE 42020:—		ISO/IEC/IEEE 42010:2011	
Clause#	**Clause name**	**Clause#**	**Activity name**	**Clause#**	**Activity name**	**Clause#**	**Clause name**
7	Variability management in design						
8	Asset management in design						
						6	Architecture frameworks and architecture description languages

Annex B
(informative)

Variability specification elements in ADL

The diagram in Figure B.1 depicts core elements of variability that should be specified when variability is described by using ADL.

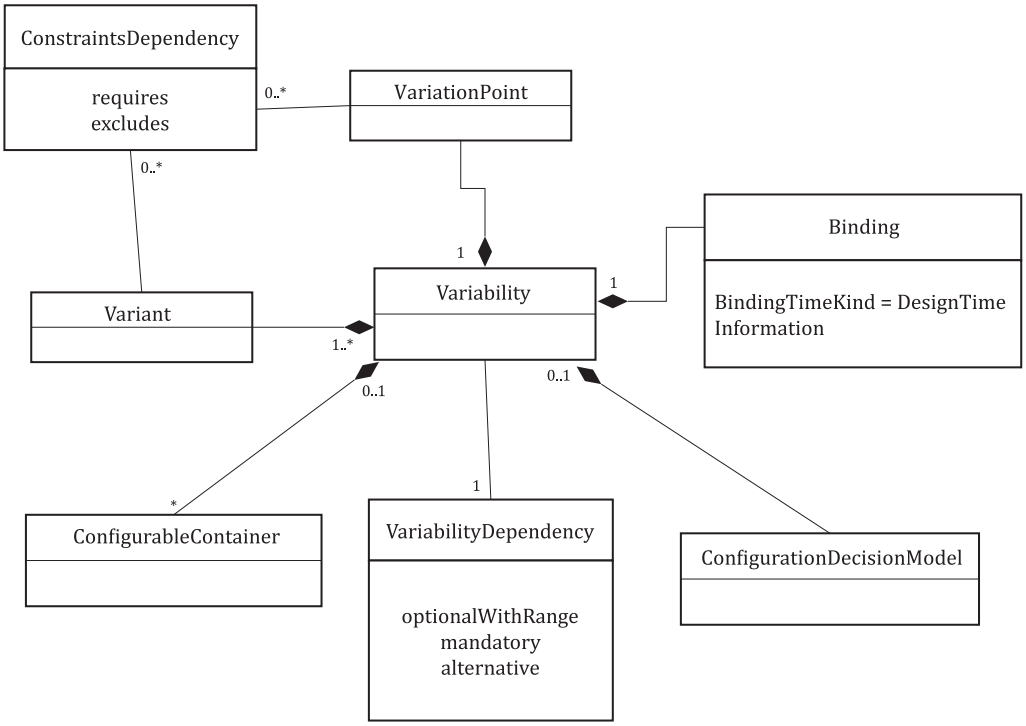

Figure B.1 — Variability specification elements in ADL

Annex C
(informative)

Architecture structure and texture example

The example architecture structure of the home automation applications is depicted in Figure C.1 that includes four layers, each of which has an internal structure consisting of subsystems.

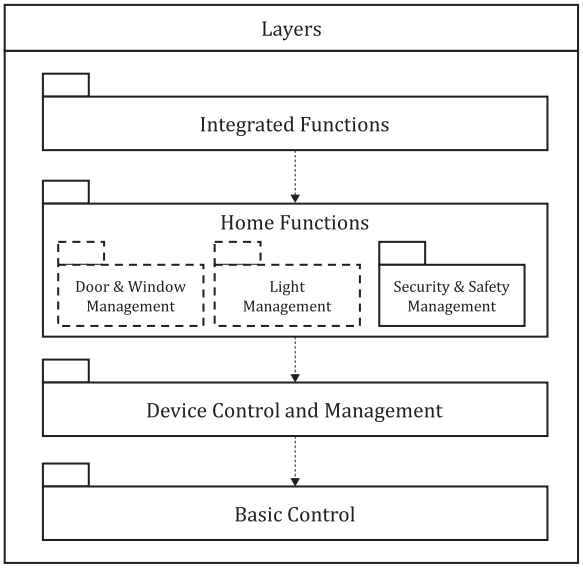

Figure C.1 — Package diagram of the home automation applications[21]

"Door & Window Management" and "Light Management" of the Home Functions layer are variable subsystems, which are depicted with dotted lines. The texture of the home automation applications in a package diagram contains:

— the use of layering in the structure, a hierarchy of layers, and subsystems, as described in Figure C.1;

— the use of the façade pattern for providing a single interface at the subsystem level;

— the use of high-priority processes for user interface handling and medium-priority processes for user functions.

Bibliography

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

[2] ISO/IEC 14102, *Information technology — Guideline for the evaluation and selection of CASE tools*

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

[4] ISO/IEC 15940, *Systems and software engineering — Software Engineering Environment Services*

[5] ISO/IEC/TR 19759, *Software Engineering — Guide to the software engineering body of knowledge (SWEBOK)*

[6] ISO/IEC 25000, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*

[7] ISO/IEC 26550, *Software and systems engineering — Reference model for product line engineering and management*

[8] ISO/IEC 26551, *Software and systems engineering — Tools and methods for product line requirements engineering*

[9] ISO/IEC 26553, *Information technology — Software and systems engineering — Tools and methods for product line realization*

[10] ISO/IEC 26554, *Information technology — Software and systems engineering — Tools and methods for product line testing*

[11] ISO/IEC 26555, *Software and systems engineering — Tools and methods for product line technical management*

[12] ISO/IEC 26556, *Information technology — Software and systems engineering — Tools and methods for product line organizational management*

[13] ISO/IEC 26557, *Software and systems engineering — Methods and tools for variability mechanisms in software and systems product line*

[14] ISO/IEC 26558, *Software and systems engineering — Methods and tools for variability modelling in software and systems product line*

[15] ISO/IEC 26559, *Software and systems engineering — Methods and tools for variability traceability in software and systems product line*

[16] ISO/IEC 26560[1)], *Software and systems engineering — Tools and methods for product line product management*

[17] ISO/IEC 26561[2)], *Software and systems engineering — Methods and tools for product line technical probe*

[18] ISO/IEC 26562[3)], *Software and systems engineering — Methods and tools for product line transition management*

[19] ISO/IEC/IEEE 42010:2011, *Systems and software engineering — Architecture description*

1) Under preparation. Stage at the time of publication: ISO/IEC FDIS 26560:2019.

2) Under preparation. Stage at the time of publication: ISO/IEC DIS 26561:2019.

3) Under preparation. Stage at the time of publication: ISO/IEC DIS 26562:2019.

[20] ISO/IEC/IEEE 42020:—[4], *Software, systems and enterprise — Architecture processes*

[21] Pohl K., Böckle G., Van Der Linden F.J. *Software Product Line Engineering: Foundations, Principles and Techniques.* Springer, 2005

[22] Linda M. Northrop, Paul C. Clements *A Framework for Software Product Line Practice, Version 5.0*. Software Engineering Institute, Carnegie Mellon University, July 2007

4) Under preparation. Stage at the time of publication: ISO/IEC/IEEE FDIS 42020:2019.

ICS 35.080

Price based on 61 pages