INTERNATIONAL STANDARD

ISO/IEC 26551

Second edition
2016-05-01

Software and systems engineering — Tools and methods for product line requirements engineering

Ingénierie du logiciel et des systèmes — Outils et méthodes pour l'ingénierie d'exigences pour gammes de produits

Reference number
ISO/IEC 26551:2016(E)

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

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 on the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the WTO principles in the Technical Barriers to Trade (TBT) see the following URL: Foreword - Supplementary information

The committee responsible for this document is ISO/IEC JTC 1, *Information technology*, Subcommittee SC 7, *Software and systems engineering*.

This second edition of ISO/IEC 26551 cancels and replaces the first edition (ISO/IEC 26551:2012), which has been technically revised.

Introduction

The main purpose of this International Standard is to establish a baseline for the capabilities of tools and methods of software and systems product line (SSPL) requirements engineering. This International Standard defines how the tools and methods can support the software and systems product line specific requirements engineering processes.

A decision for the initial boundaries of domain is made to define a product line scope before initiating domain requirements engineering processes. Domain requirements engineering is carried out in a comprehensive manner because common and variable requirements and captured variabilities have consequential impacts on member products in a product line. The outcomes of domain requirements engineering processes are transferred into the requirements of a family of products at the application requirements engineering processes. Therefore, requirements engineering tools and methods are to be considered (both engineering processes), namely domain requirements engineering, and application requirements engineering.

Product line requirements engineering can be differentiated from a single product requirement engineering because of the following reasons:

— There are two core processes in requirements engineering, domain requirements engineering and application requirements engineering. The major aims of the domain requirements engineering processes are to analyse commonality and variability for a family of products and to prepare necessary variability information for variability modeling. The major aims of the application requirements engineering processes are to define application specific requirements and bind variability defined in domain requirements engineering processes;

— It is essential to analyse the costs and benefits estimate of a product line and thereby, an organization can make a go/no-go decision. Moreover, the costs and benefits estimate plays a pivotal role as an indicator for assessing the effectiveness and efficiency of a product line.

A detailed comparison showing the differences in requirements engineeing tasks between single product and product line is described in Annex A.

This International Standard can be used in the following modes:

— by the users of this International Standard: to benefit people who develop, operate, and manage requirements engineering for software and systems product lines;

— by a product line organization: to provide guidance in the evaluation and selection for tools and methods for product line requirements engineering;

— by providers of tools and methods: to provide guidance in implementing or developing tools and methods by providing a comprehensive set of the capabilities of tools and methods for product line requirements engineering.

The ISO/IEC 26550 family of standards addresses both engineering and management processes and covers the key characteristics of product line development. The ISO/IEC 26550 family of standards provides an overview of the consecutive International Standards (i.e. this International Standard through ISO/IEC 26599), as well as the structure of the model:

ISO/IEC 26550, ISO/IEC 26551 and ISO/IEC 26555 are published. ISO/IEC 26557, ISO/IEC 26558 and ISO/IEC 26559 are to be published. ISO/IEC 26552, ISO/IEC 26553, ISO/IEC 26554, ISO/IEC 26556, ISO/IEC 26560, ISO/IEC 26561, ISO/IEC 26562, ISO/IEC 26563 are planned International Standards.

— Processes and capabilities of methods and tools for domain requirements engineering and application requirements engineering are provided in this International Standard;

— Processes and capabilities of methods and tools for domain design and application design are provided in ISO/IEC 26552 (International Standard under development);

— Processes and capabilities of methods and tools for domain realization and application realization are provided in ISO/IEC 26553 (International Standard under development);

— Processes and capabilities of methods and tools for domain testing and application testing are provided in ISO/IEC 26554 (International Standard under development);

— 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 (International Standard under development);

— 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 modeling 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 (International Standard under development);

— Processes and capabilities of methods and tools for technical probe are provided in ISO/IEC 26561 (International Standard under development);

— Processes and capabilities of methods and tools for transition management are provided in ISO/IEC 26562 (International Standard under development);

— Processes and capabilities of methods and tools for configuration management of asset are provided in ISO/IEC 26563 (International Standard under development);

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

Software and systems engineering — Tools and methods for product line requirements engineering

1 Scope

This International Standard, within the context of tools and methods of requirements engineering for software and systems product lines:

— provides the terms and definitions specific to requirements engineering for software and systems product lines and associated member products;

— defines process groups and their processes performed during product line requirements engineering (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 International Standard concerns processes and capabilities of requirements tools and methods for a family of products, not for a single system.

This International Standard is not applicable to physical artefacts. Instead, system-level artefacts and software lifecycle artefacts such as requirements documents, architectural data, validation plans, behavioural models, etc. are produced using methods and tools in this International Standard. In the case of the software components of a system, this International Standard can apply twice: once to handle the system elements of the product line and a second time to handle the software elements of the product line, if any. The product line processes are recursive within the different levels of products.

NOTE The requirements in this International Standard apply to the family of systems, software or services.

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.

3.1
application assets in requirements
application specific artefacts produced during application requirements engineering such as *application requirements specifications* (3.4) and application requirements models

3.2
application requirements elicitation
subprocess for identifying stakeholders relevant to an application, eliciting application specific requirements, and binding the appropriate variants

3.3
application requirements analysis
subprocess that understands all *application specific requirements* (3.8), scrutinizes incorrect and inconsistent application requirements through modelling, and then analyses and negotiates application requirements that cannot be satisfied through the domain requirements

3.4
application requirements specification
subprocess that documents the *application specific requirements* (3.8) and integrates it with the *domain requirements specification* (3.12) whose variants are bound

3.5
application requirements verification and validation
subprocess that confirms that the *application specific requirements* (3.8) are consistent and feasible and ensures that the bound variants satisfy the specific product's requirements

3.6
application requirements management
subprocess that manages traceability and changes on application requirements

3.7
aspect
special consideration within product line engineering process groups and tasks to which we can associate specialized methods and tools

3.8
asset proposal
artefact that includes major assets (functional areas and high-level common and variable features of all applications) that can be included in a product line with their quantified costs and benefits, and estimate results

3.9
application specific requirements
requirements specific to an application or requirements not covered in domain requirements

3.10
domain assets in requirements
reusable artefacts produced during domain requirements engineering such as *asset proposals* (3.7), *domain requirements specifications* (3.12), and domain requirements models

3.11
domain requirements elicitation
subprocess that identifies initial requirements from domain stakeholders for a product line

3.12
domain requirements analysis
subprocess that models domain requirements so as to analyse and scrutinize commonality/variability of a product line in requirements

3.13
domain requirements specification
subprocess that documents domain requirements for a product line based on domain analysis results

3.14
domain requirements verification and validation
subprocess that confirms that domain requirements are correct, consistent, and complete

3.15
domain requirements management
subprocess that manages traceability and changes with respect to domain requirements and their relevant domain/application artefacts

3.16
functional domain
categorized functions that are generally used together

3.17
production plan
description of how domain assets are to be used to develop member products in a product line

3.18
requirements traceability
traceabilities in domain and application requirements respectively and those between them

3.19
texture
architectural texture
collection of common development rules and constraints for realising the applications of a product line

3.20
variability in requirements
external and internal variability in requirements engineering

Note 1 to entry: Variability modelling and traceability with domain requirements artefacts are also addressed.

4 Reference model for product line requirements engineering

The methods and tools for product line requirements engineering should support systematic management and interaction of the domain and application requirements engineering processes. They also need to be adequately integrated with the other product line engineering lifecycle processes in order to enable traceability between all requirements artefacts and the related design, realization, and testing artefacts. In the rest of this International Standard, product line requirements engineering practices, methods, and tools are described in accordance with a framework focusing on product line requirements engineering (Figure 1).

Product line scoping leads and controls all work on a product line by creating and maintaining the product line scope through ongoing interactions with the domain and application requirements engineering.

Domain requirements engineering serves to:

— decompose features defined in product line scoping into initial requirements and elicit additional requirements and derived requirements from stakeholders and domain experts;

— analyse domain requirements with variabilities in requirements;

— model and simulate the static and behavioural constructs of domain requirements;

— document domain requirements specifications that can be bound by specific member products of a product line.

The complexity of variability grows in accordance with the complexity of a product line. Separating variability from domain requirements engineering mitigates this problem. Defining variability in requirements independently leads to a clear understanding of the necessary capabilities of tools and methods, and thus helps in selecting tools that support product line requirements engineering.

Application requirements engineering serves to:

— identify gaps between domain features and application specific features;

— reuse domain requirements from the asset repository and elicit application specific requirements;

— define application variability model by binding domain variability model and adding application specific variability;

— analyse and document application specific requirements;

— provide feedback to product line scoping and domain requirements engineering for the evolution of a product line.

The reference model for product line requirements engineering in Figure 1 is structured into five *processes: product line scoping, domain requirements engineering, variability management in requirements engineering, asset management in requirements engineering, and application requirements engineering*. Each process is divided into subprocesses to address technical management issues, 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.

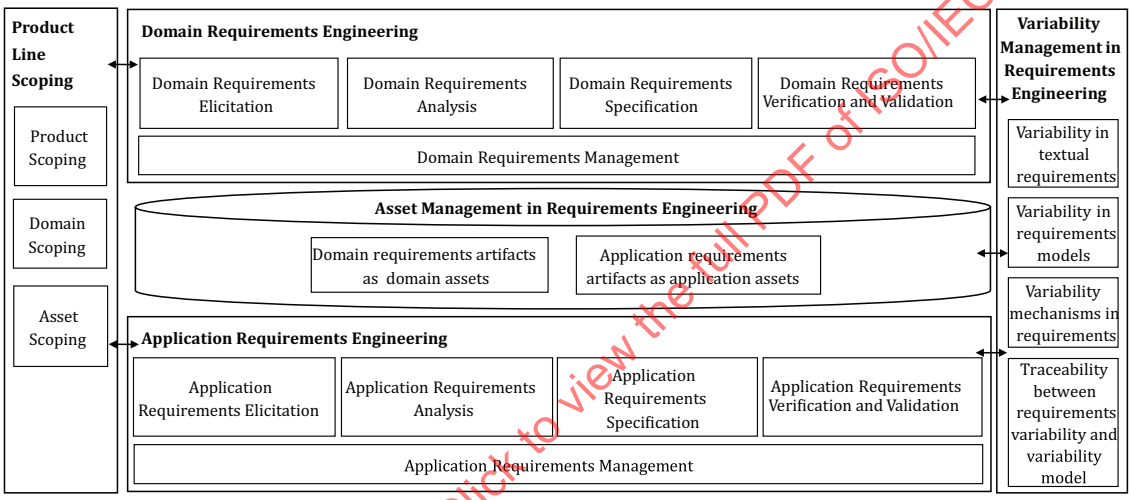

Figure 1 — Product line requirements engineering

Product line scoping defines the member products and their major (externally visible) features, analyses the products from an economic point of view, and controls and schedules producing the product line and its products. The major result of this process is the asset proposals. Asset proposal includes major assets (functional domains and high-level common and variable features) for a product line with their quantified costs and benefits, and ROIs expected from a product line development. More than one asset proposal can be made to find out an optimal set of products and assets. Domain and application requirements engineering start from the features defined in the asset proposals. Product line scoping shall serve to do the following and to define the capabilities of tools and methods for supporting them:

— *Product scoping* determines the product portfolio definition, and provides a roadmap for releasing specific applications to customers or to market;

— *Domain scoping* identifies and bounds the functional domains to provide sufficient reuse potential;

— *Asset scoping* identifies reusable assets, estimates the cost/benefit to justify the product line initiation.

Domain requirements engineering begins with using the outcomes of product line scoping. It comprehensively captures the initial domain requirements for a product line, and constructs an initial-requirements specification including a variability model. It also provides feedback on the changes required in the feature sets and the product roadmap as a whole to organizational business management process group. Domain requirements engineering documents domain requirements specifications for the later use in domain design and in application requirements engineering. Domain

requirements engineering shall serve to do the following and to define the capabilities of tools and methods for supporting them:

— *Domain requirements elicitation* captures initial domain requirements and anticipated variations of those requirements;

— *Domain requirements analysis* identifies functional and non-functional requirements with the variabilities of those requirements;

— *Domain requirements specification* documents domain requirements based on analysis results;

— *Domain requirements validation and verification* confirms that the specified domain requirements are consistent and feasible, and ensures that all products' requirements within a product line are well understood;

— *Domain requirements management* provides management services for the dual nature of the requirements engineering, i.e. domain and application requirements engineering.

Variability management in requirements engineering should be conducted in parallel with domain requirements engineering because variability models are clarified and modified gradually together with the domain and application requirements. Variability modeling starts as part of domain requirements elicitation and continuously evolves throughout the product line life-cycle. This process is responsible for a variability model which documents the external variability explicitly. As domain requirements engineering activities proceed, some additional internal variabilities may be added to the variability model. Variability management in requirements engineering shall serve to do the following and to define the capabilities of tools and methods for supporting them:

— *Variability in textual requirements* expresses and documents variability in requirements using natural language and makes them explicit;

— *Variability in requirements model* expresses and documents variability in requirements using modelling language and makes them explicit;

— *Variability mechanisms in requirements* categorize variability mechanisms that are to be used as part of requirements;

— *Traceability between requirements variability and a variability model* establishes and maintains links between *textual requirements /requirements models* and a variability model.

During asset management in requirements engineering, requirements artefacts resulting from domain requirements engineering are structured as domain assets. Variability, as well as commonality in requirements, is managed as domain assets. In addition, application requirements artefacts with high reusability potentials are identified as potential domain assets. Asset management in requirements engineering adds or develops extra elaborations and glues to requirements assets to be used effectively and efficiently. The relationship among requirements domain assets for being reused successfully or managing changes on them is also analysed in this process area. Processes for configuring domain assets and managing them in asset repository refer to asset management of ISO/IEC 26555. Asset management in requirements engineering shall serve to do the following and to define the capabilities of tools and methods for supporting them:

— *Domain requirements artefacts* as domain assets identify and develop necessary information to help application engineers reuse requirements assets in their application development;

— *Application requirements artefacts* as application assets identify and manage application requirements artefacts as assets to be referred by the application later.

Application requirements engineering identifies specific requirements for each product line member. It starts to assess the reusability of existing common and variable requirements to fully leverage a product line platform. It can also provide feedback to domain requirements engineering so as to make a decision on incorporating application requirements assets into domain assets. Application

requirements engineering shall serve to do the following and to define the capabilities of tools and methods for supporting them:

— *Application requirements elicitation* identifies gaps between domain features and application specific features, reuses domain requirements from the asset repository, and elicits application specific requirements;

— *Application requirements analysis* abstracts, organizes, and models application specific requirements. This subprocess has to ensure that all requirements of the application stakeholders are understood, and has to scrutinize the correctness, completeness, and consistencies of application requirements;

— *Application requirements specification* documents the analysed application specific requirements with the bound portion of domain requirements specification;

— *Application requirements verification and validation* confirms that the specific product requirements are consistent and feasible, and ensures that the bound variants are relevant to the specific product requirements;

— *Application requirements management* provides management services for subsequent changes of the member product's requirements.

NOTE 1 Processes of product line requirements engineering are compatible with ISO/IEC 12207 and ISO/IEC/IEEE 15288 as depicted in process mapping with ISO/IEC 12207, ISO/IEC/IEEE 15288, and ISO/IEC 29148 (Annex B).

The identification and analysis of the aspects for the product line requirements engineering enable an organization to understand the requirements engineering processes and to formulate a strategy for the successful implementation of the concept. Requirements engineering processes for product lines should be defined in terms of these aspects, and capabilities of tools and methods for supporting these processes should be identified on the bases of these aspects:

NOTE 2 The relationship between sub processes, key aspects, and associated tool and method capabilities is depicted in Annex C.

Table 1 shows the key aspects for each characteristic of product line requirements engineering.

Table 1 — Key aspects for identifying product line-specific requirements engineering tasks

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

— *Application engineering:* Domain requirements should be reused and external variability for specific application should be bound. As a result, application requirements are specified;

— *Asset:* Domain requirements engineering provides a common portion of requirements (domain requirements) to application engineering through asset repository. Therefore, the domain assets of requirements engineering and management is a distinguished aspect;

— *Binding:* Binding in requirements engineering should consider reflecting the external variability. Thus, binding is a distinct aspect of the product line development;

— *Collaboration:* Since the domain requirements engineering and application requirements engineering can be performed in parallel, collaborations are necessary between engineering teams, as well as those among processes such as domain assets, variability management, product management, scoping, etc. This makes collaboration an important aspect in a product line development environment;

— *Configuration:* Configuration of products and artefacts of product line requirements engineering can be multidimensional, i.e. exist in time and space. Maintaining the integrity of those dimensions is an important aspect;

— *Domain architecture:* Domain requirements engineering provides domain requirements as a major input for establishing reference architecture;

— *Domain engineering:* Domain requirements engineering process does not exist in a single product development. However, this process is necessary for a product line;

— *Enabling technology support:* Technologies that are needed to enable product line requirements engineering are a key success factor of product line implementation;

— *Measurement and tracking:* In product line requirements engineering, data collection, measures, and tracking need to consider domain engineering, application engineering, product management, and domain assets. This means that the measurements for product line are multidimensional and thus the required activities, roles, procedures, tools, and methods should be considered;

— *Platform:* Platform enables to reuse common elements (e.g. artefacts, components, connectors, etc.) among products. Product line requirements engineering artefacts are key elements of a platform;

— *Product management:* Requirements engineering should deal with reusability of the product line from the foreseen product line strategy. Since requirements continuously evolve in accordance with risks and opportunities, product management should monitor and support the evolution of requirements;

— *Reusability:* The reusability of assets from requirements engineering processes is closely related to achieve the overall goal of a product line. Achieving reusability throughout the product line requirements engineering is a differentiating aspect;

— *Texture:* Product line requirements are major inputs for establishing texture. Particularly for developing rules and specific detail of functional and non-functional requirements, corresponding architectural texture have to be established;

— *Traceability:* There exist various kinds of trace links between variabilities and the artefacts from domain/application requirements engineering. It is necessary to develop a traceability scheme to handle tracing;

— *Validation and verification:* In product line development environments, provisioning of objective evidences for validation and verification of requirements from various viewpoints (e.g. domain, application, variability, domain assets, etc.) is an important aspect, so differentiated schemes with single product development should be provided;

— *Variability:* Variability in requirements engineering mainly deals with external variability related to a reusability strategy, which is not a concern of single product development.

5 Product line scoping

A product line organization needs to determine with which products and features constitute a product line, and thereafter the organization determines which features are implemented by reusing or adapting legacy assets or which of them are newly developed. Using objective and quantitative endeavours, the organization estimates economic benefits expected from a product line and makes a go/no-go decision for product line initiation base on the economic benefits. A product line scoping process includes three key sub processes:

— *Product scoping* determines the potential member products and initial common and variable features of those base on market inputs;

— *Domain scoping* decomposes domains into subdomains (or functional domains) and maps the initial features determined in product scoping to the subdomains;

— *Asset scoping* identifies reusable assets, calculates the cost/benefit estimate to justify the product line initiation, and provides a roadmap of a product line.

The above three types of scoping can be iterative.

5.1 Product scoping

5.1.1 Purpose of product scoping

The purpose of product scoping is to determine product portfolio definition:

1) The products that the product line organization should be developing, producing, marketing, and selling;

2) The common and variable features that the products should provide in satisfying customer needs and reaching the long and short term business objectives of the product line organization; and

3) A schedule for introducing the products to markets.

5.1.1.1 Inputs

— *A Product definition strategy:* two main types are customer-driven and producer-driven strategies. In a customer-driven strategy, member products of a product line are defined on demand based on market inputs especially based on customer needs and wants. In a producer-driven strategy, the product line organization defines the products.

— *Market information:* for the product scoping period, which comes from the following:

  — market segments, their characteristics and expected future evolutions;

  — customer needs and their expected evolutions;

  — technologies and their expected evolutions;

  — existing and potential competitors; and

  — competitive products and their expected evolutions.

— *Other internal information:* necessary to develop and maintain an understanding and control of the product line organization as a whole.

5.1.1.2 Outcomes

— *Product portfolio definition,* i.e. a (initial) product roadmap including products, their externally visible common and variable feature sets, and a schedule for market introduction is established.

— A *High-level production plan* is defined.

5.1.1.3 Tasks

— *Structure information to be used for scoping*. Market information, information of candidate member products of a product line, and business goals are structured in a form that is easy to refer for processing further scoping activities.

— *Identify products*. Identification of potential member products, including existing ones and new ones, is performed to meet market requirements and business goals. Existing products, prototypes, and other assets that may be reused are identified and analysed. Information about the products may be gathered from internal (e.g. domain experts) and external (e.g. outside experts of component vendors) resources.

— *Analyse common and variable features*. An initial description of a set of systems is made by analysing which features are inside or outside the product line.

— *Define the product portfolio*. Member products and features that constitute a product line are determined. The decision of the product portfolio should be consistent with an organizational product definition strategy (e.g. *customer-driven or producer-driven strategies*).

5.1.2 Structure information to be used for scoping

Information to be used for scoping is extracted and structured so that participants in scoping make a correct decision.

The method should support structuring information to be used for scoping with the following capabilities:

— selecting information to be used for scoping;

— providing a template for structuring necessary information;

— figuring out relationship among selected information set and their rationale;

— reorganizing information in accordance with the provided template.

A tool should support structuring information to be used for scoping by allowing the user to do the following:

— accessing information to be used for scoping (or providing interfaces for accessing);

— allowing documentation using the template;

— supporting maintenance of the structured information;

— allowing reuse of the structured information.

5.1.3 Identify products

The goal of this task is to find member products for a product line, a release plan, and their high level features.

The method should support identifying member products with the following capabilities:

— identifying existing and/or future member products that may be produced in the product line;

— generating a possible member products list;

— analysing the generated possible member products list (using goals, market definition, etc.);

— selecting and defining an initial member products list;

— verifying the defined initial member products list.

A tool should support the identification of member products through the following means:

— supporting team collaboration;

— supporting the documentation of the process for identifying member products.

And it should allow the user to do the following:

— collect information for existing and/or future products that may be produced in the product line;

— cluster collected candidate products information;

— manage clustered candidate products information;

— visualize, model, and organize information about existing, future, and hypothetical products as candidates for inclusion in a product line.

5.1.4 Analyse common and variable features

The common and variable features that are analysed in this task have high-level abstractions. These common and variable features including product quality are used for confirming whether or not the product line satisfies an organizational expectation on cost reduction through reuse.

The method should support analysing common and variable features with the following capabilities:

— analysing features (high level) characterizing products in the product line;

— analysing product quality (that can be refined as quality attributes in domain requirement engineering);

— discriminating common and variable features;

— modelling features (using an appropriate feature modelling notation);

— evaluating common and variable features for ensuring that major features of products are identified (through workshops, brainstorming and categorizing, etc.);

— providing a document template for documenting common and variable features.

A tool should support analysing common and variable features by allowing the user to do the following:

— visualizing common/variable features and product qualities;

— allowing documentation using the template;

— allowing the analysed results being referred in the further scoping and requirements engineering activities;

— supporting documentation of the rationale relate to the decision making and making it be referred in the further scoping and requirements engineering activities.

5.1.5 Define a product portfolio

The method should support defining a product portfolio with the following capabilities:

— structuring issues concerning market information, an initial products list, and their high level features;

— making decisions using a decision process (e.g. brainstorming, Nominal Group technique, Delphi);

— verifying the decision;

— providing checklists for verifying decisions.

A tool should support defining a product portfolio through the following means:

— providing templates for assisting in decision making;

— documenting the defined product portfolio.

5.2 Domain scoping

5.2.1 Purpose of domain scoping

The purpose of domain scoping is to decompose functional domains and map initial features to the functional domains to reduce complexity and to promote reusability of a product line.

5.2.1.1 Inputs

— *Product portfolio definition.*

— *Information collected from domain experts and existing products experiences.*

5.2.1.2 Outcomes

— *A domain scope definition* (including functional (sub-functional) domains and their common and variable features) is established.

— *A set of high-level features* is defined.

— *The product portfolio definition* is refined.

5.2.1.3 Tasks

— *Identify functional domains.* Find, analyse, and categorize functions to maximize reusability. Functional domains may be derived by grouping features and hierarchically defined (i.e. including sub-areas).

— *Map features to functional domains.* Identified features are distributed to identified functional domains.

— *Define domain scope.* Assess and select functional domains that best lend themselves to reuse.

5.2.2 Identify functional domains

Functional domains are identified and classified by experts with sufficient experience and knowledge of products in a product line. The technical feasibility of functional domains and features is validated and features are further refined.

The method should support identifying functional domains with the following capabilities:

— analysing the functionalities of domains of the product line (rules of inclusion or exclusion, structure diagram, context diagram, etc. are example approaches for defining boundary of domain);

— categorizing functionalities that are used together;

— documenting and structuring identified functional domains;

— verifying categorized functional domains.

A tool should support identifying functional domains with the following capabilities:

— supporting team collaboration;

— displaying analysed and categorized results.

And it should allow the user to do the following:

— manage analysed and categorized functional domains consistently;

— document the functional domains;

— manage the relationship between functional domains and associated analyses.

5.2.3 Map features to functional domains

A mapping table for identified functional domains, products, and features is established for representing the relationships among them.

The method should support mapping functional domains to features with the following capabilities:

— analysing relationships between functional domains and features;

— mapping features to functional domains;

— refining functional domains and features;

— validating the mapping results.

A tool should support mapping functional domains to features with the following capabilities:

— displaying functional domains and features (e.g. an excel worksheet composed of a functional domains column and a features row);

— allowing documentation of the rationale for defining functional domain;

and it should allow the user to do the following:

— create a relationship between functional domains and features;

— manage the relationship between functional domains and features.

5.2.4 Define domain scope

A domain scope is defined for relationships among the functional domains, products, and features.

The method should support defining a domain scope with the following capabilities:

— assessing functional domains according to the variability and performance dimensions;

— selecting functional domains using an appropriate decision-making process (e.g. brainstorming, Nominal Group Technique (NGT), Delphi);

— managing issues relating to functional domain selection;

— verifying the defined domain scope.

A tool should support defining a domain scope with the following capabilities:

— displaying a domain scope definition;

— providing a collaboration environment for reviewing, commenting, and communicating;

and it should allow the user to do the following:

— assess, verify, and document the domain scope definition;

— raise and resolve issues relating to the domain scope.

5.3 Asset scoping

5.3.1 Purpose of asset scoping

The purpose of asset scoping is to identify potential reusable assets and to estimate the economic benefits from reusing the proposed assets. Assets are evaluated based on the effort required to engineer them, quantified using effort estimate functions derived from historical data collected from existing related products and domain experts.

5.3.1.1 Inputs

— *A Domain scope definition* (including functional domains and their common and variable features).

— *Estimate models and/or other information of current practices.*

— *Available measurement data* (i.e. historical data for related domains).

5.3.1.2 Outcomes

— *Asset proposals* (including functions, features, and effort estimates) are proposed.

— *A quantified domain definition* is established.

— *ROI estimates* are produced.

— *List of existing assets* is provided.

5.3.1.3 Tasks

— *Gather historical data from existing single products.* Identity and gather information about existing products to better understand the domain and identify potentially reusable assets.

— *Estimate additional effort required to adapt potential assets.* Estimate the additional effort required for adapting potential assets to the product line.

— *Estimate expected development effort for new products in the product portfolio definition.* Estimate the cost, effort, and risk of including new products into the product portfolio. This calculation is based on learning from existing related products, development constraints, systems/software attributes, and market attributes.

— *Estimate economic benefits from reusing proposed assets.* Estimate the economic benefit from reusing existing assets, from reusing newly developed domain assets, and from reducing rework and failure due to quality improvements.

— *Derive asset proposals from economic evaluation results.* Define the asset proposals including assets (functional domains and features) and their economic evaluation results. Establish asset proposals by extending the domain definition. Create detailed and quantified effort estimates for all assets in the domain.

5.3.2 Gather historical data from existing single products

Effort estimate can be conducted based on the historical data such as measurement data cumulated from existing relevant products for a specified period of time.

The method should support gathering historical data from existing single products with the following capabilities:

— defining what kinds of historical data should be collected;

— integrating data from various sources;

— searching data;

— verifying the validity of collected data.

A tool should support gathering historical data from existing single products by allowing the user to do the following:

— allowing documentation for the results of gathering historical data from existing single products;

— importing sources from other tools;

— providing management capability for historical data.

5.3.3 Estimate additional effort required to adapt potential assets

While some assets are reused as they are, some others can be reused after adaptation. Therefore, estimating adaptation efforts is necessary before developing them as domain assets because if adaptation efforts are too high, it is difficult to expect benefits from domain asset reuse.

The method should support estimating additional effort required to adapt potential assets with the following capabilities:

— analysing relationships between existing and new products;

— analysing factors for expected development effort for the new products;

— defining effort estimate functions for the new products;

— providing metrics that can be used for effort estimates;

— allowing adding a new metric if necessary.

A tool should support estimating expected additional efforts required to adapt potential assets with the following capabilities:

— displaying the relationship between assets, functional domains, features, and products, and its influence on economic benefit;

— providing metrics to be used to estimate so that tool users can choose among them;

— allowing adding a new metric if necessary.

5.3.4 Estimate expected development effort for new products in the product portfolio definition

Some products included in a product portfolio definition are new ones, so the efforts for them have to be estimated based on the measurement data gathered from existing relevant products. This estimate should be considered separately for estimating the whole benefits from a product line.

The method should support estimating the expected development effort for new products in the product portfolio definition with the following capabilities:

— analysing relationships between existing and new products;

— analysing factors for the expected development effort for the new products;

— defining effort estimate functions for the new products.

A tool should support estimating expected development efforts for new products in the product portfolio definition with the following capabilities:

— providing metrics for effort estimate that tool users can choose;

— allowing adding a new metric if necessary;

— displaying the relationship between assets, functional domains, features, and products, and its influence on the economic benefit.

And it should allow the user to do the following:

— document estimated results of expected economic benefits for new products.

5.3.5 Estimate economic benefits from reusing proposed assets

Effort estimate functions for calculating the costs and benefits expected from proposed assets are developed based on historical data and experiences.

The method should support estimating the economic benefits from reusing proposed assets with the following capabilities:

— collecting historical data;

— analysing historical data;

— estimating expected benefits from historical information.

A tool should support estimating benefits from reusing proposed assets with the following capabilities:

— documenting the process of estimating benefits from reusing proposed assets;

— importing historical data from various sources;

— displaying the relation among domains, sub-domains, products, and benefits.

5.3.6 Derive asset proposals from economic evaluation results

The goal of this task is to integrate the previously estimated results for deriving asset proposals that include costs and benefits due to a product line.

The method should support deriving asset proposals from economic data with the following capabilities:

— integrating effort estimates;

— making a decision on the asset proposals using a decision process (e.g. brainstorming, Nominal Group Technique (NGT), Delphi);

— verifying the decision making.

A tool should support deriving asset proposals from economic data with the following capabilities:

— documenting asset proposals;

— displaying asset proposals in a graphic user interface;

— offering automatic calculation using input data and pre-defined cost-benefit functions;

— highlighting asset lists that are necessary to develop as domain assets;

— allowing asset proposals to be used for tracing the gap between the estimated and actual cost/benefit results.

6 Domain requirements engineering

Functional and non-functional requirements are identified and documented with those external variabilities in domain requirements engineering. Domain requirements engineering should adhere to the features defined in the asset proposals produced at the product line scoping. These features are the key sources of domain requirements. The defined domain requirements are referred by other domain and application engineering processes.

Domain requirements engineering has five basic subprocesses:

— *Domain requirements elicitation.* The purpose of this subprocess is to decompose features defined in product line scoping into initial requirements and to elicit additional requirements and derived requirements from stakeholders and domain experts. This subprocess also captures variations anticipated over the product line members explicitly from domain stakeholders;

— *Domain requirements analysis.* The purpose of this subprocess is to find commonalities and identify variabilities based on the elicited domain needs in order to achieve targeted reuse goals of the product line. Domain requirements should be analysed more vigorously to find out more common

and fewer unique requirements, so as to improve economic viabilities and benefits. The product line scope is a major input to determine the boundaries of a product line;

— *Domain requirements specification.* The purpose of this subprocess is to document the analysed domain requirements, where this specification includes both common and variable requirements. Variability in domain requirements specification is bound by specific member products of a product line;

— *Domain requirements validation and verification.* The purpose of this subprocess is to validate and verify that the product line requirements are correct, complete, consistent, and they make sense for each product whereas product-specific requirements should be validated and verified at the application requirements validation and verification subprocess;

— *Domain requirements management.* The purpose of this service is to manage traceabilities and changes on domain requirements. Because of the dual nature of the product line and the domain assets that are commonly used across applications, the complexity of traceability and change management are high, so an effective counter-plan for coping with the complexities is essential.

6.1 Domain requirements elicitation

6.1.1 Purpose of the domain requirements elicitation

The purpose of the domain requirements elicitation is to decompose, discover, review, and understand stakeholders' needs and constraints related to a platform for a product line based on the outcomes of the product line scoping.

6.1.1.1 Inputs

— *Information from sources* (stakeholders of a domain).

— *Asset proposal* (from product line scoping).

— *A set of high-level features* (from product line scoping).

— *List of existing assets.*

6.1.1.2 Outcomes

— *Domain information* (each application's goals, problem definitions, current and potential users, requirements, stakeholders, etc.) is maintained.

— *Domain information sources link*s are established.

— *Context diagram* is defined.

— *Refined common and variable feature set* is produced.

— *Classified domain requirements* (initial set of common and variable requirements) are produced.

6.1.1.3 Tasks

— *Draw a context diagram.* A context diagram for the product line captures high-level entities and those relationships (e.g. product line users, physical environment, or other systems related to the product line).

— *Gather domain information.* Domain information is captured by holding workshops, examining the current systems, and interviewing domain and application engineers.

— *Identify initial domain requirements.* Domain requirements engineers review the gathered domain information and identify initial domain requirements from the users' points of view. A set of high-level features, including the common and variable requirements that are produced during product

line scoping, are revised in more detail. Variabilities are refined and linked to the related models developed as part of other processes.

— *Review the elicited initial domain requirements.* Elicited initial domain requirements are checked to see whether they violate the product line's scope and the requirements of the stakeholders. They are revised as necessary (including a comparison with asset proposals).

The representatives and relevant stakeholders for domain requirements elicitation are found (e.g. potential customers and other stakeholders such as domain experts and application engineers). Stakeholders include people who have knowledge about customers, competitors, market trends, and member products in a product line.

NOTE Examples of techniques to elicit requirements:

— Goal Question Metric (GQM);

— Quality Function Deployment (QFD);

— Kano modelling;

— Brainstorming.

6.1.2 Draw a context diagram

A context diagram describes the relationships between the domain environment and product line members. It also depicts the relationships among product line members. This task serves to extract interacting objects and their interactions within a product line.

The method should support drawing a context diagram with the following capabilities:

— identifying the existing context diagrams for each product line member;

— analysing the domain environment (domain users, stakeholders, external systems, etc.);

— analysing operational environment;

— analysing relations between a family of products and identified domain entities;

— diagramming the domain context.

A tool should support drawing a context diagram with the following capabilities:

— providing documentation templates for domain entities with respect to relations between product line members and identified domain entities;

— providing checklists for identifying domain entities;

— allowing drawing up a domain context diagram.

6.1.3 Gather domain information

Domain information is collected from a variety of sources by analysing, for example, the requirements currently met by the applications within a product line; the problems relevant stakeholders face with the current applications, and change requests that are pending for the applications.

The method should support gathering domain information with the following capabilities:

— identifying domain information sources (e.g. product portfolio, high-level production plan, domain definition, and asset proposals from scoping, interviews and surveys using context-free questions, and requirements elicitation workshops and sessions);

— identifying stakeholders relevant to the product line;

— gathering domain information (product line goals, stakeholders, goals and requirements for existing product line members, goals and problems that are supposed to be tackled by the new applications, etc.);

— validating the gathered domain information.

A tool should support gathering domain information with the following capabilities:

— importing domain information into the domain requirements elicitation workspace;

— providing templates for documenting the gathered domain information;

— providing repository (registry) for preserving domain information;

— maintaining trace links between domain information and its sources.

6.1.4 Identify initial domain requirements

The goal of this task is to capture initial requirements from the product line stakeholders.

The method should support identifying initial domain requirements with the following capabilities:

— reviewing gathered domain information;

— refining high-level features to lower level features;

— identifying requirements related to features;

— collecting additional requirements for new applications within a product line;

— describing domain requirements (e.g. ConOps, goal model, use case diagram, feature model);

— developing domain usage scenarios from the stakeholders' points of view.

A tool should support identifying initial domain requirements with the following capabilities:

— providing a browsing, reviewing, and commenting environment on domain information;

— providing templates for documenting elicited domain requirements;

— modelling elicited requirements from the stakeholders' points of view.

And it should allow the user to do the following:

— describe domain requirements (standard template);

— describe domain usage scenarios (standard template).

6.1.5 Review the elicited initial domain requirements

Captured initial domain requirements are reviewed to ensure their completeness and correctness.

The method should support reviewing the elicited initial domain requirements with the following capabilities:

— establishing traceability between elicited requirements and sources;

— verifying completeness and correctness of elicited results;

— reviewing conflicts among applications within a product line;

— resolving domain requirements problem (documenting conflicts unresolved in order to be analysed and resolved as part of requirements analysis);

— documenting elicited and classified requirements.

A tool should support reviewing the elicited initial domain requirements with the following capabilities:

— maintaining traceabilities between the elicited requirements and their sources;

— providing templates for documenting elicitation results;

— maintaining checklists for verification.

6.2 Domain requirements analysis

6.2.1 Purpose of the domain requirements analysis

The purpose of this subprocess is to define functional requirements, quality requirements, and risks in the requirements and analyse their feasibility. As the requirements are analysed, variability models are refined consistent with previous models, and a domain requirements analysis is conducted with the variability in requirements process at the same time for managing variability models as a whole.

6.2.1.1 Inputs

— *Elicited domain requirements.*

— *High-level domain requirements models* (e.g. use case models, feature models, context diagrams).

— *Initial set of common and variable features.*

— *Initial set of common and variable requirements.*

6.2.1.2 Outcomes

— *Functional and non-functional domain requirements* (including constraints, dependencies, and priorities) are established.

— *Domain requirements model* (commonality model is established by refining use case models, feature models, and context diagrams produced as part of requirements elicitation are refined and are established.

— *Information relevant to variabilities.* Variabilities in the level of domain requirements (variabilities, dependencies, constraints) are defined.

— *Technical/economic feasibility analysis results* are produced.

— *Checklist for review* is defined.

— *Conceptual test cases and scenarios for domain requirements* are produced.

6.2.1.3 Tasks

— *Classify and balance initial domain requirements.* Initial domain requirements are classified into functionality, quality attributes, constraints, and other non-functional requirements categories. The granularity levels of initial domain requirements are balanced by decomposing high level requirements into low level ones and aggregating detailed requirements into higher level ones.

— *Analyse commonalities and variabilities.* Requirements that are common for a family of products are analysed. Requirements that vary systematically from application to application (variabilities) are analysed. Variability analysis includes the identification of variation points, variants, and dependencies.

— *Model domain requirements.* Domain requirements and their interdependencies are visualized at a high level of abstraction to reveal incorrect, inconsistent, missing, and superfluous requirements.

Domain requirements models should deal with commonality and variability that are traceable to a domain variability model.

— *Create prototypes and analyse feasibility.* Prototypes can be used to evaluate the feasibility of implementing certain critical requirements from numerous viewpoints (e.g. acceptable cost and performance), and understand/determine the risks and priorities of the requirements.

— *Develop conceptual test cases and scenarios for acceptance testing.* Conceptual test cases and scenarios for acceptance testing are derived. These are commonly applied for all family members. These scenarios and cases may be derived from requirements models like use case diagrams.

— *Review the analysed domain requirements.* Analysed domain requirements are reviewed for identifying and correcting incompleteness and incorrectness.

6.2.2 Classify and balance initial domain requirements

This task serves for classifying and balancing initial domain requirements. Goals, business rules, functional requirements, quality attributes, constraints, etc. are classified with respect to their common and variable characteristics, and their granularities, incorrectness, and contradiction are coordinated for further analysis.

The method should support classifying and balancing initial domain requirements with the following capabilities:

— defining guidelines for balancing the level of abstractions of initial requirements;

— decomposing the high-level initial domain requirements into lower-level ones;

— aggregating low-level initial domain requirements to higher level ones;

— identifying conflicts among the initial domain requirements.

A tool should support classifying and balancing initial domain with the following capabilities:

— importing elicited initial domain requirements and relevant information into the domain analysis workspace;

— supporting editorial environment for decomposition or aggregation of initial domain requirements;

— allowing discrimination of common and variable initial requirements;

— providing documentation templates for describing the classified initial domain requirements.

6.2.3 Analyse commonalities and variabilities

In this task, the common and variable requirements are analysed. In case of variabilities variation points, variants, binding times, dependencies, and constraints are also defined.

The method should support analysing commonalities and variabilities with the following capabilities:

— analysing initial common and variable requirements (e.g. by mapping requirements to features; by using modelling concepts for product lines);

— performing detailed analysis of requirements in order to determine common and variable requirements across all applications (e.g. by mapping domain requirements with member products' requirements and by applying organizational rules for deciding commonality and variability);

— determining common and variable requirements;

— analysing attributes characterizing variable requirements (e.g. binding times, variation points, and their variants and types;

— providing rationale and assumptions (if necessary) for the common and variable requirements decision;

— validating analysed commonalities and variabilities (including consistencies among variabilities).

A tool should support analysing commonalities and variabilities with the following capabilities:

— providing analysis workspace for deciding commonalities and variabilities;

— providing a documentation template for describing commonalities and variabilities in the requirements level;

— importing relevant information into the domain analysis workspace;

— showing relevant commonality and variability information that are already defined.

6.2.4 Model domain requirements

Requirements modelling should be conducted for ensuring completeness and feasibility of identified domain requirements.

The method should support modelling domain requirements with the following capabilities:

— modelling requirements (by using text or natural language or visual modelling);

— analysing requirements models for their completeness, correctness, feasibility, and verifiability;

— refining functional requirements (e.g. from scenarios, use cases);

— refining non-functional requirements.

A tool should support modelling domain requirements with the following capabilities:

— supporting domain requirements modelling or providing an interface for interacting with the modelling tool;

— allowing traces with domain requirements;

— providing a documentation template for documenting the domain requirements model.

6.2.5 Create prototypes and analyse feasibility

Prototyping is conducted for requirements that are ambiguous or have technical difficulties, and thereafter feasibilities are analysed based upon the prototypes.

The method should support creating prototypes and analysing feasibility with the following capabilities:

— prioritizing domain requirements;

— choosing requirements to prototype;

— deciding what kinds of information to collect;

— prototyping;

— gathering feedback and other necessary information from stakeholders designing and using the prototypes;

— analysing feasibility using the prototyping results.

A tool should support creating prototypes and analysing feasibility with the following capabilities:

— generating and storing user defined/tool provided checklists or templates for various analyses, (e.g. technical, economical, and operational analyses);

— storing and managing contents for rationale of feasibility analysis;

— exchanging information pertinent to prototyping with external tools.

6.2.6 Develop conceptual test cases and scenarios for acceptance testing

Test cases and scenarios used in acceptance testing are developed in this task. Requirements for which no test cases can be developed are to be further clarified before resuming this task.

The method should support developing conceptual test cases and scenarios for acceptance testing with the following capabilities:

— driving conceptual test cases and scenarios to be used in acceptance testing (conceptual test cases and scenarios should deal with variable requirements) based on domain requirements (domain requirements models or spec.);

— producing reusable conceptual test cases and scenarios in application requirements engineering;

— allocating a unique ID for each test case and scenarios.

A tool should support developing conceptual test cases and scenarios for acceptance testing with the following capabilities:

— supporting automatic/semi-automatic derivation from domain requirements;

— allowing variabilities in conceptual test cases and scenarios for later binding in each member product;

— providing a documentation template for documenting derived test cases and scenarios;

— allowing users to find and access relevant information (e.g. functional and non-functional requirements, variability models);

— generating test cases semi/automatically from domain requirements.

6.2.7 Review the analysed domain requirements

The goal of this task is to ensure that domain requirements are unambiguous and include all stakeholders' requirements and that commonalities and variabilities are reviewed for ensuring correctness, economic and technological feasibilities, and completeness of common requirements.

The method should support reviewing the analysed domain requirements with the following capabilities:

— identifying the ambiguous or unverifiable analysed requirements;

— analysing them with relevant stakeholders and suggesting changes;

— analysing the impacts of the suggested changes on the analysed domain requirements;

— revising the analysed requirements as necessary.

A tool should support reviewing the analysed domain requirements with the following capabilities:

— importing domain requirements analysis results;

— providing an environment for reviewing and commenting the domain requirements via communication networks.

6.3 Domain requirements specification

6.3.1 Purpose of the domain requirements specification

Domain requirements specifications document the functional and non-functional requirements and constraints clearly and precisely in a consistent, accessible, and reviewable way. The specifications should have trace links with the associated variability models to manage the requirements, variabilities, and their changes.

6.3.1.1 Inputs

— *Functional and non-functional domain requirements* (including constraints, dependencies, and priorities).

— *Domain requirements models.*

— *Variability models (from Variability in requirements).*

— *Conceptual test cases and scenarios for domain requirements.*

— *Technical/economic feasibility analysis results.*

— *Software Requirements Specification (SRS) template.*

6.3.1.2 Outcomes

— *SRS for domain requirements* is established. Domain requirements specifications (i.e. SRS for product lines) should distinguish commonly used requirements for all member products in the product lines and application specific parts. Trace links from the requirements to the sources are established.

6.3.1.3 Tasks

— *Identify sources of domain requirements* to ensure that all stakeholders know the how and why of every requirement in the SRS, to facilitate further clarifications, and to trace each requirement back to its origin.

— *Establish traceability through explicit trace links from the requirements sources to the domain requirements.* Trace links are needed among the domain requirements and between the domain requirements and other domain artefacts (e.g. domain scoping). Trace links for managing variability are addressed in the Variability in requirements process.

— *Document domain requirements.* Domain requirements are documented using an SRS template.

— *Review the domain requirements specification.* Domain requirements specification is reviewed for ensuring documentation completeness and correctness.

6.3.2 Identify sources of domain requirements

Sources of domain requirements are identified for establishing trace links and for managing them during the requirements management.

The method should support identifying the sources of the domain requirements with the following capabilities:

— analysing domain requirements' attributes;

— identifying sources of all domain requirements;

— analysing relationships between domain requirements and other domain artefacts.

A tool should support identifying the sources of the domain requirements with the following capabilities:

— finding and referring to information generated as part of other processes (from product line scoping to domain requirements analysis);

— providing templates with predefined attributes for systematically recording relevant information about domain requirements;

— recording and maintaining information about the identified sources of domain requirements.

NOTE Examples of requirement attributes:

— Decision for commonality or variability;

— Rationale for the decision;

— Products related to each variability;

— Relevant features derived from product line scoping.

6.3.3 Establish traceability

Establishing trace links for a product line is typically more complicated than for a single application because in the product line context there are more information sources (e.g. stakeholders, existing products, markets), more domain artefacts, more interdependencies and more valid configurations of the artefacts than in the single application development context. Therefore, tools play especially important roles in the product line context.

The method should support establishing traceability with the following capabilities:

— establishing backward and forward trace links from domain requirements to requirements sources and vice-versa;

— establishing trace links among requirements artefacts including scoping artefacts;

— establishing traceability tables.

A tool should support establishing traceability with the following capabilities:

— presenting backward and forward trace links between domain requirements and their sources;

— defining explicitly typed trace links to determine whether a dependency between two artefacts is mandatory (i.e. choosing a requirement X in a product makes it mandatory to choose a requirement Y in the product), optional (i.e. choosing a requirement X in a product makes it possible but not necessary to choose a requirement Y), or mutually exclusive (i.e. choosing a requirement X in a product makes it impossible to choose the requirement Y in a product);

— displaying trace links and associated commonality and variability information between requirements artefacts graphically;

— providing templates with predefined traceability attributes for documenting traceability information.

6.3.4 Document domain requirements

The goal of this task is to make a well-structured specification of domain requirements with all necessary information.

The method should support documenting the domain requirements with the following capabilities:

— giving a unique ID to each requirement by differentiating common and variable requirements;

— defining reusable SRS template for documenting each product;

— prioritizing domain requirements;

— recording the domain level business rules (including corporate policies, government regulations, and computational algorithms) separately from SRS.

A tool should support documenting common and variable domain requirements with the following capabilities:

— providing templates for documentation (including the differentiation of common and variable requirements);

— importing requirements models from the modelling tool for documentation;

— checking the document quality and consistency through spell checking, data dictionaries, acronym tables, etc.;

— storing the SRS and associated domain requirements in a (centralized) repository.

6.3.5 Review the domain requirements specification

The goal of this task is to ascertain the completeness, correctness, unambiguity, and traceability of the domain requirements specification.

The method should support reviewing the domain requirements specification with the following capabilities:

— ensuring that domain requirements specification includes all necessary contents;

— confirming that domain requirements specification can be easily reused for deriving application requirements specification;

— resolving and revising the domain requirements specification.

A tool should support reviewing the domain requirements specification with the following capabilities:

— importing and browsing domain requirements specifications;

— providing an environment for reviewing, discussing, and commenting on the requirements specifications.

6.4 Domain requirements verification and validation

6.4.1 Purpose of the domain requirements verification and validation

The purpose of this subprocess is to ensure that product line requirements are complete, correct, consistent, and clear. Domain requirements validation should be conducted with variability model validation to maintain consistency between the requirements and variability.

6.4.1.1 Inputs

— *Domain requirements specification.*

— *Conceptual test cases and scenarios.*

— *Checklists for review.*

6.4.1.2 Outcomes

— *Conceptual test cases and scenarios for acceptance testing.* These acceptance criteria are commonly used to derive acceptance criteria for a specific product.

— *Baselined domain requirements.* Documents produced from requirements elicitation, analysis, specification, traceability, testing, and requirements models are reviewed/inspected and validated. Domain requirements are then baselined.

6.4.1.3 Tasks

— *Verify domain requirements.* Ambiguous or unverifiable domain requirements are identified, and domain requirements models (e.g. SRS, use cases, test cases, and prototypes) and other documents produced during domain requirements elicitation, analysis, and specification subprocesses are examined. Domain requirements are verified using conceptual test cases derived during domain requirements analysis.

— *Validate domain requirements.* The specified domain requirements (functional and non-functional domain requirements) are confirmed whether they are complete and correct set.

— *Validate conceptual test cases and scenarios for acceptance testing.* Conceptual test cases and scenarios are confirmed whether they are proper cases and scenarios for requirements-based system testing or acceptance testing.

— *Baseline domain requirements.* This task establishes the agreed upon and approved set of common requirements for all member products. When the task has been completed, all requirements changes are subjected to thorough change and configuration control during domain requirements management.

— *Initiate change management process.* Appropriate change management processes and configuration management systems are initiated in this task to enable well-managed changing of the domain requirements after the requirements have been baselined.

6.4.2 Verify domain requirements

This task ensures that the domain requirements are clear, understandable, and testable.

The method should support verifying domain requirements with the following capabilities:

— verifying the specified domain requirements artefacts;

— identifying ambiguous or unverifiable requirements;

— providing checklists for verification;

— verifying common and variable requirements for optimization.

A tool should support verifying domain requirements with the following capabilities:

— guiding verification process;

— showing the specified domain requirements and relevant information;

— providing multi-user write access to the same document;

— providing an environment for team reviewing, marking up, and commenting of domain requirements;

— providing a documentation template for documenting the verification results (including rationales for the review decisions and the measures taken to correct the issues found);

— notifying relevant stakeholders about the verification status.

6.4.3 Validate domain requirements

This task ensures the completeness and correctness of domain requirements.

The method should support validating domain requirements with the following capabilities:

— confirming whether missing or contradicting domain requirements exist;

— providing checklists for validation;

— checking the completeness and correctness of domain requirements;

— validating the defined variabilities including their variants and dependencies.

A tool should support validating domain requirements with the following capabilities:

— guiding the validation process;

— presenting the specified domain requirements and relevant information;

— providing multi-user write access to the same document;

— providing an environment for team reviewing, marking up, and commenting of domain requirements;

— providing a documentation template for documenting the validation results (including rationales for the review decisions and the measures taken to correct the issues found);

— notifying relevant stakeholders about the validation status.

6.4.4 Validate conceptual test cases and scenarios for acceptance testing

Conceptual test cases and scenarios commonly used by all applications within a product line are validated to ensure that they are trustworthy enough to validate the completeness and correctness of member products. During application requirements engineering, application specific test cases and scenarios are defined on top of the domain acceptance criteria.

The method should support validating conceptual test cases and scenarios for acceptance testing with the following capabilities:

— defining validation criteria for conceptual test cases and scenarios;

— validating whether variabilities are structured well in conceptual test cases and scenarios;

— confirming whether application specific test cases and scenarios for acceptance testing can be derived.

A tool should support validating conceptual test cases and scenarios for acceptance testing with the following capabilities:

— browsing and accessing the domain requirements assets for validating conceptual test cases and scenarios;

— accessing traces among domain requirements assets, variability model, and defined conceptual test cases and scenarios;

— allowing semi-/automated validation reports.

6.4.5 Baseline domain requirements

Domain requirements baseline is created and released in this task. Thereafter the changes for the baseline are controlled and managed by change management tasks.

The method should support baselining domain requirements with the following capabilities:

— establishing formal agreements for the domain requirements;

— establishing a baseline for the approved domain requirements;

— managing the configuration of the baselined domain requirements.

A tool should support baselining domain requirements with the following capabilities:

— providing an interface to interact with the change and configuration management systems;

— exporting baselined domain requirements into the repository of change and configuration management systems.

6.4.6 Initiate change management process

The change management should be initiated at this point before proceeding to further change management defined in the requirements management.

The method should support initiating a change management process with the following capabilities:

— establishing a change management environment (e.g. the change management process and a supporting change management information system);

— instituting a change management information system and a supporting configuration management system.

A change management information system and a supporting configuration management system should be instituted to support the initiation of the change management process with the following capabilities:

— initializing the repository where the baselined domain requirements can be stored when baselining has been completed;

— setting up interfaces for interacting with change management information system and configuration management system.

6.5 Domain requirements management

6.5.1 Purpose of the domain requirements management

The purpose of the domain requirements management is to schedule, coordinate, and document the domain requirements engineering process (elicitation, analysis, documentation, and validation) and deal efficiently and effectively with stakeholders' change requests during product line and product line member development.

6.5.1.1 Inputs

— *Change requests for domain requirements* (from Domain Engineering or from Application Engineering).

6.5.1.2 Outcomes

— *New baselines and versions of domain requirements documents* are established.

— *Impact analysis results* (including impacts for domain requirements and application requirements) are documented.

— *A change history for each domain requirement* is maintained.

— *Updated domain requirements trace links* are maintained.

— *Status accounting and reports* are performed.

— *Domain requirements engineering process assets* are produced.

— *Change requests for variability models* (this is addressed in variability evolution sub-process of the Variability in requirements) are documented.

6.5.1.3 Tasks

— *Manage domain requirements change.* Changes to the domain requirements are tracked and the approved changes are communicated to all affected stakeholders.

— *Manage traceability.* Traceability management allows stakeholders to follow the life of each requirement throughout the life cycle from the origin to implementation in both forward and backward direction.

— *Manage versions of domain requirements.* Versions of artefacts or work products produced during domain requirements engineering are managed.

— *Record and report status (Status accounting).* Management records, that show the status and history of the controlled domain requirements baseline are recorded and reported. Status reports should include the number of changes for domain requirements and the latest artefacts' versions produced during domain requirements engineering.

— *Manage process improvement.* The domain requirements engineering process and the enabling information systems are assessed and reviewed to understand the strengths and weaknesses of the implemented process. Improvements are implemented based on these assessments.

— *Manage feedback.* Feedback from other domain and application engineering processes is managed and resolved. Changes for variable requirements that distinguish member products in a product line from each other need to be conducted with the variability in requirements process for reflecting variability changes.

6.5.2 Manage domain requirements change

Change management in a product line is complicated because change requests for domain requirements are raised from application engineering as well as domain engineering. Because domain requirements are related to a number of applications within a product line, impact analyses of the requested changes are more difficult in the product line context than in the context of single products. Therefore, supporting tools and methods are essential.

The method should support managing requirements change with the following capabilities:

— acquiring change requests for domain requirements;

— analysing the impact on other domain requirements related changes;

— analysing the impact on other applications related changes;

— approving changes;

— tracking change requests' status to closure.

A tool should support requirements change management with the following capabilities:

— establishing a channel for requesting changes;

— uploading change requests;

— saving information about the sources of change requests;

— notifying stakeholders about the statuses of change requests;

— maintaining change histories (e.g. who changed the requirement, when and why the change was done).

6.5.3 Manage traceability

The complexity of trace links in product lines are more complicated than the links for the requirements of a single product because there are trace links in domain engineering, those in application engineering, and those between domain engineering and application engineering. This task covers product line-specific tools and methods' capabilities to cover the complexity.

The method should support managing traceability with the following capabilities:

— analysing impacts on traceability related to change requests;

— revising trace links (e.g. between domain requirements and sources);

— validating trace links.

A tool should support managing traceability with the following capabilities:

— maintaining versions of requirements trace links for recovery;

— updating the trace links;

— showing traceability information to a stakeholder in a certain role when the stakeholder requests it.

6.5.4 Manage versions of domain requirements

Applications can share a number of versions of domain requirements artefacts. Tools have to be provided to cope with these complexities.

The method should support managing versions of domain requirements with the following capabilities:

— specifying recent versions of the baselines of domain requirements artefacts;

— describing the differences between baselines of domain requirements artefacts;

— managing the history of versions for recovering from failure.

A tool should support version management of domain requirements with the following capabilities:

— generating an automatic version number;

— providing a recovery mechanism;

— providing a repository for managing version history;

— tracing the pertinent artefacts under the version control (requirements, models, etc.);

— controlling versions of the baselines of domain requirements artefacts over time.

6.5.5 Record and report status

The statuses and histories of controlled domain artefacts are continuously recorded and reported during domain requirements management so that all applications can refer to them.

The method should support status recording and reporting with the following capabilities:

— managing the requirements management records and status reports for showing the statuses and histories of controlled artefacts;

— recording the contents and statuses of domain requirements artefacts for recovering previous versions of artefacts when necessary;

— revising the status and history of each artefact if necessary;

— reporting the statuses of domain requirements artefacts.

Status reports should include the number of changes for requirements, latest versions of domain requirements artefacts, release identifiers, and the number and comparisons of releases.

A tool should support the recording and reporting of status information with the following capabilities:

— maintaining contents and statuses of domain requirements artefacts;

— controlling access of a user according to the authority and responsibility assigned to a user's role;

— maintaining revision histories (management records and status reports) of domain requirements artefacts.

6.5.6 Manage process improvement

This task specifically deals with improving the domain requirements engineering lifecycle processes.

Note that overall product line process improvements are addressed in tools and methods for product line technical management.

The method should support managing process improvement with the following capabilities:

— collecting process improvement requests;

— appraising the weaknesses and strengths of the implemented domain requirements engineering processes;

— developing domain requirements engineering process improvement recommendations based on process improvement requests and appraisal results;

— developing an action plan for implementing the requested domain requirements engineering process improvement;

— managing raised issues during the domain requirements engineering process implementation.

A tool should support managing process improvement with the following capabilities:

— maintaining domain requirements engineering process assets (e.g. process descriptions, process artefacts);

— providing templates for domain requirements engineering process appraisals;

— providing templates for developing action plans for domain requirements engineering process improvement.

6.5.7 Manage feedback

Feedback related to domain requirements needs to be collected from various sources such as other domain requirements engineering services and application requirements engineering services. This task should have the capability to deal with various kinds of feedback from all relevant sources and to communicate to the sources providing feedback about the actions taken or planned to be taken to deal with the feedback.

The method should support learning from feedback with the following capabilities:

— identifying feedback sources;

— evaluating the feasibility of feedback;

— processing feedback (if changes to domain requirements are likely, explicit change requests are established and processed);

— returning the feedback processing results.

A tool should support managing feedback with the following capabilities:

— establishing a channel for collecting feedbacks;

— capturing and saving feedback and information about the sources of feedback;

— updating the status of feedback;

— communicating the feedback processing results to the sources of feedback.

7 Variability management in requirements engineering

Variability in requirements process should be conducted in parallel with domain requirements engineering because variability models are gradually clarified and modified together with the domain and application requirements. Variability modelling is started in the domain requirements elicitation subprocess and the models evolve continuously throughout the product line life cycle.

This process establishes and maintains the external variability for the product line but part of the internal variability might be specified. Variability management in requirements engineering has the following subprocesses:

— *Variability in textual requirements* serves to express and document the variability in requirements using natural language and makes them explicit.

— *Variability in requirements models* serves to document variabilities by integrating them into requirements models.

— *Variability mechanisms in requirements* serves to categorize variability mechanisms in requirements.

— *Traceability between requirements variability and variability model* serves to establish and maintain links between requirements variability documentations and variability model defined separately.

7.1 Variability in textual requirements

7.1.1 Purpose of variability in textual requirements

The purpose of variability in textural requirements is to express and document variabilities in requirements using natural language and makes them explicit. Documenting requirements variability leaves room for ambiguity even when textual requirements are expressed using certain keywords or phrases.

7.1.1.1 Inputs

— *Textual requirements documents.*

7.1.1.2 Outcomes

— *Textual requirements documents explicitly expressing requirements variability* are defined.

7.1.1.3 Tasks

— *Define requirements variability in textual requirements* to illustrate requirements variability when requirements are expressed in natural language. Variation point and its variants including its effect on other variabilities should be depicted explicitly.

— *Document requirements variability in textual requirements* to augment structural and relational information of variability. Requirements variability has to be documented to provide the means to select or process variants in application requirements engineering.

7.1.2 Define requirements variability in textual requirements

The goal of this task is to express variability by certain keywords. The variation point and its variants have to be made explicit by adding extra annotations.

The method should support defining requirements variability in textual requirements with the following capabilities:

— making textual requirements variability explicit;

— making variation points and its variants explicit and expressing them;

— depicting the effect of a variability.

A tool should support defining requirements variability in textual requirements with the following capabilities:

— highlighting variable requirements for discrimination;

— providing notation for making requirements variability explicit.

7.1.3 Document requirements variability in textual requirements

The goal of this task is to document structural and relational information of variability in a textual format. It should provide the means to select or process variants in application requirements engineering.

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

— adding structural information to textual requirements variability necessary to select or process a variability, e.g. unique identifiers for variants;

— adding relational information among textual requirements variabilities;

— structuring the requirements variability to make processing be possible;

— validating the readability of requirements variability documentation.

A tool should support documenting requirements variability in textual requirements with the following capabilities:

— enabling trace links from a variability model defined separately;

— providing capability for identifying each variant uniquely.

7.2 Variability in requirements models

7.2.1 Purpose of variability in requirements models

The purpose of variability in requirements models is to explicitly document variabilities in requirements models. Variabilities should be expressed in various types of requirements models including feature, use case, state transition, sequence, data, and etc.

7.2.1.1 Inputs

— *Requirements model.*

7.2.1.2 Outcomes

— *Requirements models explicitly expressing requirements variability* are defined.

7.2.1.3 Tasks

— *Define requirements variability in requirements model* to explicitly express requirements variability in requirements model. Variation points with their variants and constraints are described in requirements models.

— *Document requirements variability in requirements model* to elaborate relevant information about requirements variability in requirements model and to offer clear understanding of variability in requirements models.

7.2.2 Define requirements variability in model

The goal of this task is to express variability by certain keywords or phrases. The variation point and its variants have to be made explicit by adding extra expressions.

The method should support defining requirements variability in model with the following capabilities:

— making requirements variability in requirements model explicit;

— expressing explicitly the variation point, its variants, and constraints of variants;

— expressing separately the variability of functional or non-functional requirements.

A tool should support defining requirements variability in model with the following capabilities:

— providing modelling notation to make requirements variability explicit;

— providing notation for separating the functional and non-functional requirements variability.

7.2.3 Document requirements variability in requirements model

The goal of this task is to elaborate requirements variability in the documentation of the requirements model so that requirements variability is integrated into requirements model documentations. Requirements variability is documented by using different requirements models, i.e. feature, use case, state transition, sequence, data, and etc.

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

— adding elaborations to requirements variability in requirements model;

— establishing relationships among the same requirements variability documented in different ways;

— adding glues that support linking with variability model;

— integrating the requirements variability documents documented in different requirements models;

— validating the requirements variability documentation to ensure unambiguous documentations.

A tool should support documenting requirements variability in requirements model with the following capabilities:

— enabling trace links from a variability model that is defined separately;

— supporting integration of requirements variability documentations which are documented in different types of requirements models.

7.3 Variability mechanisms in requirements

7.3.1 Purpose of variability mechanisms in requirements

The purpose of this subprocess is to categorize variability mechanisms suitable for the domain of interest, to guide the selection of variability mechanisms in requirements, and to manage the categorized variability mechanisms.

Variability in requirements subprocess is expressed in a variety of requirements models and textual requirements by variability mechanisms such as extend/include, stereotypes, tags or mark-up and so on. Most of variability that is bound at requirements stage is external variability, but some of internal variability can be bound at requirements stage. This subprocess deals with requirements engineering specific methods and tools capabilities related to variability mechanisms.

7.3.1.1 Inputs

— *Defined domain of interest.*

— *A catalog of variability mechanisms in requirements.*

— *Initial variability model.*

— *Initial decision table.*

7.3.1.2 Outcomes

— *Variability mechanisms to be used in requirements* are categorized.

— *Guidance for the use of variability mechanisms in requirements* is documented.

— *Selected variability mechanisms in requirements* are verified.

— *Improvements for variability mechanisms* are conducted.

— *Variability model added requirements specific variability mechanism information* is produced.

— *Decision table added requirements specific variability mechanism information* is produced.

7.3.1.3 Tasks

— *Identify variability mechanisms in requirements* is to categorize suitable variability mechanisms from a catalog of variability mechanisms in requirements.

— *Guide the use of variability mechanisms in requirements* is to provide documented guidance for the precise use of variability mechanisms in requirements.

— *Verify the usage of variability mechanisms in requirements* is to monitor the usage status of variability mechanisms so as to improve the accuracy of mechanism selection.

— *Improve variability mechanisms in requirements* is to improve the usage and usage support of variability mechanisms able to be used in requirements model and textural requirements.

7.3.2 Identify variability mechanisms in requirements

The goal of this task is to review and find variability mechanisms suitable to variability in requirements and requirements artefacts.

The method should support identifying variability mechanisms in requirements with the following capabilities:

— identifying variability mechanisms that have potential to be used in requirements artefacts;

— categorizing variability mechanisms in requirements and requirements artefacts;

— identifying requirements specific variability mechanism information added to initial variability model;

— identifying requirements specific variability mechanism information added to initial decision model;

— verifying the suitability of categorized variability mechanisms in requirements.

A tool should support identifying variability mechanisms in requirements with the following capabilities:

— displaying variability mechanisms in requirements by categories;

— allowing the catalogue management of variability mechanisms in requirements;

— allowing the revision of variability model for adding requirements specific variability mechanism information;

— allowing the revision of decision model for adding requirements specific variability mechanism information.

7.3.3 Guide the use of variability mechanisms in requirements

The goal of this task is to provide guidance for selecting variability mechanism and ways for performing configuration and binding in a member product. The overall guidance is provided by ISO/IEC 26557. This task supports concrete guides filled with requirement engineering specific practices.

The method should support guiding variability mechanisms in requirements with the following capabilities:

— customizing variability mechanisms guidance (defined in ISO/IEC 26557) to requirement engineering specifically;

— developing the exemplar usage of variability mechanisms for the precise use of variability mechanisms in requirements.

A tool should support guiding variability mechanisms in requirements with the following capabilities:

— documenting guidance for the precise use of variability mechanisms in requirements;

— providing exemplar usage of variability mechanisms in requirements.

7.3.4 Verify the usage of variability mechanisms in requirements

The goal of this task is to verify the effectiveness and efficiency of variability mechanisms in requirements.

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

The method should support verifying the usage of variability mechanisms in requirements with the following capabilities:

— measuring the effectiveness and efficiency of the categorized variability mechanisms in requirements;

— reviewing plan versus actual usage status of variability mechanisms;

— deriving and tracing corrective actions.

A tool should support verifying the usage of variability mechanisms in requirements with the following capabilities:

— allowing (semi-)automatic measurement data collection;

— sharing potential corrective actions with relevant participants.

7.3.5 Improve variability mechanisms in requirements

The goal of this task is to improve the established variability mechanisms that are used in requirements artefacts and to support variability that is bound at requirements stage.

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

The method should support improving variability mechanisms in requirements with the following capabilities:

— identifying improvement items for variability mechanisms in requirements;

— implementing the identified improvement items;

— tracing the status of improvement items.

A tool should support improving variability mechanisms in requirements with the following capabilities:

— documenting improvement items;

— sharing improvement results with relevant participants.

7.4 Traceability between requirements variability and variability model

7.4.1 Purpose of traceability between requirements variability and variability model

The purpose of this subprocess is to establish links between requirements variability documents and variability model defined separately.

7.4.1.1 Inputs

— *Requirements models or textual requirements documents including requirements variability.*

— *Variability models.*

7.4.1.2 Outcomes

— *Requirements variability including trace links with variability models* are defined.

7.4.1.3 Tasks

— *Define explicit links between requirements variability and variability model.* Trace links of requirements variability in requirements documents with the separated variability model are established and maintained.

7.4.2 Define explicit links between requirements variability and variability model

The goal of this task is to establish explicit trace links between requirements variabilities and variability model.

The method should support defining explicit links between requirements variability and variability model with the following capabilities:

— determining links of variability model with the requirements variability in requirements documentation;

— confirming whether requirements variability is consistent with variability model;

— checking whether the different definitions of requirements variability in requirements documentations are consistent.

A tool should support defining explicit links between requirements variability and variability model with the following capabilities:

— importing and browsing a variability model;

— storing links between requirements variability and variability model;

— supporting assessment of deviations between requirements variability and variability model.

8 Asset management in requirements engineering

Asset management in requirements engineering is a process in which the requirements artefacts resulting from domain and application requirements engineering are structured as assets. Variability, as well as commonality in domain requirements, is managed as domain assets. In addition, application requirements artefacts with high reusability potential are defined as domain assets. Asset management in requirements adds or develops extra elaborations and specifications to requirements artefacts to be reused as domain assets. The relationship among requirements domain assets for being reused successfully or managing changes on them are also analysed in this process area. Subprocesses for configuring domain assets and managing them in asset repository refer to asset management in ISO/IEC 26555.

Asset management in requirements engineering has two basic subprocesses:

— *Domain requirements artefacts* as domain assets serves to identify and develop necessary information to help application engineers reuse domain requirements assets in their application development. This process develops necessary glues that are used for integrating and orchestrating domain assets with other domain assets and for reusing the assets. Such information is created and analysed when the domain assets are derived and refined;

— *Application requirements artefacts* as application assets serves to add extra descriptions including relationships to application assets.

The domain requirements models and specifications are important domain assets as application requirements engineering reuses them to define their requirements. And they are used to refine the product line scope and business cases and determine the feasibility of requirements.

8.1 Domain requirements artefacts as domain assets

8.1.1 Purpose of domain requirements artefacts as domain assets

The purpose of this subprocess is to identify domain requirements artefacts for using them as domain assets over a number of applications.

Reusable artefacts among designated work products of the domain/application requirements engineering that place under control as domain assets are elaborated. Work products developed in domain/application requirements engineering, e.g. features, models, and textual or model-based requirements specification, are elaborated for making them as domain assets.

8.1.1.1 Inputs

— *Domain requirements models.*

— *Application requirements models that have reuse potential.*

— *Domain requirements specifications.*

— *Application requirements specifications that have reuse potential.*

— *Request from an application engineer for making requirements artefacts as envisioned domain assets.*

8.1.1.2 Outcomes

— *Configuration and annotations for domain requirements artefacts* are defined or developed.

8.1.1.3 Tasks

— *Identify domain requirements artefacts managed as domain assets.* Domain requirements artefacts such as models and specifications that are managed as domain assets are identified. This task also covers application requirements models that have reuse potential.

— *Define configuration and annotation.* Domain requirements artefacts identified as domain assets are structured according to the defined domain asset configuration and attached annotations for proper reuse.

8.1.2 Identify domain requirements artefacts managed as domain assets

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

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

— selecting domain requirements artefacts that have reuse potential;

— evaluating selected domain requirements artefacts (e.g. checklist for evaluating reusability);

— establishing backward trace links with scoping assets.

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

— importing domain requirements artefacts that have reuse potential;

— storing the list of domain requirements artefacts as domain assets.

8.1.3 Define configuration and annotation

The goal of this task is to define and develop the further structure of domain requirements artefacts that help with reuse for the application developments.

Processes to guide how to derive the requirements for an application from the domain assets in requirements engineering or how to validate the derived requirements for an application can be added. They can also guide binding variants. Each domain assets in requirements engineering should have attached processes that specify how it can be used to develop individual product's requirements.

The method should support defining configuration and annotation with the following capabilities:

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

— identifying annotations necessity to reuse domain requirements artefacts as domain assets at the application developments;

— validating configuration and annotations for domain requirements artefacts.

A tool should support defining configuration and annotation with the following capabilities:

— allowing the access of existing information for defining configuration and annotation (the configuration includes trace links with the relevant scoping assets);

— allowing the access of domain requirements artefacts;

— providing an editor for template definition.

8.2 Application requirements artefacts as application assets

8.2.1 Purpose of application requirements artefacts as application assets

The purpose of this subprocess is to establish the structure of application assets in requirements as constituent relationships with other application engineering processes.

8.2.1.1 Inputs

— *Application requirements artefacts.*

8.2.1.2 Outcomes

— *Configuration and annotations for application requirements artefacts* are defined or developed.

8.2.1.3 Tasks

— *Identify application requirements artefacts managed as application assets.* Application requirements artefacts that have relationships with successive application engineering processes are selected and determined.

— *Define configuration and annotation for application requirements assets.* The structure and annotations of application requirements assets that are necessary to maintain them within the asset repository or to reuse them at each application development are defined and developed.

8.2.2 Identify application requirements artefacts managed as application assets

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

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

— selecting application requirements artefacts used at the later processes of application development.

A tool should support identifying domain requirements specification managed as domain assets with the following capabilities:

— importing application requirements artefacts;

— supporting reviewing and editing application requirements artefacts.

8.2.3 Define configuration and annotation for application requirements assets

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

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

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

— identifying annotations necessary to use application requirements assets for the application developments;

— validating configuration and annotations for application requirements assets.

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

— allowing the access of existing information for defining configuration and annotation;

— allowing the access of application requirements assets;

— providing an editor for template definition;

— supporting trace links among application requirements assets;

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

9 Application requirements engineering

Application requirements come from the stakeholders of a member product. The domain variability model and domain requirements help elicit the application specific requirements. The requirements of a member product that can be selected from domain requirements should be bound properly. The gap between domain requirements and a member product's requirements are analysed. Unsatisfied requirements stemming from the gap may be met by implementing application specific requirements artefacts. Application requirements artefacts to be integrated in the domain requirements artefacts, and requests for additional or altered requirements, are issued during the application requirements engineering. When high priority application requirements are shared by many applications, domain engineering should be requested to ensure that the next platform release satisfy the requirements.

Application requirements engineering has five basic subprocesses:

— *Application requirements elicitation* identifies stakeholders relevant to an application, elicits application-specific requirements and binds the appropriate variants;

— *Application requirements analysis* first ensures that all requirements of the application stakeholders are understood and scrutinized for incorrectness, omissions, and inconsistencies through requirements modelling. Requirements that cannot be satisfied through the domain requirements, that is the gap between domain and application requirements, are then documented, analysed, and negotiated;

— *Application requirements specification* documents application requirements by adding a member product specific requirements to the specification of the bound domain requirements;

— *Application requirements validation and verification* review and test the requirements relevant to the bound variants and the remaining gap and examines their rationales. Application specific requirements and the bound domain requirements are verified to ensure they make sense for the product;

— *Application requirements management manages traceabilities and changes of application requirements.* Traceability management deals with traceabilities among application requirements, and traceabilities between requirements and application variability models. Changes in the application requirements can impact the related domain assets so that impacts should be carefully analysed.

9.1 Application requirements elicitation

9.1.1 Purpose of the application requirements elicitation

The purpose of the application requirements elicitation is to identify the appropriate stakeholders and application specific requirements sources and to elicit application specific functional and non-functional requirements from them. Application requirements engineers use variation points and variants to communicate with stakeholders and let them select the variants that best meet their requirements. Application requirements engineers bind the selected domain requirements. Thus a domain variability model can be used to guide the application elicitation ensuring systematic reuse.

9.1.1.1 Inputs

— *Application stakeholders and other information sources related to the application.*

— *Domain assets related to the application.*

— *Variability models related to the selected domain assets including trace links.*

— *Binding information.*

9.1.1.2 Outcomes

— *Domain assets* are selected.

— *Variant bindings including the rationales of variability resolutions* are conducted.

— *Application specific requirements lists (in elicitation service)* are made.

— *Elicited requirements gaps between application and domain requirements* are defined.

— *Application variability model in requirements elicitation* is defined.

9.1.1.3 Tasks

— *Draw a context diagram for an application.* A context diagram for an application captures high-level entities and those relationships (e.g. users of a member product, physical environment, or other systems related to a member product).

— *Identify the requirements gaps between domain and application requirements.* The gaps and their impacts are analysed and evaluated. Application specific features defined in product line scoping should be decomposed into initial application requirements and additional application specific initial requirements and its derived requirements should be elicited.

— *Bind the best matching variants.* Variants that meet the application requirements to the maximum possible extent are selected.

— *Select domain assets.* A set of domain assets associated with an application are identified from asset repository.

— *Review the elicited initial application requirements* to see whether they violate the application's scope and the requirements of the stakeholders and revise them as necessary (including a comparison with the asset proposals)

9.1.2 Draw a context diagram for an application

A context diagram describes the relationships between an application and its environment. This task serves to find application users, relevant systems for interoperability, and other relationships.

The method should support drawing a context diagram for an application with the following capabilities:

— analysing the application environment (e.g. application users, stakeholders, external systems);

— analysing the relations among identified entities;

— diagramming the application context under the product line context.

A tool should support drawing a context diagram for an application with the following capabilities:

— providing documentation templates;

— providing checklists for analysing the entities involved in application environment and their relations;

— drawing an application context diagram.

9.1.3 Identify the requirements gaps between domain and application requirements

Application specific initial requirements which are not in domain requirements are elicited in this task. The application requirements are assumed to significantly overlap with the domain requirements. However, detailed requirements elicitation and analysis only needs to be conducted for those application requirements that cannot be implemented by using the product line platform.

The method should support identifying the requirements gaps between domain and application requirements with the following capabilities:

— deriving initial gaps based on application specific features defined in product line scoping;

— decomposing application specific features into application initial requirements;

— identifying additional gaps with the help of the domain requirements and variability model;

— analysing the impacts of the gaps;

— reviewing the gaps with respect to the required change efforts;

— creating additional application specific initial requirements lists (including preliminary classifications – functional/non-functional);

— leveraging the domain variability model to help stakeholders understand domain requirements and revise their requirements and expectations as necessary.

A tool should support identifying requirements gaps between the domain and application requirements with the following capabilities:

— browsing the relevant scoping assets, domain requirements assets, and variability model;

— providing a document template for documenting the gap resolutions and their rationales;

— providing a documentation template for documenting the information for the application specific initial requirements that cannot be met by the bound variants;

— recording sources of application specific initial requirements.

9.1.4 Bind the best matching variants

This task serves to determine external variants selected by customers and other stakeholders and to bind relevant internal variants for all selected external variants. Binding information provides information about to which variants the application are related. A set of domain assets is delivered from the asset repository after binding.

The method should have the following capabilities to support binding the best matching variants:

— analysing the relevant domain variability models and the domain requirements artefacts linked to the variability models with trace links;

— binding the internal (technical) variants associated with the external variants;

— validating the bound variants.

A tool should support binding the best matching variants with the following capabilities:

— importing the relevant variability models;

— importing other relevant domain assets into the application workspace;

— showing the variability models (e.g. variants and dependencies) for binding variability;

— indicating the variants that best meet stakeholders' needs and storing rationales for the selections;

— checking the variability dependencies and constraint dependencies of the selected variants;

— recording the rationales for the decision.

9.1.5 Select domain assets

A set of domain assets related to an application is elicited. When domain assets are selected, the relevant domain assets for the different product line stages should be elicited. The goal of this task is to mine a set of domain assets from the asset repository associated with an application.

The method should support selecting domain assets with the following capabilities:

— finding the relevant domain requirements artefacts with the help of the application variability model;

— evaluating them from the viewpoints of application and the reusability requirements imposed by the product line.

A tool should support selecting domain assets with the following capabilities:

— providing an interface to interact with the asset repository;

— enabling the stakeholders to view the variability model and associated domain requirements (preferably visually through graphical models);

— importing the relevant domain requirements and variability models into the application engineering workspace;

— recording the rationales for the selections.

9.1.6 Review the elicited application requirements

The captured application requirements are reviewed to ensure that they are correct, complete, and understandable.

The method should support reviewing the elicited application requirements with the following capabilities:

— checking trace links between the elicited requirements and their sources;

— integrating the requirements into the relevant domain requirements document;

— verifying the completeness and correctness of the elicited requirements.

A tool should support reviewing the elicited application requirements with the following capabilities:

— allowing trace between the elicited requirements and sources;

— providing an editing environment to integrate the domain requirements with the application specific requirements;

— maintaining the checklists for verification.

9.2 Application requirements analysis

9.2.1 Purpose of the application requirements analysis

The purpose of the application requirements analysis is to refine the application stakeholders' requirements and constraints in order to extend the specification of the bound variants into an application requirements specification that fully meets the application requirements. Relevant tasks include building prototypes, evaluating feasibility, and negotiating priorities. Because the relevant domain assets are typically reused as they are, only the application specific requirements are analysed and refined in detail, saving substantial resources compared to traditional single-system engineering.

The traceabilities among domain assets should be established and well maintained in domain requirements engineering. In this subprocess, the relevant domain assets are retrieved by using the traceability information. All outcomes of this subprocess deal with application requirements holistically, that is, both application specific requirements and the domain requirements associated with the bound variants.

9.2.1.1 Inputs

— *Elicited application requirements.*

— *Relevant domain assets.*

9.2.1.2 Outcomes

— *Functional/non-functional application requirements* (including constraints, dependencies, and priorities) are refined.

— *Application requirements models* (e.g. use case, feature model, context diagram, and data dictionary) are defined.

— *Technical and economic feasibility of the application (technical prototypes)* are determined.

— *Application test cases and scenarios* are defined.

9.2.1.3 Tasks

— *Classify and balance application specific initial requirements.* Application specific initial requirements are classified into functionality, quality attributes, constraints, and other non-functional requirements categories. The granularity levels of application specific initial requirements are balanced by decomposing high level requirements into low level ones and aggregating detailed requirements into higher level ones.

— *Analyse commonalities and variabilities.* Requirements that are specific to a member product are analysed. An application variability model may include new variation points and/or variants.

— *Model application specific requirements.* Application specific requirements and their interdependencies are visualized to reveal incorrect, inconsistent, missing, and superfluous requirements. Application specific requirements models should be traceable to both domain requirements assets and an application variability model.

— *Create prototypes and analyse feasibility.* Prototypes can be used to evaluate the feasibility of implementing certain critical application specific requirements from numerous viewpoints (e.g. acceptable cost and performance).

— *Develop conceptual test cases and scenarios for acceptance testing.* Conceptual test cases and scenarios for acceptance testing of a member product are derived. These scenarios and cases may be derived from application specific requirements models like use case diagrams.

— *Review the analysed application requirements.* Analysed application requirements are reviewed for identifying and correcting incompleteness and incorrectness.

9.2.2 Classify and balance application specific initial requirements

This task classifies application specific initial requirements and then coordinates the granularity of the initial requirements for further analysis.

The method should support classifying and balancing application specific initial requirements with the following capabilities:

— defining guidelines for balancing the level of abstractions of initial requirements;

— decomposing the high-level application specific initial requirements into lower-level ones by balancing with the bound domain requirements;

— aggregating low-level application specific initial requirements to higher level ones by balancing with the bound domain requirements;

— identifying conflicts among the application specific initial requirements.

A tool should support classifying and balancing application specific initial requirements with the following capabilities:

— importing the domain requirements classification results for integration;

— importing elicited application specific initial requirements and relevant information into the application requirements analysis workspace;

— supporting editorial environment for decomposition or aggregation of application specific initial requirements;

— allowing discrimination of derived requirements from domain requirements with application specific initial requirements;

— allowing documentation of the classified application specific initial requirements.

9.2.3 Analyse commonalities and variabilities

This task analyses application specific requirements variabilities. Commonalities can be analysed for considering adaptation. An application variability model may include new variabilities in accordance with the decisions.

The method should support analysing commonalities and variabilities with the following capabilities:

— analysing application specific requirements variabilities under the leverage of commonalities and domain variability model;

— analysing inconsistencies of bound variabilities and application specific requirements;

— analysing commonalities similar to the application specific requirements for considering adaptation;

— defining application specific variation points and/or variants if necessary.

A tool should support analysing commonalities and variabilities with the following capabilities:

— allowing reference to the elicited application specific requirements and bound variable requirements;

— allowing reference to the collected information related to the application specific requirements;

— allowing reference to the commonalities and domain variability model;

— supporting documentation for application specific variation points and variants.

9.2.4 Model application specific requirements

The goal of this task is to ensure completeness and feasibility of the application specific requirements through modelling. Application models with defined variability bounds in domain requirements models are integrated into the requirements models.

The method should support modelling application specific requirements with the following capabilities:

— refining the functional application specific requirements (e.g. from domain scenarios, use cases);

— refining the non-functional application specific requirements;

— modelling the application specific requirements in a consistent way with domain requirements models;

— analysing the requirements models for their completeness, correctness, feasibility, and verifiability;

— prototyping if necessary to analyse feasibility.

A tool should support modelling application specific requirements with the following capabilities:

— supporting requirements modelling;

— allowing tracing the models with application specific requirements;

— storing and content management of the rationale of the feasibility analysis;

— exchanging information pertinent to prototyping with external prototyping tools.

9.2.5 Create prototypes and analyse feasibility

Prototyping is conducted only for the bound variants, application specific requirements, and their integration that are not conducted during domain requirements engineering to distinguish ambiguous and feasible requirements. If the bound variants have been fully implemented as part of the domain realization process, prototypes can be realized quickly on top of the realized variants.

The method should support creating prototypes and analysing the feasibility for application with the following capabilities:

— prioritizing the application requirements considering the priority of relevant domain requirements;

— choosing application specific requirements to prototype;

— deciding what kinds of information to collect;

— prototyping;

— gathering the necessary information;

— analysing the feasibility of the requirements using the prototyping results.

A tool should support creating prototypes and analysing feasibility for application with the following capabilities:

— storing and generating user defined and tool provided checklists and templates for technical, economical, and operational analyses;

— storing and managing the results of the feasibility analysis and their rationale;

— exchanging information pertinent to prototyping with external tools.

9.2.6 Develop conceptual test cases and scenarios for acceptance testing

The goal of this task is to develop application specific requirements test cases and scenarios and integrate them with the domain requirements test cases and scenarios developed in domain engineering and associate with the bound variants in order to test application requirements holistically.

The method should support developing conceptual test cases and scenarios for acceptance testing with the following capabilities:

— developing application specific requirements conceptual test cases and scenarios and analysing their feasibility;

— allocating a unique ID for each application test case in a consistent way with the domain's;

— deciding on the expected results for each test case;

— linking the "prior to" and "after" relationships among test cases and scenarios;

— integrating the application specific test cases with the domain test cases.

A tool should support developing conceptual test cases and scenarios for acceptance testing with the following capabilities:

— retrieving domain requirements conceptual test cases from the asset repository for testing the bound variants;

— deriving application specific test cases and scenarios based on domain test cases and scenarios;

— managing the test cases by their IDs;

— storing the "prior to" and "after" relationships among test cases and scenarios.

9.2.7 Review the analysed application requirements

The goal of this task is to ensure that application requirements are unambiguous and include all application stakeholders' requirements. It should be particularly ensured that application requirements derived by binding variability and application specific requirements are well-harmonized.

The method should support reviewing the analysed application requirements with the following capabilities:

— identifying the ambiguous or unverifiable analysed application specific requirements;

— analysing whether bound variant agrees with application stakeholder's requirements;

— reviewing whether bound variant and application specific requirements agree with each other.

The tool should support reviewing the analysed application requirements with the following capabilities:

— importing application requirements analysis results;

— providing an environment for reviewing and commenting the application requirements via communication networks;

— allowing an environment for checking relations between bound variant and application specific requirements.

9.3 Application requirements specification

9.3.1 Purpose of the application requirements specification

The purpose of the application requirements specification is to document the application specific requirements and integrate them with the domain requirements associated with the bound variants. All outcomes of this subprocess thus deal with application requirements holistically, i.e. that both application specific requirements and the domain requirements associated with the bound variants are covered.

9.3.1.1 Inputs

— *Domain requirements specification.*

— *Application variability model.*

— *Functional and non-functional application requirements.*

— *Application requirements models.*

— *Technical and economic feasibility of the application.*

— *Application requirements' test cases and scenarios.*

9.3.1.2 Outcomes

— *SRS for the application* is produced.

— *Trace links to application requirements sources including the domain requirements associated with the bound variants* are established.

9.3.1.3 Tasks

— *Identify sources of application specific requirements.* This task serves to ensure that stakeholders for the application specific requirements know why and how every application requirement belongs to the SRS and to facilitate further clarification. The results are then integrated with the domain's result.

— *Establish traceabilities for application requirements.* Traceabilities relevant to application requirements are established.

— *Document application specific requirements.* The functions and capabilities that are required by a specific application should be provided along with required constraints and the results are integrated with the domain requirements document as a whole (this document is an application requirements specification).

— *Document the rationale for variability decision.* Rationales of decision making related to the variability decision and their trade-offs are documented.

— *Review the application requirements specification.* Application requirements specification is reviewed to ensure its completeness and correctness.

9.3.2 Identify sources of application specific requirements

Sources of the application specific requirements are identified for establishing trace links between requirements and their sources.

The method should support identifying the sources of application requirements for the bound variants with the following capabilities:

— analysing the domain requirements' annotations (can use the requirements annotations definition) associated with the bound variants;

— identifying the sources of application specific requirements;

— adding annotation to application specific requirements in a consistent way with the domain requirement.

A tool should support identifying the sources of requirements for the bound variants with the following capabilities:

— allowing reference to information from the application requirements elicitation to the application requirements analysis;

— providing templates for recording the requirements annotations.

9.3.3 Establish traceabilities for application specific requirements

Most trace links that are established during domain requirements engineering are reused as they are, and in this task, only the trace links for the bound variants and application specific requirements are established.

The method should support establishing traceability for the bound variants with the following capabilities:

— establishing trace links to their sources, domain requirement, and to the application variability model.

A tool should support establishing traceability for the bound variants with the following capabilities:

— storing the application requirements trace links;

— showing the application requirements trace links at specific points.

9.3.4 Document application specific requirements

The goal of this task is to produce a well-structured and organized application requirements specification. For this purpose, the domain requirements specification should also be well structured.

The method should support documenting application requirements for the bound variants with the following capabilities:

— assigning values for all requirement annotations;

— aggregating documentation from the application requirements elicitation and analysis;

— integrating the document with the relevant domain requirements artefacts.

A tool should support documenting application requirements for the bound variants with the following capabilities:

— browsing the relevant domain requirements artefacts;

— entering application specific requirements(for the bound variants);

— integrating the application specific requirements documentation with the bound domain requirements specification;

— checking the document quality and consistency (e.g. through spell checking, data dictionaries, and acronym tables).

9.3.5 Document the rationale for variability decision

This task is an application specific one to document the rationale for decision making concerning the binding of the variants.

The method should support documenting the rationale of the variability decision with the following capabilities:

— identifying all the rationales for the decision;

— documenting the rationales for the variability bindings.

A tool should support documenting the rationale of the variability decision with the following capabilities:

— allowing reference to the rationales with the relevant decisions;

— providing document templates for documenting the rationales.

9.3.6 Review the application requirements specification

The goal of this task is to ensure the correctness, completeness, and unambiguity of the specified application requirements. Because the domain requirements specification is reviewed during domain requirements engineering, in this task, only the application specific requirements specification is reviewed.

The method should support reviewing the specified application requirements with the following capabilities:

— analysing the contents of application requirements specification;

— resolving issues and revising application requirements specification.

A tool should support reviewing the specified application requirements with the following capabilities:

— importing and browsing the application requirements specification;

— providing an environment for reviewing, discussing, and commenting on the requirements specification.

9.4 Application requirements verification and validation

9.4.1 Purpose of the application requirements verification and validation

The purpose of this subprocess is to ensure the completeness, correctness, consistency, and unambiguity of the application specific requirements and that bound variants have considered all the defined dependencies and constraints of domain requirements.

9.4.1.1 Inputs

— *All artefacts of the application requirements engineering tasks.*

— *Relevant domain requirements artefacts.*

— *Relevant variability models.*