INTERNATIONAL STANDARD

ISO/IEC 26555

Second edition
2015-12-01

Software and systems engineering — Tools and methods for product line technical management

Ingénierie du logiciel et des systèmes — Outils et méthodes pour le management technique des gammes de produits

Reference number
ISO/IEC 26555:2015(E)

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, *Systems and software engineering*.

This second edition cancels and replaces the first edition (ISO/IEC 26555:2013), of which it constitutes a minor revision.

Introduction

The major purpose of this International Standard is to deal with the capabilities of tools and methods of software and systems product line (SSPL) Technical Management. This International Standard defines how the tools and methods can support for the software and systems product line-specific technical management processes. Since product lines deal with multiple products that have similarities, product lines have an unprecedented level of technical management complexities. This arises from the following sources:

— There are inherent differences in technical considerations because there are parallel development processes, domain and application engineering, in a product line and the two processes are tightly related with each other around assets.

— The close relationships among domain engineering, application engineering, and assets require the highly matured managerial capabilities for addressing relationships among them.

— There are lack of tools and methods to support the product line-specific technical management.

Technical management provides management support for a timely and proper deployment of product line in balance with pre-defined product line objectives such as reusability, reducing cost, improving quality, etc., as well as its planned cost, schedule, and resources. Technical management addresses actual means used to support, monitor, and control the activities of both domain engineering and application engineering of a product line.

There are needs for defining product line-specific technical management processes that integrate the involved product line disciplines with those for a single product. Furthermore, support of tools and methods are required so that a product line organization can perform technical management under the systematic control of complexities. This International Standard addresses the product line-specific processes in technical management by dividing those into *process management, variability management, asset management, and support management* areas with the guidance of a set of tools and methods capabilities for supporting tasks for product line technical management.

This International Standard is intended to benefit people who acquire, supply, develop, operate, and maintain tools and methods for product line technical management. This International Standard can be used in one or more of the following modes:

— By an organization intended to implement product lines—to understand, adopt, and enact the processes, tools, and methods for product line technical management. This also helps the organization to evaluate and select relevant tools and methods based on business and user-related criteria.

— By a tool vendor who facilitates or leverages product line engineering practices—to provide a set of tool capabilities that should be embodied in a tool for supporting product line technical management.

ISO/IEC 26550 addresses both engineering and management processes and covers the key characteristics of product line development. ISO/IEC 26550 provides an overview of the consecutive international standards (i.e. ISO/IEC 26551 to ISO/IEC 26556), as well as the structure of the model:

— Processes and capabilities of methods and tools for product line scoping, domain requirements engineering, and application requirements engineering are provided as ISO/IEC 26551.

— Processes and capabilities of methods and tools for domain design and application design are provided as ISO/IEC 26552.

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

— Processes and capabilities of methods and tools for domain verification and validation and application verification and validation are provided in ISO/IEC 26554.

— Processes and capabilities of methods and tools for technical management are provided in this International Standard.

— Processes and capabilities of methods and tools for organizational management are provided in this International Standard.

Software and systems engineering — Tools and methods for product line technical management

1 Scope

This International Standard deals with the tools and methods of technical management for software products, software services, software-intensive systems (including System Architecture and excluding hardware) within a product line. The scope of this International Standard is as follows:

— Enable the users of this standard to holistically understand, adopt, and enact the processes, tools, and methods for product line technical management. In addition, this International Standard helps the users evaluate and select relevant tools and methods based on business and user-related criteria.

— Help product line engineers, developers, and tool vendors become informed about capabilities of tools and methods that are required for supporting product line implementation from technical aspects.

— Provide product line-specific processes and capabilities of tools and methods in technical management.

This International Standard does not concern processes and capabilities of tools and methods for technical management for a one-of-a-kind system but rather deals with those belonging to a family of systems.

NOTE System Architecture is a set of logical and physical principles used to achieve a mission within a given environment. From System Architecture are derived components that can be subsystems, software products, human-based products like crew or operators or hardware product like mechanical structures, electronic boards, chemicals, etc. The scope of the International Standard spans from the system, to sub-systems, and software products. Other types of components and especially those related to human beings and to hardware parts are not within the scope of this International Standard.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

There are no normative references cited in this document.

3 Terms and definitions

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

3.1
application engineering process
processes for developing a member product in a product line

3.2
attached process
process definitions how each asset will be used in application

Note 1 to entry: The set of attached processes are those for orchestrating the assets together into a coherent whole application.

3.3
binding time
moment of variability resolution

3.4
external variability
variability that is visible to customers

3.5
domain engineering process
processes for domain asset development

3.6
internal variability
variability defined from an engineer's perspective and is not visible to customers

3.7
variability binding
act of determining the variant of the variation point defined in the variability model

3.8
variability documentation
detailed description of variability models for being used across the member products within a product line

3.9
variability in space
variation that occurs at the same time with different shape

3.10
variability in time
variation that occurs at different times

3.11
variability mechanism
variability implementation methods in a product line for supporting assembly of domain assets

3.12
variability traceability
trace links established for a variability model both with domain assets and with application assets where variants are bound.

4 Reference model for product line technical management

The technical management provides processes for addressing technical issues that arise during implementation of a product line, deployment of processes, and development of assets and products within a product line in balance with accomplishing the objectives of a product line, the given costs, schedule, and so forth. Technical management process group deals with variability management, asset management, process management, support management process for managing technical capabilities, resources, issues, and skills that are required for successful implementation of product lines.

Tools and methods in technical management support the systematic development of product lines. Because of profound complexity of product lines, proper tool support for domain engineering lifecycle, application engineering lifecycle, technical management, and organizational management should be provided so that a product line organization can control the technical complexities.

The reference model for technical management in Figure 1 is structured into four processes: process management, variability management, asset management, and support management. Each process is divided into subprocesses and each subprocess is described in terms of the following attributes:

— the title of the subprocess;

— the purpose of the subprocess;

— the inputs to produce the outcomes;

— the tasks to achieve the outcomes;

— the outcomes of the subprocess.

The capabilities of tools and methods are a list of the required support of tools and methods for performing the tasks properly.

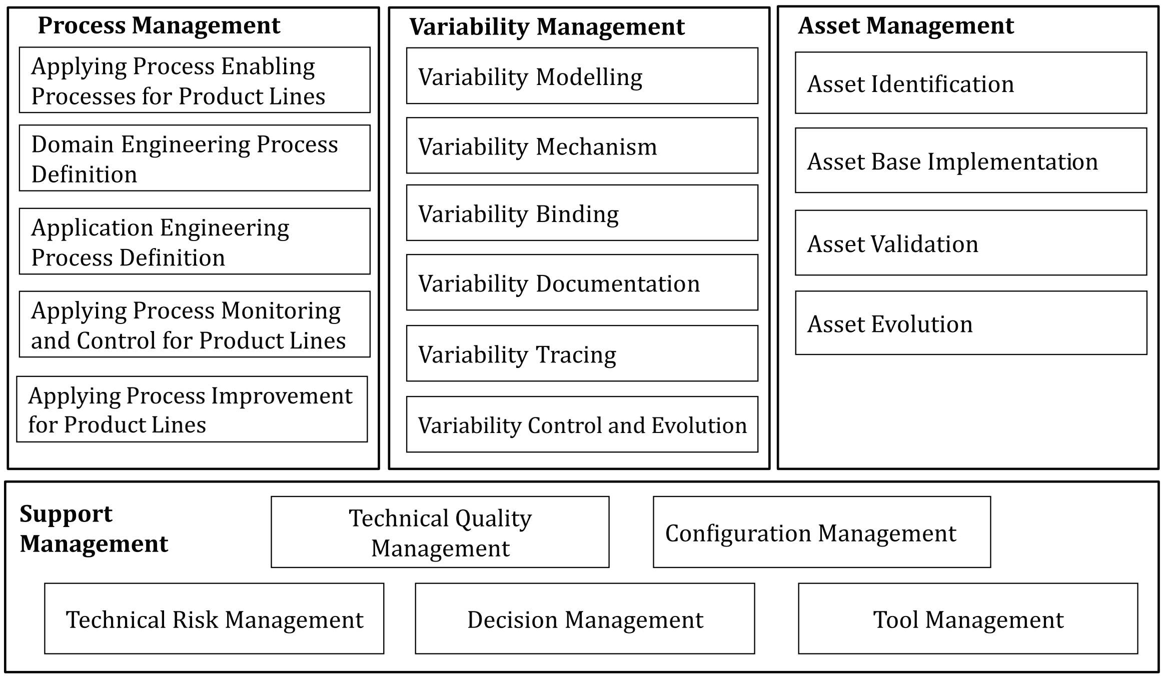
Process Management
Applying Process Enabling Processes for Product Lines
Domain Engineering Process Definition
Application Engineering Process Definition
Applying Process Monitoring and Control for Product Lines
Applying Process Improvement for Product Lines
Variability Management
Variability Modelling
Variability Mechanism
Variability Binding
Variability Documentation
Variability Tracing
Variability Control and Evolution
Asset Management
Asset Identification
Asset Base Implementation
Asset Validation
Asset Evolution
Support Management
Technical Quality Management
Configuration Management
Technical Risk Management
Decision Management
Tool Management

Figure 1 — Product line technical management

Product line technical management shall be conducted from a multidimensional viewpoint that is domain and application engineering. As managerial targets vary according to each viewpoint, management processes for each of them differ. Moreover, assets are used across all applications, and hence, applications within a product line shall share common processes for deriving their applications from assets.

Product line development tends to be performed in a parallel or distributed manner, and therefore technical management processes and related capabilities of tools/methods should be defined by considering these aspects. As for variabilities, they can be managed orthogonally in order to avoid complexity of a product line. Variabilities can be managed orthogonally in order to avoid complexity of a product line. Therefore, the modeling and evolution of variabilities and their traceabilities with domain/application assets should be engineered systematically balancing with its cost, effectiveness, and supported over the product line life cycle.

Project processes and software support processes provided in ISO/IEC 12207 are compatible with the technical management dealt in this International Standard. However, the life cycle model management process that is categorized in organizational project-enabling processes provided in ISO/IEC 12207 is addressed in the process management part of this International Standard.

Because there are collaborations among groups of people in engineering and managing product lines, some relevant processes shall support these collaborations. Process management process serves for establishing and managing a product line organization’s capabilities of implementing product line processes. It shall serve to do the following and to define the capabilities of tools and methods for related support:

— *Applying process enabling processes for product lines* addresses the organization’s capabilities for initiation, execution, control, and improvement of product line processes.

— *Domain engineering process definition* establishes and maintains domain engineering processes for product line platform development.

— *Application engineering process definition* defines and maintains application specific processes for developing each application based on product line platform.

— *Applying process monitoring and control for product lines* aims to align domain/application engineering processes to achieve product line goals and objectives.

— *Applying process improvement for product lines* serves for the assessment and improvement of domain/application engineering processes.

Variability throughout the domain and application engineering lifecycle processes of a product line should be managed properly. Variability management process defines how member products in a product line can vary and includes variability model management and the explicit documentation of variability. Variability management process shall serve to do the following and to define the capabilities of tools and methods for related support:

— *Variability modelling* supports to maintain the integrity of domain variability model and application variability models.

— *Variability mechanism* supports to realize the variability of a product line in accordance with the determined binding times, rules and constraints, and domain artifacts.

— *Variability binding* maintains necessary binding information to achieve the efficient development of a member product and proactive reuse of product line platform.

— *Variability documentation* supports detailed descriptions of variability models to provide the rationales of variability related decisions.

— *Variability tracing* establishes and maintains trace links between elements of the variability model and the associated domain/application assets.

— *Variability control and evolution* deals with change requests, change impact analysis, change execution, and verification/validation for the change.

The asset management process covers the establishment of asset base and change management for the assets. Domain engineering is responsible for creating and maintaining the reusable domain assets that are stored and managed in the asset base. It also manages the application assets that are worth generalizing and incorporating into domain assets. Asset management process includes management activities for application assets produced and maintained for each member product in a product line. Asset management process shall serve to do the following and to define the capabilities of tools and methods for related support:

— *Asset identification* selects asset candidates (e.g. features, models, specifications, and test cases) developed in domain/application engineering.

— *Asset base implementation* configures the structure of domain/application assets and makes them easy to mine, retrieve, and manage.

— *Asset validation* aims to assure whether the defined strategy of asset structure has been reflected.

— *Asset evolution* serves to deal with change requests from domain and application lifecycle processes.

Support management process deals with generic subprocesses that support the implementation, qualification, and automation/semi-automation of other processes. This includes subprocesses that can be generally applied across the product line organization. Support management process shall serve to do the following and to define the capabilities of tools and methods for related support:

— *Technical quality management* provides assurance about whether assets from domain and application engineering comply with predefined quality criteria and whether domain and application engineering processes comply with predefined provisions.

— *Configuration management* controls the configurations of product line platform as well as configurations of each member product in a product line.

— *Decision management* aims to select the best option where alternatives exist. Quantitative criteria should be considered to achieve an objective decision.

— *Technical risk management* deals with risk identification, assessment, prioritization, and mitigation to prevent failures that inhibit the achievement of business values and product line objectives.

— *Tool management* serves to improve productivity and quality by automation or semi-automation of domain engineering, application engineering, technical management, and organizational management processes.

The identification and analysis of the aspects involved in product line technical management process group will enable an organization to understand the technical management process group and to formulate a strategy for the successful implementation of a product line. The technical management process group shall be defined from these aspects and product line-specific tasks shall be identified on the basis of these aspects.

Table 1 shows the key aspects for each characteristic of product line technical management process group:

Table 1 — Key aspects for identifying product line-specific technical management tasks

Category	**Aspects**
Reuse management	application engineering, 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, verification and validation

— *Application engineering:* Application engineering processes provide processes for developing applications based on assets. Discrimination of domain and application engineering processes is a unique technical aspect of product line development.

— *Asset:* Technical management provides managerial capabilities necessary for managing assets. Asset management is a distinguished aspect of product line development.

— *Binding:* Technical management serves to prepare sufficient information necessary to bind variants, so that each application can focus on only on application specific parts. Variability binding is an aspect that distinguishes technical management of the product line development from that of single product development.

— *Collaboration:* Collaboration is essential in a product line because product line engineering uses artifacts or products developed in different organization units. Technical management enables that people do their technical roles and responsibilities within agreed-on commitments.

— *Configuration:* Configurations of assets for a product line are distinguished technical aspects of product line development. Maintaining integrity of assets is an important aspect. Configurations of products and artifacts of a product line can be multidimensional, i.e. exist in time and space.

— *Domain architecture:* Technical management enables the decision for obtaining individual component assets of the domain architecture.

— *Domain engineering:* Domain engineering processes are a technical aspect that distinguishes product line development from single product development.

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

— *Measurement and tracking:* Technical management defines tasks, tool capabilities, and method capabilities for the product line measurement and tracking. The performance of the product line processes should be measured aligning with the overall product line objectives. The measurement results should be collected, and they shall be used to control product lines for the better achievement of the product line objectives.

— *Platform*: Technical management enables the development of a platform and the development of applications based on the platform.

— *Product management:* Product management should have capability for defining and analysing the measures that make it possible to evaluate designed reusability and productivity and thereafter coordinate a product line towards achieving its goals. These are possible under the support of technical management.

— *Reusability:* Technical management monitors and controls whether the desired level of reusability is achieved in a product line. Providing managerial support for achieving desired level of reusability is a key aspect peculiar to product line development.

— *Texture:* Technical management enables that application can obey the rules defined in textures. Managerial support for obeying textures are dealt in technical management.

— *Traceability:* There exist trace links for asset management and variability management in technical management.

— *Validation and verification:* Validation and verification of assets, platform, and variability model are an aspect that distinguishes product line development from single development.

— *Variability:* Technical management provides managerial capabilities for variability. Variability management is a distinguished aspect of product line development.

5 Process management

Because product line requires a group of people working together and calls for repeated, ongoing, disciplined interactions among separate organizational units, it is essential to make participants adhere a process. Product line engineering relies on a high quality process definition and its effective enactment to achieve required fidelity. In particular, because changes on assets impact more than one application, configuration and changes should be controlled in a disciplined way and asset developers should follow the disciplines carefully. The subprocesses of process management are as follows:

— *Applying process enabling processes for product lines* serves to ensure the readiness for product line process. This subprocess provides capabilities for establishing product line process leadership, resources, and collaboration environment, and supporting product line process assessment and improvement.

— *Domain engineering process definition* serves to define the commonly used processes within a product line and make the participants share them. This subprocess is composed of tasks such as defining, validating, and deploying domain engineering processes.

— *Application engineering process definition* serves to define the application specific processes for developing applications within a product line context. Application engineering processes address how application developers develop their applications based on platform, assets, and variability models. Application engineering processes should be tailored (if necessary) or harmonized with domain engineering processes for right use of assets.

— *Applying process monitoring and control for product lines* serves to measure the performance of processes and if necessary, planning corrective actions to resolve problems for fixing the deviations. Processes for identifying measurement objectives and process-performance objectives are provided. Measures correspond to the overall product line objectives are defined and their measurement results are collected for controlling processes.

— *Applying process improvement for product lines* is normally proceeded by organizational process assessment which identifies the weaknesses and strengths of the domain and application engineering processes. Process improvement should be systematic to migrate from an as-is state toward a to-be state.

5.1 Applying process enabling processes for product lines

Purpose of applying process enabling processes for product lines

The purpose of applying process enabling processes for product lines is to get ready for process implementation and improvement such as relevant-policies, steering groups (e.g. SEPG) for process assessment and improvement, and resources for executing them.

Because a group of people who belong to different organization units, sometimes working at geographically different locations, is involved in a product line, a product line organization requires support for communicating about domain engineering processes and obtaining agreement on processes. Moreover, a centralized organization responsible for commitment, assessment, and improvement is highly required for providing integrated process enabling environment.

Inputs

— Domain and application process assets (e.g. documented policies, process assessment and improvement assets used in the single system development, organizational roles and responsibilities, etc.).

— Resource inventory with availability.

— Size and complexity of a product line.

Outcomes

— *Product line process management group is established.*

— *Communication and collaboration work environments are facilitated.*

— *Process improvement goals and strategies are documented.*

— *Requirements for infrastructure and resources for enabling processes are defined.*

— *Infrastructure for process-enabling is implemented and maintained.*

Tasks

— *Establish process management group* is to establish the sponsorship and accountability for leading the process management and improvement.

— *Align resources for process improvement* is to provide adequate resources for developing process assets, performing process improvement, facilitating an environment that enables communication about processes, and forming a basis for cooperating to solve a problem of processes. Necessary supports for executing the established process improvement goals and strategies are identified and provided.

— *Govern process improvement* is to monitor and control the status of process improvement and implementation from the overall product line perspective. Process management group as a process control tower determines appropriate actions for steering process improvement.

5.1.1 Establish process management group

The goal of this task is to establish and maintain the sponsorship and accountability for the process management and improvement. Process management group (PMG) has responsibilities for leading the remaining tasks of process enabling process and making necessary decisions.

Process improvement goals and strategies for the product line are established and maintained by product line management group. The process improvement goals and strategies reflect the improvement needs for better achievement of product line objectives such as improving reusability, cost saving by enhancing reusability, and etc., as well as for better achievement of organization's business goals.

The method should support establishing product line process management group with the following capabilities:

— defining process improvement goals and strategies including high-level measures for monitoring them based on product line objectives;

— simulating achievable reusability improvement, cost reductions, efficient variability management and etc. through the process improvement (e.g. Delphi);

— coordinating the status of domain, application, and collaboration processes (e.g. AHP, ANP, CSF, Delphi).

A tool should support establishing product line process management group by allowing the user to do the following:

— promoting collaboration (or communication) environments for knowledge sharing regardless of geographical location;

— sharing the status of the separated and concurrent development processes, i.e. domain engineering processes, application engineering processes;

— sharing deliverables and milestones of process improvement.

5.1.2 Align resources for process definition and improvements

The goal of this task is to provide the product line organization with necessary resources for implementing and improving processes. The resources contain skills, budgets, and facilities for executing goals and strategies.

Since in product line engineering there are parallel processes, i.e. domain engineering and application engineering, and a product line organization is cross-functional, collaborations among organization-units or teams that have different roles and responsibilities are occurred continuously. Therefore, facilities enabling communication and collaboration among product line participants are required.

The method should support aligning resources for process definition ad improvements with the following capabilities:

— reviewing the resources need to perform process improvement;

— harmonizing resources in the processes for the different lifecycles, i.e. domain engineering, application engineering, and management;

— reallocating resources assigned to domain engineering and therefore the resources are shared with application engineering (elevating resource reusability);

— establishing the plans and commitments to provide the resources.

A tool should support aligning resources for process definition and improvements by allowing the user to do the following:

— getting feedbacks (or providing feed-forwards) about resources from product line participants (e.g. Web-based environment for collecting opinions about resources);

— accessing resource list with their status (resources allocated to domain engineering, all application engineering within a product line, and management);

— providing the usage status of resources for domain engineering, application engineering, and management respectively for reuse.

5.1.3 Govern process definition and improvement

The goal of this task is to monitor and control the status of domain and application engineering process improvement from the process management group, which is a process control tower. Product line management group decides the appropriate corrective actions for process improvement and they get a common understanding for those with the executive level managers.

The method should support governing process definition and improvement with the following capabilities:

— approving, measuring, and managing process improvement;

— aligning process improvement in the product line organization;

— reviewing the status of the process improvement.

A tool should support governing process definition and improvement by allowing the user to do the following:

— supporting status reports for checking whether the process improvement are on track (in the case of commonly used domain engineering processes among applications, this capability is crucial);

— allowing reference of measurements, their integration results, and work products related to process improvement to make informed decisions.

5.1.4 Prepare process management and improvement

The goal of this task is to provide appropriate resources and control for process management and improvement.

The necessary resources for developing domain engineering process assets, performing process improvement, facilitating an environment that enables communication about domain engineering processes among application engineers, forming a basis for cooperating to solve a problem related to domain engineering processes, and executing the established process improvement goals and strategies are identified and provided.

Process improvement goals and strategies reflect the improvement needs for better achievement of product line objectives such as improving reusability, cost saving by enhancing reusability, and etc. as well as for better achievement of organization’s business goals.

The method should support preparing process management and improvement with the following capabilities:

— defining process improvement goals and strategies including high-level measures for monitoring them based on product line objectives;

— simulating achievable reusability improvement, cost reductions, efficient variability management and etc. through process improvement (e.g. Delphi);

— estimating and procuring necessary resources;

— coordinating the status of domain, application, and collaboration processes (e.g. AHP, ANP, CSF, Delphi);

— optimizing resources in the processes for different lifecycles, i.e. domain engineering, application engineering, and management.

A tool should support preparing process management and improvement by allowing the user to do the following:

— promoting collaboration (or communication) environments for knowledge sharing regardless of geographical location;

— getting feedbacks (or providing feed-forwards) about resources from product line participants (e.g. Web-based environment for collecting options (or alternatives) about resources);

— sharing the status of separated and concurrent development processes, i.e. domain engineering processes, application engineering processes;

— sharing deliverables and milestones of process improvement.

5.2 Domain engineering process definition

Purpose of domain engineering process definition

The purpose of this subprocess is to manage the organization's capability to establish a set of processes for domain engineering. Because product lines call for the disciplined interactions among the different organizational units to develop assets and products, there are processes that all participants should follow (be reused as they are) and processes that can be reused after tailoring. The fundamental goal of product lines is reuse, and hence processes are also reusable objects. When a product line organization defines processes they should provide architecture, detail description, and implementation guides for processes.

Inputs

— A set of organization's standard processes, which has been published for developing a single product.

— Tailoring guidelines used for developing a single product.

Outcomes

— *A set of domain engineering processes is defined.*

— *Tailoring guidelines for processes are established.*

— *Harmonization criteria and guidelines are documented.*

— *All of above are shared within the product line organization.*

Tasks

— *Define domain engineering processes* establish and maintain a set of common processes that should be followed by the participants of domain engineering. Because within product lines people work together cooperatively and each product is developed utilizing the same platform, every participant does their job within the agreed processes.

— *Validate domain engineering processes* ensure that the defined processes address the issues of domain engineering in the product line development such as reusability, assets, variability, traceability, and so on.

— *Deploy the domain engineering processes for reuse* to ensure that all participants within the product line organization are made aware that there is a set of industrial-strength standard processes, organizational standard processes, and tailoring guidelines available to participants.

5.2.1 Define domain engineering processes

The goal of this task is to define common processes and to develop process assets that will be commonly used by domain engineers for developing domain assets. Domain engineering processes provide processes for developing assets and most of them are reused in application engineering. Domain engineering processes have to provide common process architecture and detailed process definitions to minimize the deviations among participants of domain engineering.

The method should support defining domain engineering processes with the following capabilities:

— providing templates for describing domain engineering processes (process architecture, detailed design of processes, and implementation of processes);

— defining purpose, inputs, outcomes, entry criteria, tasks, verification, measurement, and exit criteria of domain engineering processes;

— decomposing processes into domain engineering process elements for domain asset development;

— defining processes for supporting assembly of domain assets;

— providing processes that would guide the collaborations in a product line;

— modeling and documenting processes (providing templates for documentation).

A tool should support defining domain engineering processes by allowing the user to do the following:

— providing integrated process support environments to automate the required processes;

— allowing electronic process documentation for the user of a process (e.g. a web-based process handbook or an electronic process guide).

Process descriptions should contain guides to handle variability, binding, reuse, and collaboration among different organization units.

5.2.2 Validate domain engineering processes

The goal of this task is to ensure the capabilities of domain engineering processes required to address domain engineering specific issues such as commonalities, variabilities, assets, and traceabilities.

The method should support validating domain engineering processes with the following capabilities:

— reviewing whether the defined domain engineering processes address necessary purpose, outcomes, inputs, entry criteria, tasks, verification, measurement, and exit criteria;

— ensuring that the defined domain engineering processes address variability relevant tasks;

— confirming that the defined domain engineering processes can produce assets and a platform that will be reused within a product line;

— reviewing whether the defined domain engineering processes direct the establishment of appropriate traceabilities.

A tool should support validating domain engineering processes by allowing the user to do the following:

— guiding validation tasks (e.g. providing web-based checklist);

— making the defined domain engineering process assets available via online;

— generating reports for the validation results.

5.2.3 Deploy the domain engineering processes

The goal of this task is to deploy the defined common processes for use. This task ensures that the domain engineering processes and process assets are made available to perform domain engineering and its accompanying management.

The method should support deploying the domain engineering processes with the following capabilities:

— designing and implementing the common process asset library;

— specifying the harmonization criteria and guidelines for reuse in application engineering;

— specifying the procedures for deploying processes;

— making domain engineering process assets available for use.

A tool should support deploying the domain engineering processes by allowing the user to do the following:

— supporting domain engineering process asset repository for sharing domain engineering process assets, tailoring guides, and harmonization criteria with participants of product lines;

— monitoring whether an application adheres to the relevant domain engineering processes;

— allowing ability to integrate feedbacks from member applications.

5.3 Application engineering process definition

Purpose of application engineering process definition

The purpose of this subprocess is to establish the processes that fit to each application. Application engineering processes are defined by reflecting application specific needs, goals, and etc. and by harmonizing with the domain engineering processes.

There are processes each member product shall follow whereas, there are those flexible in accordance with member products. Furthermore, those processes collaborate with each other or are performed in parallel or independently. Fundamentally, processes should be systematically reused in the product line organization. When a process is reused, there are possibilities of a process mismatch among organization units, products, and so forth.

Inputs

— Organizational standard processes.

— Domain engineering process assets.

— Application specific needs.

— Tailoring guidelines.

Outcomes

— *A set of application engineering processes is defined.*

— *Application engineering process assets are established and maintained.*

Tasks

— *Define application engineering processes.* Application development processes are defined for developing an application by reusing domain assets, binding variabilities, and addressing application specific needs such as application specific quality requirements, technical difficulties, people’s skills, and so forth. Application engineering processes should be harmonized with domain engineering processes.

— *Validate the conformance of application engineering processes with domain engineering processes*. Consistency with domain engineering processes should be assured so that the domain objectives and strategies are appropriately addressed.

— *Deploy the application engineering processes* to ensure that all participants of an application development use the defined application engineering processes.

5.3.1 Define application engineering processes

The goal of this task is to define application specific processes that will be used to develop each application by reusing domain assets, binding variabilities, and developing application specific parts.

While harmonizing with domain engineering processes application engineering processes should address the contextual variables of an application such as cost, schedule, quality tradeoffs, technical difficulties, and experience of the participants. Moreover, because applications interact with variability models for variability binding, the defined application engineering processes should cover this.

With the application specific processes defined, learning about the performance of application engineering processes can be shared with other applications within a product line and they might be fed into domain engineering processes for improving them.

The method should support defining application engineering processes with the following capabilities:

— providing templates for describing application engineering processes (process architecture, detailed design of processes, and implementation of processes);

— defining purpose, outcomes, inputs, entry criteria, tasks, verification, measurement, and exit criteria of application engineering processes;

— defining processes for dealing with asset reuse;

— defining processes for considering variability binding;

— defining processes for addressing application specific parts;

— modeling and documenting application engineering processes (providing templates for documentation);

— harmonizing application engineering processes with domain engineering processes.

A tool should support defining application engineering processes by allowing the user to do the following:

— providing templates for describing application engineering processes;

— supporting modeling and documenting application engineering processes;

— allowing traceability with asset base to refer reuse information;

— allowing traceability with variabilities including models, annotations, etc.;

— supporting traceability with defined domain engineering processes for harmonization.

5.3.2 Validate the conformance of application engineering processes with domain engineering processes

The goal of this task is to assure that application engineering processes conform to domain engineering processes, so that each application is consistent with domain engineering processes and thereafter the performance of processes and quality of a product can be predictable.

The method should support validating conformance of application engineering processes with domain engineering processes with the following capabilities:

— reviewing whether the defined application engineering processes address necessary purpose, outcomes, inputs, entry criteria, tasks, verification, measurement, and exit criteria;

— ensuring that the application engineering processes adhere to mandatory parts of domain engineering processes;

— ensuring whether or not application engineering processes reflects the application engineering context in the product line development;

— ensuring that the defined application engineering processes satisfies the goals of a product line organization.

A tool should support validating conformance of application engineering processes with domain engineering processes by allowing the user to do the following:

— providing an environment that supports the coordination of domain and application engineering processes;

— guiding validation procedure step by step;

— sharing domain engineering process assets and assets with annotations such as attached processes, constraints, and etc.;

— supporting documentation for non-compliance issues.

5.3.3 Deploy the application engineering processes

The goal of this task is to deploy the defined application processes for use. This task ensures that the application engineering processes and process assets are made available to perform application engineering and its accompanying management.

The method should support deploying the application engineering processes with the following capabilities:

— designing and implementing the application specific process asset library;

— specifying the harmonization criteria and guidelines to reuse domain engineering process assets;

— specifying the procedures for deploying processes;

— mining domain engineering process assets for reuse.

A tool should support deploying the application engineering processes by allowing the user to do the following:

— mining domain engineering process asset base for sharing domain engineering process assets, tailoring guides, and harmonization criteria;

— monitoring whether an application adheres to the relevant domain engineering processes;

— allowing ability to provide feedbacks to domain engineering.

5.4 Applying process monitoring and control for product lines

Purpose of process monitoring and control

The purpose of applying process monitoring and control for product lines is to support the organization's capabilities in planning, performance measurement, and reusability of processes.

Inputs

— Product line organization's objectives.

— Product line process assessment framework.

— Collected data or information.

Outcomes

— Achieved degree of process reusability is determined.

— Process performance is determined,

Tasks

— *Plan for process monitoring and control*. Process performance objectives and plans of process monitoring and control for confirming whether or not the implemented processes are comply with the standard processes are established and maintained. A product line objective that is able to be characterized by processes is selected. And it is traced through tracking the process performance.

— *Define process performance measures*. Measures for analyzing the process performance are defined and thereafter measurements are initiated.

— *Measure and manage process performance*. Process performance results are characterized and the results are fed into the relevant processes for promoting the accomplishment.

— *Coordinate processes for improving reusability.* Processes in the product line development are major assets reused, and therefore reusability of processes are monitored and coordinated for promoting the overall reusability of product lines.

5.4.1 Plan for process monitoring and control

The goal of this task is to identify the process performance objectives and establish plans for monitoring the actual results achieved by following processes. In addition, processes that will be monitored and controlled are selected.

Product line objectives established at the product line initiation phase, can be characterized by process and product measures. The product line objectives such as reusability, productivity, and cost saving should be monitored and controlled by measuring process-performance.

The method should support planning for process monitoring and control with the following capabilities:

— determining domain/application engineering process performance objectives (e.g. process performance objectives that are drawn from the product line objectives and process improvement strategy and goals);

— selecting a set of processes that will be monitored;

— reviewing and negotiating the objectives with the relevant stakeholders;

— assigning resources for performing the monitoring and control.

A tool should support planning for process monitoring and control by allowing the user to do the following:

— providing different planning guides by what plans are for (i.e. for domain engineering processes or application engineering processes);

— supporting environment for collaborating and sharing plans among relevant stakeholders, such as domain and application managers;

— sharing plans for monitoring and controlling performance of domain engineering processes;

— browsing plans of monitoring and control for reusing them among applications.

5.4.2 Define process performance measures

The goal of this task is to define the measures used for analysing and controlling the process performance. Through the process cascading, relevant processes and their quantitative responsibilities, namely performance objectives for each relevant process are assigned.

The method should support defining process performance measures with the following capabilities:

— defining measures that will be used to monitor the processes toward the defined objectives;

— cascading the performance objectives to the relevant processes;

— specifying and assigning roles for measuring process performance;

— aligning the assignment results with stakeholders;

— providing measurement templates reflecting characteristics of domain and application engineering processes.

A tool should support defining process performance measures by allowing the user to do the following:

— providing storage for commonly used process performance measures so as to reuse and share them;

— providing access services for measures and their relevant processes;

— sharing measurement results by measurement procedures who are in charge of measurement of domain engineering processes.

5.4.3 Measure and manage process performance

The goal of this task is to analyze and translate the measurement results into the value of process performance and take corrective actions to improve process performance.

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

— characterizing the measurement results for analyzing process performance (e.g. using process performance models);

— analysing causes that tackles the achievement of process performance objectives;

— reviewing and negotiating the corrective actions with relevant stakeholders;

— tracking the status of corrective actions.

A tool should support managing process performance by allowing the user to do the following:

— calculating achievement results based on process performance models of each of application, domain, or overall product line;

— integrating performance data from each of the application engineering processes for calculating the achievement of domain engineering processes;

— displaying achievement results in accordance with the roles and responsibilities (e.g. views for application, domain, or overall product line).

5.4.4 Coordinate processes for improving reusability

The goal of this task is to refine processes for maximizing reusability. In product lines processes are one of major reusable assets and process reuse is important for effective reuse of core assets.

The method should support coordinating processes for improving reusability with the following capabilities:

— assessing process reusability;

— deriving opportunities for improving process reusability;

— reviewing and negotiating the process improvement with relevant stakeholders;

— performing process simulation for validating the effectiveness.

A tool should support coordinating processes for improving reusability by allowing the user to do the following:

— supporting calculation of process reusability;

— tracking what processes are reused;

— facilitating (web-based) environments for the coordination among for improving process reusability;

— providing hyper-navigation service between application engineering process and relevant domain engineering process.

5.5 Applying process improvement for product lines

Purpose of process improvement

The purpose of applying process improvement for product lines is to provide organizational capabilities in assessment, change impact estimation, improvement plan, improvement results evaluation of processes.

Inputs

— Change requests for domain engineering processes.

— Actual domain and application process-performance.

Outcomes

— *Assessment results* are documented.

— *Action plans for improvement* are documented.

— *Changes on domain engineering processes* are documented.

Tasks

— *Assess processes*. Product lines rely heavily on the adherence to processes in order to build products successfully within a product line. Process assessment should be conducted for both domain engineering processes, which develop domain assets, and application engineering processes, which are application specific processes.

— *Estimate the impact of changes on processes.* Changes on a process have impact either on processes that cooperate with it or on processes reused by products that are developed in parallel within a product line. Therefore, the impact due to process changes have to be estimated before making changes or enacting changed processes.

— *Plans for process improvement*. Since the domain engineering processes are shared among product line participants planning for improvement should be established carefully. The priorities of improvements should be determined in accordance with their contributions to the achievement of the organization's product line objectives.

—— *Implement process improvements*. Pilots through structuring domain subset using representative domain engineering processes should be conducted before deploying the changed processes. Each application also conducts pilots in advance, so that the problems due to the changes should be detected before implementing the processes throughout the product line.

— *Evaluate process improvements*. Product line organization ensures whether the action plans for product improvement are appropriately implemented to all relevant parts of a product line. Metrics for monitoring and measuring the process improvement status of a product line should be defined.

5.5.1 Assess processes

The goal of this task is to appraise the domain engineering processes periodically in order to better understand the strengths and weaknesses of the processes. Process assessment is conducted from both domain and application engineering aspects. Process improvement action plans should be reviewed and negotiated with the organizational units responsible for implementing the process action plans. Action plans for improving domain engineering processes should be reviewed thoroughly because the changes on domain engineering processes affect to the relevant application engineering processes.

The method should support assessing processes with the following capabilities:

— defining appraisal scope of domain and application engineering processes;

— determining appraisal forms for domain and application engineering processes;

— planning appraisal for domain and application engineering processes;

— conducting appraisal for domain and application engineering processes (providing appraisal templates for each of domain and application engineering process appraisal);

— defining action items to implement the domain and application engineering process improvements.

A tool should support assessing processes by allowing the user to do the following:

— allowing electronic data recording (e.g. spreadsheet) by domain and application engineering processes;

— supporting interfaces to open or to refer the domain artifacts and application artifacts that each process produces;

— providing real-time updates of artifact lists for the parallel assessment of domain and application engineering processes.

5.5.2 Estimate the impact of changes on processes

The goal of this task is to analyse the impacts due to changes on domain engineering processes. This analysis includes application engineering processes related to the process changes. Because changes on domain engineering processes have impacts on multiple processes of member applications impact analysis should be performed more considerately.

The method should support estimating the impact of changes on processes with the following capabilities:

— determining and prioritizing candidate process improvements alternatives;

— analyzing impacts due to the changes on domain engineering processes;

— comparing alternative domain engineering processes;

— identifying the process improvements that will be implemented;

— performing pilots by choosing applications (applications can be chosen in accordance with the pilot strategy).

A tool should support estimating the impact of changes on processes by allowing the user to do the following:

— maintaining bidirectional links between domain engineering processes and their relative application engineering processes;

— recoding impact analysis results for supporting decisions of domain engineering process improvements;

— storing rationales for decisions of domain engineering process improvement.

5.5.3 Plan process improvement

The goal of this task is to establish and maintain plans for product line process improvements based on the process assessment and monitoring results. Process improvement performs improvement tasks in accordance with plans and evaluates the results against plans.

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

— identifying strategies for process improvement, approaches, and actions to address the identified process improvements;

— documenting activities for process improvements;

— reviewing and negotiating plans with relevant stakeholders.

A tool should support planning process improvement by allowing the user to do the following:

— sharing process improvement plans among stakeholders;

— reporting the status of plan implementation;

— updating plans.

5.5.4 Implement process improvements

The goal of this task is to establish and implement action plans for improving domain and application engineering processes.

The method should support implementing process improvements with the following capabilities:

— establishing action plans for domain and application engineering process improvements;

— defining metrics for measuring the results of the domain and application engineering process improvement;

— tracking progress against action plans;

— deploying process improvement throughout the organization;

— monitoring implementation of process improvements.

A tool should support implementing process improvements by allowing the user to do the following:

— sharing process improvement action plans among stakeholders;

— integrating measurement data into the defined metrics;

— tracing the measurement results with the relevant improvement plans;

— documenting and reporting analysis results;

— supporting status reports for checking whether the process improvement are on track (in the case of commonly used domain engineering processes among applications, this capability is crucial);

— supporting version management for domain engineering processes with assets;

— sharing improvements of domain engineering processes that should be commonly used by member applications;

— allowing reference of measurements, their integration results, and work products related to process improvement to make informed decisions;

— reporting the status of action plan implementation;

— updating and sharing action plans,

5.5.5 Evaluate process improvement

The goal of this task is to confirm the effectiveness and efficiency of process improvement. Metrics for evaluating the effectiveness and efficiency of product line process improvement should be defined.

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

— defining metrics;

— measuring the results;

— analysing the measurement results;

— incorporating lessons learned into the product lines.

A tool should support evaluating process improvement by allowing the user to do the following:

— integrating measurement data into the defined metrics;

— tracing the measurement results with the relevant improvement plans;

— documenting and reporting analysis results.

6 Variability management

Product line variability visualizes how the member products in a product line differ with each other. The basis for managing variability is the explicit documentation of variabilities. The primary way of documenting variability is to use graphical models. This process supports modeling, exploiting, tracing, and evolving variability throughout the domain and application engineering. Variability management encompasses the following roles:

— structuring variability model;

— managing the evolution of the variability model including the domain variability model and application variability models;

— ensuring consistencies among variability models;

— managing the trace links among variability models and associated assets.

Variability management process should be conducted in parallel with domain engineering processes because variabilities are clarified and modified gradually together with the domain and application engineering. Variability management is started in the domain requirements elicitation process and the models evolve continuously throughout the product line lifecycle.

Variability management supports the following:

— *Variability modelling* supports to develop and validate variability models by using variability information derived from the domain engineering. This subprocess serves to collect variability information for identifying variability elements and bases information for constructing variability models. Validation of variability models is served for ensuring that the models consider variability dependencies and constraints derived during domain engineering.

— *Variability mechanism* is a mean for realizing the variability of product line. Because variability is introduced at the whole domain engineering lifecycle phases and binding occurs at the whole application engineering lifecycle phases, variability mechanisms that support the determined binding times, rules and constraints, and domain artifacts should be provided.

— *Variability binding* maintains information that is necessary to resolve variability appropriately. This subprocess manages variability binding results established during domain and application engineering.

— *Variability documentation* supports stakeholders to document variability models in detail with annotations. It also supports to *s*pecify variability models with their traceabilities for the right use and reviews of variability documentation.

— *Variability tracing* manages to *e*stablish and maintain trace links of variability models with domain and application assets according to the established policies for traceability management. By following the trace links between variability and product line assets, it is possible to know how a given variant is realized in the product line and check if different definitions of variability are consistent.

— *Variability control and evolution* serves to manage changes of variability models and their established traceabilities. The tasks for managing feedbacks about the feasibility of realizing variability fed from the preceding processes are also provided.

6.1 Variability modelling

Purpose of variability modelling

The purpose of this subprocess is to manage and support variability modeling for a product line. There exist two distinct approaches for variability modeling; ad-hoc (conventional) variability modeling that includes variability information in some software modeling notation (e.g. UML, feature, etc.) and orthogonal variability modeling that integrates all the variability information in a single model. The variability models alone cannot represent the full contents of variability information in both approaches for variability modeling. The development artifacts of domain/application engineering and the variability model need to be traced.

Inputs

— Variability related information from domain engineering.

— Validation criteria (e.g. aspects should be checked) for variability models.

Outcomes

— Variability modeling policy is established.

— Variability models are maintained with versions.

Tasks

— *Establish variability modeling policy.* Variability modeling approach and its notations are determined.

— *Collect variability information.* Variability information elicited and analyzed during domain and application engineering is referred to make variability models.

— *Verify variability models.* The variability model is reviewed with respect to variability dependency types and their cardinalities, rationale, and constraint dependencies.

— *Share and maintain variability models*. Domain variability model and application variability models are shared and evolved.

6.1.1 Establish variability modeling policy

Variability models should be structured to decrease the complexity of the product line engineering. Necessary information for the variability modeling policy should be identified and analysed. The traceability between the variability model and the development artifacts of domain/application engineering should be considered for the decision of the policy.

The method should support establishing variability modeling policy with the following capabilities:

— comparing alternative approaches of variability modeling;

— reviewing relationships with other variability models;

— reviewing relationships between domain/application assets and variability models;

— deciding on the variability representation and notations;

— evaluating and selecting variability modeling tools.

A tool should support establishing variability modeling policy with the following capabilities:

— supporting multi-criteria decision making;

— sharing the variability modeling policy with relevant participants.

A tool for variability modeling should support the following capabilities:

— providing modeling notations and a workspace for constructing variability models;

— storing relationships with other variability models and domain assets;

— allowing automatic construction of variability models based on the recorded variability information;

— storing configurations of variability models;

— supporting consistency checking;

— generating a unique ID for each variability model;

— providing format for exchanging variability models among different applications;

— maintaining variability dependencies;

— maintaining constraint dependencies.

6.1.2 Collect variability information

In this task, variability information identified and analyzed during domain engineering is delivered for later construction of variability models. This task identifies variability elements from the collected information (e.g. variation points and their variants, binding times for all variation points, dependencies, constraints dependencies) to develop variability models.

The method should support collecting variability information with the following capabilities:

— collecting variability information from domain assets;

— checking if the associated variability elements (e.g. variation points and their variants, binding times for all variation points) are included;

— validating consistencies among identified variability relevant information;

— maintaining links between variability information and its sources.

A tool should support the collection and manipulation of variability information with the following capabilities:

— accessing and browsing repository of domain assets to collect variability information;

— storing dependencies and constraints among variability elements for maintaining consistencies;

— providing a function for an automatic consistency check (e.g. by using arithmetic constraint check);

— providing a template for recording variability information;

— tracing variability relevant information with its sources;

— supporting manipulation of variability information within the variability modeling workspace;

— allowing synchronization with variability modeling workspace.

6.1.3 Verify variability models

Variability models are verified to ensure that each model is clear, complete, and correct.

The method should support verifying the variability models with the following capabilities:

— collating variability models with the identified variability information;

— verifying consistencies of variability dependencies and constraint dependencies;

— analysing conflicts and inconsistencies within the variability model and between the variability models and associated domain assets.

A tool should support verifying the variability models with the following capabilities:

— displaying variability models and associated variability information for verification;

— providing guidance for proceeding verification (including checklists);

— supporting verification reports.

6.1.4 Share and maintain variability models

Domain variability model and application variability models are shared by stakeholders and evolved based on the formal change requests.

The method should support sharing and maintaining the variability models with the following capabilities:

— maintaining variability information for effective and efficient sharing;

— managing versions of variability models;

— managing change request resolution process;

— analysing collected change requests.

A tool should support sharing and maintaining variability models with the following capabilities:

— collecting change requests;

— supporting change request resolution process;

— notifying changes of variability models;

— changing versions of variability model;

— disseminating variability information to relevant stakeholders.

6.2 Variability mechanism

Purpose of variability mechanism

The purpose of this subprocess is to provide required supports for the efficient and effective management and operation of variability mechanisms in accordance with domain of interest.

Inputs

— *Initial list of variability mechanisms.*

— *Draft variability model.*

— *Draft decision table.*

Outcomes

— *Variability mechanism management policy* is established and maintained.

— *Variability mechanisms pool (catalog)* is implemented and managed.

— *Guidance for variability mechanism* use is defined.

— *Draft variability model* is revised in variability mechanism specifically.

— *Draft decision* table is revised in variability mechanism specifically.

Tasks

— *Establish variability mechanism management policy* is to determine policies for managing and operating variability mechanisms.

— *Operate variability mechanisms* is to use variability mechanisms in accordance with the defined procedures, rules, and constraints.

— *Support variability mechanisms operation* is to provide proper supports for managing and operating variability mechanisms.

6.2.1 Establish variability mechanism management policy

The goal of this task is to define technical supports (e.g. methods, tools, materials, technical trainings, and so on) for realizing variability mechanism operation.

The method should support establishing variability mechanism management policy with the following capabilities:

— developing an understanding of variability mechanisms (e.g. complexity of mechanisms, necessary information types for the proper use of mechanisms) used for establishing policies;

— deciding level of details for variability mechanism management;

— establishing a variability mechanism management policy.

A tool should support establishing variability mechanism management policy with the following capabilities:

— allowing documentation of a variability mechanism management policy;

— sharing a variability mechanism management policy with relevant participants.

6.2.2 Operate variability mechanisms

The goal of this task is to provide required technical support to meet the requirements of variability mechanism operation considering applicable variability mechanisms, procedures to select and use, tools, and techniques.

The method should support operating variability mechanisms with the following capabilities:

— categorizing variability mechanisms by binding times and the types of domain assets;

— defining guidance (e.g. selection guide, procedures for use) for operating variability mechanisms;

— specify variability mechanisms.

A tool should support operating variability mechanisms with the following capabilities:

— storing and accessing variability mechanisms catalogue;

— allowing the access of guidance for using variability mechanisms;

— allowing the specification of variability mechanisms.

6.2.3 Support variability mechanisms operation

The goal of this task is to provide technical supports (e.g. methods, tools, materials, skilled people, and so on) for the proper operation of variability mechanisms.

The method should support variability mechanism operation with the following capabilities:

— establish relations from variability mechanisms to relevant variability models;

— defining decision variables related to variability mechanisms;

— refining the values of decision variables related to variability mechanisms used;

— revising a decision table in variability mechanisms specifically;

— supporting realizing configuration from variability mechanisms aspect.

A tool should support variability mechanism operation with the following capabilities:

— establishing and maintaining relations from variability mechanisms to relevant variability models;

— providing an editor for revising a decision table in order to add variability mechanism specific information;

— providing tool support for realizing configuration.

6.3 Variability documentation

Purpose of variability documentation

The purpose of this subprocess is to add detailed descriptions on the orthogonal variability models. This process enables stakeholders to document variability models in detail by annotating the elements of the variability models.

Inputs

— *Variability models.*

— *Information relevant to variabilities.*

Outcomes

— *Variability models* are documented.

Tasks

— *Establish policies for variability documentation* establishes and maintains policies for forward and backward traceability links between variability models and between variability models and associated domain assets for maintaining consistencies.

— *Collect annotations of variability models*. Each element (e.g. variation points, variants, and dependencies) of a variability model is annotated with appropriate information (e.g. owners or related domain assets that need to use the element).

— *Validate the variability documentation*. Variability documentation is reviewed for ensuring correctness and completeness.

6.3.1 Establish policies for variability documentation

The goal of this task is to establish a product line organization's policies for variability documentations. The purpose of policies for variability documentation is to define what should be done in variability documentation.

A method should support establishing policies for variability documentation with the following capabilities:

— gathering existing practices or rules regarding variabilities to be adhered in the product line organization;

— establish practices or rules regarding variability documentation;

— reviewing the results for ensuring correctness.

A tool should support establishing policies for variability documentation with the following capabilities:

— providing templates for documenting policies;

— notifying policies for variability documentation to stakeholders within a product line.

6.3.2 Collect annotations of variability models

In this task, annotations of variability models are collected with information that helps stakeholders understand, use, and manage the elements and associated domain assets.

The method should support collecting annotations of the variability models with the following capabilities:

— determining which information should be attached to each element of variability models (e.g. owner, related artifacts);

— reviewing annotations of the elements of variability models.

A tool should support collecting annotations of the variability models with the following capabilities:

— finding and referring to the information necessary to develop annotations;

— providing a workspace for relevant stakeholders to develop the annotations and attach them to respective elements of the variability model.

6.3.3 Validate the variability documentation

Variability documentation is reviewed to ensure that it is clear, complete, correct, and understandable.

The method should support reviewing the variability annotation with the following capabilities:

— validate that all the variabilities of the different domain assets related to the same variability model are linked;

— confirming the consistencies of variability annotation;

— checking the correctness of variability documentation;

— reviewing the completeness of variability documentation.

A tool should support reviewing the variability annotation with the following capabilities:

— browsing the variability annotation for reviewing;

— importing and browsing the relevant information from other tools if it is necessary to review them.

6.4 Variability binding

Purpose of variability binding

The purpose of variability binding management is to manage bound variability introduced during domain engineering in application engineering.

Binding time of variability is the time (or application development phase) that the value of a variation point is determined. Binding time can significantly affect the flexibility, runtime performance, and production costs of product line members. Because these decisions are influential, alternatives of binding time shall be reviewed carefully.

Inputs

— Platforms related to binding.

— Alternative binding times.

— Historical data related to binding times.

Outcomes

— *Binding times and their rationales* are documented.

— *Variability mechanisms* are defined.

Tasks

— *Establish binding policy* determines policies for binding time decision and trade-off analysis on binding time.

— *Guide trade-off analysis among alternatives of binding time* analyzes the trade-offs for each recommended binding time and compares the results. Variability is introduced at the particular time of domain engineering lifecycle process and it is bound at the particular time of application engineering lifecycle processes. Decision of optimal binding time is important for flexibility and development/maintenance cost of a product line.

— *Guide binding time decision* identifies the moment of deciding the values of variation point. Binding times can be compile time, link time, build time, or runtime.

— *Maintain binding information*. Binding information reported from application engineering lifecycle processes is maintained for the change management and evolution.

6.4.1 Establish binding policy

Binding polish should support optimal binding time decision and trade-off analysis on binding time.

The method should support establishing binding policy with the following capabilities:

— defining a common binding process for a product line;

— providing a validation guide for optimal binding decision;

— providing a guide for the documentation and reporting of binding decision.

A tool should support establishing binding policy with the following capabilities:

— sharing the binding policy with relevant stakeholders.

6.4.2 Guide trade-offs analysis among alternatives of binding time

The goal of this task is to guide for analyzing alternatives of binding time.

The method should support guiding trade-offs analysis among alternatives of binding time with the following capabilities:

— identify possible process/sub-process for binding for each variability;

— defining aspects for determining binding time (e.g. quality attributes or product line goals such as flexibility, maintenance cost, and so on);

— analysing trade-offs between different alternatives of binding times based on defined aspects.

A tool should support guiding trade-offs analysis among alternatives of binding time with the following capabilities:

— displaying alternatives for each of binding times;

— providing environment for integrating trade-off analysis results of each of applications.

6.4.3 Guide binding time decision

The goal of this task is to guide determination for the moment of variability resolution. Because if binding time is too early, flexibility of the product line gets lower, and if it is too late, the maintenance costs get higher, so the binding times should be determined carefully.

The method should support guiding binding time decision with the following capabilities:

— determining binding times of variation points;

— documenting decisions and their rationales;

— selecting variability mechanism.

A tool should support guiding binding time decision with the following capabilities:

— displaying variation point structures according to the binding of variation points;

— describing the variation point structure at the specific development phase;

— making variability selection mechanism explicitly.

6.4.4 Maintain binding information

The goal of this task is to collect and maintain binding information reported from application engineering lifecycle processes.

The method should support maintaining binding information with the following capabilities:

— recording the rationales for the binding decision;

— maintaining binding information for effective and efficient sharing.

A tool should support maintaining binding information with the following capabilities:

— collecting binding information from application engineering;

— providing verification scheme for binding decision;

— allowing binding decision to be recorded;

— sharing binding information with relevant stakeholders.

6.5 Variability tracing

Purpose of variability tracing

The purpose of variability tracing is to establish and maintain forward and backward traceability links between variability models and associated domain assets for maintaining consistencies.

Inputs

— *Domain and application artifacts related to variability.*

— *Information related to variability models and associated domain artifacts.*

NOTE Domain or application artifacts related to variability means variability models, variability documentation, domain models, or application models that incorporate variabilities or variability relevant information.

Outcomes

— *Traceability management policies* are established.

— *Trace links among domain or application assets related to variability are established.*

— *Trace links between elements of the variability model, and the associated domain and application assets are established.*

Tasks

— *Establish policies for traceability management*. An economic analysis should be conducted to align the level of detail of traceability management with respective costs and benefits. As the result, economically viable traceability management policies for planning and performing traceability management are established and maintained.

— *Define links between variability model and domain assets.* Variability model and its relevant domain assets should be traced with each other for being referred by every product line lifecycle processes for collecting relevant domain assets when a variability selects a variant.

— *Manage the changes of the defined trace links.* Changes on the defined trace links are managed for maintaining consistencies between them in accordance with the evolution of a product line.

6.5.1 Establish policies for traceability management of variability models

The goal of this task is to establish and maintain organizational norms, expectations, and policies of traceability management for *domain or application assets related to variability*.

The method should support establishing policies for traceability management of variability models with the following capabilities:

— analyzing the complexity of variability models;

— developing a holistic understanding of the feasible levels of traceability management and suggesting a level;

— evaluating the suggested level of detail of traceability management;

— deciding on the level of detail of traceability management;

— establishing traceability management policies.

A tool should support relevant stakeholders in establishing policies for traceability management of variability models with the following capabilities:

— presenting evaluation criteria for evaluating levels of traceability management;

- providing a multi-user access environment for reviewing, commenting, and communicating about existing and envisioned traceability management policies;
- supporting publish of the traceability management policies for sharing among stakeholders.

6.5.2 Define links between variability model and domain assets

The goal of this task is to establish explicit trace links between elements of domain or application assets related to variability.

A method should support defining links between variability model and domain assets with the following capabilities:

- identifying the variabilities of the domain assets related to the variability model;
- identifying a variant of the variability model that will be linked with the variability of the domain assets;
- defining links between a variant of the variability model and the associated domain assets;
- reviewing the results for ensuring correctness.

A tool should support defining links between elements of the variability model and the associated domain assets with the following capabilities:

- importing and browsing domain assets in order to establish links with the variability model;
- storing links between elements of the variability model and the associated domain assets.

6.5.3 Manage the changes of the defined trace links

The goal of this task is to establish and ensure the integrity of trace links of domain or application assets related to variability in accordance with the product line evolution.

The method should support maintaining the changes of the defined trace links with the following capabilities:

- tracking and analyzing trace links of variabilities influenced by changes of domain or applications;
- restructuring the trace links of variabilities;
- checking for consistencies of the restructured trace links.

A tool should support maintaining the changes of the defined trace links with the following capabilities:

- tracing impacts due to changes;
- providing a roll-back function for restore trace links to their former state.

6.6 Variability control and evolution

Purpose of variability control and evolution

The purpose of variability control and evolution is to manage change requests/feedbacks for variability models and improving variability evolution process and maintain traceability among variability models and between variability models and domain assets.

Inputs

- *Change requests for variability models.*
- *Feedbacks from domain and application engineering for changes.*

Outcomes

— *Change history of variability models* is maintained.

— *Traceability maps* are changed.

— *Feedback statuses* are notified.

Tasks

— *Identify and analyze the evolution needs of variants*. Decisions for change or evolution needs are made after receiving those needs from domain or application engineers.

— *Add or remove variants*. A new variant can be added or an existing variant can be removed. Change requests to variability models are traced and their impacts are analyzed. Approved changes are communicated to all affected participants.

— *Add or remove dependencies and constraints*. Change needs of dependencies and constraints are accepted and reflected or affected dependencies and constraints in accordance with the changes of variants are reflected.

— *Change binding time* supports changes of binding time in accordance with the evolution of a product line.

— *Maintain the affected traceabilities*. Defined traceabilities are maintained consistently even after the changes have taken place.

— *Provide feedback related to variability models* (from application and domain engineering). Domain and application engineers send feedback to improve variability models and the variability evolution process. This task manages the statuses of feedback documents and the changes taken based on processing the feedback.

6.6.1 Identify and analyse the evolution needs of variants

The goal of this task is to identify, select, and analyse the evolution or change requests for variabilities and then decide which parts of variability models can be changed.

The method should support identifying and analysing the evolution needs of variants with the following capabilities:

— analysing change requests of variants from variability evolution needs;

— analysing impacts of a variant change (using a variability mechanism);

— establishing the necessary change list.

A tool should support identifying and analysing the evolution needs of variants with the following capabilities:

— displaying relations among variabilities explicitly;

— showing affected elements of variability models due to changes.

6.6.2 Add or remove variants

The goal of this task is to add or remove variants at any given variation point in time. Change management for variability models is conducted during this task.

The method should support adding or removing variants with the following capabilities:

— introducing requested changes of variants;

— controlling versions of the affected assets (e.g. the changed variability models, variability documentations, or domain/application artifacts that include variabilities);

— controlling changes of variants in a manner that does not affect the existing product lines.

A tool should support adding or removing variants with the following capabilities:

— tracing the statuses of change requests;

— storing the change history of variability models;

— storing versions of the variability models and variability documentation;

— maintaining and tracking versions of variability models;

— notifying the statuses of change requests to the requested sources.

6.6.3 Add or remove dependencies and constraints

The goal of this task is to add or remove dependencies and constraints affected by changes of variants or according to the changed needs for them.

The method should support adding or removing dependencies and constraints with the following capabilities:

— introducing the changes of affected dependencies and constraints due to changes of variants;

— analysing impacts of a variant change (using a variability mechanism);

— making the necessary change list.

A tool should support adding or removing dependencies and constraints with the following capabilities:

— displaying relations among variabilities explicitly;

— showing affected elements of variability models by changes.

6.6.4 Change binding time

When a product line evolves, the binding time of a variation point can be changed. The goal of this task is to support changes of binding time for increasing the flexibility of the resulting system as the product line evolves.

The method should support changing binding time with the following capabilities:

— assessing impacts of binding time change on other variabilities;

— analysing effect of binding time change;

— adjusting binding times of variabilities affected by change of binding time;

— reorganizing variability mechanism.

A tool should support changing binding times with the following capabilities:

— tracking changes of variabilities according to the changes of binding time;

— tracing the changes of binding time;

— supporting reorganization of variability mechanism.

6.6.5 Maintain the affected traceabilities

Trace links among variability models and between a variability model and the associated domain assets are managed in this task.

The method should support maintaining the affected traceabilities with the following capabilities:

— finding, analysing, and updating all trace links affected by variability changes;

— checking that the updated links are consistent;

— maintaining old trace links to enable roll-backing when necessary.

A tool should support maintaining the affected traceabilities with the following capabilities:

— representing those traceability links affected by variability changes (traceability tracking);

— providing a roll-back function for canceling the traceability changes.

6.6.6 Provide feedback for variabilities and the variability evolution process

Feedbacks concerning the variabilitiess and the variability evolution process are obtained from domain and application engineering and are managed during this task.

The method should support providing feedback concerning the variabilitiess and the variability evolution process with the following capabilities:

— analysing the feedback to determine to which parts are related to variabilities or the variability evolution process;

— tracing the feedback to closure;

— managing feedback history with related changes.

A tool should support providing feedback concerning the variabilities and the variability evolution process with the following capabilities:

— collecting and notifying relevant stakeholders about feedback;

— providing templates for feedbacks and feed-forwards;

— storing and displaying the feedback statuses.

7 Asset Management

Asset management is a process that establishes an asset base and uses it to store and manage variability models and domain/application assets. The asset base will also be used in the subsequent phases of domain engineering to store and manage all domain assets, because domain engineers should design domain assets to be reusable and fulfill the predefined variability needs, and in application engineering to find and reuse appropriate domain assets. All member applications developed in a product line have separate application asset repositories for storing and managing application specific assets produced during application engineering and during the later stages of the application engineering life-cycle. Application specific assets are originally developed for being used by particular applications, not for reuse. However, some application specific assets can be generalized as domain assets later in accordance with the decision at Organization Management. This is especially true in the initial stages of product line development when product line organizations have not yet been able to populate extensively their asset repositories. Application asset repositories are thus crucial resources for establishing and maintaining the asset repository. It is important that application assets with high reuse potential are identified and their trace links are managed in order to maintain the consistencies of the product line throughout all assets. Asset management not only establishes and maintains repositories for domain/application assets but also includes subprocesses that shall be conducted together with the other processes defined in this International Standard as necessary.

Asset management has the following six basic subprocesses:

— *Asset identification* identifies and evaluates asset candidates (e.g. features, models, specifications, and test cases) developed in domain/application engineering. All reusable domain artifacts developed during domain engineering are identified assets and stored in the repository. The purpose of this process is to identify reusable domain artifacts and extract assets and relevant information from the existing application asset repositories and other sources, evaluate them, and arrange necessary information to help application engineers in their reuse.

— *Asset base implementation* configures an internal and external structure of assets that makes it easy to mine, retrieve, and manage assets. Each asset should have a well-organized and structured configuration.

— *Asset validation* ensures that asset base reflects the defined asset structure and is easy to be changed and controlled and that the asset structure and annotations facilitate effective reuse.

— *Asset evolution* manages and controls change requests and feedback, traceability, and versioning of assets after baselining.

7.1 Asset identification

Purpose of asset identification

The purpose of this subprocess is to identify domain/application artifacts that should be captured as assets.

Inputs

— *All artifacts produced during domain/application engineering.*

— *Information related to asset candidates (e.g. type, attached information for enabling correct reuse, owner).*

Outcomes

— *Organizational policies for assets* are established.

— *Asset list* is established.

Tasks

— *Set up and maintain organizational policies for managing assets*. Organizational policies for evaluating, validating, mining, and managing assets are established and maintained.

— *Identity asset candidates*. Asset candidates that are planned for systematic reuse and are included in the domain/one or more application artifact repositories are identified. Information (e.g. artifact type, owners, and relations between artifacts) about asset candidates necessary for evaluating them is identified as well.

— *Estimate efforts necessary to create, reuse, and update domain assets*. Economic values for creating, reusing, and updating domain assets are estimated.

— *Determine assets*. Assets are selected after reviewing the effectiveness and efficiency of candidates from economical and other relevant viewpoints.

— *Elicit information necessary to reuse assets*. Information necessary to reuse an asset correctly, such as the usage process, glues (i.e. interfaces used to integrate assets with other assets or facilitate reuse of the assets), guidelines, and the owner/user/caretaker is created. Application engineering seldom has the need and resources to create such information because application artifacts are not meant to be reusable.

7.1.1 Set up and maintain organizational policies for managing assets

This task establishes organizational norms, expectations, and policies for the reuse and management of assets.

The method should support setting up and maintaining organizational policies for managing assets with the following capabilities:

— establishing organizational goals for assets;

— establishing an organizational policy for planning and developing assets for reuse by the product line members;

— establishing an organizational policy for making assets available for the product line members;

— establishing an organizational policy for asset evolution.

A tool should support setting up and maintaining organizational policies for managing assets with the following capabilities:

— providing templates for helping stakeholders to establish and maintain the policies;

— providing a workspace where stakeholders can share, evaluate, negotiate, and revise organizational policies for managing assets;

— distributing the agreed upon organizational policies to relevant stakeholders.

7.1.2 Identity asset candidates

This task serves to find and retrieve asset candidates from the domain/application asset base.

The method should support identifying asset candidates with the following capabilities:

— selecting asset candidates from the domain/application artifact repositories (a planned artifacts list for application engineering);

— collecting information for each asset candidate (e.g. artifact type, owners, users, relations between artifacts);

— assigning unique identifiers to the candidates.

A tool should support identifying asset candidates with the following capabilities:

— finding, selecting, and Importing asset candidates from the application artifact repositories into the asset management workspace;

— storing information about asset candidates;

— generating unique identifiers to distinguish the candidates.

7.1.3 Estimate efforts necessary to create, reuse, and update domain assets

The goal of this task is to calculate efforts that will be invested to create, reuse, and update domain assets.

Creating reusable domain artifacts is much more costly than creating artifacts that have no reusability and variability. Even if existing application artifacts show high reusability potential, transforming them into domain artifacts typically requires substantial rework and extensive documentation. Efforts and costs thus need to be estimated in order to decide whether asset candidates should be transformed into and managed as assets.

The method should support effort estimation to determine the economic feasibility of creating, reusing, and updating domain assets with the following capabilities:

— providing effort estimation functions (costs, benefits, risks, etc.) and estimation databases containing productivity information about similar past transformation projects;

— collecting information for effort estimating and storing it into estimation databases;

— estimating the efforts required to transform asset candidates into assets;

— integrating the estimates with the analysis results obtained in identifying the asset candidates.

A tool should support estimating the efforts necessary to create, reuse, and update domain assets with the following capabilities:

— maintaining historical productivity information about software projects for effort estimation;

— offering a GUI for inputting values necessary to estimate the efforts;

— generating effort estimates according to the predefined estimation functions and productivity information from similar past projects;

— representing integrated results for helping further decisions.

7.1.4 Determine assets

Asset candidates are evaluated according to the evaluation criteria and explicitly justified decisions are taken to determine the assets.

The method should support determining assets with the following capabilities:

— offering evaluation criteria for determining assets;

— evaluating the asset candidates;

— taking decision and recording its rationale;

— implementing the chosen candidates into assets in domain/application engineering;

— reviewing the decision.

A tool should support determining the assets with the following capabilities:

— providing a workspace for reviewing, discussing, and commenting the asset candidates;

— displaying criteria and guidelines for evaluating the candidates;

— transforming (in domain engineering) the determined artifact into domain assets that fulfill the reusability and variability for domain artifacts;

— storing the domain assets into the asset base;

— storing rationales for decisions.

7.1.5 Elicit information necessary to reuse assets

The goal of this task is to develop information necessary for successful reuse of domain assets and to attach information to the relevant assets. This information is usually developed by the relevant domain/application engineers.

The method should support eliciting information necessary to correctly and effectively reuse assets with the following capabilities:

— reviewing relevant asset information (collected and created during the identification of asset candidates) for each asset;

— arranging the information and attaching it for each asset;

— validating the results.

A tool should support eliciting information necessary to correctly and effectively reuse assets with the following capabilities:

— providing an environment to review, communicate, and comment;

— showing criteria and guidelines for reviewing and validating assets;

— maintaining rationales for decisions;

— automatically generating and maintaining system relevant information (e.g. when a asset was created and by whom);

— entering and storing creator-defined information.

7.2 Asset base implementation

Purpose of asset base implementation

Asset base implementation provides processes and tools/methods capabilities to implement the repository and to mine and retrieve assets from it. Its structure and services should reflect the pre-defined asset configurations.

Inputs

— Domain artifacts.

— Application artifacts.

Outcomes

— *Asset mining system* is established.

— *Asset base* is established.

Tasks

— *Establish the mining (or retrieval) mechanism for assets*. A mechanism supporting the mining of highly relevant assets (especially from the viewpoint of application engineering) is established.

— *Define and implement the CRUD method for assets*. The procedures for creating, retrieving, updating, and deleting assets are defined.

— *Establish asset base*. A domain/application asset base is built and evolved. It includes the storage and services for mining and accessing the repository and leverages the mining mechanism and CRUD.

— *Evaluate the asset base* to confirm that it has been correctly established for storing, accessing, mining, and maintaining assets.

7.2.1 Establish the mining (retrieval) mechanism for assets

The mining mechanism for assets has a close relation with the asset configuration system.

The method should support establishing and maintaining of the mining (retrieval) service for assets with the following capabilities:

— determining the classification for assets;

— choosing the attributes for retrieving assets. The attributes are chosen from the internal or external attributes of assets;

— determining how to compose the chosen attributes for mining.

A tool should support establishing and maintaining of the mining (retrieval) service for assets with the following capabilities:

— enabling access to the pre-defined asset configuration;

— simulating the defined mining mechanism;

7.2.2 Define and implement the CRUD method for assets

The goal of this task is to define the CRUD method for creating, retrieving, updating, and deleting assets. CRUD is related with the asset configuration system and the defined mining mechanism.

The method should support defining and implementing the CRUD method with the following capabilities:

— determining a CRUD strategy for assets;

— defining the procedures for creating, reusing, updating, and deleting assets;

— implementing the CRUD method.

A tool should support defining and implementing the CRUD method with the following capabilities:

— providing interfaces for creating, retrieving, updating, and deleting assets;

— creating and saving assets and the baselines of sets of assets into the repository;

— searching and retrieving assets and baselines with the help of the external attributes of the assets;

— updating assets and baselines under the control of change management and configuration management;

— deleting assets.

7.2.3 Establish asset base

The goal of this task is to establish and maintain asset base by leveraging the defined and implemented mining mechanism and CRUD procedures.

The method should support establishing asset base with the following capabilities:

— designing the structure of asset base to enable the mining mechanism;

— implementing the defined mining mechanism;

— implementing the defined CRUD method;

— specifying appropriate access rights for assets with different statuses (e.g. login, check in, check out for read, check out for modification, change request, change approved/rejected, change implemented, change released, change audited);

— specifying appropriate access rights for different granularities of assets (e.g. line, code, function, module);

— maintaining the log for all accesses of the configuration items by access right and granularity.

A tool should support establishing asset base with the following capabilities:

— storing assets according to the defined asset structure;

— recovering archived versions of assets;

— controlling access rights to asset base by user, group, role, or the type of transaction (e.g. representing, specifying, searching, and changing);

— enabling the asset base access by means of the mining service and CRUD and in accordance with the access rights;

— generating the log files for all accesses;

— providing appropriate encryption methods to ensure secure usage and transmission of data.

7.2.4 Evaluate asset base

Asset base is reviewed and validated to confirm that it appropriately uses the asset configuration, mining mechanism, and CRUD processes.

The method should support evaluating the asset base with the following capabilities:

— reviewing the structure of the asset base;

— evaluating whether the asset base reflects the configuration, mining mechanism, and CRUD;

— taking corrective and improvement actions as necessary;

— checking the integrity of the established asset base.

A tool should support evaluating the asset base with the following capabilities:

— accessing information related to asset base and its structure;

— checking consistencies automatically.

7.3 Asset validation

Purpose of asset validation

Asset validation aims at ensuring that asset base reflects the defined asset configuration and is easy to mine, change, and control.

Inputs

— *Assets*

— *Asset configuration*

Outcomes

— *Validation results for the asset configuration are produced.*

— *Baselines of assets are established.*

Tasks

— *Review the selected assets* confirms that proper assets have been selected, created, and specified.

— *Review the asset configurations* ensures that they reflect the defined assets attributes, relations, and constraints.

— *Create and release baselines of assets* establishes sets of assets designated for systematic reuse.

— *Storing assets into asset base* serves to store assets into the relevant workspace of the repository and to mine and retrieve assets from it

7.3.1 Review the selected assets

Baselined assets are reviewed and validated to ensure that each asset and baseline is clearly defined, complete, correct, and understandable.

The method should support reviewing the selected assets with the following capabilities:

— identifying the evaluation criteria for the assets;

— evaluating whether the selected assets are proper according to the criteria.

A tool should support reviewing the selected assets with the following capabilities:

— accessing assets with related information;

— providing guidance for proceeding with the review process;

— providing templates and checklists for reviewing each asset.

7.3.2 Review asset configurations

Asset configurations are reviewed and validated to ensure they are clear, complete, correct, and understandable.

The method should support reviewing the asset configuration with the following capabilities:

— identifying the evaluation criteria for asset configuration;

— reviewing whether all the necessary internal and external attributes of the assets have been defined;

— confirming the baselines for the assets;

— revising the baselines for the assets as necessary;

— committing the baselines for assets.

A tool should support reviewing asset configuration with the following capabilities:

— guiding the review process;

— enabling access to asset base for reviewing the baselines, assets, and their structures.

7.3.3 Create and release baselines of assets

The goal of this task is to create and release baselines of assets and document the baselined sets of assets in asset base. Asset baselines are agreed-upon sets of assets designated for systematic reuse during application engineering and further evolution during domain engineering. Baselining includes the building, storing, tracing, access controlling, and reporting of baselines.

The method should support creating baselines of assets with the following capabilities:

— creating (initiating) baselines from assets;

— documenting the set of assets contained in a baseline;

— storing baselines of assets;

— establishing traceability.

A tool should support creating baselines of assets with the following capabilities:

— providing a specification template for specifying the baselines in accordance with baseline attributes (e.g. creation date, owner, acceptance criteria, and change history);

— supporting communication related to the planning of baselines and the authorizations of baselines;

— storing the baseline specifications into asset base;

— placing associated tool information for versioning, building, and releasing.

7.4 Asset evolution (including change management)

Purpose of the asset evolution

The purpose of asset evolution is to manage and control change requests and feedback, traceability, and versioning for assets after baselining.

Inputs

— *Asset base.*

— *Change requests for assets.*

Outcomes

— *Change history of assets is maintained.*

— *Differences between baselines are maintained.*

— *Traceability between assets is established.*

— *Statuses for assets are fed back.*

Tasks

— *Manage asset changes*. Change requests to the assets and their impacts are analyzed and traced. The approved changes are communicated to all affected participants.

— *Maintain traceability of assets*. Bidirectional traceability between assets and between attributes within assets is maintained.

— *Manage feedback*. Feedback for assets from domain and application engineering is managed and action items are traced to closure.

— *Transform the existing assets into assets to rehabilitate asset base*. Assets with high reusability potential are identified from existing systems that do not belong to the product line, transformed into assets in domain engineering, and used to rehabilitate the repository.

— *Dispose assets from asset base.* Unwanted or invalid assets within asset base are removed.

7.4.1 Manage asset changes

This task serves to track and control changes to assets that are contained in a baseline.

The method should support managing asset changes with the following capabilities:

— analysing change requests for assets;

— analysing impacts of changing assets;

— controlling changes in the assets;

— documenting the differences between baselines due to changes;

— controlling the versions of assets and baselines;

— controlling the releases of baselines.

A tool should support managing asset changes with the following capabilities:

— tracing the statuses of change requests;

— finding and presenting items that are affected by those changes;

— storing the change history for each asset;

— providing a document template for describing the differences between baselines;

— tracking versions of assets;

— tracking the releases of baselines;

— locking and unlocking the versions from the version branches with respect to check-outs and check-ins;

— displaying the versions and version branches in a tree or graph form.

7.4.2 Maintain traceability of assets

The goal of this task is to maintain and ensure the integrity of established traceabilities among assets.

The method should support maintaining the traceability of assets with the following capabilities:

— tracking and analyzing traceability links influenced by asset changes;

— restructuring the traceability links;

— checking for consistencies.

A tool should support maintaining the traceability of assets with the following capabilities:

— tracing impacts due to asset changes (Traceability tracking);

— providing a roll-back function for restoring traceability links related to assets.

7.4.3 Manage feedback and take appropriate evolution actions

This task manages feedback related to assets and to the asset evolution process.

The method should support managing feedback and taking appropriate evolution actions with the following capabilities:

— analysing feedback to determine unforeseen relations between the assets and to identify improvement opportunities related to assets and the asset evolution process;

— analysing impacts expected to materialize if the feedback is accepted;

— determining whether or not to accept the feedback;

— taking appropriate actions and tracing the feedback to closure;

— managing the history of feedback with related changes;

— notifying the relevant stakeholders about the feedback statuses.

A tool should support managing feedback and taking appropriate evolution actions with the following capabilities:

- collecting feedback and alerting relevant stakeholders about it;
- storing the feedback statuses and notifying stakeholders about the actions taken;
- enabling access to other tools or asset base to analyse and process the feedback.

7.4.4 Transform the existing assets into assets to rehabilitate asset base

Valuable assets of existing systems that are available to the product line organization but do not belong to the product line are identified, transformed into assets in domain engineering, and used to rehabilitate asset base (typically in the early stages of product line adoption when the product line organization does not yet have adequate assets in asset base). This task is very similar to the process described in 5.5.1 where suitable application artifacts within a product line are generalised into assets. It is thus not explained in detail in this International Standard.

The method should support transforming existing assets into assets to rehabilitate asset base with the following capabilities:

- identifying the existing assets to be transformed into and managed as assets;
- evaluating the existing assets according to defined evaluation criteria;
- estimating efforts necessary to transform such assets into assets.

A tool should support transforming the existing assets into assets to rehabilitate asset base with the following capabilities:

- interfacing the existing asset base;
- providing estimation templates;
- allowing asset configuration and annotation of assets.

7.4.5 Dispose assets from asset base

The goal of this task is to dispose unwanted or invalid assets from the asset base.

The method should support disposing assets from repository with the following capabilities:

- analysing the relations of the asset that will be disposed within asset base;
- analysing the products related to the asset;
- removing the asset from repository without affecting the remaining asset configurations.

A tool should support disposing assets from repository with the following capabilities:

- indicating affected asset;
- indicating relevant products;
- sharing the statements of the list of assets and why the assets are no longer supported;
- tracking the retirement schedule.

8 Support management

Support management covers the processes that provide necessary support for performing domain/application engineering, asset management, and variability management. The support management addresses processes that are used to perform other processes. This process deals with processes that are specialized for product lines consistent with the support management of ISO/IEC 12207. This process consists of quality assurance, configuration management, verification and validation, and tool support for automation and/or reducing complexity level of control. The method for carrying out and adherence to processes shall be defined and specified at the process discipline.

Support management has three basic subprocesses:

— *Technical quality management* serves to implement quality management schemes and to provide processes for ensuring the qualities of product line artifacts and processes for domain engineering, application engineering, organizational management, and technical management.

— *Configuration management* serves to manage configurations to cope with the complexity of the number of member applications consisting of many parts in different versions. Configuration items such as domain artifacts, assets, platforms, application artifacts, and trace links among configuration items are managed.

— *Decision management* serves to determine the most beneficial option among business or technical decision alternatives. This subprocess responds to a request for a decision encountered during product line lifecycle.

— *Technical risk management* serves to resolve technical risk issues that could jeopardize the accomplishment of business goals and objectives. Critical technical risks should be (e.g. lack of domain knowledge, uncertain or volatile domain requirements, lack of prevented or mitigated by the tasks of this subprocesses.

— *Tool management* serves to automate or semi-automate the product line engineering and management processes. Thus, it serves to improve productivity/quality and to reduce cost/time-to-market for a product line organization.

8.1 Technical quality management

Purpose of technical quality management

The purpose of this subprocess is to provide assurance that the development processes and artifacts of a product line are consistent with the predefined quality requirements of an organization.

Product line approach is based on the systematic and large scale reuse of assets, and hence the quality of assets have direct impact on whole products within a product line. Therefore, quality management is critical and complex due to the special properties of commonality and variability of product lines.

Inputs

— Goals and objectives of technical quality management.

— Defined domain and application engineering processes.

— Development artifacts of product lines (assets, products, application artifacts).

Outcomes

— *Quality assurance* results are documented.

Tasks

— *Establish technical quality management policy*. Technical quality management should establish necessary policy for quality assurance to ensure that work products and processes produced or implemented in a product line comply with predefined provisions and plans.

— *Establish and maintain criteria for quality assurance*. In addition to the work products of single system development, qualities of assets, variability models, and domain artifacts should be ensured through quality assurance in product line development. Criteria for evaluating quality have to be established.

— *Perform quality assurance according to criteria* assures that work products derived from a product line context and implemented processes have fulfilled the constraints and are consistent. Furthermore, whether the derived applications produced by reusing relevant assets are ensured.

— *Communicate and ensure resolution of noncompliance issues* shares the results of quality assurance and to resolve the raised noncompliance issues.

8.1.1 Establish technical quality management policy

Technical quality management policy needs to guide on quality assurance process, product assurance, process assurance, and conformance criteria for assurance. It also needs to coordinate with other processes such as verification and validation. Defined functionalities, quality attributes, artifacts, processes, and plans should be assessed based on technical quality assurance criteria.

The method should support establishing quality management policy with the following capabilities:

— defining technical quality assurance process;

— identifying target areas of technical quality assurance;

— providing a guidance for assurance and conformance criteria;

— providing a guidance for coordination with verification and validation.

A tool should support establishing quality management policy with the following capabilities:

— sharing the technical quality management policy with relevant participants.

8.1.2 Establish and maintain criteria for quality assurance

The goal of this task is to establish and maintain criteria for the evaluations of processes, domain and application assets, and variability models. Criteria for evaluating domain engineering processes adhered by all member applications of a product line, domain assets, adherence of architectural textures, variability models and bindings, application processes, and derived application assets.

The method should support establishing and maintaining criteria for quality assurance with the following capabilities:

— selecting processes, domain/application artifacts, or applications to be objectively evaluated;

— defining evaluation criteria for the two separation of engineering processes, domain and application engineering.

A tool should support establishing and maintaining criteria for quality assurance with the following capabilities:

— sharing domain artifacts;

— allowing access of quality criteria that are defined for domain/application engineering;

— allowing modification and reuse of the defined quality criteria between domain and application engineering.

8.1.3 Perform quality assurance according to criteria

The goal of this task is to perform objective assessments on product line processes and artifacts by using pre-defined criteria. Especially this task assures, that assets have fully satisfied the common requirements and the defined variabilities have guaranteed the flexibilities of applications within a product line.

This task also evaluates that application derived from a product line context has fulfilled requirements for each application and that each application adheres to reuse constraints. Furthermore, whether the derived applications have produced by reusing relevant assets is ensured.

The method should support performing quality assurance according to criteria with the following capabilities:

— evaluating performed domain/application engineering processes;

— evaluating domain/application artifacts;

— evaluating assets reused;

— identifying and documenting noncompliance issues (providing documentation templates).

A tool should support performing quality assurance according to criteria with the following capabilities:

— accessing reuse history of assets and information relevant to reuse of assets;

— sharing the variability model for ensuring its quality or confirming whether a right binding is performed;

— referring binding results of domain artifacts that entail variabilities;

— supporting documentation for noncompliance issues from each of domain, application, and process perspectives.

8.1.4 Communicate and ensure resolution of noncompliance issues

The goal of this task is to confirm whether quality issues are resolved.

In product lines, many cases of noncompliance issues are analysed from multi-dimensional aspects like domain engineering, application engineering, assets, and management because they are closely related, so the resolutions are required from a couple of aspects.

The method should support communicating and ensuring resolution of noncompliance issues with the following capabilities:

— resolving each noncompliance issues with the relevant stakeholders;

— tracking noncompliance issues unresolved until they are resolved;

— recoding QA reports (providing report templates).

A tool should support communicating and ensuring resolution of noncompliance issues with the following capabilities:

— providing collaboration environments among domain engineers, application engineers, management, and QAs;

— tracking the status of noncompliance issues.

8.2 Configuration management

Purpose of configuration management

The purpose of this subprocess is to maintain the integrity of domain and application assets.

Configuration management in product lines covers variations among member products of a product line, namely variation in space. In addition, configurations associated with product line evolution, namely, variability in time, should be managed. Therefore, product line development should manage configurations in both time and space. There might be complex relationships among configurations of variability in both time and space including commonality, so it makes difficult to manage configurations for product lines.

The difficulties for configuration management can be coped with the ability to manage the complexity of the product line. Tools and methods for product line configuration management shall support development/operating environment for various versions of common and variable product line artifacts plus various versions of product-specific artifacts. Moreover, development/operation of product line is conducted in parallel in a distributed environment, so product line configuration management shall define and set up appropriate policies and strategies to cope with it.

The tools, methods, processes, and environments for product line configuration management shall support for:

— variations in space as well as in time;

— domain engineering and application engineering by different team members;

— the distributed nature of product line engineering;

— build and release management of assets and product-specific artifacts;

— disciplined change management.

Inputs

— Configuration management policies.

— Artifacts produced during product line engineering.

Outcomes

— *A product line configuration management plan* is developed.

— *Configuration for a member product* is identified.

— *Configuration baselines* are established.

— *Product line configuration tree* is established.

— *Product line configuration tree is controlled.*

— *The status accounting is available.*

— *Changes are formally controlled.*

***Ta*sks**

— *Identify configurations of member products.* Configurations can vary in accordance with the member products to be developed under the same product line environment. The differences between configurations are identified, and they are used to establish configuration trees.

— *Establish configuration tree for a product line.* Configuration trees reflecting the differences among member products are established and maintained.

— *Manage configurations of variability in space.* The integrity of configurations of variability in space can be controlled through the variability model management and the formal change management.

NOTE Planning activities for configuration management are not covered in this International Standard because this International Standard only deals with SSPL-specific processes and tasks.

8.2.1 Identify configurations of member products

The goal of this task is to identify configuration of each member product including configuration items, components, and related work products in a space dimension that will be managed as configurations and establish their baselines. Each configuration includes binding results for a member product and an application's specific items.

The separation of common/variable and time/space concerns makes easy to deal with the complexity of configuration management in the product line.

The method should identify configurations of member products with the following capabilities:

— creating or releasing baselines of assets and platforms (they are produced by one team and used in parallel by several others);

— identifying configurations of member product in different shapes at the same time (e.g. using application variability model);

— informing the differences among the configurations of member products within a product line;

— managing and tracing the differences among the configurations of member products.

A tool should support identifying configurations of member products with the following capabilities:

— importing candidate configuration items into the configuration management environment;

— allowing the separations of baselines into time and space dimensions;

— managing the baseline of variability models including application variability models for member products;

— dealing with configuration items for hardware, software, and documents of all varieties.

8.2.2 Establish configuration tree for a product line

The goal of this task is to establish configuration tree for a product line. Differences among configurations of each member product are reflected into the configuration tree.

The configuration tree in single-system development provides a mechanism for tracing and controlling versions of an artifact. Whereas, in a product line, the configuration tree is established for multiple products. Application variability model, including binding results and application specific artifacts provides information about how member products in a product line vary in space. Thus, the configuration tree for a product line can be constructed based on the differences among application variability models. However, each member product can vary in time, so configurations of each member product should be established. In addition, configuration tree should be established in a form that can provide a mechanism for tracing and controlling configurations of each member product.

The method should support establishing configuration tree for a product line with the following capabilities:

— defining the configuration tree for a product line whose elements are member products' configurations;

— maintaining configuration trees including relationships among tree elements;

— establishing trace and control mechanism for a product line's configuration tree, for configuration of each member product, and for the relationship between the two of them.

The tool should support establishing configuration tree for a product line with the following capabilities:

— allowing the definition of member products' configurations;

— allowing the definition of a configuration tree for a product line based on the member products' configurations;

— supporting traces among configuration tree elements in accordance with the differences among member products;

— providing capability for tracing each member product's configuration from the configuration tree for a product line;

— supporting capability for tracing each member product's configuration in time dimension.

8.2.3 Manage configuration of variability in space

The goal of this task is to manage configuration of variability in space by an authorized way. Integrity of a domain variability model with application variability models should be ensured.

The method should support managing configuration of variability in space with the following capabilities:

— establishing a parallel and distributed configuration management system;

— tracking the versions of variability models, e.g. a domain variability model and application variability models;

— tracking the versions of assets with the versions of variability models;

— controlling the configurations of variability models in accordance with product line evolution;

— controlling the configurations of assets with variability model.

A tool should support managing configuration of variability in space with the following capabilities:

— maintaining formal configuration management schemes;

— allowing access right control of domain and application engineering as separate rights;

— maintaining versions of assets, variability models, and applications;

— maintaining the configurations of platforms and variability models in accordance with product line evolution;

— providing repository for separating assets, variability models, and applications with trace links among them;

— supporting forward and backward propagation for applying configuration changes.

8.3 Decision management

Purpose of decision management

Decision management is to select the most optimal alternative among technical or management decision alternatives. Since software and systems in many domains are increasingly critical the failures of decision management may cause catastrophic consequences. Examples of some critical decisions in software and systems product line include scoping, commonality vs. variability, binding time, platform architecture, domain testing strategy, etc. This subprocess guides tasks, methods, and tools for decision management to be safe from catastrophic consequences.

Inputs

— Decision needs.

— Decision environment.

— Historical information relevant to the decision.

Outcomes

— Defined decision goal is established.

- Goal achievement criteria are defined.
- *Tailored decision process* is defined.
- *The best decision option* is selected.
- *Measures of the decision performance* are defined.
- *Reviewed results of the gap between expected and actual performance* are documented.
- *Root causes of the gap* are analysed.
- *Lessons learned* are documented.

Tasks

- *Establish decision management policy*. Decision management should establish necessary policy for decision making to select the best option.
- *Tailor decision procedure* customizes generic decision procedure to meet specific decision needs.
- *Guide the decision execution* provides the guidance on the documentation of decision rationale, the measurement of decision effectiveness, gap analysis between expected and actual performance for the selected decision option, and root cause analysis of the gap.
- *Learn from execution results* guides learning mechanism, e.g. Baysian.

8.3.1 Establish decision management policy

Decision management policy establishes necessary grounds for reasonable decision making that can be analysed by quantitative/qualitative measures.

The method should support establishing decision management policy with the following capabilities:

- defining generic decision procedure;
- identifying quantitative/qualitative decision measures;
- establishing decision criteria with threshold value;
- providing a guidance for overall decision procedure.

Example generic decision procedure

- formulate decision goals,
- define goal achievement measures,
- generate alternatives,
- converge alternatives,
- evaluate alternatives,
- select the best alternatives,
- document the rationale,
- activate the decision,
- measure the performance of the decision results,
- review the gap between expected and actual goal achievement,
- analyze root cause of the gap, and

— learn by learning mechanism.

A tool should support establishing decision management policy with the following capabilities:

— sharing the decision management policy with relevant participants;

— providing multi-criteria decision supports.

8.3.2 Tailor decision procedure

The goal of this task is to customize generic decision procedure to align with specific decision such as scoping, commonality vs. variability, binding time, platform architecture, domain testing strategy, etc.

The method should support tailoring decision procedure with the following capabilities:

— specifying to the decision needs;

— scaling up or down each procedure;

— adding or deleting some of the procedures.

A tool should support tailoring decision procedure with the following capabilities:

— providing a template for tailoring.

8.3.3 Guide the decision execution

The goal of this task is to provide the guidance on the documentation of decision rationale, the measurement of decision effectiveness, gap analysis between expected and actual performance for the selected decision option, and root cause analysis of the gap.

The method should support guiding the decision execution with the following capabilities:

— documenting decision rationale;

— measuring decision effectiveness.

A tool should support guiding the decision execution with the following capabilities:

— providing multi-criteria decision supports;

— providing documentation templates.

8.3.4 Learn from execution results

The goal of this task is to guide learning mechanism of decision making.

The method should support learning from execution results with the following capabilities:

— analysing the gap between expected and actual decision performance;

— analysing root causes;

— learning from gap analysis and root cause analysis.

A tool should support learning from execution results with the following capabilities:

— supporting learning mechanism, e.g. Baysian;

— providing documentation templates.

8.4 Technical risk management

Purpose of technical risk management

Technical risk management deals with risk issues that can jeopardize the achievement of product line goals and objectives. Severe technical risks (e.g. lack of domain knowledge, uncertain or volatile domain requirements, lack of historical data for effort estimation, etc.) are likely to materialize without adequate technical risk management, so product line risk management should establish necessary mitigation and/or contingency plans to cope with technical risks.

Inputs

— probable technical risk sources (e.g. domain engineering (for core asset development), application engineering (for product development), management, etc.);

— technical plans.

Outcomes

— Technical risks for the product line approach and their priorities and mitigation plans for the most important risk areas are documented.

Tasks

— *Identify technical risks*. Severe technical risks are identified.

— *Assess technical risks*. The severity and impacts of technical risks are evaluated and their priorities are determined.

— *Develop technical risk mitigation plans*. Risk mitigation plans are established and if necessary contingency plans are defined.

— *Activate the mitigation plan*. Mitigation plans for technical risks are initiated and the implementation status of plans are monitored and controlled.

8.4.1 Identify technical risks

The goal of this task is to identify severe technical risks of product lines.

The method should support identifying technical risks with the following capabilities:

— identifying common technical risks possible to happen for carrying out each process of overall product line process areas;

— identifying technical risks associated with reusability goal achievement;

— analysing risks for determining probability of occurrence and impact from overall product line, domain engineering, and application engineering perspectives (risks related to domain engineering have a ripple effect throughout the product line so they should be analyzed with particular care);

— analysing interrelationships between risks for later collaboration among the different teams.

A tool should support identifying technical risks with the following capabilities:

— allowing access to domain artifacts by providing interfaces with domain development tools;

— sharing risk sources;

— accumulating historical data for capturing and assessing risks.

8.4.2 Assess technical risks

The goal of this task is to evaluate the technical risks by estimating risk probability and its consequences.

The method should support assessing technical risks with the following capabilities:

— evaluating the risks as to likelihood and consequences;

— assessing the options for accommodating the risks;

— analysing relations among technical risks identified from domain and application engineering respectively;

— prioritizing risks in accordance with their severity;

— establishing a risk baseline and mitigation plans.

A tool should support assessing technical risks with the following capabilities:

— sharing technical risks related to core asset development and variability management with the geographically distributed stakeholders;

— supporting documentation of technical risks' assessment results including their priorities.

8.4.3 Develop technical risk mitigation plans

The goal of this task is to prepare risk mitigation or contingency plans.

The method should support developing technical risk mitigation plans with the following capabilities:

— determining the technical risk mitigation efforts;

— developing technical risk mitigation plans;

— aligning collaboration among different organization units;

— tracking the technical risk mitigation efforts.

A tool should support developing technical risk mitigation plans with the following capabilities:

— providing collaboration environments for parallel management of risks.

8.4.4 Activate the mitigation plan

The goal of this task is to implement the mitigation plan for technical risks.

A method should support activating the mitigation plan with the following capabilities:

— initiating the mitigation plans;

— monitoring technical risk status for the product lines;

— tracking technical risk mitigation action items.

A tool should support activating the mitigation plan with the following capabilities:

— showing the technical risk mitigation status;

— tracking technical risk mitigation action items.

8.5 Tool management

Purpose of tool management

The purpose of tool management is to identify subprocesses or tasks that require tool support. This subprocess deals with generic tool support whereas one purpose of this International Standard is to deal with tool capabilities for product line-specific tasks.

NOTE Selection and evaluation for CASE tool refer to ISO/IEC 14102.

Input

— Requests for tool support.

Outcome

— *Requests for tool support* are documented.

— *Tool requirements specific to a product line* (tool criteria) are specified.

— *Tool lists including their capabilities* are documented.

Tasks

— *Identify needs for tool support* elicits needs for tool support. Product line processes or tasks that are essential for tool support are identified and their relevant business and technical needs are specified and analysed.

— *Select and adapt tools. Avail*able tools are selected and evaluated about their suitability to support the business and technical needs of a product line organization. If necessary, tools for a single product are adapted in order to meet the needs.

— *Set-up and maintain tools*. The chosen and adapted tools are operated and if necessary, they are maintained. Because tools operated within a product line should support for concurrently creating, maintaining, and using multiple versions of product line artifacts, a development environment requires coordination of domain/application engineering and organization/technical management processes. In addition sharing of product line artifacts is essential. Therefore, tools should be maintained continuously as the product line processes evolve and are sophisticated.

8.5.1 Identify needs for tool management

The goal of this task is to identify tool needs with their relevant business and technical needs. Since managerial complexities of product lines are high and there are two tracks of development process, domain and application engineering processes which should collaborate closely with each other, tool support is essential. Furthermore, collaborations among tools have to be ensured because product lines should control the development of multiple products.

Especially, the managerial complexity of configuration (change) management, variability management, and traceability management cannot be dealt without tool support. Variability management tools, product line architecture design tools, and product line testing tools are examples of product line specific types of tool support.

The method should support identifying needs for tool support with the following capabilities:

— analysing technical and managerial complexities, which are sources of tool requests;

— prioritizing the tool needs in accordance with the importance of each requests;

— analysing detailed requirements of tool needs (including criteria of tool's abilities);

— specifying tool requirements (providing templates).

A tool should support identifying needs for tool support with the following capabilities:

— allowing access to tool needs and their relevant process information for aligning tool selection and processes;

— supporting documentation of tool requirements.

8.5.2 Select and adapt tools

The goal of this task is to select available tools and evaluate their fitness for the needs. If adaptation is necessary, effort analysis should be embodied.

Product line tools require capabilities such as reuse, evolution, binding, consistency checking, and platform support. Furthermore, product line tools have to be interoperable with other tools. For these, adaptation or development of add-ons is usually required.

The method should support selecting and adapting tools with the following capabilities:

— searching and selecting candidate tools;

— evaluating candidate tools based on tool requirements specification or defined criteria (e.g. quality attributes of the entire tool-support environment);

— assessing compatibility with currently operating tools;

— determining tools and adapting them if necessary.

A tool should support selecting and adapting tools with the following capabilities:

— providing communication environments for discussion;

— recoding and retrieving adaptation information;

— sharing rationales for tool decision.

8.5.3 Set up and maintain tools

The goal of this task is to set up tools and maintain them for coping with the tool requirements changes. Because product line evolves continuously, product line support tools should be flexible for coping with future expansion.

The method should support setting up and maintaining tools with the following capabilities:

— allowing interoperation among tools in a product line environment;

— monitoring the uses of entire tools;

— maintaining tools or inserting new tools into current product line environment in accordance with product line evolution.

A tool should support setting up and maintaining tools with the following capabilities:

— installing tools by being interoperable with other tool environments;

— integrating a tool into its operation environment (including the existing tools in which it incorporates);

— tracking the usage status of tools;

— collecting feedbacks about usage of tools;

— maintaining the versions of currently operating tools.

Annex A
(informative)

Technical management and technical readiness levels (TRL)

In the world of Systems, it is quite typical to handle long development time that span over 10 to 15 years. The time-to-deliver aircraft carriers, constellations of satellites or high-speed railways systems is typically in the magnitude of one or two decades. In this case, the development efforts are also spread across the time. This reduces the financial impact of developing everything at the same time, but also enable to take advantage of the future. In this sense, the System will be delivered as increments which are usually providing partial missions before the entire system is fully operational.

However, the design of the first System increment should already freeze all major interfaces (mechanicals, electrical, networks, etc.). This is a major challenge for the System architects to design a System which subsystems will be designed several years later. That unavoidable concern requires provisions against failures of the future to deliver the anticipated technologies.

Systems Product Lines do apply even in these very large systems as reusability is a major concern.

For instance, a high-speed railways System may be delivered in 10 years. At the very beginning of the implementation, the railways stations and the tracks are built. The stations are operational after three years, and will serve with existing tracks and trains. The ground segment (e.g. rails and tracks) will be operational seven years later and will be connected then, to be operated with existing low-speed trains. Then, the design of the high-speed train and related signals can start with the current technologies and not with the ones available at the beginning of the project, which will be obsolete.

However, the design of the high-speed railways System, done the first year, should freeze many interfaces, even if the high-speed trains are not designed in details: provisions for power supply, communication, physical size, anticipated speed, etc. shall be made. Therefore, the System architects shall take into account some prospective technologies availabilities. The Technical Readiness Level is a way to indicate how mature are the envisioned technologies and a path to reduce the risks of using them. Technical Readiness Levels are usefull to bringing them from a low-level (prospective) to a higher one (trusted).

If the use of TRL is important for a single high-speed train, it is even more when several high-speed systems are being delivered in the next 10 to 50 years with overlaps between the different yards, to avoid duplicated efforts and costly incompatibilities. Product Line is a way to optimize the costs and risks of those Systems and Projects.

The Technical Readiness Level is a scale ranging from one (some interesting phenomenon has been observed) to nine (the technology has been used in real projects). The intermediate levels indicate how this phenomenon is turned into usable laws and principles, leaving the Research labs to enter the world of Engineers.

The Technical Management Process is responsible for handling the sequencing of the engagement of the different technologies, as well as their maturity. The outcomes of the Technical Management process using the Technical Readiness Levels are major inputs for risks studies, as well as for prospectives studies on the future of Product Lines.

Annex B
(informative)

Relationship with ISO/IEC 12207 processes

Table B.1 — Process Mapping from ISO/IEC 12207 to ISO/IEC 26555

Category and process of ISO/IEC 12207		**Relevant process and subprocess of ISO/IEC 26555 (technical mgt.)**		
Agreement	1. Acquisition			
	2. Supply			
Org. project enabling	3. Life cycle model mgt.	1. Process mgt.	Applying process enabling processes for product lines	S1
			Domain engineering process definition	N
			Application engineering process definition	N
			Process monitoring and control	S1
			Process improvement	S1
	4. Infrastructure mgt.		Applying process enabling processes for product lines	S1
	5. Project portfolio mgt.			
	6. Human resource mgt.		Applying process enabling processes for product lines	S1
	7. Quality mgt.	4. Support mgt.	Technical quality mgt.	S1
Project	8. Project planning	4. Support mgt.	Technical planning and control	N
	9. Project assessment and control	4. Support mgt.	Technical planning and control	S1
	10. Decision mgt.	4. Support mgt.	Decision management	S0
	11. Risk mgt.	4. Support mgt.	Technical risk mgt.	N
	12. Configuration mgt.	4. Support mgt.	Configuration mgt.	S2
	13. Information mgt.			
	14. Measurement	4. Support mgt.	Process monitoring and control	S2

'N' means "Newly Added Subprocess".

'S' means "Supplemented Subprocess":

— S0: conceptually same with ISO/IEC 12207;

— S1: a little supplement;

— S2: considerable supplement;

— S3: extensive supplement.

Table B.1 *(continued)*

Category and process of ISO/IEC 12207		Relevant process and subprocess of ISO/IEC 26555 (technical mgt.)		
Technical	15. Stakeholder requirements definition			
	16. System requirements analysis			
	17. System architectural design			
	18. Implementation			
	19. System integration			
	20. System qualification testing			
	21. Software installation			
	22. Software acceptance support			
	23. Software operation			
	24. Software maintenance			
	25. Software disposal			
SW implementation	26. Software implementation			
	27. Software requirements analysis			
	28. Software architectural design			
	29. Software detailed design			
	30. Software construction			
	31. Software integration			
	32. Software qualification testing			
SW support	33. SW documentation mgt.			S0
	34. SW configuration mgt.	4. Support mgt.	Configuration mgt.	S2
	35. SW quality assurance	4. Support mgt.	Technical quality mgt.	S2
	36. SW verification			S0
	37. SW validation			S0
	38. SW review			S0
	39. SW audit			S0
	40. SW problem resolution			S0
Software Reuse	41. Domain engineering			
	42. Reuse program mgt.	Technical Mgt.		S3
	43. Reuse asset mgt.	3. Asset mgt.	Asset base implementation	S3
			Asset identification	N
			Asset annotation	N
			Asset validation	N
			Asset evolution	N

'N' means "Newly Added Subprocess".

'S' means "Supplemented Subprocess":

— S0: conceptually same with ISO/IEC 12207;

— S1: a little supplement;

— S2: considerable supplement;

— S3: extensive supplement.

Table B.1 *(continued)*

Category and process of ISO/IEC 12207		**Relevant process and subprocess of ISO/IEC 26555 (technical mgt.)**		
		2. Variability mgt.	Variability model mgt.	N
			Variability binding mgt.	N
			Variability documentation mgt.	N
			Variability tracing	N
			Variability control and evolution	N
		4. Support mgt.	Tool mgt.	N

'N' means "Newly Added Subprocess".

'S' means "Supplemented Subprocess":

— S0: conceptually same with ISO/IEC 12207;

— S1: a little supplement;

— S2: considerable supplement;

— S3: extensive supplement.

Annex C
(informative)

Construct for process, method, tool, and aspect

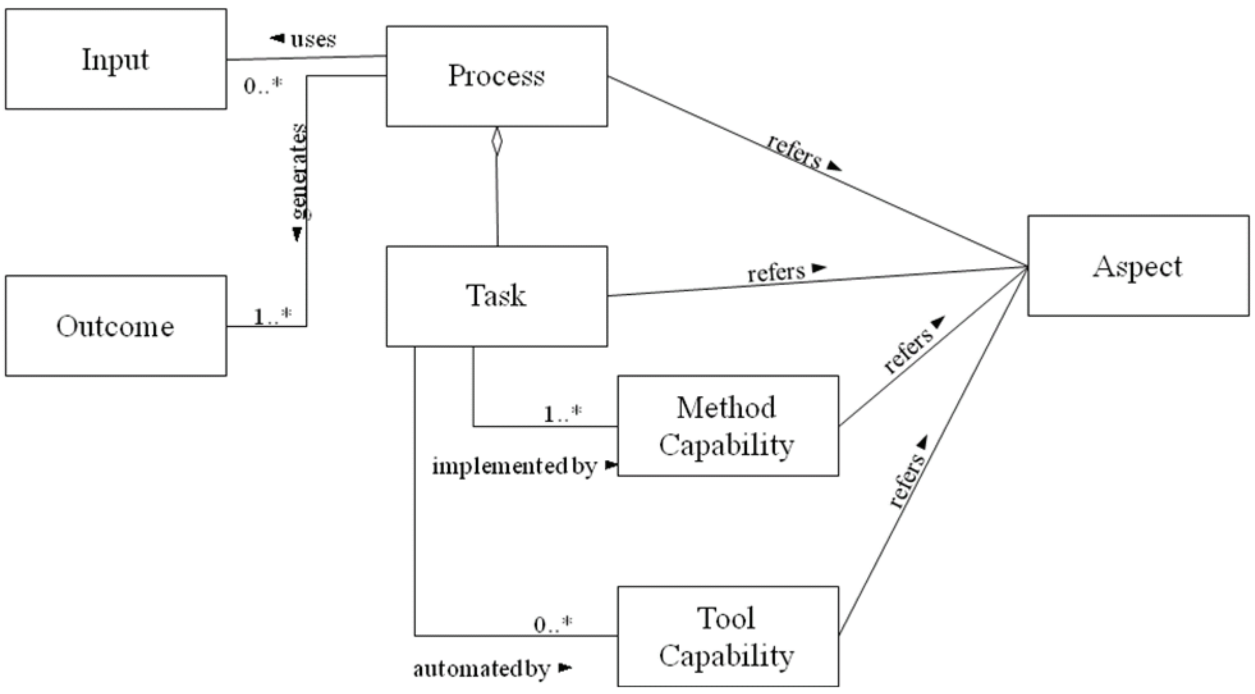

Figure C.1 — Construct for process, method, tool, and aspect

Bibliography

[1] ISO/IEC 15940:2013, *Systems and software engineering — Software Engineering Environment Services*

[2] ISO/IEC 14102:2008, *Information technology — Guideline for the evaluation and selection of CASE tools*.

[3] ISO/IEC 25000:2014, *Systems and software engineering — Systems and software Quality Requirements and Evaluation (SQuaRE) — Guide to SQuaRE*

[4] ISO/IEC/TR 19759, *Software Engineering — Guide to the Software Engineering Body of Knowledge (SWEBOK)*

[5] Klaus Pohl, Günter Böckle, Frank J. van der Linden. Software Product Line Engineering: Foundations, Principles and Techniques. Springer 2005.

[6] NORTHROP Linda M., & CLEMENTS Paul C. A Framework for Software Product Line Practice, Version 5.0. Software Engineering Institute, Carnegie Mellon University, July 2007.

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

ICS 35.080

Price based on 61 pages