INTERNATIONAL STANDARD

ISO/IEC 26557

First edition
2016-12-15

Software and systems engineering — Methods and tools for variability mechanisms in software and systems product line

Ingénierie du logiciel et des systèmes — Méthodes et outils pour les mécanismes de variabilité dans les chaînes de production de logiciels et de systèmes

Reference number
ISO/IEC 26557:2016(E)

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

Contents

Page

Foreword

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

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular the different approval criteria needed for the different types of ISO documents 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 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 World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see the following URL: www.iso.org/iso/foreword.html.

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

Introduction

Software and systems product line (SSPL) engineering and management creates, exploits and manages a common platform to develop a family of products (e.g. software products, systems architectures) at lower cost, reduced time to market and with better quality. As a result, it has gained increasing global attention since 1990s.

Variability, which differentiates a member product from other products within a product line, plays an important role in SSPL. Variability mechanism means ways to implement variability; it realizes variability in the product line artefacts. Variability mechanisms differ in accordance with the binding time of variability, and variability of a product line is introduced from product line scoping through product line testing and its binding can occur at any stages of product line development. Thus, variability mechanism should be systematically managed for the right operation in domain engineering and for the right binding in application engineering. Furthermore, variability mechanisms should support easy variability management and traceability management. Accordingly, this document provides processes with their supporting methods and tools capabilities for variability mechanism operationalization and for managerial supports for the right use of variability mechanisms at domain engineering stages and the right bindings at application engineering stages.

This document can be used in the following modes:

— by the users of this document: to benefit people who want to adopt SSPL for producing their products by guiding variability mechanism operationalization, variability mechanism management and variability mechanism supports;

— by a product line organization: to provide guidance in the evaluation and selection for methods and tools for the tasks of providing variability mechanism operationalization, variability mechanism management and variability mechanism supports;

— by providers of tools and methods: to provide guidance in implementing or developing tools and methods by providing a comprehensive set of the capabilities of methods and tools for supporting variability mechanism operationalization, variability mechanism management and variability mechanism supports.

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

ISO/IEC 26550, ISO/IEC 26551 and ISO/IEC 26555 are published. ISO/IEC 26558 and ISO/IEC 26559 are under preparation. ISO/IEC 26552, ISO/IEC 26553, ISO/IEC 26554, ISO/IEC 26556, ISO/IEC 26560, ISO/IEC 26561, ISO/IEC 26562 and ISO/IEC 26563 are planned.

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

— Processes and capabilities of methods and tools for domain design and application design are provided in ISO/IEC 26552 (planned).

— Processes and capabilities of methods and tools for domain realization and application realization are provided in ISO/IEC 26553 (planned).

— Processes and capabilities of methods and tools for domain testing and application testing are provided in ISO/IEC 26554 (planned).

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

— Processes and capabilities of methods and tools for organizational management are provided in ISO/IEC 26556 (planned).

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

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

— Processes and capabilities of methods and tools for product management are provided in ISO/IEC 26560 (planned).

— Processes and capabilities of methods and tools for technical probe are provided in ISO/IEC 26561 (planned).

— Processes and capabilities of methods and tools for transition management are provided in ISO/IEC 26562 (planned).

— Processes and capabilities of methods and tools for configuration management of asset are provided in ISO/IEC 26563 (planned).

— Others (ISO/IEC 26564 to ISO/IEC 26599): planned.

Software and systems engineering — Methods and tools for variability mechanisms in software and systems product line

1 Scope

This document, within the context of tools and methods of variability mechanisms for software and system product lines**:**

— provides the terms and definitions related to variability mechanisms for software and systems product lines;

— defines processes and their subprocesses for operating variability mechanisms at each product line life cycle stages and those for providing managerial supports. Those processes are described in terms of purpose, inputs, tasks and outcomes;

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

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

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

2 Normative references

There are no normative references in this document.

3 Terms and definitions

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

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

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

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

3.1
application configuration
derivation for a member product specific executables from domain assets in *realization* (3.10)

Note 1 to entry: The specific configuration of an application is the *binding* (3.3) results for the *variation points* (3.19) with the selected *variants* (3.17).

3.2
aspect
special consideration within product line engineering process groups and tasks to which we can associate specialized methods and tools

3.3
binding
task for making a *decision* (3.7) on relevant *variants* (3.17) using domain *variability model* (3.16) and *decision tables* (3.8)

3.4
binding time
moment of variability resolution

Note 1 to entry: The choice of binding time is independent from variability modelling. It is the consequence of *decisions* (3.7) made from requirements through *run time* (3.11). Demands for flexibility and the support of tools allow late binding times or even the use of variable binding times.

3.5
binding time decision
selection for *variability* (3.13) defined in platforms in accordance with the functional distinction between variability in time and variability in space

3.6
configurability
degree of how well a *variability mechanism* (3.14) supports the configuration of a member product

3.7
decision
types of statements in which a choice between two or more possible outcomes controls which set of actions will result

3.8
decision table
table that specifies decision variables

Note 1 to entry: It also includes rules, constraints and relevancy among variables.

3.9
post-compile time
collective name for link time and load time that are right after the compilation of components

3.10
realization
stage for detailed design and construction

3.11
run time
stage that a member product is executed

Note 1 to entry: Components can be developed, compiled, linked and loaded separately. Only at run time are they combined into a working system.

3.12
texture
architectural texture
collection of common development rules and constraints for realizing the applications of a product line

3.13
variability
characteristics that may differ among members of a product line

Note 1 to entry: The differences between members may be captured from multiple viewpoints such as functionality, quality attributes, environments in which the members are used, users, constraints and internal mechanisms that realize functionality and quality attributes.

Note 2 to entry: It is important to distinguish between the concepts of system and software variability and product line variability. Any system partially or fully composed of software can be considered to possess software variability because software systems are inherently malleable, extendable or configurable for specific use contexts. Product line variability is concerned with the variability that is explicitly defined by product management. This document is primarily concerned with product line variability.

EXAMPLE 1 In the case of a home automation system, in accordance with business strategy, the use of a LAN as an alternative to the European Installation Bus (EIB) in a home automation system might be a competitive advantage for the company since it allows the use of low-cost components.

EXAMPLE 2 Annex B provides variability examples in accordance with variability types.

3.14
variability mechanism
variability representation/implementation technique for the product line variability

Note 1 to entry: It deals with variabilities based on the *binding time* (3.4) at the specific life cycle stage.

3.15
variability mechanism operationalization
VMO
adequate provision or *binding* (3.3) of *variability mechanisms* (3.14) at each specific domain or application engineering life cycle stage

3.16
variability model
explicit definition for product line variability

Note 1 to entry: It introduces *variation points* (3.19), types of variation for the variation points, *variants* (3.17) offered by the variation points, variability dependencies and variability constraints. Variability models may be orthogonal to or integrated in other models such as requirements or design models. There are two types of variability models: application variability models and domain variability models.

3.17
variant
option or an alternative that may be used to realize particular *variation points* (3.19)

Note 1 to entry: One or more variants should correspond to each variation point to represent V_VP relationship. Selection and *binding* (3.3) of variants for a specific product determine the characteristics of the particular *variability* (3.13) for the product.

3.18
variant selection
decision making for a choice of a *variant* (3.17) in a *variation point* (3.19)

Note 1 to entry: *binding* (3.3), variability resolution.

3.19
variation point
representation corresponding to particular variable characteristics of products, domain assets and application assets in the context of a product line

Note 1 to entry: Variation points show which of the product line element varies. Each variation point can have multiple V_VP relationships.

4 Variability mechanisms for software and systems product line (SSPL)

4.1 Overview

Variability mechanism means a method for implementing the variability of a product line and it incorporates variability into the product line development artefacts. Because variability is introduced at the whole product line life cycle stages and so binding does, variability mechanisms for implementing variability in accordance with its binding time should be provided. It is necessary to classify variability mechanisms by binding times so the users of this document can choose proper variability mechanisms in accordance with the binding time decisions. This subclause provides a set of variability mechanisms used at given binding times.

Bindings can occur at requirements stage. In the case of requirements, binding decisions for whether external requirements variabilities exist or not, are made. Variability mechanism operationalization for requirements differ in accordance with requirements artefacts. Variability mechanism operationalization for requirements can be summarized as follows:

— SSPL specifically defined mechanism: feature variability notation;

— language extension mechanism: stereotype in activity diagram, sequence diagram and state machine diagram, coloured notations in state machine diagram;

— language supported mechanism: extends and includes in use case model, swim lane and notes in activity diagram, notes in textual specification and sequence diagram, tags or markups in textural use case scenario.

During architecture design, bindings also occur. In the case of design time, binding elements that compose architectural structure such as components, ports and connectors are selectable. In accordance with the variant selection, architectural structure differs. Components and interfaces are also substituted with another. Variability mechanism operationalization for design can be summarized as follows:

— SSPL specifically defined mechanism: plug-ins in component framework diagram, composite structure diagram, package diagram and deployment diagram, architecture reorganization in process table;

— language extension mechanism: stereotype in component diagram, class diagram, E-R diagram and communication diagram;

— language supported mechanism: notes or tagged values in component diagram, frameworks, pre- and post-conditions in Object Constraint Language (OCL).

At realization stage, additional variation points can be introduced and bindings can occur, so variability mechanisms supporting variability in realization stage are required. Variability mechanism operationalization for realization can be summarized as follows:

— SSPL specifically defined mechanism: orthogonal feature set in model-driven approaches (e.g. KobrA);

— language extension mechanism: stereotype in entity model, stereotype in detailed design level model;

— language supported mechanism: generalization/specialization (extension) in classes, aggregation (optional) in class, abstract class designed as parameterized class (template) and concrete classes (optional), tagged values in entity model, generics in codes, pointcut and advice concepts in Aspect Oriented Programming (AOP), framework implementation methods (e.g. hotspots and hooks) in framework.

There exists variability binding during compilation and in some cases, all bindings can be delayed after compilation. Most bindings occurred when compilation are difficult to change in subsequent stage. Variability mechanism operationalization at compile time can be summarized as follows:

— SSPL specifically defined mechanism: break (variation point), adapt (include) and select (variants) in XVCL, Markup (vp, insert_before, insert_after, insert) and highlighted variant elements code in frame technology;

— language extension mechanism: none;

— language supported mechanism: macro, #ifdef and directives mark in codes for conditional compilation, parameters in makefile.

After compilation, bindings can occur for generating different linked configurations. Such bindings at compile time and at link time are also difficult to change in subsequent stage. Variability mechanism operationalization at link time can be summarized as follows:

— SSPL specifically defined mechanism: configuration space including rules and constraints for different bindings at link time;

— language extension mechanism: none;

— language supported mechanism: configuration files or linker directives for linking static libraries, parameters in makefiles.

Bindings also occur for loading different executable files. Variability mechanism operationalization at load time can be summarized as follows:

— SSPL specifically defined mechanism: rules and constraints description for bindings at load time;

— language extension mechanism: none;

— language supported mechanism: import in source code or external makefile for load time dynamic linking.

For supporting different installation in accordance with the customer preferences or system environments, bindings occur at deployment/installation time. Variability mechanisms at deployment/installation time can be summarized as follows:

— SSPL specifically defined mechanism: rules and constraints description for bindings at deployment/installation time;

— language extension mechanism: none;

— language supported mechanism: options for conditional installation.

Some variabilities are bound at run time. In the case of highly reconfigurable systems, bindings occur only at run time. In normal case, bindings occur at run time for reflecting the running conditions. Variability that is bound at run time tends to be easily rebound or unbound in accordance with the user selections or context conditions. Unlike binding at compile or link time, bindings at run time can easily be changed. Variability mechanism operationalization at run time can be summarized as follows:

— SSPL specifically defined mechanism: rules and constraints description for bindings at run time;

— language extension mechanism: none;

— language supported mechanism: dynamic libraries.

Variabilities should be encoded into test artefacts for addressing test variations among member products. Bindings at testing stage are closely related to the bindings of member product's development stage. Variabilities can also be added for achieving test variations among member products. Variability mechanism operationalization for test artefacts can be summarized as follows:

— SSPL specifically defined mechanism: segmentation and fragmentation mechanisms with notes for test case scenario in sequence diagram;

— language extension mechanism: decision point and its branches for test model in extended activity diagram;

— language supported mechanism: note for test plan in natural language.

NOTE 1 Variability mechanisms for the selected software development activities are depicted in Annex A.

NOTE 2 Binding time suggestions in accordance with variability types and their exemplar variability are depicted in Annex B.

4.2 Reference model for variability mechanisms in product line

The methods and tools for variability mechanisms for software and systems product line should support systematic implementation and management of variability mechanisms in accordance with the determined binding times. They also need to adequately be used in domain engineering life cycle processes in order to enable right bindings in application engineering life cycle processes.

The reference model specifies the structure of supporting processes and subprocesses for variability mechanisms in product line. As shown in Figure 1, variability mechanism in product line can be structured into three processes: variability mechanism management, variability mechanism operationalization and variability mechanism support. In the rest of this document, tasks, methods and tools are described in terms of processes and subprocesses defined in the reference model.

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

— the title of the subprocess;

— the purpose of the subprocess;

— the inputs to produce the outcomes;

— the tasks to achieve the outcomes;

— the outcomes of the subprocess;

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

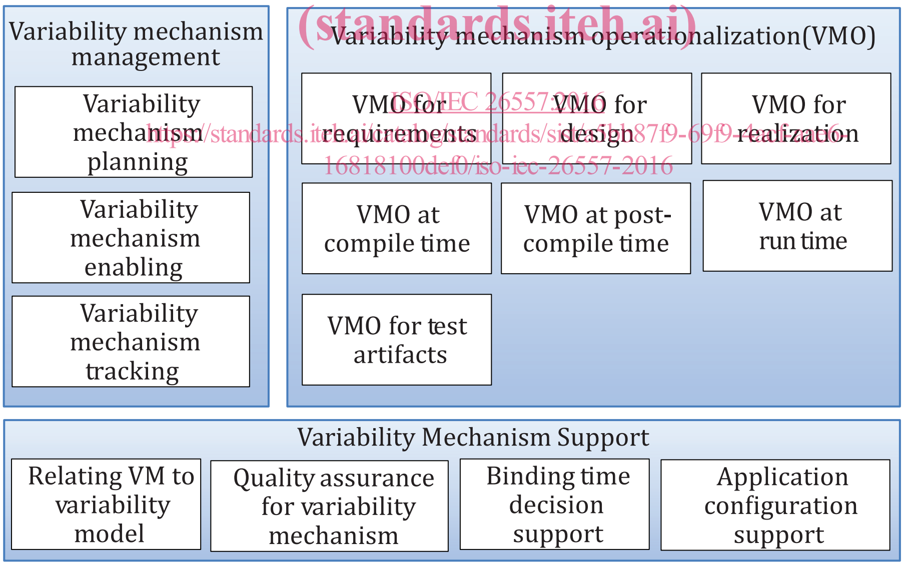

Figure 1 — Variability mechanisms for SSPL

Variability mechanism management shall serve to do the following and to define the capabilities of tools and methods for supporting them.

— *Variability mechanism planning* establishes an organization level plan to utilize variability mechanisms.

— *Variability mechanism enabling* defines, maintains and assures the availability of strategies for variability mechanism operationalization, guidance for variability mechanism selection and enabling supports such as human skills and tools for variability mechanism operationalization.

— *Variability mechanism tracking* provides integrated management for variability mechanism operationalization in both domain engineering and application engineering.

Variability mechanism operationalization shall serve to do the following and to define the capabilities of tools and methods for supporting them.

— *Variability mechanism operationalization for requirements* provides variability mechanisms used for dealing with the variabilities that will be bound at requirements stage.

— *Variability mechanism operationalization for design* provides variability mechanisms used for dealing with the variabilities at architecture design stage.

— *Variability mechanism operationalization for realization* provides variability mechanisms used for dealing with the variabilities that will be bound at detailed design and implementation stage;

— *Variability mechanism in compilation* provides variability mechanisms used for dealing with the variabilities that will be bound at pre-compile and compile time.

— *Variability mechanism operationalization at post-compile time* provides variability mechanisms used for dealing with the variabilities defined for different sequence of linkages, for loading different executable files and for installing or deploying different configurations.

— *Variability mechanism operationalization at run time* provides variability mechanisms used for dealing with the variabilities defined for accessing different files (or components, interfaces) at run time.

— *Variability mechanism operationalization for test artefacts* provides variability mechanisms used for dealing with variabilities that should be realized in test assets.

Variability mechanism support shall serve to do the following and to define the capabilities of tools and methods for supporting them.

— *Relating variability mechanism to variability model* establishes and maintains information (e.g. binding time, rules and constraints adhered) necessary to conduct right binding and right configuration by using right variability mechanisms.

— *Quality assurance for variability mechanism* helps ensure whether the artefacts and processes of variability mechanism management and operation comply with predefined provisions and plans.

— *Binding time decision support* deals with the required provisions necessary to conduct the right binding at the planned binding stages (e.g. architecture rules to constrain the use of macro or makefile parameters).

— *Application configuration support* deals with the realization of configurability by determining ways to derive a member product such as compiler parameters.

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

Table 1 — Key aspects for identifying product line-specific variability mechanism tasks

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

The following are the descriptions for each aspect concerning variability mechanisms for product lines. The variability mechanism relevant processes and tasks shall the identified on the basis of these