DRAFT INTERNATIONAL STANDARD

ISO/IEC DIS 26557

ISO/IEC JTC **1**/SC **7**	Secretariat: **SCC**
Voting begins on: **2015-08-10**	Voting terminates on: **2015-11-10**

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

Titre manque

ICS:

Reference number
ISO/IEC DIS 26557:2015(E)

COPYRIGHT PROTECTED DOCUMENT

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

Table of Content

Page

Foreword

ISO (the International Organization for Standardization) is a worldwide federation of national standards bodies (ISO member bodies). The work of preparing International Standards is normally carried out through ISO technical committees. Each member body interested in a subject for which a technical committee has been established has the right to be represented on that committee. International organizations, governmental and non-governmental, in liaison with ISO also take part in the work. ISO collaborates closely with the International Electrotechnical Commission (IEC) on all matters of electro-technical standardization.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.

Draft International Standards adopted by the technical committees are circulated to the member bodies for voting. Publication as an International Standard requires approval by at least 75 % of the member bodies casting a vote.

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.

(draft) International Standard ISO/IEC 26557 was prepared by Technical Committee **ISO/IEC JTC1**, *Joint Technical Committee 1,* Subcommittee SC 7, *Systems and Software 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 artifacts. 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 phases 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 International Standard provides processes with their supporting methods and tools capabilities for variability mechanism operation and for managerial supports for the right use of variability mechanisms at domain engineering phases and the right bindings at application engineering phases.

This International Standard can be used in the following modes:

-- By the users of this International Standard – to benefit people who want to adopt SSPL for producing their products by guiding variability mechanism operation, 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 operation, 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 operation, variability mechanism management, and variability mechanism supports.

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

1 Scope

This International Standard deals with the tools and methods of variability mechanisms for software and system product line. The scope of this International Standard is as follows**:**

- provide the terms and definitions related to variability mechanisms for software and systems product lines.
- define 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.
- define method capabilities to support the defined tasks of each process.
- define tool capabilities to automate/semi-automate tasks or defined method capabilities.

This International Standard 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

- ISO/IEC 26550:2013 Software and systems engineering – Reference model for product line engineering and management
- ISO/IEC 26551:2013 Software and systems engineering – Tools and methods for product line requirements engineering
- ISO/IEC 26552: Software and systems engineering – Tools and methods for product line architecture (TBD)
- ISO/IEC 26555:2013 Software and systems engineering – Tools and methods for product line technical management

3 Terms and Definitions

For the purposes of this document, the following terms and definitions apply. The general terms common to all software and systems product line have been excluded. For more information concerning the general terms, the reader should refer to “ISO/IEC 26550 Software and systems engineering —Reference model for product line engineering and management”.

3.1 application configuration
derivation for a member product specific executables from domain assets in realization

NOTE The specific configuration of an application is the binding results for the variation points with the selected variants.

3.2 binding
task for making a decision on relevant variants using domain variability model and decision tables

3.3 binding time
moment of variability resolution

NOTE The choice of binding time is independent from variability modeling. It is consequence of decisions made from requirements through runtime. Demands for flexibility and the support of tools allow late binding times or even the use of variable binding times.

3.4 binding time decision
selection for variability defined in platforms in accordance with the functional distinction between variability in time and variability in space

3.5 configurability
degree of how well a variability mechanism supports the configuration of a member product

3.6 decision
determination for the value of decision variables in a decision table for describing the individual member product

3.7 decision table
table that specifies decision variables (including rules, constraints, and relevancy among variables) that will be determined by application engineers

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

3.9 realization
stage for detailed design and construction

3.10 run time
stage that a member product is executed

NOTE Components can be developed, compiled, linked and loaded separately. Only at run-time they are combined into a working system.

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

NOTE 1 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 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 International Standard 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 EIB (European Installation Bus) in a home automation system might be a competitive advantage for the company, since it allows the use of low-cost components.

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

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

NOTE It deals with variabilities based on the binding time at the specific life cycle stage

3.13 variability model
explicit definition for product line variability

NOTE It introduces variation points, types of variation for the variation points, variants 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.

3.14 variant
an option or an alternative that may be used to realize particular variation points

NOTE One or more variants must correspond to each variation point. Each variant has to be associated with one or more variation points. Selection and binding of variants for a specific product determine the characteristics of the particular variability for the product.

3.15 variant selection
decision making for a choice of a variant in a variation point

Syn: binding, variability resolution

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

NOTE Variation points show what of the product line element varies. Each variation point should have at least one variant.

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 artifacts. 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 International Standard can choose proper variability mechanisms in accordance with the binding time decisions. This subsection provides a set of variability mechanisms used at given binding times.

Bindings can occur at requirements phase. In the case of requirements binding decisions for whether external requirements variabilities exist or not are made. Variability mechanisms in requirements differ in accordance with requirements artefacts. Variability mechanisms in 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, colored 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 mechanisms in 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 OCL(Object Constraint Language).

At realization phase additional variation points can be introduced and bindings can occur, so variability mechanisms supporting variability in realization phase are required. Variability mechanisms in 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 AOP(Aspect Oriented Programming), 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 during compilation are difficult to change in subsequent phase. Variability mechanisms 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 phase. Variability mechanisms 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 mechanisms 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 runtime 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 mechanisms 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 artifacts for addressing test variations among member products. Bindings at testing phase are closely related to the bindings of member product's development phase. Variabilities can also be added for achieving test variations among member products. Variability mechanisms in testing 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.

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. In the rest of this document, product line engineering practices, methods, and tools are described in accordance with a framework focusing on variability mechanisms in product line (Figure 1).

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

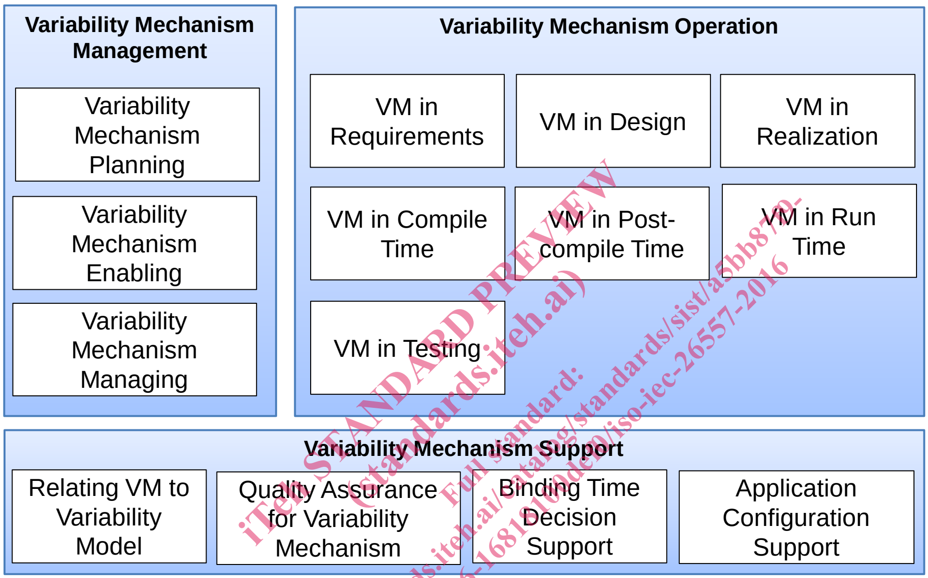

Fig. 1 Variability mechanisms for SSPL

Variability mechanism management shall serve to do the followings 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 operation, guidance for variability mechanism selection, and enabling supports such as human skills and tools for variability mechanism operation.
- *Variability mechanism managing* provides integrated management for variability mechanism operations in both domain engineering and application engineering.

Variability mechanism operation shall serve to do the followings and to define the capabilities of tools and methods for supporting them:

- *Variability mechanism in requirements* provides variability mechanisms used for dealing with the variabilities that will be bound at requirements phase.
- *Variability mechanism in design* provides variability mechanisms used for dealing with the variabilities at architecture design phase.
- *Variability mechanism in realization* provides variability mechanisms used for dealing with the variabilities that will be bound at detailed design and implementation phase.
- *Variability mechanism in compilation* provides variability mechanisms used for dealing with the variabilities that will be bound at pre-compile and compile time.