FINAL
DRAFT

INTERNATIONAL
STANDARD

**ISO/IEC
FDIS
26580**

ISO/IEC JTC **1**/SC **7**

Secretariat: BIS

Voting begins on:
2021-01-22

Voting terminates on:
2021-03-19

Software and systems engineering — Methods and tools for the feature-based approach to software and systems product line engineering

Reference number
ISO/IEC FDIS 26580:2021(E)

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

Published in Switzerland

Contents

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.

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) or the IEC list of patent declarations received (see patents.iec.ch).

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

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

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

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

Introduction

Feature-based software and systems product line engineering ("feature-based PLE" for short) is a specialization of software and systems product line (SSPL) engineering and management that is described in ISO/IEC 26550. ISO/IEC 26550 describes a very generalized approach to SSPL, focusing on the benefits of exploiting a common platform of reusable assets for a product family. Each organization that adopts SSPL under ISO/IEC 26550 is free to do so using their preferred techniques and methods.

What is the motivation for creating a standard for a specialization of SSPL? As the SSPL field has matured and achieved widespread attention in the industry, a specific and repeatable approach to SSPL has emerged that takes advantage of commercial off-the-shelf industrial-strength tools and technology, along with robust best practices for methods and processes, that automate and formalize many of the processes in domain and application engineering. The result is that less upfront analysis, design, and implementation effort is required prior to gaining the benefits from the approach.

While SSPL in general provides significant benefits, it also requires a significant investment of time and effort to adopt and to ultimately achieve those benefits. The feature-based PLE specialization is a more narrowly defined solution that can be supported by off-the-shelf tools and methods, which has resulted in lower investments when an organization adopts SSPL. Feature-based PLE embodies lessons learned about SSPL practices that have been shown to provide some of the highest benefits and returns (see, for example, References [2] and[8]).

This document provides a reference model consisting of an abstract representation of the key technical elements, tools, and methods of feature-based PLE. The predominant specializations of general SSPL that characterize feature-based PLE are:

a) a mapping from features to asset variation points that is sufficient to drive a fully automated configurator that produces assets specific to member products;

b) a methodological shift of all design and implementation effort, change management, and configuration management to domain engineering, so that application engineering is reduced to automated configuration of member product instances and testing of configured member products and member-product-specific assets.

This document embodies a distinct separation of concerns between the feature-based PLE technology providers and feature-based technology users. For each of these stakeholder concerns, the scope of this document is to define only what is necessary and sufficient to enable feature-based PLE practice. For technology providers, this imparts flexibility in how the necessary and sufficient technical capabilities are provided, as well as the opportunity to offer more expansive capabilities that are possible in an ideal solution. For technology users, this provides flexibility to select among the technology providers and to apply the methods that best match their technical and business objectives for feature-based PLE.

Software and systems engineering — Methods and tools for the feature-based approach to software and systems product line engineering

1 Scope

This document is a specialization of the more general reference model for software and systems product line engineering and management described in ISO/IEC 26550. The specialization defined herein addresses a class of methods and tools referred to as feature-based software and systems product line engineering, or feature-based PLE, which has emerged as a proven and repeatable product line engineering and management (PLE) practice supported by commercial tool providers.

This document:

— provides the terms and definitions specific to feature-based PLE;

— defines how feature-based PLE is a specialization within the general ISO/IEC 26550 reference model for product line engineering and management;

— defines a reference model for the overall structure and processes of feature-based PLE and describes how the elements of the reference model fit together;

— defines interrelationships, and methods for applying the elements and tools of the product line reference model;

— defines required and supporting tool capabilities.

In this document, products of feature-based PLE include digital work products that support the engineering of a system. Some of the artefacts are actually part of the delivered products, while other artefacts can be non-deliverable, such as physical or digital design models.

The intended audience for this document comprises:

— technology providers who wish to provide automated tool support for the reference model and processes described in this document;

— champions within an organization who wish to introduce feature-based PLE throughout that organization;

— IT staff within a PLE organization who will introduce and maintain the necessary technology to support feature-based PLE;

— practitioner stakeholders who will use the provided technology to practice feature-based PLE;

— technical and business managers who will sponsor and direct the methods necessary to practice feature-based PLE;

— university professors, researchers, corporate trainers, and other educators who will create and share pedagogical materials about feature-based PLE and its benefits.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ISO/IEC/IEEE 15288, *Systems and software engineering — System life cycle processes*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC/IEEE 12207, ISO/IEC/IEEE 15288, and the following apply.

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

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

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

NOTE 1 For additional terms and definitions in the field of systems and software engineering, see ISO/IEC/IEEE 24765, which is published periodically as a "snapshot" of the SEVOCAB (Systems and software engineering – Vocabulary) database and is publicly accessible at www.computer.org/sevocab.

NOTE 2 Because feature-based PLE is a specialization of the more general product line engineering approach described in ISO/IEC 26550, some of the terminology used herein is noted as a specialization of the terminology from ISO/IEC 26550, with further details provided in Annex A.

3.1
bill-of-features
specification for a *member product* (3.8) in the *product line* (3.16), rendered in terms of the specific *features* (3.4) from the *feature catalogue* (3.5) that are chosen for that member product

3.2
bill-of-features portfolio
collection comprising the *bill-of-features* (3.1) for each *member product* (3.8) in a *product line* (3.16)

3.3
domain superset
collection comprising the *feature catalogue* (3.5) and *shared asset supersets* (3.18)

3.4
feature
characteristic of a *member product* (3.8) in a *product line* (3.16) that distinguishes it from other member products in the product line

Note 1 to entry: Features can a) express the customer-visible or end-user-visible variability among the member products in a product line, or b) distinguish implementation variability not directly visible to a customer or end user except through non-functional differences such as price, performance, noise, weight, energy and more.

Note 2 to entry: In feature-based PLE, features express differences among member products. A capability or other characteristic common to all member products in the product line is not modelled as a feature.

Note 3 to entry: See Annex A for the definition of this term in ISO/IEC 26550.

3.5
feature catalogue
model of the collection of all the *feature* (3.4) options and *feature constraints* (3.6) available across the entire *product line* (3.16)

3.6
feature constraint
formal relationship between two or more *features* (3.4) that is necessarily satisfied for all *member products* (3.8)

3.7
feature language
syntax and semantics for the formal representation, structural taxonomy, and relationships among the concepts and constructs in the *feature catalogue* (3.5), *bill-of-features portfolio* (3.2), and *shared asset superset* (3.18) *variation points* (3.20)

3.8
member product
product belonging to the *product line* (3.16)

[SOURCE: ISO/IEC 26550:2015, 3.15, modified — The preferred term "application" has been removed.]

3.9
mutually exclusive
alternatives from which at most one is selected

3.10
mutually inclusive
alternatives from which zero or more are selected

3.11
PLE factory
technological, organizational, and business infrastructure and processes to support a *PLE factory configurator* (3.12) producing *product asset instances* (3.14) from *shared asset supersets* (3.18) based on a *bill-of-features* (3.1) for a *member product* (3.8)

3.12
PLE factory configurator
automated mechanism that produces assets for a specific *member product* (3.8) by processing the *bill-of-features* (3.1) for that member product, and exercising the *shared assets'* (3.17) *variation points* (3.20) in light of the *feature* (3.4) selections made in that bill-of-features

3.13
PLE factory development environment
toolset for creating, organizing, assembling, and maintaining a collection of elements in a *feature catalogue* (3.5), *bill-of-features portfolio* (3.2), *shared asset supersets* (3.18), and a hierarchy of a *product line* (3.16) of product lines

3.14
product asset instance
instantiation of a *shared asset* (3.17) specific to a *member product* (3.8), automatically produced by the *PLE factory configurator* (3.12), corresponding to a *bill-of-features* (3.1) for that member product

Note 1 to entry: A product asset instance is analogous to an application asset (ISO/IEC 26550) with the proviso that it is produced by the PLE factory configurator.

3.15
product instances
collection comprising the *bill-of-features portfolio* (3.2) and *product asset instances* (3.14)

3.16
product line
family of similar products with variations in *features* (3.4)

Note 1 to entry: See Annex A for the definition of this term in ISO/IEC 26550.

3.17
shared asset
software and systems engineering lifecycle digital artefacts that compose a part of a delivered *member product* (3.8) or support the engineering process to create and maintain a member product

Note 1 to entry: A shared asset is analogous to a domain asset (ISO/IEC 26550).

Note 2 to entry: Typical shared assets are requirements, design specifications or models for mechanical, electrical, and software, source code, build files or scripts, test plans and test cases, user documentation, repair manuals and installation guides, project budgets, schedules, and work plans, product calibration and configuration files, mechanical bills-of-materials, electrical circuit board and wiring harness designs, engineering management plans, engineering drawings, training plans and training materials, skill set requirements, manufacturing plans and instructions, and shipping manifests.

3.18
shared asset superset
representation of a *shared asset* (3.17) that includes all content needed by any of the *member products* (3.8)

3.19
variant
alternative that can be used to realize a particular *variation point* (3.20)

[SOURCE: ISO/IEC 26550:2015, 3.28, modified — the word "one" at the beginning of the definition has been removed; "may" has been changed to "can"; "particular variation points" has been changed to "a particular variation point"; note 1 to entry has been removed.]

3.20
variation point
identification of a specific piece of *shared asset superset* (3.18) content and a mapping from *feature* (3.4) selection(s) to the form of that content that appears in a *product asset instance* (3.14)

Note 1 to entry: In this document, all features express characteristics that differ among *member products* (3.8), which according to ISO/IEC 26550 would also make every feature a variation point. To avoid this redundancy, this document does not call out features as variation points.

Note 2 to entry: See Annex A for the definition of this term in ISO/IEC 26550. The definition in this document is more specific to feature-based PLE and the *PLE factory configurator* (3.12) approach of producing product asset instances from shared asset supersets.

4 Overview of feature-based product line engineering

4.1 General

This clause gives a brief informational overview of feature-based product line engineering, as a way of introducing key concepts that are important for understanding the purpose and content of this document.

Software and systems product line engineering emerged some time ago as way to engineer a portfolio of related products in an efficient manner, taking full advantage of the products' similarities while respecting and managing their differences. Here "engineer" means all of the activities involved in planning, producing, delivering, deploying, sustaining, and retiring products. Born in the software engineering field in the 1970s and 1980s and based largely on the concept of software reuse, PLE has long been known for delivering significant improvements in development time, cost, and quality of products in a product line[1].

Early attempts to capture and codify best practices recognized a dichotomy between two sides: product content that applies to multiple products and product content specific to a single product. Some referred to the two sides as domain engineering and application engineering, respectively, while others referred to core asset development vs. product development[1]. Application engineering was often said to include creating any content that happened to be used only in a single product, and promoting that content to domain status only if subsequently used in more. Application engineering included the obligation to choose and carry out a production strategy – that is, a way to turn the shared assets into products – that was potentially different for each type of shared asset, and was often manual and therefore labour-intensive and error-prone. Whatever they were called, the two sides stood on roughly equal footing in terms of the effort required to execute.

However, starting in the early 2000s, the advent of industrial-strength and commercially available technology designed specifically to support PLE enabled the rapid emergence of a specialization of PLE practices. This approach is called feature-based software and systems product line engineering[5].

Under this approach, the technology just mentioned allows application engineering, which was so important under the "classic" PLE approach, to shrink to almost nothing. Products are produced through the use of high-end industrial-strength automation that configures the shared assets appropriately for each product. Feature-based PLE explicitly declines to configuration-manage or change-manage product versions. Instead, the shared assets (and not the individual products or systems) are managed under CM (configuration management). A new version of a product is not derived from a previous version of the same product, but from the shared asset supersets themselves. Additionally, any defects are fixed in the shared assets, not the products. The affected products will then be regenerated, into a form suitable for testing and deployment. Since regeneration has a low and fixed cost, it matters very little whether 2 or 200 or 2000 products need to be regenerated. Thus, fixing a defect, making a systematic enhancement, or carrying out any other kind of multi-product change becomes much more economical[4].

4.2 to 4.4 describe a few more of the key ways that feature-based PLE differs from its antecedents.

4.2 Shared assets

Shared assets are the "soft" artefacts associated with the engineering lifecycle of the products, the building blocks of the products in the product line. Assets can be whatever artefacts that are representable digitally and either compose a product or support the engineering process to create a product. Examples include, but are not limited to, the following:

- requirements;
- design specifications and design models for mechanical, electrical, and software:
- software source code and build files;
- test plans and test cases;
- user documentation, repair manuals, and installation guides;
- project budgets, schedules, and work plans;
- product calibration and configuration files;
- data models;
- process descriptions;
- parts lists and mechanical bills-of-materials;
- engineering drawings;
- electrical circuit board and wiring harness designs;
- training plans and training materials;
- skill set requirements;
- manufacturing plans and instructions;
- shipping manifests;
- marketing brochures;
- product descriptions;
- contract proposals.

Shared assets are engineered to be used (shared) across the product line. All of this was true before, but in feature-based PLE, shared assets take the form of supersets, meaning that any asset content used in any product is included. For example, a shared asset of requirements contains all of the requirements across the product line; a shared asset of computer software source code contains all of the source code; and so forth.

The virtue of this approach is that every piece of content for the product line, no matter in which products it is used, is only created, stored, and maintained once. There is no duplication or replication of asset content at all. This elimination of duplication (and elimination of replication of work across duplicates) is where feature-based PLE derives its savings[5].

The shared asset supersets include variation points, which are places in the asset that denote content that is configured according to the feature selections made for the product being built. When a product is built, a statement of the product's distinguishing characteristics – its features – is applied to "exercise" these variation points (that is, cause the content associated with each variation point to be configured in a way to meet the needs of the product).

Configuration capabilities typically include inclusion or omission of the content; selection from among mutually exclusive content alternatives; generation of content from feature specifications; and feature-based transformation of content from one form into another.

This approach enforces consistent treatment of all of the variation points in all of the shared assets; for example, variation points are specified using the same feature-based configuration language. This consistent treatment of variation across all shared assets is a hallmark of feature-based PLE. The result is that a full set of demonstrably consistent supporting artefacts can be systematically produced for each product.

In contrast, consider approaches where each domain asset type has its own variation mechanism. For example, a requirements asset engineering team has embraced a PLE requirements management technique based on tagging requirements in a requirements database with attributes that differentiate feature variations in requirements. Further, the design team has adopted a SysML tool and has embraced inheritance as the mechanism for managing their design asset variations. The software asset development team is using a feature-oriented domain analysis (FODA)[7] feature model drawn in a graphical editor, plus #ifdefs, build flags and configuration management branches to manage implementation variations. Finally, the test team has adopted clone-and-own (i.e. make a copy and take ownership of modifications to the copy) of test plan sections, stored in appropriately named file system directories to manage their test asset variations.

Now consider what would be needed to create a complete PLE lifecycle solution from these unrelated asset variation mechanisms that integrates into a larger business process model. How do the requirements asset attributes and tagged requirements relate and trace to the subtypes and supertypes in the design assets? How do these attributes and supertypes relate and trace to the #ifdef flags, CM branches, FODA features in the software assets, and test asset clone directories? Trying to translate between the different representations and characterizations of features and variations among the domain assets creates convoluted dissonance at the boundaries between stages in the domain asset engineering lifecycle.

To resolve this quagmire of multiple disparate mechanisms for managing product line asset variation, a key aspect of feature-based PLE is consistent and traceable treatment of variation points across all shared asset types, using feature choices as the basis of a common language of variation[1].

4.3 Features

As the name implies, features play a central role in feature-based PLE. Features express differences among products in a product line. A capability or other characteristic common to all products in the product line is not modelled as a feature. A feature is a distinguishing characteristic of a product, usually directly visible to the customer or user of that product, but can also express a distinguishing implementation variability in any phase of the lifecycle that is not directly visible to a customer or end user except through non-functional differences such as price, performance, noise, weight, energy and more. An example is a function that one product can perform that others cannot. The concept of

"feature" allows experienced domain experts and systems engineers to employ a consistent abstraction when defining and making selections from a whole product configuration, all the way down to the deployment of components and parts within a low-level subsystem in the architecture[3]. A bill-of-features (analogous to a bill-of-materials, but defining a product in terms of its features rather than its parts) can be the communication vehicle among business, product marketing, and engineering units.

4.4 Automated means of production

At the centre of feature-based PLE is an automated mechanism called a configurator that exercises the shared assets' variation points to produce configured variants that, together, constitute the artefact set for each of the products in the product line. PLE configurators have been commercially available since the early 2000s.

Previous PLE approaches have always made allowances for automation, but large-scale PLE demands it. The complexities of modern product lines are growing to astronomical proportions. For example, an automotive product line may comprise thousands or millions of products[8]. Automated generation of products is essential. Once this technology is in place, products are instantiated rather than manually created.

5 A feature-based specialization of software and systems product line engineering

ISO/IEC 26550 describes software and systems product line engineering (SSPL), a general approach for efficiently engineering the full lifecycle of digital assets for a family of similar products or systems. The approach derives its benefits from reusing and sharing the digital engineering assets across the products in the family and by systematically managing variations among the digital assets for the different products. A reference model for this overall approach is defined in ISO/IEC 26550. The primary illustration from ISO/IEC 26550 is shown in Figure 1.