Variability Management in Software Product Line Engineering

Klaus Pohl

Lero (The Irish Software Engineering Research Centre)
University of Limerick
Limerick, Ireland
+ 353-(61)-202-706
pohl@lero.ie

Software Systems Engineering
University of Duisburg-Essen
Schützenbahn 70
45117 Essen, Germany
+49-(201)-183-4660
pohl@sse.uni-due.de

Andreas Metzger

Software Systems Engineering
University of Duisburg-Essen
Schützenbahn 70
45117 Essen, Germany
+49-(201)-183-4650
metzger@sse.uni-due.de

ABSTRACT

By explicitly modeling and managing variability, software product line engineering provides a systematic approach for creating a diversity of similar products at low cost, in short time, and with high quality. This tutorial focuses on the two principle differences of software product line engineering when compared to single systems development: The differentiation of two key development processes (domain engineering and application engineering) and the explicit representation and management of variability. We characterize the two processes and their main activities and introduce the orthogonal variability modeling approach (OVM). We further illustrate the OVM approach in the product line requirements engineering and product line testing activities.

Categories and Subject Descriptors

D.2.10 [**Software Engineering**]: Design – *methodologies;* D.2.13 [**Software Engineering**]: Reusable Software – *domain engineering, reusable libraries*

General Terms

Documentation, Design, Languages, Verification.

Keywords

Requirements engineering, software product lines, testing, variability management, variability modeling

1. INTRODUCTION

Software product line engineering (SPLE) has proven to be *the* approach for developing a diversity of similar software products and software-intensive systems at low costs, in short time, and with high quality. Numerous reports document the significant achievements of introducing software product lines in industry.

To facilitate the efficient development of a diversity of products, two important kinds of activities are performed in software product line engineering:

1. The developers identify and describe where the applications of the product line vary in terms of the features they provide or the requirements they fulfill such that the customers can easily select the features they desire.
2. The developers create reusable artifacts of a product line in such a way that these artifacts are sufficiently adaptable (variable) to efficiently derive individual applications from them.

Variability modeling, which is used to document the variability of the product line, is a powerful tool for managing the complexity that is involved with the above kinds of activities.

After a brief introduction of a proven SPLE framework, the tutorial focuses on product line variability. More precisely, we discuss the modeling as well as the management of variability across software product line development artifacts.

The tutorial is based on our recent text book on software product line engineering [1].

2. A FRAMEWORK FOR SOFTWARE PRODUCT LINE ENGINEERING

The tutorial is structured along the SPLE framework in Fig. 1, which has been defined based on our experiences and the results of the European SPLE research projects ESAPS, CAFÉ, and FAMILIES [2]. The framework shows the two key product line engineering processes: *domain engineering* and *application engineering*.

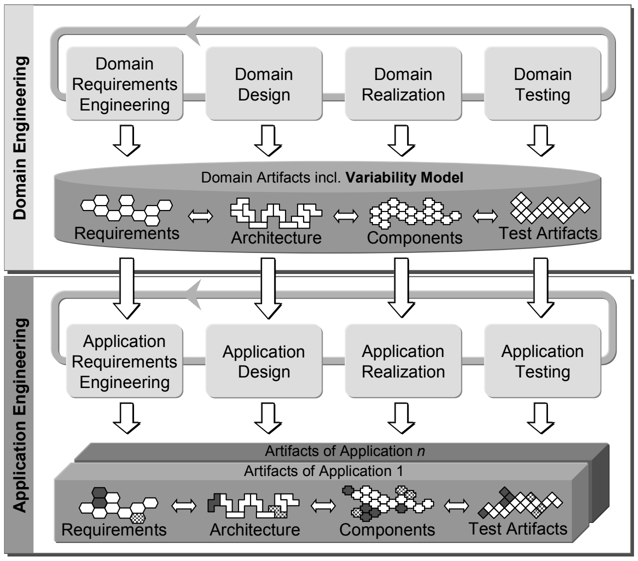

Fig. 1: SPLE framework (adapted from [1])

The *domain engineering* process (shown in the upper half of Fig. 1) is responsible for defining the commonality and the variability of the product line, and thus for establishing the reusable artifacts. Domain engineering has to ensure that the available variability is appropriate for producing the applications that are within the scope of the product line, which also involves mechanisms for realizing variability in the respective development artifacts (e.g., by designing configurable components).

The *application engineering* process (shown in the lower half of the figure) is responsible for deriving product line applications from the reusable artifacts. Application engineering exploits the variability of the reusable artifacts by binding the variability according to application specific needs.

In this tutorial, we will explain in detail how product line variability can be handled during the product line requirements engineering and product line testing activities (see Fig. 1):

- During the *domain requirements engineering* sub-process, common and variable requirements of the product line are elicited and documented. The output of this sub-process comprises reusable textual and model-based requirements, and in particular, the variability model of the product line.
- During *application requirements engineering*, requirements specifications of the individual, customer-specific applications are developed by reusing the requirements artifacts from domain requirements engineering.
- In *domain testing,* early tests are performed to ensure a high quality of the reusable artifacts of the product line. Additionally, reusable test cases are created.
- In the *application testing* sub-process, individual application test cases are derived from the reusable test cases, and further tests are performed to validate and verify the individual applications of the product line.

3. MODELING VARIABILITY

For managing the variability of the product line, an adequate documentation of variability information is essential. Variability information has to be available both during domain engineering when deciding on the variability and commonality of the product line, as well as during application engineering when the variability is bound to develop individual applications.

3.1 Explicit Documentation of Variability

To be useful, the documentation of variability should at least answer the following questions:

- *What does vary?* The answer of this question leads to a variable item or a variable property of an item, called *variation point*; e.g., the kind of alarm notification of an alarm system.
- *How does it vary?* Answering this question identifies the possible instances of a variable item or a variable property, called *variants*. For example, variants of the variation point 'alarm notification' can be 'siren', 'flasher', and 'police call'.
- *Why does it vary?* This question leads to the rationale behind a variation point or variant. In the example, different regional laws might prohibit certain kinds of alarm notification.
- *Who is it documented for?* This question identifies the target group of a variation point or variant; e.g., the customers.

3.2 Orthogonal Variability Model (OVM)

Existing approaches for explicitly documenting variability, e.g., FODA or FORM [3], are adequate for specifying both common and variable features of a product line. However, this can lead to huge and complex models. With our orthogonal variability modeling approach (OVM [1]), a significant reduction of model size and complexity can be achieved, because only the variable aspects of a product line are documented in the OVM models.

Additionally, the OVM approach can be used to document the variability of arbitrary development artifacts, ranging from textual requirements to design models and even test cases. Also, the documentation of variability is separated from the technical realization of variability in the development artifacts and as such provides a further level of abstraction that aids developers in managing complexity.

In this tutorial, we introduce the OVM approach in detail. Fig. 2 shows parts of the meta-model for the OVM language.

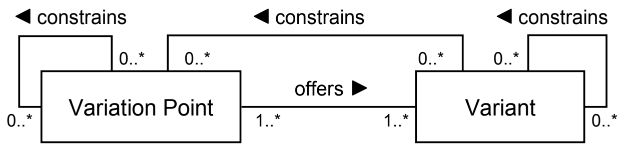

Fig. 2: Excerpt of the OVM meta-model (adapted from [1])

The core concepts of the OVM language are *variation point* ("what does vary") and *variant* ("how does it vary"). Each variation point has to offer at least one variant (*offers*-association). Additionally, the *constrains*-associations between these elements describe dependencies between variable elements.

Employing the OVM approach has three major advantages:

- *Improved communication*: High-level abstractions of variable artifacts are available to simplify the communication of product line variability.
- *Improved decision making*: Engineers have to make their rationales for introducing variability explicit.
- *Improved traceability*: The dependencies between variable artifacts can easily be traced by employing the *constrains*-associations in the OVM models.

4. REFERENCES

[1] Pohl, K., Böckle, G. and van der Linden, F. *Software Product Line Engineering – Foundations, Principles, and Techniques*. Springer, Berlin, Heidelberg, New York, 2005.

[2] van der Linden, F. Software Product Families in Europe: The Esaps & Café Projects, *IEEE Software*, 19(4), 2002, 41–49.

[3] Kang, K.C., Kim, S., Lee, J. et al. FORM: A feature-oriented reuse method with domain-specific reference architectures. *Annals of Software Engineering*, 5, 1998, 143–168.