Using Feature Models for Product Derivation*

Olaf Spinczyk
University of Dortmund
Olaf.Spinczyk@udo.edu

Holger Papajewski
pure-systems GmbH
Holger.Papajewski@pure-systems.com

Abstract

In general the implementation of a software product line leads to a high degree of variability within the software architecture. For an effective development and deployment it is necessary to resolve variation points within the architecture and source code automatically during product/variant derivation. Given the complexity of most software systems tool support is necessary for these tasks. This tutorial familiarizes the audience with different tools and their underlying concepts that help to automate the product derivation process. After beginning on the level of features models and feature selections the tutorial covers various topics down to level of source code organization and language issues. Thereby, the presenters provide valuable insights about how to put product line engineering into practice.

1. Motivation

Feature models are the most commonly used modeling concept for the description of commonality and variability in the context of software product lines. However, often these models are merely used to document the results of the domain analysis. In contrast to this, with proper tool support it is possible to connect feature models with the solution space, i.e. the architecture and code of the product line implementation, and, thereby, gain traceability and even an automated feature-driven product derivation process. New product variants can be derived simply by the selecting desired features. This reduces the application engineering costs and helps to keep the product line platform manageable.

Today several approaches to feature-driven product derivation can be used. There are simple ones such as feature-driven asset selection or more complex approaches such as the combination of feature models with model-driven software development. For product line architects and developers it is crucial for long-term success to know the benefits as well as the limits of these feature model based derivation techniques. The aim of this tutorial is to provide an overview on the most important product derivation techniques and, thus, help product line architects and developers to choose the right technique for their specific SPLD demands.

2. Example

This tutorial shows how feature models combined with appropriate tools can provide this support [3]. An example for simple feature-driven file selection is shown in figure 1. The screenshots were taken from the pure::variants[1] variant management system. In step (1) the user selects all desired features from the *feature model*. The *solution space* consists of a *family model* and a *component repository*. The *family model* maps features on *logical implementation components* (2), which in turn are mapped on *physical implementation files*. The thereby determined set of implementation files for a concrete configuration is copied into a target directory and finally compiled and linked into the actual product variant, which is a weather station software in this example.

3. Contents

At the beginning of the tutorial the importance of the separation of problem space modeling and solution space modeling is discussed. This followed by a description of various techniques that connect both spaces. We start with simple tool supported asset selection and move on to approaches with increasing generative power such as feature-driven aspect selection or feature-driven code generation. Each of the approaches is evaluated with a common scheme that helps to understand the specific pros and cons. Eventually, an overview on all covered approaches provides a direct comparison.

The tutorial is accompanied by short demonstrations of the presented concepts with freely available tools, namely AspectC++ [6], Feature Modeling Plug-In [1], oAW [4],

*This work was supported by the German Research Council (DFG) under grant no. SCHR 603/4 and SP 968/2-1.

[1] www.pure-systems.com

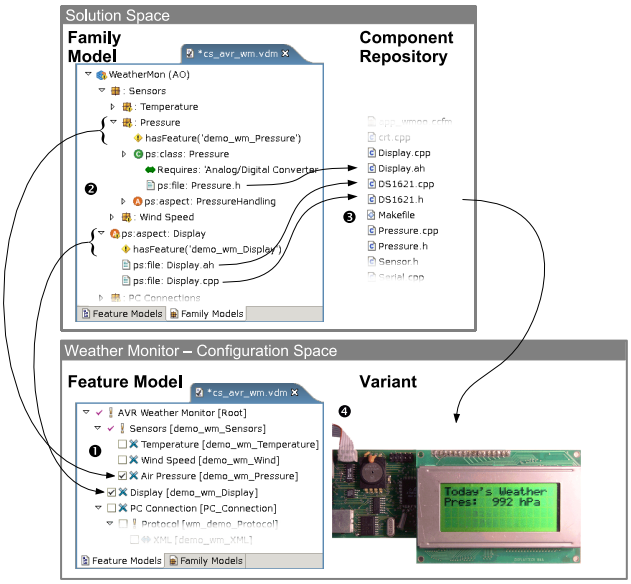

Figure 1. Example: problem space/solution space connection in the pure::variants variant management system.

pure::variants [2], XVCL [5] and, thereby, gives attendees pointers to useful tools when the concepts shall be put into practice.

4. Presenters

Olaf Spinczyk is a Professor of Computer Science at the University of Dortmund, Germany. He received his PhD from the University of Magdeburg, Germany, in 2002 for his research on the combination of aspect-orientation and product-line concepts for the construction of embedded operating system families. An important part of this work was the design of the AspectC++ language, which he started in 2001. AspectC++ is an aspect-oriented language extension for C++ in the style of AspectJ. Today he is the main designer and developer of the AspectC++ weaver, which he presented at various AOSD and OOPSLA demonstrations and tutorials. His current research is focused on the combination of generic and generative programming with AOP in AspectC++, and on applying these techniques in the implementation of the embedded CiAO operating system family.

Holger Papajewski is CTO of the pure-systems GmbH. pure-systems is a software company specialized in services and tool development for the application of product line technologies in embedded software systems. Before he joined pure-systems he worked as researcher at the University of Magdeburg and was involved in network software and real-time operating system development. Among other things he was responsible for the development of OSEK-compatible family members of an object-oriented operating systems family. At pure-systems he is responsible for the management of a software product line development with several hundred thousands lines of code, written in C++ and Java. Holger has (co-) authored several scientific articles during his academic career and also taught students in the field of operating system development.

5. Intended Audience

The tutorial is intended for practitioners from industry (SPL Novice to Intermediate). Attendees should have a basic understanding of software design. Some knowledge about software product lines in general is helpful.

References

[1] M. Antkiewicz and K. Czarnecki. Featureplugin: Feature modeling plug-in for eclipse. In *Proceedings of the 2004 OOPSLA Workshop on Eclipse Technology eXchange (Eclipse '04)*, pages 67–72, Vancouver, British Columbia, Canada, 2004. ACM Press.

[2] D. Beuche. Variant management with pure::variants. Technical report, pure-systems GmbH, 2003. http://www.pure-systems.com/.

[3] D. Beuche, H. Papajewski, and W. Schröder-Preikschat. Variability management with feature models. *Science of Computer Programming*, 53(3):333–352, 2004.

[4] S. Efftinge, M. Völter, A. Haase, and B. Kolb. The pragmatic code generator programmer. In *TheServerSide.COM*, Sept. 2006.

[5] National University of Singapore. XML-based Variant Configuration Language (XVCL), 2004.

[6] O. Spinczyk and D. Lohmann. The design and implementation of AspectC++. In *Journal on Knowledge-Based Systems, Special Issue on Creative Software Design*. Elsevier North-Holland, Inc., 2007. (to appear).