Modeling and Building Software Product Lines with Eclipse

Olaf Spinczyk
University of Erlangen-Nuremberg
Martensstr. 1
D-91058 Erlangen
olaf.spinczyk@informatik.uni-erlangen.de

Danilo Beuche
pure-systems GmbH
Agnetenstr. 14
D-39106 Magdeburg
danilo.beuche@pure-systems.com

General Terms

Design, Management

Categories and Subject Descriptors

D.2.13 [**Software Engineering**]: Reusable Software—*Domain engineering, Software product lines*; D.2.9 [**Software Engineering**]: Management—*Software configuration management*

Keywords

software product lines, variant management, model-driven software development, embedded systems

1. INTRODUCTION

Software product line development (SPLD) can reduce the overall software production costs, but imposes extra complexity on the development process. Dealing with the commonalities and variabilities of the product variants and with the flexible software archiecture makes product line development a real challenge. Developers can only be successful with adequate tool support for product line development, which is still not the ”state of the art”.

The demonstration shows the integration of software product line modeling based on feature models into the popular integrated development environment Eclipse. We will demonstrate that a flexible product line architecture can be build by using the modeling capabilities provided by pure::variants, an eclipse plugin for software product line development.

The plugin covers all steps of product line development from requirements and variability analysis to product generation. Extended feature models are used for modeling of problem domain. Family models are used to represent the variable architecture of product line solution domains independent of the programming or modeling languages used for product line implementation.

Along some concrete examples from the domain of embedded software product lines we will make a round-trip through all steps of the development from domain analysis to product deployment and demonstrate how pure::variants facilitates the entire process. We will show how features are used to model problem domains, how product line architectures are connected with domain models and finally how to integrate concepts of model driven development into the process of product derivation.

OOPSLA'04, Oct. 24-28, 2004, Vancouver, British Columbia, Canada.
ACM 1-58113-833-4/04/0010.

2. TOOL SUPPORT FOR SOFTWARE PRODUCT LINES

Despite some quite successful demonstrations of software product line development it is still in the beginning of attracting more and more interest. It will take a long time before there will be a common set of best pratices in SPLD and precise instructions how to implement them in reality.

Development of software product lines is a complex problem, involving high-level issues like development organisation structure and processes, but also software architecture, design and implementation. One of the cross-cutting and most crucial tasks in product line development is the variant and variability management. However, most available software development tools, including but not limited to Eclipse [2], do not offer support for explicit handling of this task.

Several important issues have to be considered for tools supporting the complete process of variability management:

- Easy, but universal model(s) for expressing variabilities and commonalities should be supported.
- Variability at all levels must be manageable.
- Introduction of new variability expression techniques should be possible and easy.

The pure::variants tool chain was developed to meet all these requirements. It is the successor of the research prototype presented in [1]. It is structured in a client/server architecture with an Eclipse plugin as its main graphical front end.

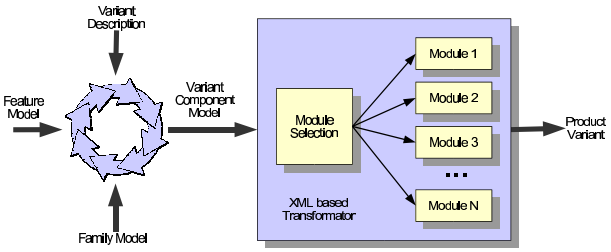

Figure 1: Overview of models and derivation process

The pure::variants-based tools are used in different phases of the software development process. The development of product lines is basically divided into two steps. In the first step the problem and

solution domains are analyzed, common assets are identified and realized (domain engineering). In the second step the individual products are derived from the product line (application engineering).

Several model types are used to capture the information required to manage variability and variants on the different levels of domain knowledge, software design and implementation.

Feature models [3] play a key role in this. They allow a uniform representation of variabilities and commonalities of the products of the entire product line. Compared to orignal works on feature models, pure::variants supports an extended version of this concept.

Implementations of the product line are described by family models. They enable the mapping of the problem space to the different implementations. This model type was developed especially for the pure::variants technology, since existing modeling techniques such as UML or SDL were not suitable for this purpose.

The variant description model is used to describe a individual product. It describes the products features and values associated with those features, and it is used to derive the final product from the family models.

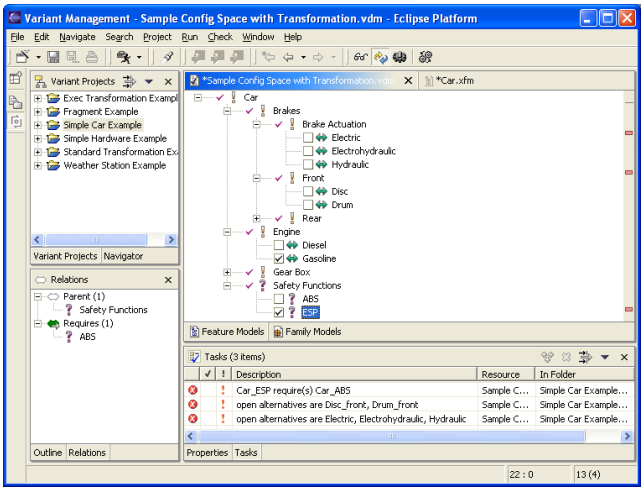

Figure 2: pure::variants plugin during variant derivation

Figure 1 depicts the simplified process of variant derivation with pure::variants. Most steps are be performed automatically once the various models have been created. The product line developers have to provide feature models, family models, and the implementations itself. To derive a specific variant features from the feature models have to be selected and possibly some associated values specified. Once this variant description is sucessfully validated the the transformer generates the variant realization from the variant model using a product line specific transformation process.

2.1 Benefiting from the Eclipse principles

The Eclipse platform is a successful open-source project with a high impact on the way how software tools are developed. The basic idea of Eclipse is that extensibility is the core for successful development environments. This idea was transformed into a very powerful and highly extensible framework by IBM and many contributors from the community.

For the pure::variants project using Eclipse as base for the graphical user interface allowed for reuse in many ways. The already available infra-structure could be reused for many standard task like resource handling, problem list, error display etc.. Also the already existing user interface elements provided a guide for own implementations.

But most importantly the pure::variants plugin uses the Eclipse extensibility mechanisms to provide interfaces for user specific implementation of views, editors and other components using the basic pure::variants infrastructure.

3. CONCLUSIONS

Crucial factors in the development of software product lines are adequate models for the domain analysis and design, programming languages that support the modular implementation even of cross-cutting concerns, and an integration of all this in a user-friendly development environment, which supports the whole process. The demonstration will show that the pure::variants tool chain in combination with the Eclipse platform addresses these factors successfully.

Further Information on pure::variants including a free community edition can be found on [4].

4. REFERENCES

[1] D. Beuche, H. Papajewski, and W. Schröder-Preikschat. Variability Management with Feature Models. In *Proceedings of the Software Variability Management Workshop*, pages 72–83, University of Groningen, The Netherlands, Feb. 2003. Technical Report IWI 2003-7-01, Research Institute of Mathematics and Computing Science.

[2] Eclipse Project Homepage. see http://www.eclipse.org.

[3] K. Kang, S. Cohen, J. Hess, W. Nowak, and S. Peterson. Feature Oriented Domain Analysis (FODA) Feasibility Study. Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, PA, USA, Nov. 1990.

[4] pure-systems Homepage. see http://www.pure-systems.com.