Cross-checking disambiguated product line variability models

Patrick Heymans
University of Namur
phe@info.fundp.ac.be

Andreas Metzger
University of Duisburg-Essen
andreas.metzger@sse.uni-due.de

Klaus Pohl
University of Duisburg-Essen & Lero
klaus.pohl@sse.uni-due.de

Pierre-Yves Schobbens
University of Namur
pys@info.fundp.ac.be

Germain Saval
University of Namur
gsa@info.fundp.ac.be

Many industry sectors face the challenge of how to satisfy the increasing demand for individualized software systems and software-intensive systems. The software product line (PL) engineering paradigm (SPLE, see [5]) has proven to empower organizations to develop a diversity of similar systems at lower cost, in shorter time, and with higher quality compared to single system development [5].

Key to SPLE is to exploit the commonalities of the systems that belong to the PL and to handle the *variability* (i.e., the differences) between those systems. *Commonalities* are properties and qualities that are shared by all systems of the PL [1]; e.g., all mobile phones let users make calls.

In SPLE, two kinds of variability can be distinguished: Software variability and PL variability.

Software variability refers to the "ability of a software system or artefact to be efficiently extended, changed, customized or configured for use in a particular context" [7]. This kind of variability is well known from the development of single systems. As examples, an abstract Java super-class allows different specializations to be used where the super-class is used; an interface allows different implementations to be chosen.

PL variability [1, 5, 3] is specific to SPLE and describes the variation between the systems that belong to a PL in terms of properties and qualities, like features that are provided or requirements that are fulfilled. It is important to understand that defining PL variability, i.e., determining what should vary between the systems in a PL and what should not, is an explicit decision of product management (see [3, 5]). As an example, product management might have decided that the mobile phones of their PL should either offer the GSM or the UMTS protocol.

A challenging task in SPLE is to map the PL variability to software variability. This means that the reusable artefacts from which the systems of the PL are built (called the *core assets*, which constitute the *PL platform* [5]) should be constructed flexibly enough to allow for efficiently and effectively building those systems [2, 6]. The decisions to be made are crucial and mutually influence each other: which systems to offer as part of the PL (i.e., what the scope of the PL should be [6]), and how to design the reusable artefacts to support this scope [3].

A lack of flexibility in the reusable artefacts, or a scope that lacks awareness of the technical realizability, can severely undermine the SPLE process. At best, time-consuming and expensive changes of the reusable artefacts or the scope will be required. Therefore, it is essential to ensure that PL variability and software variability are consistent from the beginning. But since all changes cannot be anticipated, co-evolution of both variabilities over time should be facilitated too.

In this presentation, we will introduce language and tool support for these tasks. To disambiguate the documentation of variability, we propose to record PL variability and software variability in separate models and to interrelate them. We use Feature Diagrams and Orthogonal Variability Models (OVM) respectively. Equiped with formal syntax and semantics, those models are amenable to automatic analysis in isolation. But, most importantly, since the cross-links between them are also formalized, the models can be cross-checked. We devise a set of checks that are straightforward for the stakeholders to interpret, like whether all planned systems of the PL can be realized, or whether the flexibility of the reusable artefacts is useful. A prototypical implementation relying on a SAT solver is reported.

This presentation is based on a paper by the same authors that was recently published in the proceeding of the 15th IEEE International Symposium on Requirements Engineering (RE'07) [4].

References

[1] J. Coplien, D. Hoffman, and D. Weiss. Commonality and variability in software engineering. *IEEE Softw.*, 15(6):37–

45, November 1998.

[2] I. John, J. Lee, and D. Muthig. Separation of variability dimension and development dimension. In *1st Int'l Workshop on Variability Modelling of Software-intensive Systems, VaMoS 2007, Limerick, Ireland, January 16-18, 2007*, pages 45–49. Lero, 2007.

[3] K. C. Kang, J. Lee, and P. Donohoe. Feature-oriented project line engineering. *IEEE Softw.*, 19(4):58–65, 2002.

[4] A. Metzger, P. Heymans, K. Pohl, P.-Y. Schobbens, and G. Saval. Disambiguating the documentation of variability in software product lines: A separation of concerns, formalization and automated analysis. In I. C. S. Press, editor, *Proceedings of 15th IEEE International Requirements Engineering Conference*, 2007.

[5] K. Pohl, G. Böckle, and F. van der Linden. *Software Product Line Engineering: Foundations, Principles and Techniques*. Springer, 2005.

[6] K. Schmid. A comprehensive product line scoping approach and its validation. In *22rd Int'l Conference on Software Engineering, ICSE 2002, 19-25 May 2002, Orlando, Florida, USA*, pages 593–603. ACM, 2002.

[7] M. Svahnberg, J. van Gurp, and J. Bosch. A taxonomy of variability realization techniques. *Softw. Pract. Exper.*, 35(8):705–754, July 2005.