Analysis Strategies for Software Product Lines: A Classification and Survey

Thomas Thüm,[1] Sven Apel,[2] Christian Kästner,[3] Ina Schaefer,[4] and Gunter Saake[1]
[1] University of Magdeburg, Germany
[2] University of Passau, Germany
[3] Carnegie Mellon University, USA
[4] Technische Universität Braunschweig, Germany

Abstract: Software-product-line engineering enables the efficient development of similar software products. Instead of developing each product from scratch, products are generated from common artifacts. However, the product generation is a challenge for the analysis of correctness properties. Applying traditional analysis techniques, such as type checking and model checking, to each product involves redundant effort and is often not feasible due to the combinatorial explosion of products. Approaches to scale analysis techniques to product lines have been presented in unrelated research areas with a different terminology each. We propose a classification of analysis strategies and classify a corpus of more than 100 approaches. Based on our insights, we develop a research agenda to guide research on product-line analyses.

With feature-oriented software product lines, software products can be generated automatically based on a selection of features [CE00, ABKS13]. The automated generation gives rise to a potentially huge number of software products, which raises the question of how to efficiently analyze all these products. Software analyses include static techniques, such as type checking and model checking, but also dynamic techniques, such as testing and runtime monitoring.

The need for the analysis of software product lines has been identified by several independent communities: static analysis for aspect-oriented programming [KPRS01], feature-interaction detection with model checking [PR01], compositional model checking for feature modules [FK01, NCA01], type checking in the presence of preprocessor directives [APB02], and theorem proving in model-based refinement [Pop07] – just to name a few. Nevertheless, many of these approaches share similar ideas, which are hard to recognize due to different terminologies.

Recently, we have proposed a classification to identify the strategy to deal with product-line variability during the analysis [TAK+14]. We have surveyed the literature on product-line analyses and classified all approaches accordingly [TAK+14]. In this talk, we illustrate the problem of the analysis of software product lines. We present underlying strategies of existing product-line analyses and discuss their benefits and drawbacks. We close our talk by sharing our insights on open problems on the analysis of software product lines.

A distinguishing property of each product-line analysis is whether it operates at the level of domain artifacts or at the level of generated products. An approach that operates only on generated products is called *product-based analysis*. For approaches operating only on

domain artifacts, we distinguish between *feature-based analyses*, which analyze artifacts for each feature in isolation, and *family-based analyses*, which analyze domain artifacts by incorporating the knowledge on allowed feature combinations (e.g., the feature model). Beside these three basic strategies, we found that many existing analysis approaches pursue combinations of strategies. For example, a *feature-product-based analysis* consists of a feature-based analysis, whose results are used in a subsequent product-based analysis. However, there is no strategy that is superior to all others, and further empirical evaluations are required to recommend a strategy for a given product line based on certain metrics.

As classifying the research literature is a continuous process, we set up a website to enable others to apply our classification to future approaches.[1] We hope that this talk sheds light on the diverse research area of product-line analyses, motivates researchers to systematically investigate new approaches, and helps practitioners to apply existing approaches.

References

[ABKS13] Sven Apel, Don Batory, Christian Kästner, and Gunter Saake. *Feature-Oriented Software Product Lines: Concepts and Implementation*. Springer, Berlin, Heidelberg, 2013.

[APB02] Lerina Aversano, Massimiliano Di Penta, and Ira D. Baxter. Handling Preprocessor-Conditioned Declarations. In *Proc. Int'l Working Conference Source Code Analysis and Manipulation (SCAM)*, pages 83–92, Washington, DC, USA, October 2002. IEEE.

[CE00] Krzysztof Czarnecki and Ulrich Eisenecker. *Generative Programming: Methods, Tools, and Applications*. ACM/Addison-Wesley, New York, NY, USA, 2000.

[FK01] Kathi Fisler and Shriram Krishnamurthi. Modular Verification of Collaboration-Based Software Designs. In *Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE)*, pages 152–163, New York, NY, USA, 2001. ACM.

[KPRS01] Herbert Klaeren, Elke Pulvermueller, Awais Rashid, and Andreas Speck. Aspect Composition Applying the Design by Contract Principle. In *Proc. Int'l Symposium Generative and Component-Based Software Engineering (GCSE)*, pages 57–69, Berlin, Heidelberg, 2001. Springer.

[NCA01] Torsten Nelson, Donald D. Cowan, and Paulo S. C. Alencar. Supporting Formal Verification of Crosscutting Concerns. In *Proc. Int'l Conf. Metalevel Architectures and Separation of Crosscutting Concerns*, pages 153–169, London, UK, 2001. Springer.

[Pop07] Michael Poppleton. Towards Feature-Oriented Specification and Development with Event-B. In *Proc. Int'l Working Conf. Requirements Engineering: Foundation for Software Quality (REFSQ)*, pages 367–381, Berlin, Heidelberg, 2007. Springer.

[PR01] Malte Plath and Mark Ryan. Feature Integration Using a Feature Construct. *Science of Computer Programming (SCP)*, 41(1):53–84, September 2001.

[TAK+14] Thomas Thüm, Sven Apel, Christian Kästner, Ina Schaefer, and Gunter Saake. A Classification and Survey of Analysis Strategies for Software Product Lines. *ACM Computing Surveys*, 47(1):6:1–6:45, June 2014.

[1] `http://fosd.net/spl-strategies/`