# Variability Models in Large-Scale Systems: A Study and a Reverse-Engineering Technique

Thorsten Berger[1] and Sarah Nadi[2]

[1]University of Waterloo, [2]Technical University of Darmstadt

**Abstract:** Highly configurable systems can easily have thousands of configuration options, together with intricate configuration constraints. Variability models—higher-level representations of options and constraints—facilitate the development of large, highly configurable systems. Since models are difficult to create and to maintain, we strive to support both activities, automating them as much as possible. To this end, we present an empirical study of real-world variability models, and static code-analysis techniques that support reverse-engineering and consistency-checking of such models.

**Introduction.** Customizing software systems is becoming increasingly important. A common approach is to introduce configurations options, set at build or startup time, to tailor a system to specific needs (e.g., hardware, functionality, performance). Large systems can easily have thousands of such options, together with intricate configuration constraints among them. Declaring options and constraints in higher-level representations, so-called variability models, tackles complexity. Such models facility development, configuration, and validation and verification of large, highly configurable systems [BNR+14].

However, creating and maintaining variability models is laborious and error-prone, given the complexity of constraints. To improve the situation, we need to (i) increase the empirical understanding of such models in large-scale systems, and (ii) conceive approaches to adopt and maintain models, ideally using automated techniques. This talk presents our work addressing these two goals. First, we study real-world models, their underlying languages, and develop tools to make these models available to off-the-shelf reasoners [BSL+13]. Second, we develop static analysis techniques to extract constraints (the main ingredients of variability models) from source code and evaluate the feasibility of automatically reverse-engineering complete variability models [NBKC14, NBKC15].

**Real-World Variability Models.** Conceiving a model reverse-engineering approach requires a corpus of realistic models. Unfortunately, commercial models are hardly available to researchers, due to confidentiality issues. Existing model repositories, such as S.P.L.O.T., contain only small academic models. Thus, we study the (feature-model-like) variability models of twelve industry-strength, open-source projects from the systems software domain, including among others the Linux kernel and the eCos embedded operating system. We analyze and reverse-engineer the formal semantics of their modeling languages and, given their expressiveness, develop propositional abstractions to make these models available to off-the-shelf reasoners (SAT solvers). Our language analysis further reveals that there are intricate semantics and modeling concepts found in practice that are not covered in academic modeling languages and rarely considered in research. Our model analysis identifies a set of large and complex models–ideal candidates to evaluate scalable model reverse-engineering and analysis techniques.

**Mining Configuration Constraints.** We hypothesize that many of the configuration constraints declared in the models arise from low-level source-code restrictions. To investigate this, we develop scalable static analysis techniques to extract constraints from the codebase of highly configurable systems. We use our analysis to study the origin of configuration constraints and to investigate to what extent we can reverse-engineer and consistency-check models from code. Our static analysis relies on two rules to derive constraints. The first rule expresses that every configuration should build correctly; that is, it should pre-process, parse, type-check, and link. The second rule expresses that each valid configuration should yield a lexically unique system; that is, no two configurations lead to exactly the same configured system (i.e., source code). Since a naive approach—analyzing all possible combinations of options—would not scale, we extend an existing variability-aware infrastructure which allows us to analyze all variants of the system at once. Our infrastructure FaRCE (FeatuRe Constraints Extraction) is publicly available [NBKC14].

We apply our analysis on four of our previously investigated highly configurable systems with variability models (Linux kernel, eCos, uClibc, Busybox) to evaluate the accuracy of our approach and to determine which constraints are recoverable from the code. We find that our approach is highly accurate (93 % and 77 % respectively for our two rules) and that we can recover 28 % of existing constraints [NBKC15]. The extracted constraints can then be used to synthesize actual variability models using our previously developed algorithms [SLB+11]. Finally, we qualitatively investigate samples of non-recovered constraints, determining that 15 % of the constraints require further analyses (e.g., data-flow, control-flow, or dynamic analyses), 16 % could not be recovered due to limits of our tooling, and that at least 20 % of the constraints are purely domain knowledge—indicating that creating a complete model requires further substantial domain knowledge and testing.

## References

[BNR+14] Thorsten Berger, Divya Nair, Ralf Rublack, Joanne M. Atlee, Krzysztof Czarnecki, and Andrzej Wasowski. Three Cases of Feature-Based Variability Modeling in Industry. In *MODELS*, 2014.

[BSL+13] Thorsten Berger, Steven She, Rafael Lotufo, Andrzej Wasowski, and Krzysztof Czarnecki. A Study of Variability Models and Languages in the Systems Software Domain. *IEEE Transactions on Software Engineering*, 39(12):1611–1640, 2013.

[NBKC14] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. Mining Configuration Constraints: Static Analyses and Empirical Results. In *ICSE*, 2014.

[NBKC15] Sarah Nadi, Thorsten Berger, Christian Kästner, and Krzysztof Czarnecki. Where do Configuration Constraints Stem From? An Extraction Approach and an Empirical Study. Technical report, GSDLAB-TR 2015-01-27, University of Waterloo, 2015. http://gsd.uwaterloo.ca/sites/default/files/constraints-2015-nadi.pdf.

[SLB+11] Steven She, Rafael Lotufo, Thorsten Berger, Andrzej Wasowski, and Krzysztof Czarnecki. Reverse Engineering Feature Models. In *ICSE*, 2011.