Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems

Mukelabai Mukelabai[1], Damir Nešić[2], Salome Maro[1],
Thorsten Berger[1], and Jan-Philipp Steghöfer[1]

Abstract: Hundreds of dedicated analysis techniques for highly configurable systems have been conceived, many of them able to analyze properties for all possible system configurations. Unfortunately, it is largely unknown whether these techniques are adopted in practice, whether they address actual needs, or which strategies practitioners apply. We present a study [MNM+18] of analysis practices and needs in industry based on surveys and interviews. We confirm that properties considered in the literature (e.g., reliability) are relevant and that consistency between variability models and artifacts is critical, but that the majority of analyses for specifications of configuration options (a.k.a., variability model analysis) is not perceived as needed. We identified pragmatic analysis strategies, including practices to avoid the need for analysis. We discuss analyses that are missing and synthesize our insights into suggestions for future research.

Keywords: Highly configurable systems, software product lines, analysis

Engineering highly configurable systems such as software product lines is challenging due to variability—the number of configurations and system variants grows exponentially with the number of configuration options. Therefore, analysis techniques to validate *general properties* such as reliability or performance, *code properties* such as scattering or nesting, *variability model properties* such as absence of dead options, and *consistency properties* such as absence of dead code are necessary. Due to the need to analyse all possible variants, analysis techniques for highly configurable systems differ from traditional analyses. However, we lack empirical data whether the proposed analyses for highly configurable systems are adopted in practice, whether they address actual needs or find errors, or what analysis strategies are actually applied.

We present the results of a study which relied on a survey with 27 practitioners engineering highly configurable systems and follow-up interviews with 15 of them, covering 18 different companies from eight countries. Our study design relied on categorizing analysis techniques from the literature and identifying properties analyzed by them; we used these to elicit the need for and the severity of analyzing the properties. We also elicited industrial practices. Since it is intrinsically difficult to objectively understand the real practices and map them to the state of research, we triangulated results from the survey and interviews.

[1] Chalmers | Unviversity of Gothenburg, Sweden. mukelabai.mukelabai@gu.se, salome.maro@gu.se, thorsten.berger@gu.se, jan-philipp.steghofer@gu.se

[2] KTH, Stockholm, Sweden. damirn@kth.se

doi:10.18420/se2019-21

Our results show that common properties suggested in the literature are indeed relevant for highly configurable systems. However, most of the variability-model-related analysis properties are not seen as important by our practitioners. The proposed change-impact analyses are not seen as sufficient, because they are confined to the model and its configuration space, not providing holistic insights on impacts on implementation artifacts. Assuring consistencies between artifacts (especially variability model and source code) is considered highly critical, as well as identifying unwanted feature interactions.

We observed (as expected) testing as the dominant practice. Interestingly, the configuration sampling criteria that are necessary for testing primarily rely on experience. Hardly any systematic sampling or random sampling is used. Our results also suggest that the latter are not even applicable given the configuration spaces that would still leave too many irrelevant variants. Furthermore, hardly any formal method is used (apart from limited model checking). Besides testing, manual work, such as code reviews, is exercised, because often the variability models required for more sophisticated analyses do not exist or are not expressed in a form that can be used as an input. The lack of integrated tool chains is also a factor, since artifacts required for performing analyses are managed in different tools. Interestingly, the experience of the developers and rules, such as coding standards, but also engineering practices such as modularization of code, often alleviate the need for sophisticated analyses of the highly configurable system.

These findings suggest a number of research directions to support practical needs. Techniques to *avoid analyses* can help avert the substantial investment often required for more formal approaches. If analyses have to be conducted, they should be *lightweight* and account for the diversity of artifacts and tool chains. This also includes *unifying variability management and modeling concepts*. *Hybrid analysis* approaches that combine manual reviewing with variability-aware analyses allow practitioners to weed out false positives from static analysis approaches and allow focusing on relevant cases. The latter can also be achieved with *experience-based configuration sampling*. In terms of concrete analysis techniques, *automated test-case generation*, *test-case selection based on quality properties* as well as *traceability management* for failed tests can support practitioners in improving coverage and finding issues quickly. Traceability management can also support *change-impact analyses* that are not confined to the variability model. Finally, quicker and more flexible *consistency checking* and *feature-interaction analysis* for large and loosely-coupled systems will reduce the length of feedback cycles and support continuous integration and deployment.

References

[MNM+18] Mukelabai Mukelabai, Damir Nešić, Salome Maro, Thorsten Berger, Jan-Philipp Steghöfer: Tackling combinatorial explosion: a study of industrial needs and practices for analyzing highly configurable systems. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering (ASE'18), September 2018, pp. 155–166, ACM. https://doi.org/10.1145/3238147.3238201