Jens Knoop, Uwe Zdun (Hrsg.): Software Engineering 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 97

Morpheus: Variability-Aware Refactoring in the Wild

Jörg Liebig[1], Sven Apel[2], Andreas Janker[3], Florian Garbe[4], and Sebastian Oster[5]

Abstract: Today, many software systems are configurable with conditional compilation. Just like any software system, configurable systems need to be refactored during their evolution. The inherent variability of configurable systems induces an additional dimension of complexity that is not addressed properly by current academic and industrial refactoring engines. Even simple refactorings, such as RENAME IDENTIFIER, are not handled well by existing refactoring engines and may introduce errors in some variants of the configurable system to be refactored. To improve the state of the art, we propose a variability-aware refactoring approach that relies on a *canonical variability representation* and *variability-aware analysis*. The goal is to preserve the behavior of all variants of the configurable system, without compromising general applicability and scalability. To demonstrate practicality, we developed MORPHEUS, a sound variability-aware refactoring engine for C code with preprocessor directives. We applied MORPHEUS to three substantial real-world systems (*Busybox*, *OpenSSL*, and *SQLite*) showing that variability-aware refactoring is practical (i.e., scalable, sound, and complete) in the presence of conditional compilation.

Keywords: configurable systems, refactoring, preprocessor

For more than 40 years software developers implement configurable software systems in the programming language C using conditional compilation with the C preprocessor CPP. Using preprocessor directives, such as `#ifdef`s, developers write optional and alternative code fragments, which form the basis to tailor a configurable system to different application scenarios and use cases. Many developers are familiar with `#ifdef`s, and practically every software system written in C is configurable with conditional compilation [Li10]. To evolve and maintain software systems, software developers usually rely on refactoring engines as part of integrated development environments, such as ECLIPSE. A refactoring engine is a tool, which (semi-)automatically applies code restructings with the goal to improve the internal code structure while preserving the external program behavior [Me02]. However, evolving and maintaining configurable systems is challenging, as the behavior not only of a single system, but of multiple system variants have to be considered, something which is not adressed properly in current refactoring engines.

To assess the state-of-the-art of refactoring engines for configurable systems, we conducted an empirical study to classify strategies of how existing refactoring engines (industrial, open-source, and academic) handle refactorings for configurable systems [Li15]. Overall, we found that there are five different strategies: code restructurings using standard editor facilities (*find/replace*), applying refactorings to single variants only (*single variant*), applying refactorings to multiple variants in isolation (*variant-based*), support for source code with a

[1] Method Park, joerg.liebig@methodpark.de
[2] University of Passau, apel@fim.uni-passau.de
[3] University of Passau, janker@fim.uni-passau.de
[4] University of Passau, fgarbe@fim.uni-passau.de
[5] Method Park, sebastian.oster@methodpark.de

limited set of `#ifdef` usage patterns in configurable code (*limited patterns*), and involving heuristics to reasons about code restructurings in the presence of preprocessor directives (*heuristics*). All strategies suffer from certain limitations that hinder their applicability in practice: they are error-prone (*find/replace*), are incomplete (*single variant* and *limited patterns*), do not scale (*variant-based*), or are unsound (*heuristics*). For example, even a simple refactoring, such as EXTRACT FUNCTION, does not work properly in the popular development environment ECLIPSE in the presence of `#ifdefs`. In ECLIPSE, which applies a *single-variant* strategy, we observed a different program behavior after code restructurings.

As existing strategies and refactoring engines have serious shortcommings, which hinder their application in practice, we developed our own strategy, called *variability-aware refactoring* [Li15]. The idea is to use variability-aware data-structures and algorithms [Wa14, Li13, Kä11] for this task, i.e., data-structures and algorithms, which incorporate variability in the form of `#ifdefs` directly during code restructurings. To assess the applicability of variability-aware refactoring, we implemented variability-aware versions of three common refactorings (RENAME IDENTIFIER, EXTRACT FUNCTION, and INLINE FUNCTION) as part of our own refactoring engine: MORPHEUS. In experiments with three, real-world case studies (*Busybox*, *OpenSSL*, and *SQLite*), we could show that variability-aware refactoring scales well to practical configurable systems, while preserving the behavior of all variants. The average time for applying a single refactoring is in the order of milliseconds. With variability-aware refactoring, we close a gap in tool-support for configurable software systems and show that, for the first time, a scalable, sound, and complete refactoring engine for C (including preprocessor directives) is possible.

References

[Kä11] Kästner, C.; Giarrusso, P.; Rendel, T.; Erdweg, S.; Ostermann, K.; Berger, T.: Variability-Aware Parsing in the Presence of Lexical Macros and Conditional Compilation. In: Proceedings of the Conference on Object-Oriented Programming, Systems, Languages, and Applications (OOPSLA). ACM, pp. 805–824, 2011.

[Li10] Liebig, J.; Apel, S.; Lengauer, C.; Kästner, C.; Schulze, M.: An Analysis of the Variability in Forty Preprocessor-Based Software Product Lines. In: Proceedings of the International Conference on Software Engineering (ICSE). ACM, pp. 105–114, 2010.

[Li13] Liebig, J.; von Rhein, A.; Kästner, C.; Apel, S.; Dörre, J.; Lengauer, C.: Scalable Analysis of Variable Software. In: Proceedings of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE). ACM, pp. 81–91, 2013.

[Li15] Liebig, J.; Janker, A.; Garbe, F.; Apel, S.; Lengauer, C.: Morpheus: Variability-Aware Refactoring in the Wild. In: Proceedings of the International Conference on Software Engineering (ICSE). ACM, pp. 380–391, 2015.

[Me02] Mens, T.: A State-of-the-Art Survey on Software Merging. IEEE Transactions on Software Engineering, 28(5):449–462, 2002.

[Wa14] Walkingshaw, E.; Kästner, C.; Erwig, M.; Apel, S.; Bodden, E.: Variational Data Structures: Exploring Tradeoffs in Computing with Variability. In: Proceedings of the International Symposium on New Ideas in Programming and Reflections on Software (Onward!). ACM, pp. 213–226, 2014.