Gregor Engels, Regina Hebig, Matthias Tichy (Hrsg.): SE 2023,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2023 39

Classifying Edits to Variability in Source Code – Summary

Paul Maximilian Bittner[1], Christof Tinnes[2], Alexander Schultheiß[3], Sören Viegener[4], Timo Kehrer[5], Thomas Thüm[6]

Abstract: We report about recent research on edit classification in configurable software, originally published at the 30th Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering (ESEC/FSE) 2022 [Bi22]. For highly configurable software systems, such as the Linux kernel, maintaining and evolving variability information along changes to source code poses a major challenge. While source code itself may be edited, also feature-to-code mappings may be introduced, removed, or changed. In practice, such edits are often conducted ad-hoc and without proper documentation. To support the maintenance and evolution of variability, it is desirable to understand the impact of each edit on the variability. We propose the first complete and unambiguous classification of edits to variability in source code by means of a catalog of edit classes. This catalog is based on a scheme that can be used to build classifications that are complete and unambiguous by construction. To this end, we introduce a complete and sound model for edits to variability. In about 21.5 ms per commit, we validate the correctness, relevance, and suitability of our classification by classifying each edit in 1.7 million commits in the change histories of 44 open-source software systems automatically.

Keywords: software evolution, software variability, feature traceability, software product lines, mining version histories

Summary

In configurable software systems, such as the Linux kernel, certain code should only be present in certain *variants* of the software. For instance, parts of the code base may be platform dependent or a feature should only be available to a subset of customers. Maintaining and evolving variability information along changes to source code poses a major challenge for developers. One aspect thereof is keeping track of changes to variable parts of the code base that implement different *features* or feature interactions of the configurable software. While source code itself may be edited, *feature-to-code mappings* may also be introduced, removed, or changed.

Awareness of edits to variability is crucial in the development of configurable software. For instance, edits to software product lines might introduce type errors in certain variants or alter

[1] University of Ulm, Germany, paul.bittner@uni-ulm.de
[2] Siemens AG, München, Germany, christof.tinnes@siemens.com
[3] Humboldt-University of Berlin, Germany, alexander.schultheiss@informatik.hu-berlin.de
[4] University of Ulm, Germany, soeren.viegener@uni-ulm.de
[5] University of Bern, Switzerland, timo.kehrer@unibe.ch
[6] University of Ulm, Germany, thomas.thuem@uni-ulm.de

the set of available variants in an unintended way. In clone-and-own development, where each variant of a software is developed as a separate copy of the software (e. g., using branching or forking), changes to variants have to be tracked to update other variants accordingly. Variation control systems and managed clone-and-own methods inspect edits paired with information on edited features to recover knowledge about variability incrementally or to reintegrate edits to a hidden unified code base. In practice, however, the variability of the code base is often edited ad-hoc and without proper documentation.

To this end, we present a complete, unambiguous, and automatic classification of edits to variability in source code. We first introduce *variation trees* as a formalization for variability in source code. We then introduce *variation diffs* as a formalization for edits to variation trees, thus describing edits to variability in source code. We prove that variation diffs are complete and sound regarding variation trees, meaning that any possible edit to a variation tree is described by a variation diff and that every variation diff represents an actual edit to variation trees. By classifying all structures within variation diffs, we are able to classify all edits to variability in source code. We present a set of edit classes which we prove to be complete and unambiguous on variation diffs.

To validate the suitability and potential for automation of our concepts and classification, we classified about 45 million edits to source code fully automatically. In effectively 70 min, we processed about 1.7 million commits from the histories of 44 open-source software systems. 99.89% of the considered commits were processed in less than a second, making our method feasible in practical scenarios, such as continuous integration. We found that 0.2% of the patches submitted by developers contain syntactically incorrect preprocessor annotations. All other edits were classified and each of our edit classes occurs in practice.

Data Availability

The original publication is publicly accessible with the DOI 10.1145/3540250.3549108. A preprint is also available online at `https://github.com/SoftVarE-Group/Papers/raw/main/2022/2022-ESECFSE-Bittner.pdf`. Our artifact is available on Github (`https://github.com/VariantSync/DiffDetective/tree/esecfse22`) and Zenodo (DOI: 10.5281/zenodo.7110095).

Bibliography

[Bi22] Bittner, Paul Maximilian; Tinnes, Christof; Schultheiß, Alexander; Viegener, Sören; Kehrer, Timo; Thüm, Thomas: Classifying Edits to Variability in Source Code. In: Proc. Europ. Software Engineering Conf./Foundations of Software Engineering (ESEC/FSE). ACM, New York, NY, USA, pp. 196–208, November 2022.