Jens Knoop, Uwe Zdun (Hrsg.): Software Engineering 2016,
Lecture Notes in Informatics (LNI), Gesellschaft für Informatik, Bonn 2016 81

On Facilitating Reuse in Multi-goal Test-Suite Generation for Software Product Lines[1][2]

Malte Lochau[3], Johannes Bürdek[3], Stefan Bauregger[3], Andreas Holzer[4], Alexander von Rhein[5], Sven Apel[5], Dirk Beyer[5]

Abstract: Software testing is still the most established and scalable quality-assurance technique in practice today. However, generating effective test suites remains computationally expensive, consisting of repetitive reachability analyses for multiple test goals according to a coverage criterion. This situation is even worse when it comes to testing of entire software product lines (SPL). An SPL consists of a family of similar program variants, thus testing an SPL requires a sufficient coverage of all derivable program variants. Instead of considering every product variant one-by-one, family-based approaches are variability-aware analysis techniques in that they systematically explore similarities among the different variants. Based on this principle, we propose a novel approach for automated product-line test-suite generation incorporating extensive reuse of reachability information among test cases derived for different test goals and/or program variants. The developed tool implementation is built on top of CPA/TIGER which is based on CPACHECKER. We further present experimental evaluation results, revealing a considerable increase in efficiency compared to existing test-case generation techniques.

Software product line (SPL) engineering aims at developing families of similar, yet well-distinguished software products built upon a common core platform. The commonality and variability among the family members (product variants) of an SPL are specified as *features*. In this regard, a feature corresponds to user-configurable product characteristics within the problem domain, as well as implementation artifacts being automatically composeable into implementation variants. The resulting extensive *reuse* of common feature artifacts among product variants facilitates development efficiency as well as product quality compared to one-by-one variant development. However, for SPLs to become fully accepted in practice, software-quality assurance techniques have to become *variability-aware*, too, in order to benefit from SPL reuse principles. In practice, systematic software testing constitutes the most elaborated and wide-spread assurance technique, being directly applicable to software systems at any level of abstraction. In addition, testing enables a controllable trade-off between effectiveness and efficiency. In particular, white-box test generation consists of (automatically) deriving input vectors for a program under test with respect to predefined *test goals*. The derivation of sufficiently large *test suites* is, therefore, guided by test selection metrics, e.g., structural coverage criteria like *basic block coverage* and *condition coverage* [Be13]. These criteria impose multiple test goals, thus requiring *sets* of test input vectors for their complete coverage [Be13]. In case of mission-/safety-

[1] This is a summary of a full article on this topic that appeared in Proc. FASE 2015 [Bü15].

[2] This work was partially supported by the DFG (German Research Foundation) under the Priority Programme SPP1593: Design For Future – Managed Software Evolution.

[3] TU Darmstadt, Germany
[4] TU Wien, Austria
[5] University of Passau, Germany

critical systems, it is imperative, or even enforced by industrial standards to guarantee a particular degree of code coverage for every delivered product. Technically, automated test input generation requires expensive reachability analyses of the program state space. Symbolic model checking is promising approach for fully automated white-box test generation using counterexamples as test inputs [Be04]. Nevertheless, concerning large sets of complex test goals, scalability issues still obstruct efficient test case generation when being performed for every test goal in separate. This problem becomes even worse while generating test inputs for covering entire product line implementations. To avoid a variant-by-variant (re-)generation of test cases potentially leading to many redundant generation runs, an SPL test-suite generation approach must enhance existing techniques.

In [Bü15], we presented a novel technique for efficient white-box test-suite generation for multi-goal test coverage of product-line implementations. The approach systematically exploits reuse potentials among reachability analysis results by means of similarity among test cases (1) derived for different test goals [Be13], and/or (2) derived for different product variants [Ci11]. The combination of both techniques allows for an incremental, coverage-driven exploration of the state space of entire product lines under test implemented in C enriched with feature parameters. We implemented an SPL test-suite generator for arbitrary coverage criteria on top of the symbolic software model checker CPACHECKER [Be13]. We evaluated our technique considering sample SPL implementations of varying size. Our experiments revealed the applicability of the tool to real-world SPL implementations, as well as a remarkable gain in efficiency obtained from the reuse of reachability analysis results. compared to test suite generation approaches without systematic reuse. As a future work, we plan to improve reuse capabilities by applying multi-property model-checking techniques of CPACHECKER which allows for reachability analyses of multiple test goals in a single run.

Literaturverzeichnis

[Be04] Beyer, Dirk; Chlipala, Adam J.; Henzinger, Thomas A.; Jhala, Ranjit; Majumdar, Rupak: Generating Tests from Counterexamples. In: ICSE. pp. 326–335, 2004.

[Be13] Beyer, Dirk; Holzer, Andreas; Tautschnig, Michael; Veith, Helmut: Information Reuse for Multi-goal Reachability Analyses. In: ESOP, pp. 472–491. Springer, 2013.

[Bü15] Bürdek, Johannes; Lochau, Malte; Bauregger, Stefan; Holzer, Andreas; von Rhein, Alexander; Apel, Sven; Beyer, Dirk: Facilitating Reuse in Multi-Goal Test-Suite Generation for Software Product Lines. In: FASE. Springer, 2015.

[Ci11] Cichos, Harald; Oster, Sebastian; Lochau, Malte; Schürr, Andy: Model-based Coverage-Driven Test Suite Generation for Software Product Lines. In: MoDELS. Springer, pp. 425–439, 2011.