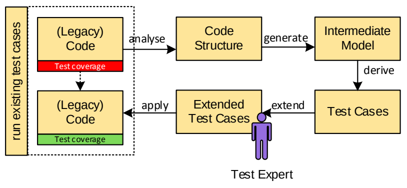
Semi-Automated Test Case Generation for Legacy Code



The generic approach enables a semi-automated generation of test code based on existing legacy code. Results of a source code analysis step include the derivation of an abstract intermediate model as foundation for generating test cases based on equivalence classes and for automated test execution.

Goal

Legacy systems and recently developed source code often include software code without related software tests. However, completing tests afterwards often requires a high effort. The generic approach for semi-automated test case generation supports engineers and developers to provide missing test cases efficient and in a systematic way. These (generated) test cases can also be used in migration projects to evaluate the product quality of migrated products effective and efficient. A language independent code model, i.e., an abstract syntax model, supports different programming languages.

Implementation

Software Quality Lab GmbH and TU Wien developed a prototype for semi-automated test case generation. Based on a simple example, existing software code can be analyzed and an abstract syntax tree can be derived. An equivalence class editor uses the abstract syntax tree to generate test cases, to be executed automatically. Fig. 1 illustrates the basic process of the approach.

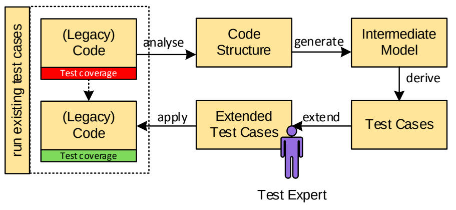


Fig. 1: Basic Process Approach.

Use Case

The developed research prototype focuses on a test automation process (see Fig. 2) including six main process steps:

1. **Parse Code:** Analysis of source code, e.g., in C, and generation of an abstract syntax tree.
2. **Code Classification:** Different code complexity levels (e.g., simple conditions, loops, dependencies, or global variables) require different approaches for automation (currently out of scope of the prototype).
3. **Test Approach:** In context of test approach, the prototype focuses on equivalence classes with tool support.
4. **Test Case Generation** focuses on (a) the generation of Test-Studs and (b) the generation of test code.
5. **Derive Test Code:** In context of the prototype use case, test code will be generated automatically and can be complemented manually (if needed).
6. **Run & Analyze Test Code**: The prototype focuses on using test cases executable in the Eclipse development environment.

The generic approach enables two important application scenarios (S):

- S1: Generation and development of missing tests and, therefore, increase test coverage of legacy code.
- S2: Support of migration projects due to reuse of test cases and the abstract syntax tree of the code.

Technical Specification

- Process for a semi-automated derivation of test cases based on code analysis.
- Consideration of Software Engineering Best-Practices.
- Flexibility and extensibility based on modular and component-oriented architecture.
- Integrated tool chain based on real-world customer use cases.

Key Characteristics

- Support of different test strategies and approaches.
- Improved test cases based on semi-automated generation of test cases complemented with manual test cases to identify also error defects.
- Support of different Unit-Test-Frameworks that could be integrated in the process.

Your Benefits

- Support to systematically construct of currently missing Software Tests.
- Quality assurance support in migration projects based on reusable test cases.
- Application of Best-Practices.
- Language independent modeling of software code (abstract syntax tree)

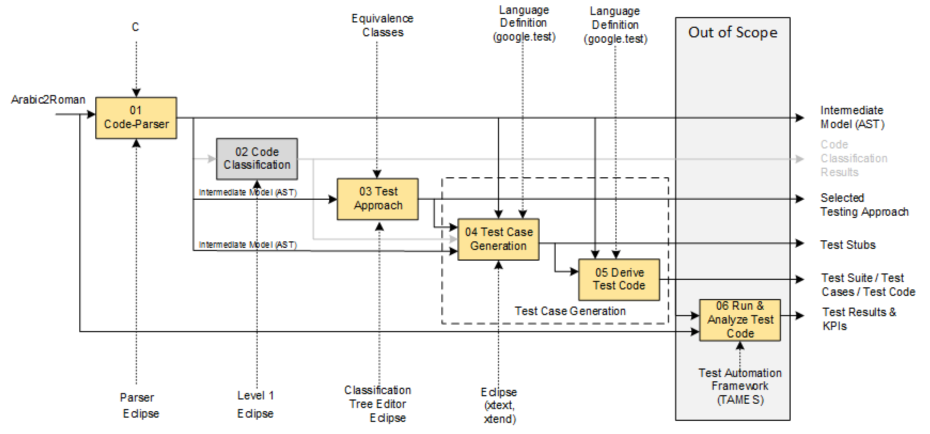


Fig. 2: Process for a semi-automated Generation of Test Cases.

Contact:

Prof. Dr. Stefan Biffl
TU Wien
stefan.biffl@tuwien.ac.at
qse.ifs.tuwien.ac.at

Johannes Bergsmann
Software Quality Lab GmbH
johannes.bergsmann@software-quality-lab.com
www. software-quality-lab.com