Enhancing Precision and Speed in Regression Test Selection by Identifying Safe Code Changes

Master's Thesis

Examiner: Prof. Dr. Alexander Pretschner
Supervisor: Roland Würsching
Email: `roland.wuersching@tum.de`
Starting date: 01.07.2024

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17362
https://www4.in.tum.de

Context

Regression testing is the main quality assurance approach of today, and is a fundamental part of any software development process. It involves re-running test suites after code modifications to make sure the system's functionality has not been affected by these changes. The common approach is to re-run all tests (also known as *RetestAll*) after any code modification to ensure that no issues have been introduced to the system.

However, as the application grows and more extensive tests with larger coverage are added, the time taken to run all tests increases [5]. Throughout development, the tests are usually run multiple times during the implementation of any feature or bug fix, which becomes impractical and time consuming as the system grows. Therefore, multiple techniques were created to find practical ways of selecting a subset of tests to run that would be sufficient to ensure the new code additions did not cause issues to existing code.

Regression test selection (RTS) expedites the regression testing process by only rerunning tests that execute updated code or that may be impacted by recent code changes. On the other hand, some types of code modifications—like adding new classes or methods or rearranging class members—are extremely unlikely to have an impact on test results. It might be feasible to choose pertinent tests considerably more precisely and at very little additional cost by identifying and eliminating these code modifications from the test selection process.

Liu et al. [3] performed an extensive study with the goal of enhancing analysis-based RTS techniques. They created a list of different types of sematics-modifying code changes, that they argue do not require re-running tests that depend only on them, which can be used to make RTS less costly and more precise through manual analysis of 250 revisions of 5 open-source projects. They enhanced the techniques EKSTAZI [1] and STARTS [2] using the results of their analysis and found that the enhanced version of EKSTAZI -called FINEEKSTAZI- reduces the end-to-end regression testing time and the number of selected tests by as much as 55.4% (average: 33.7%) and 80.8% (average: 41.7%). Additionally, FINESTARTS, their enhanced version of STARTS, had an improvement of 60.6% (average: 28.7%) and 71.2% (average: 31.8%).

Furthermore, they compared their enhanced algorithms to state-of-the-art techniques that are based on machine learning (ML) and concluded that currently the performance of analysis-based RTS surpasses that of ML-based RTS. This proves to be a new line of improvement for RTS techniques that would make RTS more feasible for adoption in the industry. However, the findings in this study were based on a small sample -5 projects- and therefore a lot of work can be done to extend their findings and make sure they generalize properly to other projects.

Goal

The goal of this thesis is to develop a new RTS approach with higher precision and cost reduction than current state-of-the-art techniques without compromising safety. This will be performed through the following: **(1)** extending the list created by Liu et al. by identifying additional types of semantics-modifying code modifications that do not affect the test result, and **(2)** investigating additional RTS techniques -like EALRTS [4], HyRTS [6], etc.- to assess the effect of reasoning about our analysis findings in their test selection process.

The identification of these code changes will be performed through the manual analysis of multiple revisions of various open-source projects and extracting these new types of code changes. An extended list of these findings will be created and, after analyzing a number of current RTS techniques, will used to develop the new RTS approach. The performance will then be measured and evaluated against current state-of-the-art RTS techniques.

The success of the newly developed RTS approach will be judged according to the reduction in the number of selected tests and the speed-up of the end-to-end regression testing time while preserving safety.

Working Plan

1. Research: Review the literature about state-of-the-art RTS techniques.
2. Implementation: Manual analysis of open-source projects.
3. Methodology: Extend list of code changes that can enhance RTS algorithms with findings from manual analysis.
4. Implementation: Develop RTS approach based on findings.
5. Evaluation: Run new RTS approach on open-source projects used.
6. Evaluation: Evaluate precision, cost reduction and safety of new approach.
7. Evaluation: Compare the results to state-of-the-art RTS algorithms.

Deliverables

- Source code of the implementation in a form that makes further use possible.
- Final thesis report written in conformance with TUM guidelines.

References

[1] Milos Gligoric, Lamyaa Eloussi, and Darko Marinov. Ekstazi: Lightweight test selection. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 2, pages 713–716, 2015.

[2] Owolabi Legunsen, August Shi, and Darko Marinov. Starts: Static regression test selection. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*, pages 949–954, 2017.

[3] Yu Liu, Jiyang Zhang, Pengyu Nie, Milos Gligoric, and Owolabi Legunsen. More precise regression test selection via reasoning about semantics-modifying changes. In *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ISSTA 2023, page 664–676, New York, NY, USA, 2023. Association for Computing Machinery.

[4] Erik Lundsten. Ealrts: A predictive regression test selection tool, 2019.

[5] Mark Ivey Pooja Gupta and John Penix. Testing at the speed and scale of google. `http://google-engtools.blogspot.com/2011/06/testing-at-speed-andscale-of-google.html`, 2011. Accessed: 2024-06-24.

[6] Lingming Zhang. Hybrid regression test selection. In *Proceedings of the 40th International Conference on Software Engineering*, pages 199–209, 2018.

Fakultät für Informatik
Lehrstuhl 4
Software & Systems Engineering
Prof. Dr. Alexander Pretschner

Boltzmannstraße 3
85748 Garching bei München

Tel: +49 (89) 289 - 17362
https://www4.in.tum.de