Automatically Detecting Flaky End-to-End Tests in Multi-Language Systems Using Differential Coverage

Master's Thesis

Supervisor: Prof. Dr. Alexander Pretschner
Advisor: Fabian Leinen, Roland Würsching
Email: `fabian.leinen@tum.de, roland.wuersching@tum.de`
Starting date: 15.11.2022

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

When testing software, we assume that a test passes if the code under test is free from faults for the input defined in the test and assume that the test fails if not. In other words, we expect a test to be deterministic, which doesn't always hold true. This kind of tests, that fail inconsistently without changes to the code under test or the test itself are called flaky tests [1]. According to a keynote from 2017, almost 16% of all tests at Google are flaky and they spend between 2 and 16% of their compute resources only on re-running flaky tests [2].

The problem of flakiness is particularly significant in User Interface (UI) tests because (1) these types of tests are often quite large and (2) involve many highly asynchronous actions like handling user input and downloading multiple resources required by the interface [3]. Although this type of testing is so problematic, it has not been well researched to date.

Developers of the interactive learning platform Artemis recently complain about frequent flickering of UI tests. As Artemis is a non-commercial project, resources for Continuous Integration (CI) are limited, so the standard approach of rerunning failing flaky tests in the CI is not appropriate. Also, it is assumed that the limited computing power in combination with the asynchronous nature of UI tests is the reason for the high amount of test flickerings. Nevertheless, computing power is available at night and on weekends for the analysis of test flakiness.

Artemis uses the popular tech stack with Java on the backend and JavaScript on the frontend, making it an ideal subject of study for research in this field.

Goal

The goal of this thesis is **(1)** to build a tool to collect coverage information for end-to-end tests and **(2)** to decide if failing tests could be flaky based on the new information and changes.

For **part 1**, a solution has to be found that allows collecting coverage for e2e tests across both the client and server part of Artemis. For this, first existing solutions will be considered and if they should not be usable for our problem, the chairs new approach will be used.

In order to be able to make statements about the scope of the problem for **part 2**, the tests for Artemis will then be extended with the coverage collection solution. After coverage is collected, it can be compared to the changes that triggered the test and an estimation can be given if test fails are due to flakiness [4]. If a test only covers files that where not changed and passed before, it is likely that the test is flaky and fails nondeterministically. The quality of the estimation can then be evaluated by running commits with test fails multiple times to see if the fail was flaky and comparing the result to our estimation.

Working Plan

1. Research: Review the literature about flaky tests.
2. Research: Learn about Cypress[1].
3. Implementation: Setup Artemis locally and run all Cypress tests[2].
4. Implementation: Build a solution to collect coverage information for E2E tests.
5. Implementation: Compare coverage information with latest git changes.
6. Evaluation: Identify relevant commits in Artemis by browsing commits and issues.
7. Evaluation: Apply the tool on relevant commits in Artemis.
8. Evaluation: Compare the results to state of the art literature and developer voices to be able to make a statement about its correctness.
9. Evaluation: Identify limitations of the approach and document them.

[1] `https://www.cypress.io/`
[2] `https://github.com/ls1intum/Artemis/tree/develop/src/test/cypress`

Deliverables

- Source code of the implementation in a form that makes further use possible.
- Technical report with comprehensive documentation of the implementation, i.e. design decision, architecture description, API description and usage instructions.
- Final thesis report written in conformance with TUM guidelines.

References

[1] Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov. 2014. An empirical analysis of flaky tests. In Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering (FSE 2014). Association for Computing Machinery, New York, NY, USA, 643–653. https://doi.org/10.1145/2635868.2635920

[2] The State of Continuous Integration Testing @Google https://research.google/pubs/pub45880/sla

[3] Alan Romano, Zihe Song, Sampath Grandhi, Wei Yang, and Weihang Wang. 2021. An Empirical Analysis of UI-based Flaky Tests. In Proceedings of the 43rd International Conference on Software Engineering (ICSE '21). IEEE Press, 1585–1597. https://doi.org/10.1109/ICSE43902.2021.00141

[4] Bell, Jonathan, Owolabi Legunsen, Michael Hilton, Lamyaa Eloussi, Tifany Yung, and Darko Marinov. 'DeFlaker: Automatically Detecting Flaky Tests'. In Proceedings of the 40th International Conference on Software Engineering, 433–44. Gothenburg Sweden: ACM, 2018. https://doi.org/10.1145/3180155.3180164.

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