A Large-Scale Longitudinal Study of Flaky Tests

Wing Lam[1], Stefan Winter[2], Anjiang Wei[3], Tao Xie[4], Darko Marinov[5], Jonathan Bell[6]

Abstract: Flaky tests that non-deterministically pass or fail without any code changes constitute an impediment to regression testing. To understand when and how flaky tests can be detected most efficiently, we analyzed the commit histories of known flaky tests. We find that 75% of flaky tests are flaky when added, indicating substantial value for developers to run detectors specifically on newly added tests. The percentage of flaky tests that can be detected early increases to 85% when detectors are run on both newly added and directly modified tests.

Keywords: Flaky Tests; Regression Testing

Introduction: Flaky tests are software tests than can both pass and fail without changes to the code under test (CUT) or the test code. They constitute a major impediment to regression testing as their failures do not indicate actual software regressions in the CUT. Such false alarms limit the utility of automated regression tests and adversely affect development productivity, because they can mislead developers to debug failures in the wrong place and prevent the CUT's timely integration with other components of the project. The problem has attracted significant attention in the scientific community (see [Pa21] for an overview).

Because the root causes of flaky tests are manifold [Lu14], a variety of detectors have been proposed to assist with their automatic detection and removal. However, applying these detectors is costly, as they commonly require large numbers of test re-executions. Therefore, it is important to understand whether flaky tests are already flaky by the time they are added to the test code base or whether they only become flaky later as the project evolves. In the former case it would be sufficient to apply flaky test detectors when new tests are added to the test code base, whereas in the latter case detectors would need run periodically as the (test) code base evolves. To answer when flaky tests become flaky, we have conducted an empirical study on a large dataset of projects with known flaky tests.

Study Design: Our study has been based on the iDFlakies dataset of tests that are known to be flaky at a certain commit (*iDFlakies-commit*) in the projects' git histories. For each such flaky test, we performed the following steps (illustrated and summarized in Fig. 1) to determine when in the project history it becomes flaky.

[1] George Mason University, USA
[2] LMU Munich, Germany
[3] Stanford University, USA
[4] Peking University, China
[5] University of Illinois at Urbana-Champaign, USA
[6] Northeastern University, USA

1. As flaky tests can manifest differently in different execution environments, we reconfirmed that the tests are indeed flaky in the iDFlakies-commit. For this purpose we employed two state-of-the-art flaky test detectors, iDFlakies and NonDex, which target different types of flaky tests.
2. In a second step, we made sure that the identified flaky tests can be reproduced as flaky. To achieve a more fine-grained categorization of identified flaky tests, we replaced iDFlakies with two more targeted detectors (Isolation and OBO).
3. We identified the *test introducing commit (TIC)* for each such test and checked whether the test was flaky at that commit.
4. For the tests that we did not identify as flaky at their TIC, we searched the commit history between the TIC and the iDFlakies-commit for the *flakiness introducing commit* (FIC), at which the test first becomes detectable as flaky.

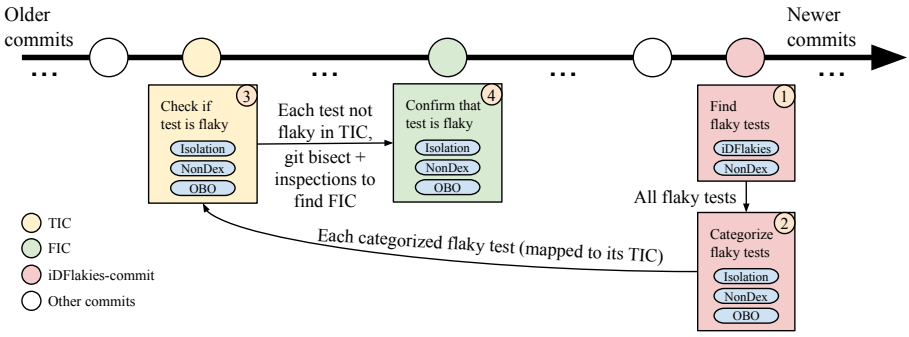

Fig. 1: Overview of the study design.

Results: In the first step of our study we identified 684 flaky tests in their iDFlakies-commit, out of which we were able to confirm and categorize 432. For 245 of these, we were able to compile and run the tests on their TIC and 184 of them (75%) were identified as flaky by detectors in our study. This indicates substantial value for developers to run detectors specifically on newly added tests. For the remaining 61 tests, we identified and inspected their FIC, which revealed that 24 of them become flaky when their test class is altered. Therefore, running detectors on newly added *or directly modified* tests would yield a 85% detection rate for the flaky tests in our study. Running detectors upon test class modifications would detect 8 more flaky tests, but with a median distance of 61 commits or 22 days (median) after their FIC. The remaining 29 flaky tests could be detected 3 commits or 3.6 days after their FIC, but only if detectors run on all tests whenever any test code is changed.

Data Availability: The original article detailing our study [La20] has been published in PACMPL, which is a Gold Open Access journal. The data and data processing scripts from our study are available on a dedicated project website at `https://sites.google.com/view/first-commit-flaky-test`. The data has been integrated in the Illinois Dataset of Flaky Tests (IDoFT) at `http://mir.cs.illinois.edu/flakytests/`.

Bibliography

[La20] Lam, Wing; Winter, Stefan; Wei, Anjiang; Xie, Tao; Marinov, Darko; Bell, Jonathan: A Large-Scale Longitudinal Study of Flaky Tests. Proc. ACM Program. Lang., 4(OOPSLA), 2020.

[Lu14] Luo, Qingzhou; Hariri, Farah; Eloussi, Lamyaa; Marinov, Darko: An Empirical Analysis of Flaky Tests. In: Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering. FSE 2014, pp. 643–653, 2014.

[Pa21] Parry, Owain; Kapfhammer, Gregory M.; Hilton, Michael; McMinn, Phil: A Survey of Flaky Tests. ACM Trans. Softw. Eng. Methodol., 31(1), oct 2021.