Coping Strategies for Unreliable Regression Tests

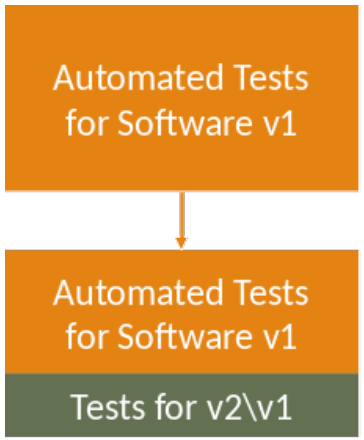
Stefan Winter

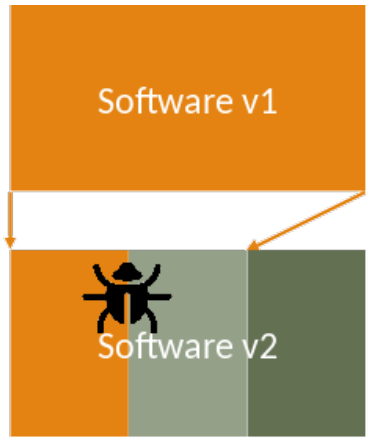
sw@stefan-winter.net
LMU Munich

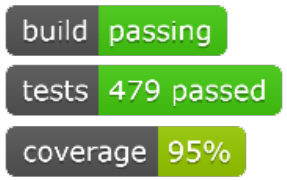
Slides Available

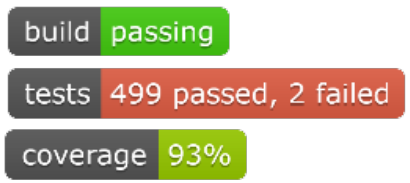
https://www.stefan-winter.net/presentations/flaky_tests_coping.html

Regression Testing

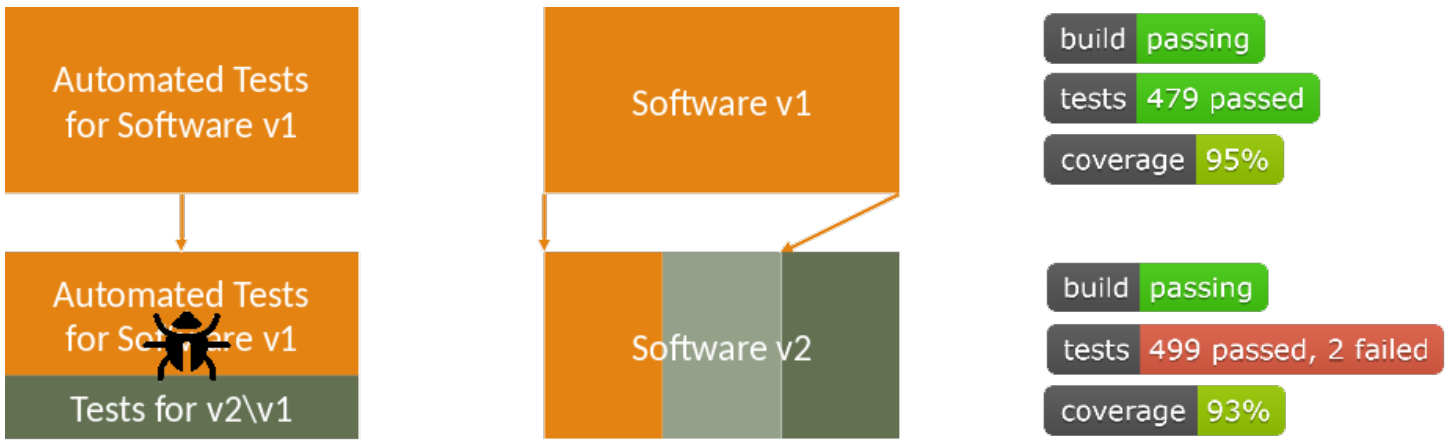








Regression Testing



Flaky tests: *Non-deterministic* failures of automated regression tests in continuous integration pipelines that are not caused by software regressions.

- Waste developer time
- Block and delay releases
- Diminish trust in testing

Comprehensive root cause analyses (Luo et al. 2014; Gruber et al. 2021; Hashemi, Tahir, and Rasheed 2022)

Flaky Tests

Example: `libkdumpfile`

```
1 #! /bin/sh
2
3 name=xlatmap
4 resultfile="out/${name}.result"
5 expectfile="$srcdir/$name.expect"
6
7 echo -n "Checking... "
8 ./xlatmap >"$resultfile"
```

Excerpt from:

https://github.com/ptesarik/libkdumpfile/blob/c54a90c2756e0ca7f9b45662ad3c987403ee7360/tests/xlatmap-check

Test Interference

Example: `libkdumpfile`

```
 1 #! /bin/sh
 2
 3 name=xlatmap
 4 resultfile="out/${name}.result"
 5 expectfile="$srcdir/$name.expect"
 6
 7 mkdir -p out
 8
 9 echo -n "Checking... "
10 ./xlatmap >"$resultfile"
```

Excerpt from accepted fix:

https://github.com/ptesarik/libkdumpfile/blob/e6c5fde6ac7201185292539bef7203c9618ac773/tests/xlatmap-check

Flaky Test Coping Strategy in Industry: Rerun

- Google: 10x
- Mozilla: 10x + 5x + “CHAOS_MODE”
- Spotify: 3-5x with coverage instrumentation
- Dropbox Athena: Mark tests in pre-submit, re-run in post-submit
- Microsoft Azure: Configurable test/pipeline reruns

If detected: Proceed with integration (no regression),

skip execution of test in the future

Viability of the Strategy

(Wendler and Winter 2024)

1. Select Java projects from IDoFT with flaky tests that have known *flakiness-introducing commit (FIC)* (Lam et al. 2020)
2. Run test suites from FIC to iDFlakies-commit & repeat 30 times

Result: *5 flaky tests in the study reveal regressions in the commit history*
→ Quarantining tests diminishes test suite power
→ Better approaches than “flag + skip” desirable

Flaky Test Coping Strategies in Academia

- Detection and repair: costly, but massive improvements with recent work (Eder and Winter 2024)
- Avoidance: (Silva et al. 2024)
- ~~ML-based prediction~~ (tests can be both flaky and regression-revealing (Wendler and Winter 2024))

Detection and Repair

- Requires hypothesis on root cause
- Systematic search for sensitivity to root cause
  - Order dependencies (Lam et al. 2019)
  - Implementation dependencies (unordered collections, etc.) (Shi et al. 2016)
  - Resource dependencies (Silva, Teixeira, and d'Amorim 2020; Silva et al. 2024)

Order Dependencies (OD)

	"intermit"	"flak"	Total
All commits	859	270	1,129
Commits about flaky tests	708	147	855
LDFFT commits	399	87	486
Inspected commits	167	34	201
Async Wait	62	12	74
Concurrency	26	6	32
Test Order Dependency	14	5	19
Resource Leak	9	2	11
Network	10	0	10
Time	5	0	5
IO	4	0	4
Randomness	2	2	4
Floating Point Operations	2	1	3
Unordered Collections	1	0	1
Hard to classify	34	6	40

Source: (Luo et al. 2014)

Order Dependencies (OD)

Root Cause	relative	total
Infrastructure	28 %	2 158
Test Order Dependency	59 %	4 461
victims		3 168
brittles		738
could not be analyzed		555
Non-order-dependent	13 %	952
sample		100
Network		42
Randomness		37
IO		7
Time		4
Async Wait		3
Concurrency		3
Resource Leak		2
Test Case Timeout		1
Unordered Collections		1
Too Restrictive Range		0
Platform Dependency		0
Test Suite Timeout		0

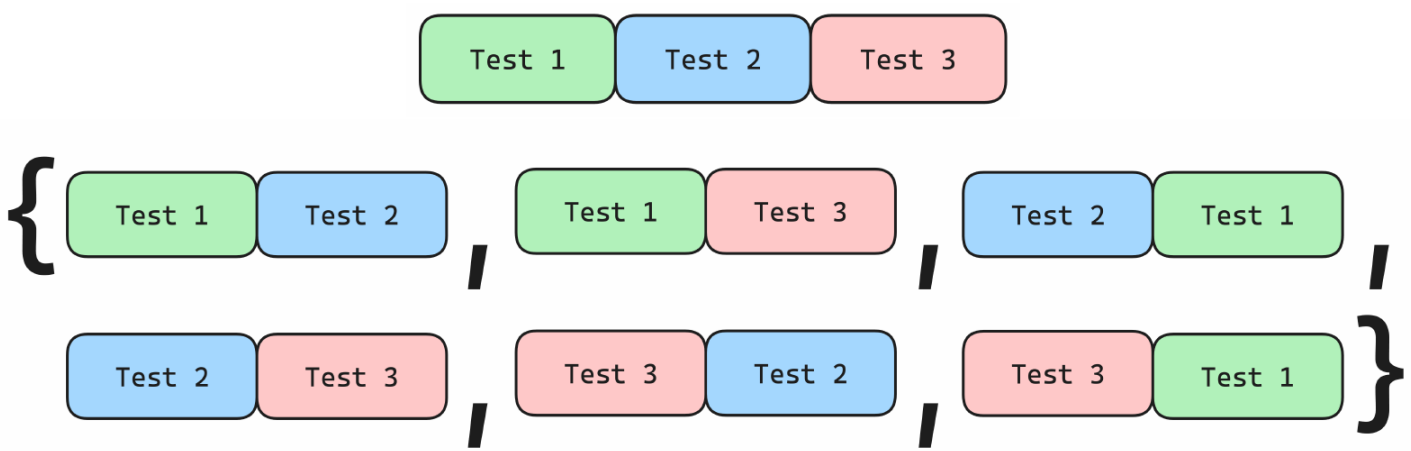
Source: (Gruber et al. 2021)

OD Detection Overhead

- Complete detection: Run all test suite permutations ($n!$)
  - `libkdumpfile` has 184 tests
  - 2.2×10^{338} test suite permutations
  - 22s per test suite run → 4.9×10^{339}s
    (estimated age of universe: 4.3×10^{17}s)

OD Detection Overhead

- Empirical results: Pairwise permutations mostly suffice $(n \cdot (n-1))$ (Zhang et al. 2014; Shi et al. 2019)
  - Factorial down to quadratic complexity
  - 33,672 test pair runs and > 2h for `libkdumpfile`
  - Still too long to run in CI

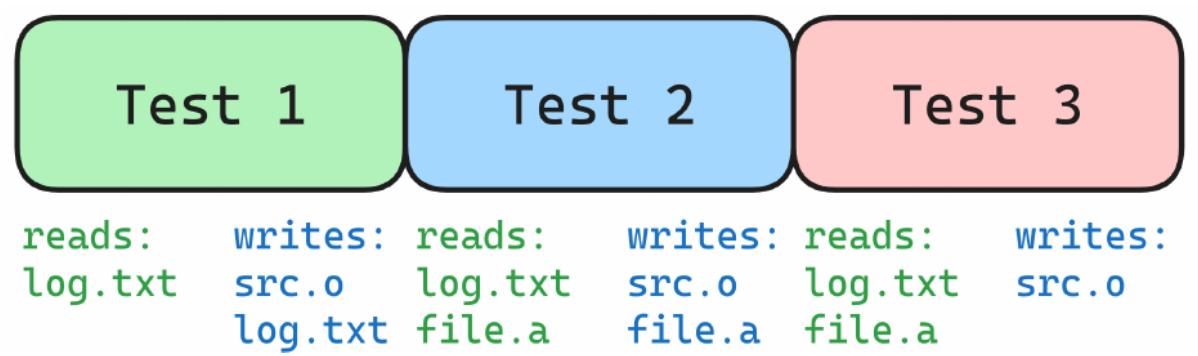


Reducing OD Detection Cost

(Eder and Winter 2024)

Insight: No shared resource access → no order dependency

Idea: Run every test once and record access rights on files, sockets, …

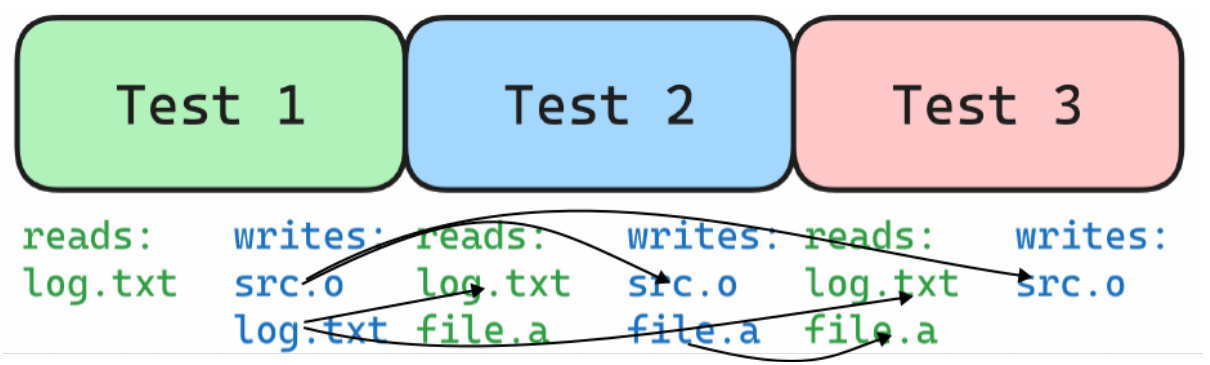


Reducing OD Detection Cost

(Eder and Winter 2024)

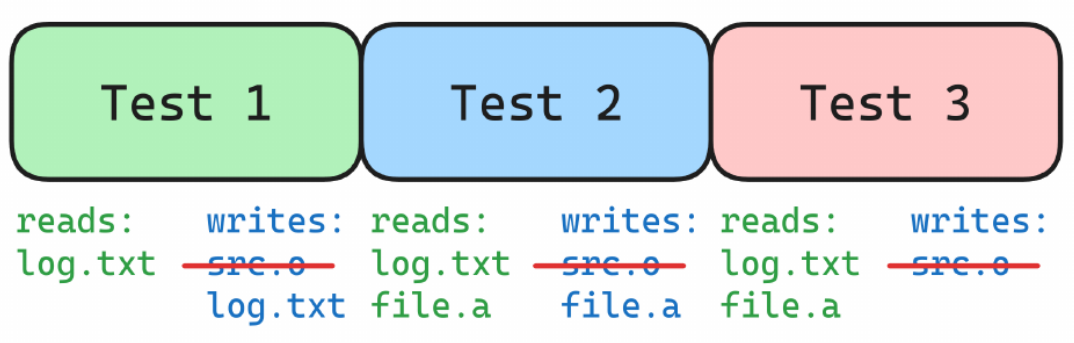
Insight: No shared resource access → no order dependency

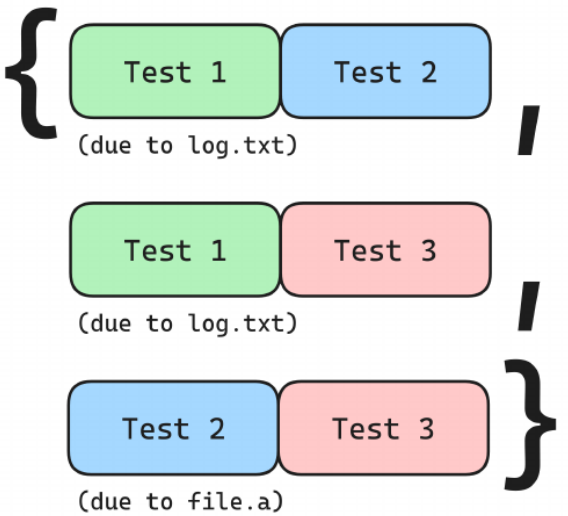
Idea: Run every test once and record access rights on files, sockets, …



Reducing OD Detection Cost

File Descriptors Filtered (FDF)

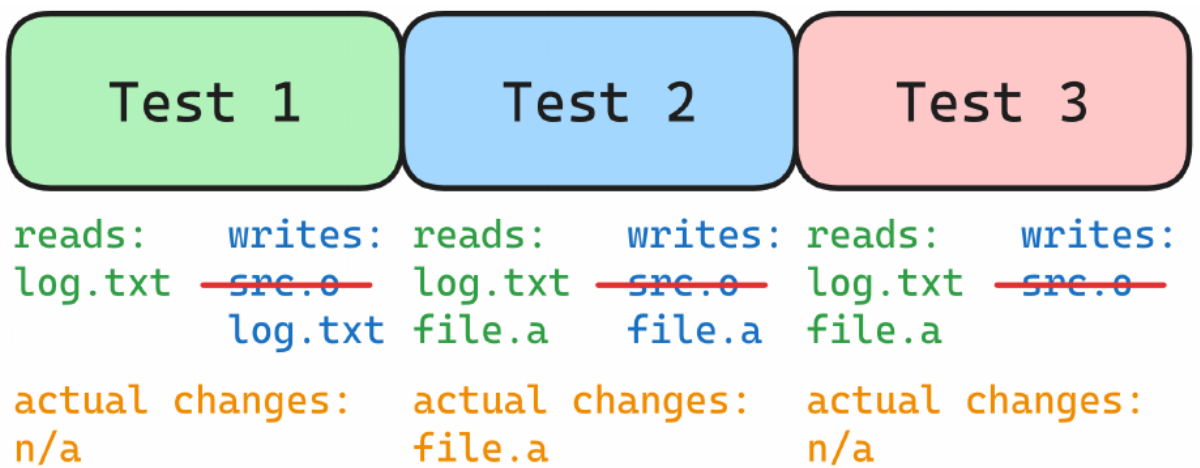




Reducing OD Detection Cost

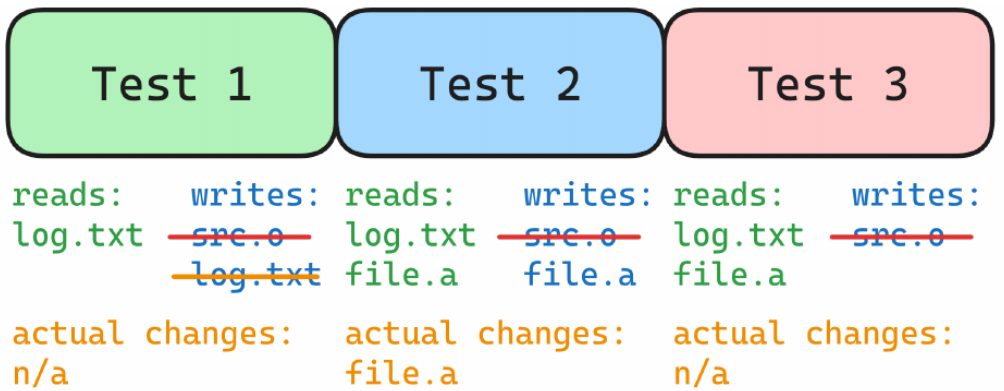
Overlay FS + FDF (OFSFDF)

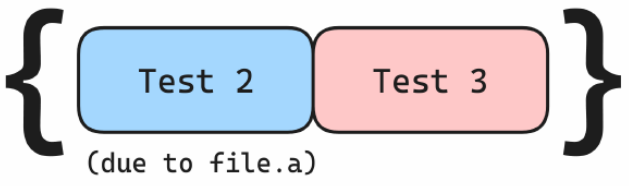
- **Insight:** Not every write *permission* leads to an actual change
- **Idea:** Snapshot filesystem before/after test run



Reducing OD Detection Cost

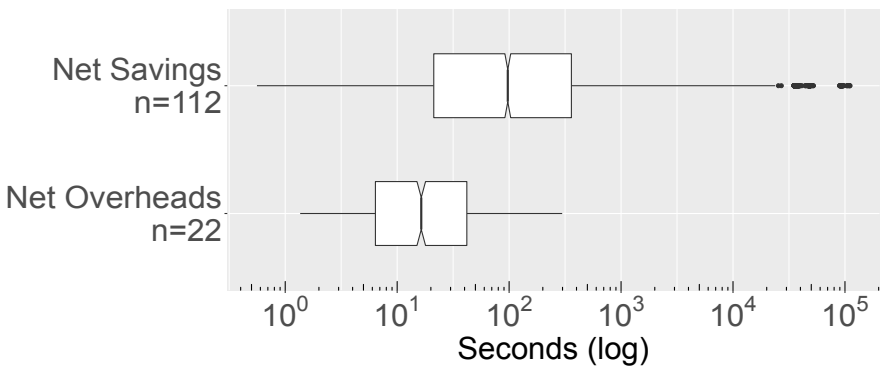
Overlay FS + FDF (OFSFDF)

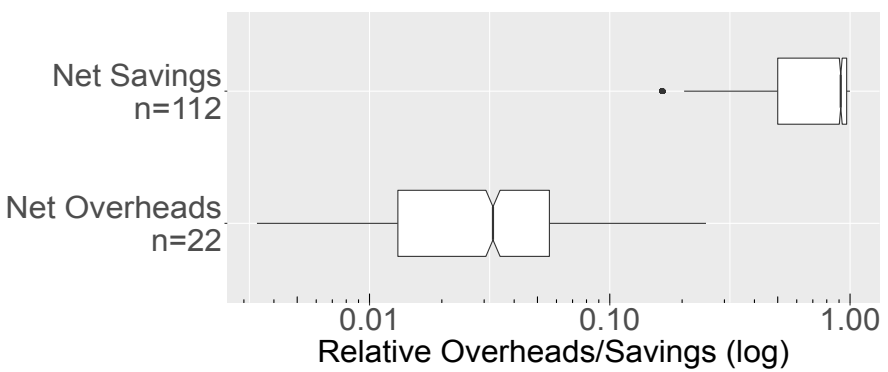




Results

Test Order Reductions





`libkdumpfile`:

33,672 test pair runs and > 2h

→

4 test pair runs and < 1s

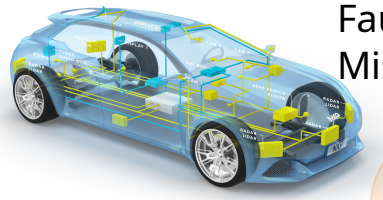
Summary

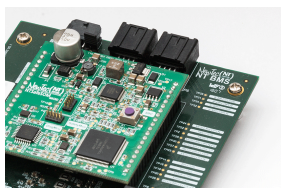
Flaky tests threaten regression testing.
Coping strategies:

- Rerun + skip (considered harmful (Wendler and Winter 2024))
- Prediction (considered imprecise (Wendler and Winter 2024))
- Avoidance (discussed in (Silva et al. 2024))
- Detection + repair (costly, but recently improved (Eder and Winter 2024))

Research Overview

Research focus: Software Dependability, Software Testing, Reproducibility





Fault Injection Tests for Mission-Critical Software

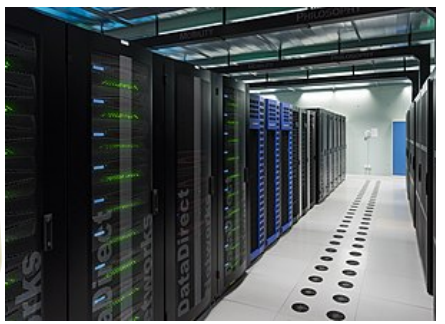
[ICSE'11] [ASE'17] [DSN'12] [ISORC'15] [DSN'13] [EDCC'15] [ISSTA'14] [TSE'20]

Research Artifacts

[FSE'20] [FSE'22]

SW Assessment Methodology

[ICSE'15] [ICST'17] [DSN'20] [ASE'24] [PRDC'18] [ISSRE'20] [FTW'24] [ISSRE'19] [OOPSLA'20] [EASE'25] [ISSTA'19] [TSE'24]



Application domains:

- Automotive, CPS, robotics
- Operating systems
- Automated verifiers

Collaboration Opportunities

Testing and reproducibility are cross-cutting concerns in CS.

Technology-induced relations:

- Robotics and embedded systems (Prof. Henrich)
- Parallel and distributed systems (Prof. Rauber)

Critical application domains:

- Ambient Assisted Living & Medical Assistance Systems (Prof. Leutheuser)
- Cybersecurity (Prof. Roth)

Reproducibility and research software: Prof. Koschmider (information systems)

References

Eder, Florian, and Stefan Winter. 2024. “Efficient Detection of Test Interference in C Projects.” In *Proceedings of the 39th IEEE/ACM International Conference on Automated Software Engineering (to Appear)*. ASE ’24. New York, NY, USA: Associatioan for Computing Machinery. https://doi.org/10.1145/3691620.3694995.

Gruber, Martin, Stephan Lukasczyk, Florian Kroiß, and Gordon Fraser. 2021. “An Empirical Study of Flaky Tests in Python.” In *2021 14th IEEE Conference on Software Testing, Verification and Validation (ICST)*, 148–58. https://doi.org/10.1109/ICST49551.2021.00026.

Hashemi, Negar, Amjed Tahir, and Shawn Rasheed. 2022. “An Empirical Study of Flaky Tests in JavaScript.” In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 24–34. https://doi.org/10.1109/ICSME55016.2022.00011.

Lam, Wing, Reed Oei, August Shi, Darko Marinov, and Tao Xie. 2019. “iDFlakies: A Framework for Detecting and Partially Classifying Flaky Tests.” In *2019 12th IEEE Conference on Software Testing, Validation and Verification (ICST)*, 312–22. https://doi.org/10.1109/ICST.2019.00038.

Lam, Wing, Stefan Winter, Anjiang Wei, Tao Xie, Darko Marinov, and Jonathan Bell. 2020. “A large-scale longitudinal study of flaky tests.” *Proc. ACM Program. Lang.* 4 (OOPSLA). https://doi.org/10.1145/3428270.

Luo, Qingzhou, 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*, 643–53. FSE 2014. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2635868.2635920.

Shi, August, Alex Gyori, Owolabi Legunsen, and Darko Marinov. 2016. "Detecting Assumptions on Deterministic Implementations of Non-deterministic Specifications." In *2016 IEEE International Conference on Software Testing, Verification and Validation (ICST)*, 80–90. https://doi.org/10.1109/ICST.2016.40.

Shi, August, Wing Lam, Reed Oei, Tao Xie, and Darko Marinov. 2019. "IFixFlakies: A Framework for Automatically Fixing Order-Dependent Flaky Tests." In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 545–55. ESEC/FSE 2019. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3338906.3338925.

Silva, Denini, Martin Gruber, Satyajit Gokhale, Ellen Arteca, Alexi Turcotte, Marcelo d'Amorim, Wing Lam, Stefan Winter, and Jonathan Bell. 2024. "The Effects of Computational Resources on Flaky Tests." *IEEE Transactions on Software Engineering* 50 (12): 3104–21. https://doi.org/10.1109/TSE.2024.3462251.

Silva, Denini, Leopoldo Teixeira, and Marcelo d'Amorim. 2020. "Shake It! Detecting Flaky Tests Caused by Concurrency with Shaker." In *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, 301–11. https://doi.org/10.1109/ICSME46990.2020.00037.

Wendler, Philipp, and Stefan Winter. 2024. "Regression-Test History Data for Flaky-Test

Research.” In *Proceedings of the 1st International Workshop on Flaky Tests*, 3–4. FTW ’24. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/3643656.3643901.

Zhang, Sai, Darioush Jalali, Jochen Wuttke, Kıvanç Muşlu, Wing Lam, Michael D. Ernst, and David Notkin. 2014. “Empirically Revisiting the Test Independence Assumption.” In *Proceedings of the 2014 International Symposium on Software Testing and Analysis*, 385–96. ISSTA 2014. New York, NY, USA: Association for Computing Machinery. https://doi.org/10.1145/2610384.2610404.