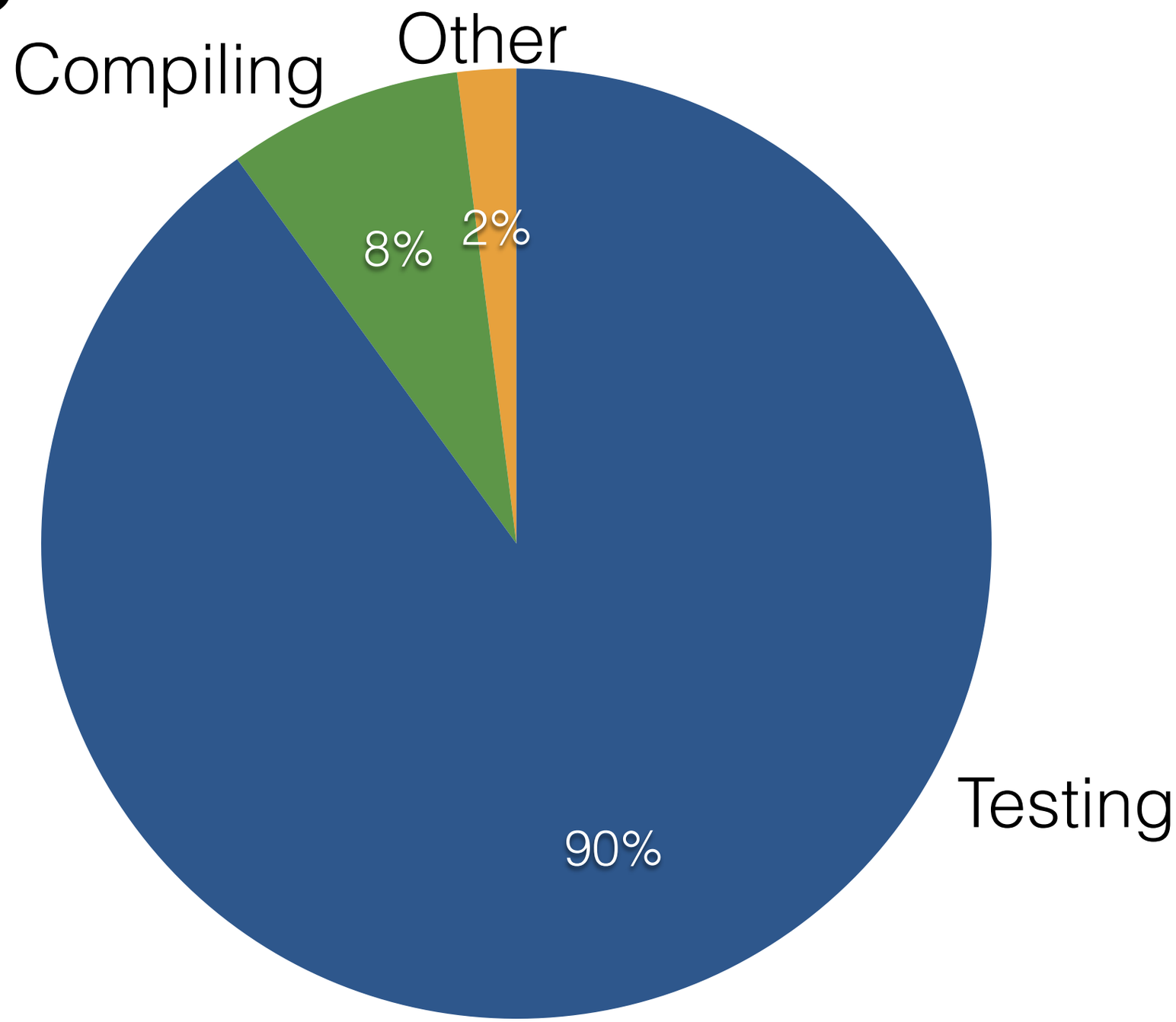
Detecting and Debugging Flaky Tests

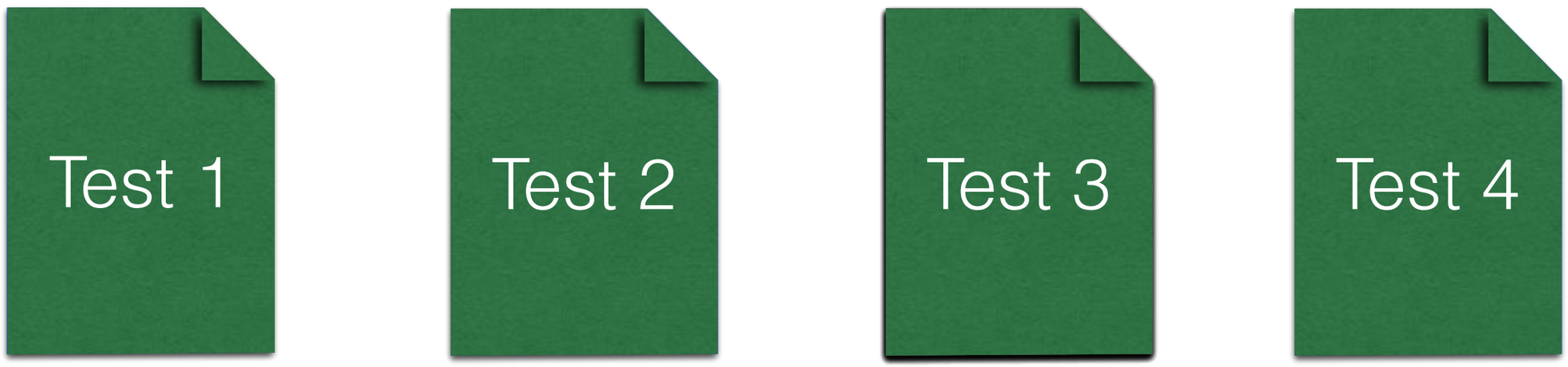
Prof Jonathan Bell (http://jonbell.net/)
George Mason University, USA
bellj@gmu.edu
@_jon_bell_

Testing Dominates Build Times

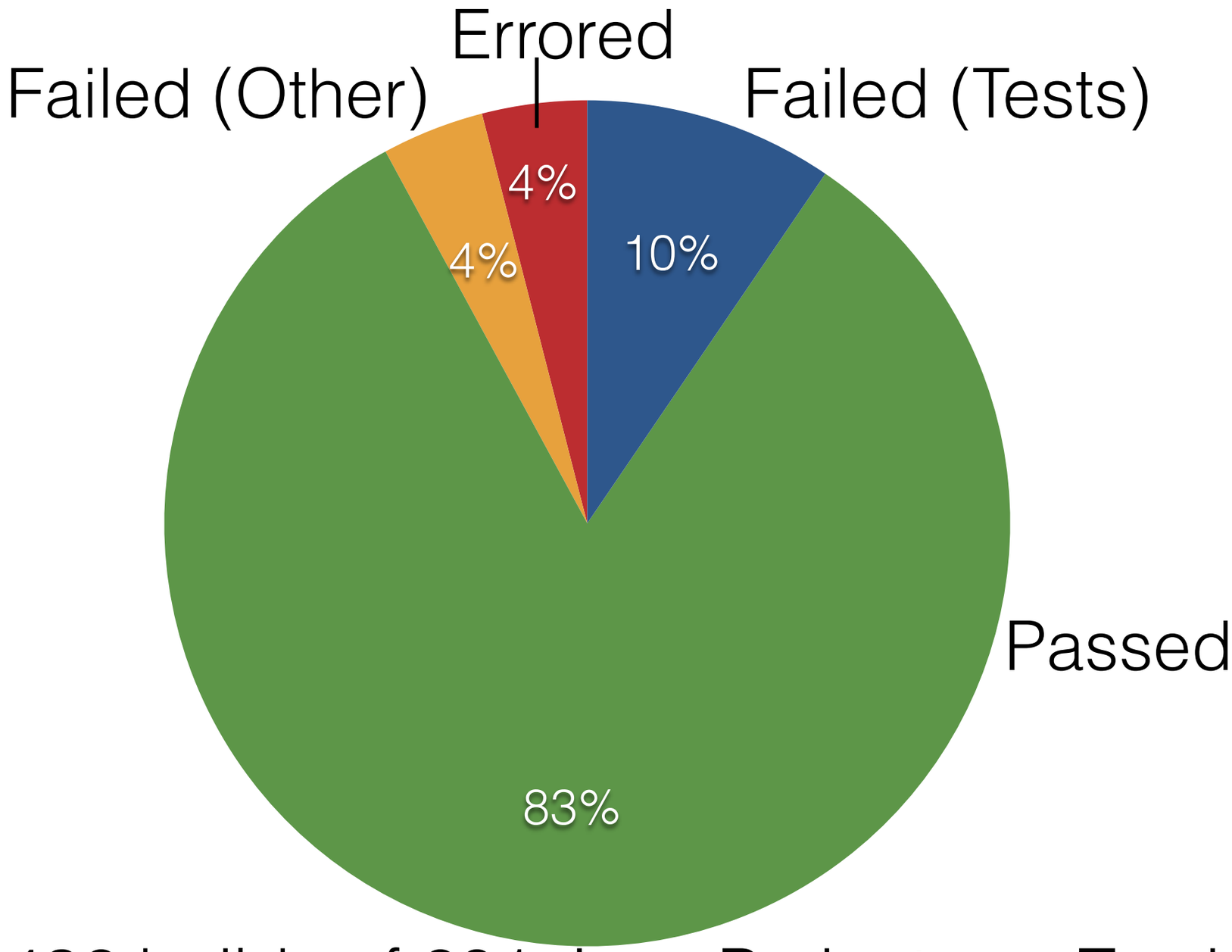


Projects taking > 1 hour to build on GitHub using Maven

Flaky Tests



Flaky Tests Fail Builds

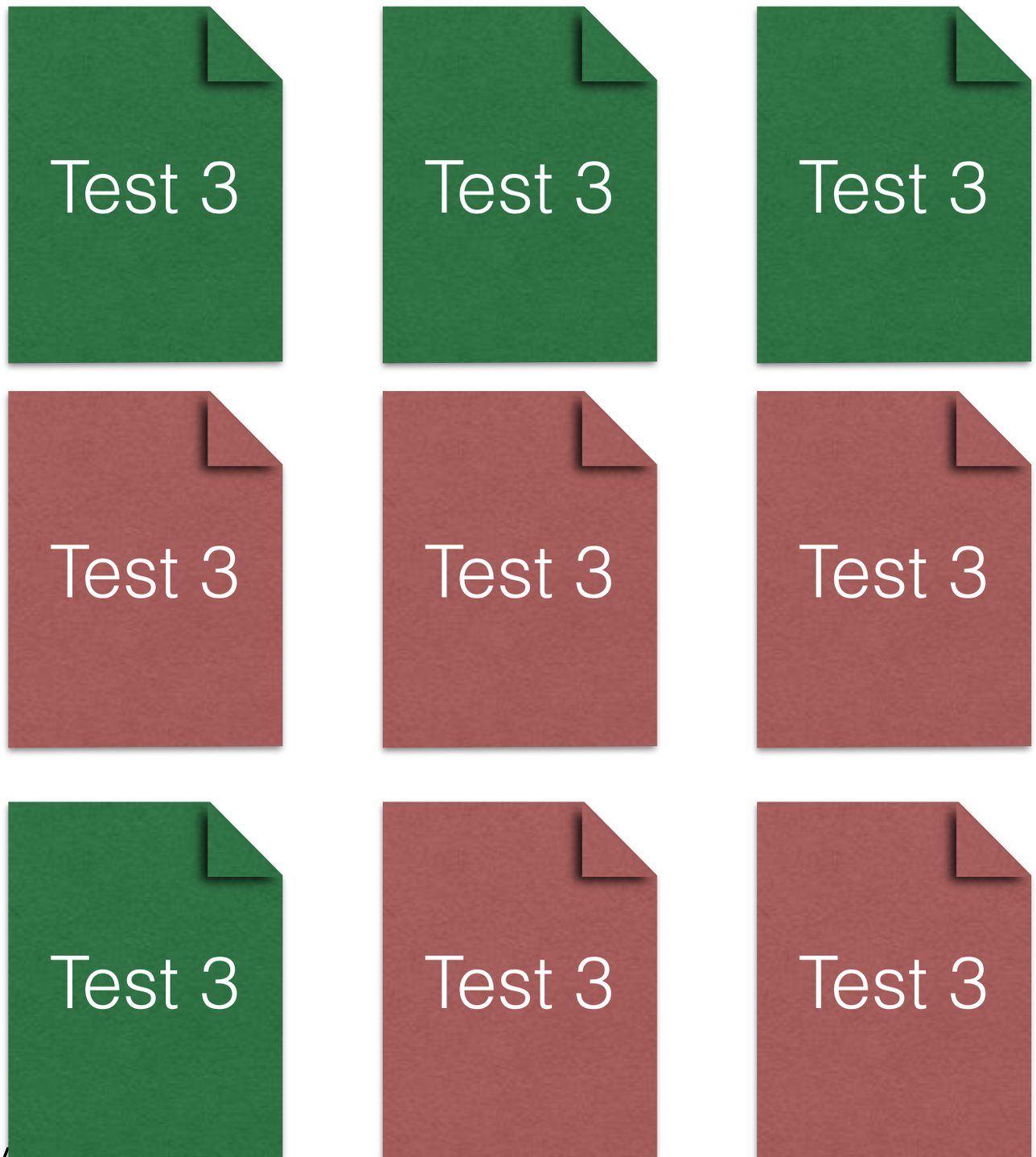


8,432 builds of 201 Java Projects on Travis CI
[Beller, Gousios and Zaidman '15]

Flaky Tests

- Test might pass or fail given the SAME code
- Google: 16% of tests are “flaky” in some way
- How do you handle these flaky tests?
  - Typical fix: if you think something is flaky, run it again and again - outcome is only decided from the complete status

Flaky Tests



"Test is OK!"

"Test failed!"

"Test outcome is unknown!"

Proactively Detecting Flaky Tests

- If we can identify which tests are likely to be flaky, then we can alert developers
- The best flaky test is the one that you find before it ever fails!
- How do we find flaky tests, before they fail?
- Many different causes of flaky tests, one cause we investigated in this work: test order dependencies

Fork me on Github

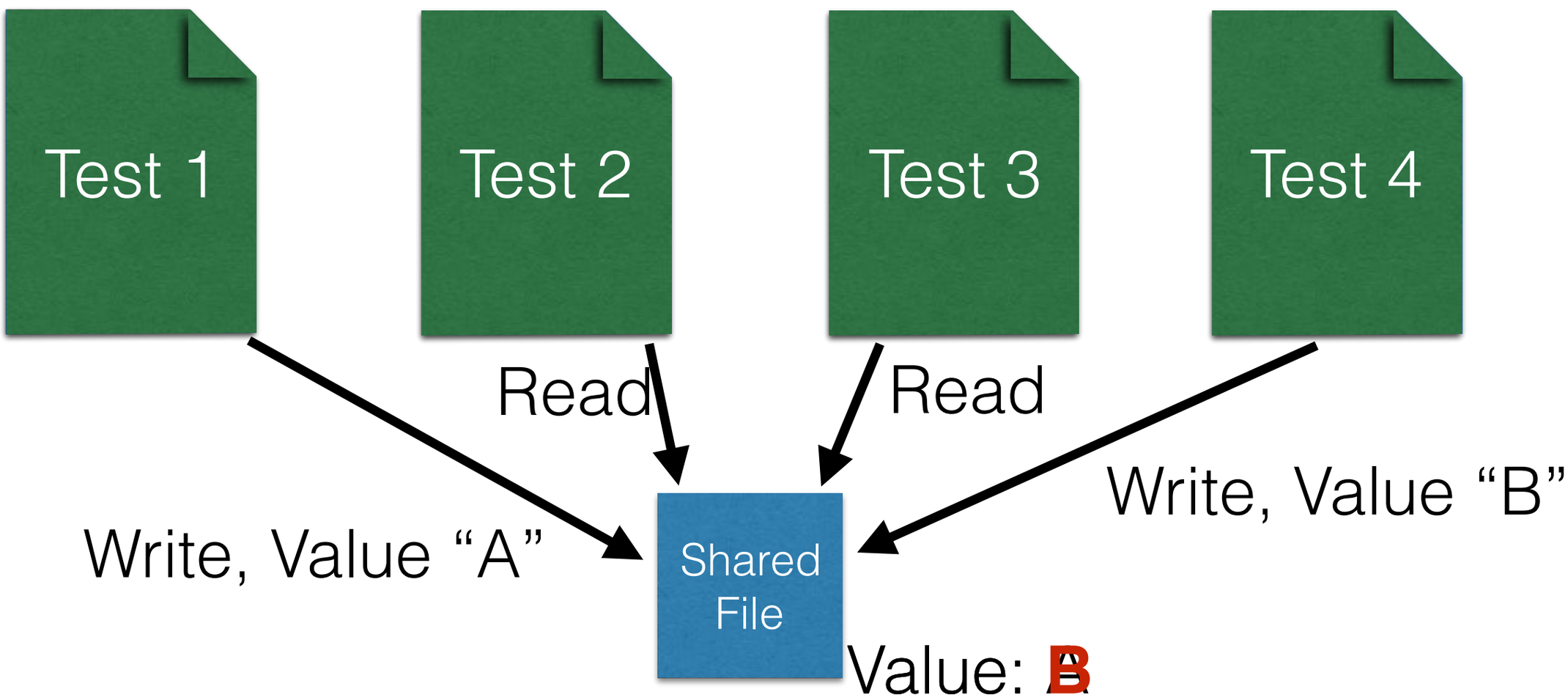
Practical Test Dependency Detection

Alessio Gambi, **Jonathan Bell**, Andreas Zeller
Passau University, George Mason University, Saarland University

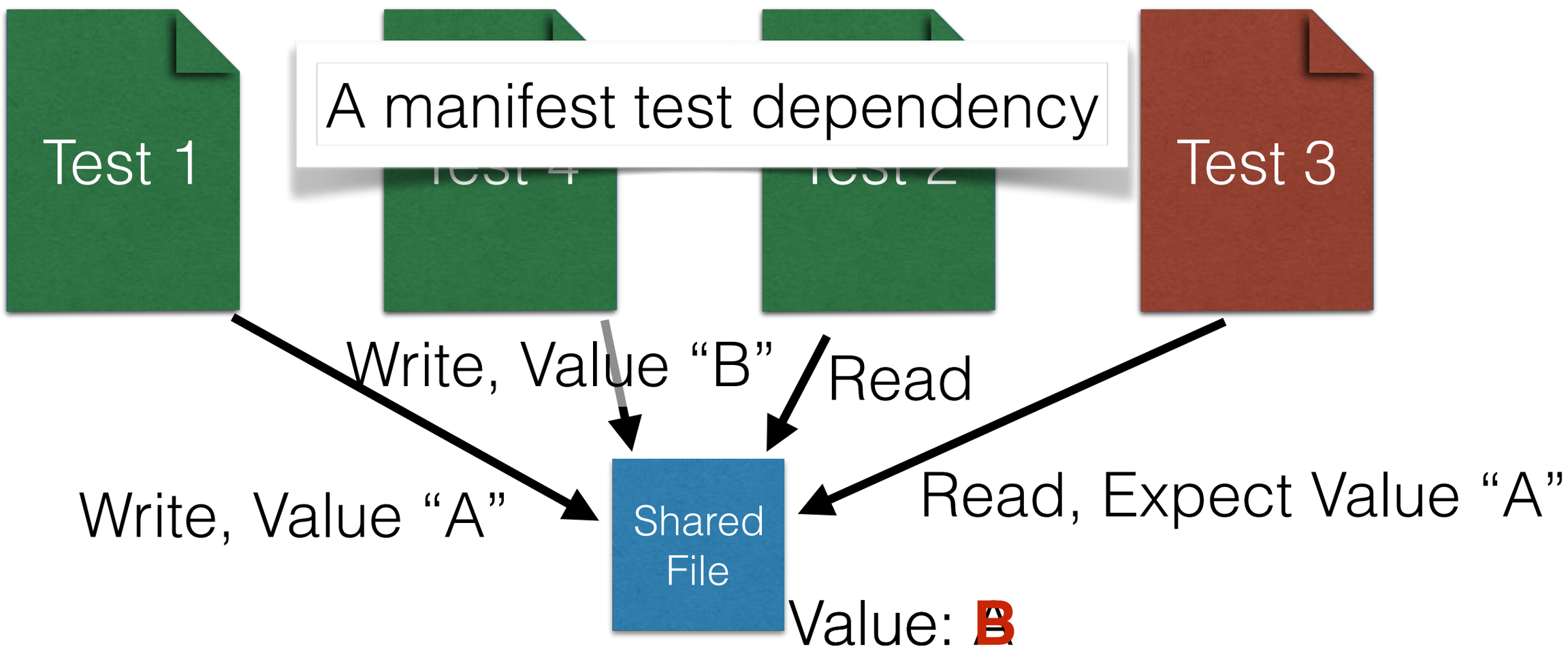
[ICST 2018, talk tomorrow at 11:00am, Research Track 1]



Test Dependencies



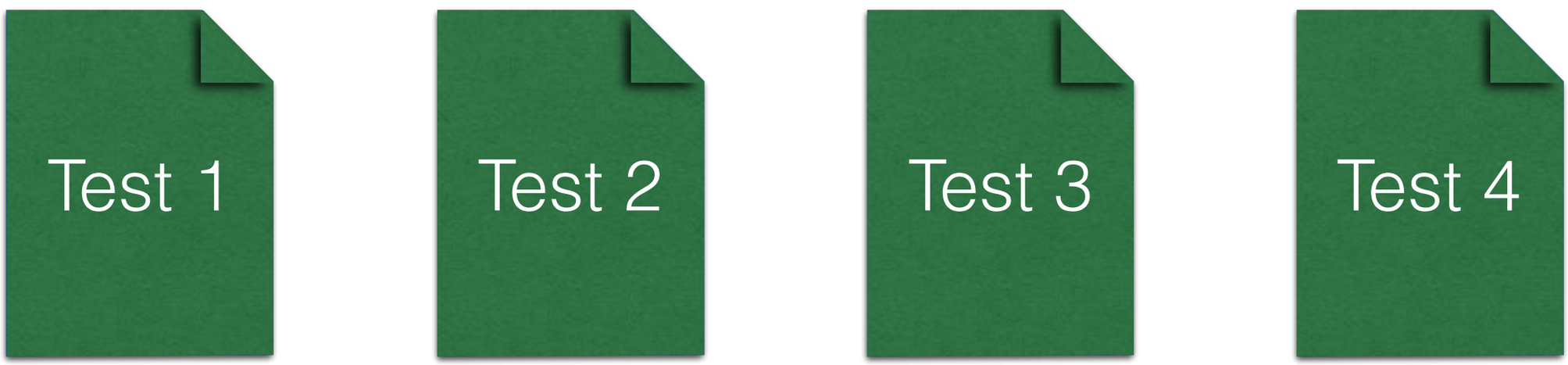
Test Dependencies



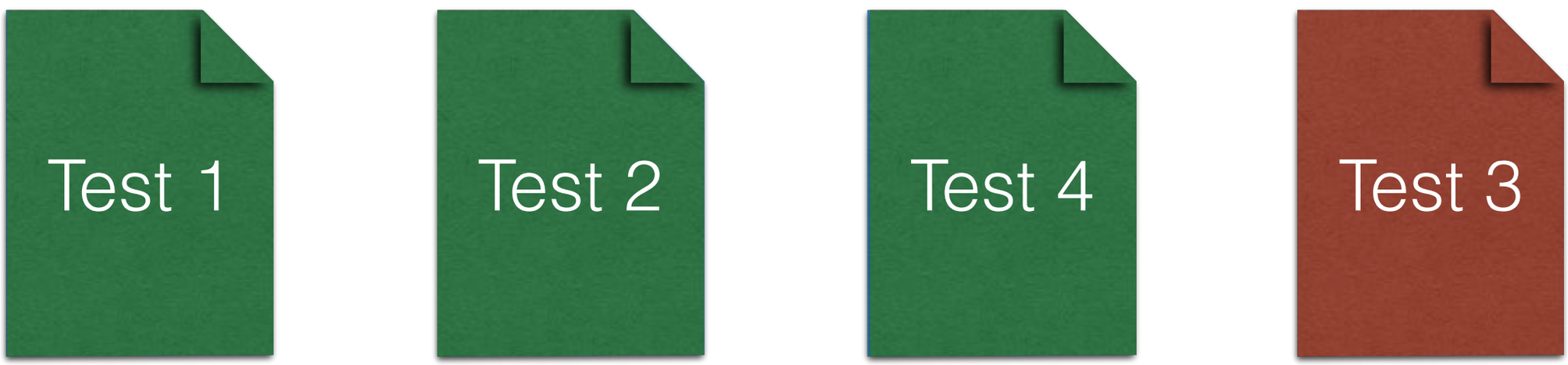
Test Dependencies

- Really exist in practice (Zhang et al. found 96, Luo et al. found 14), lead to **flaky tests**
- Existing techniques to detect:
  - Combinatorially run tests, precise, but slow [Zhang, et al ’14]
  - Run tests once, collect data dependencies: fast, imprecise [Bell, et al ’15]

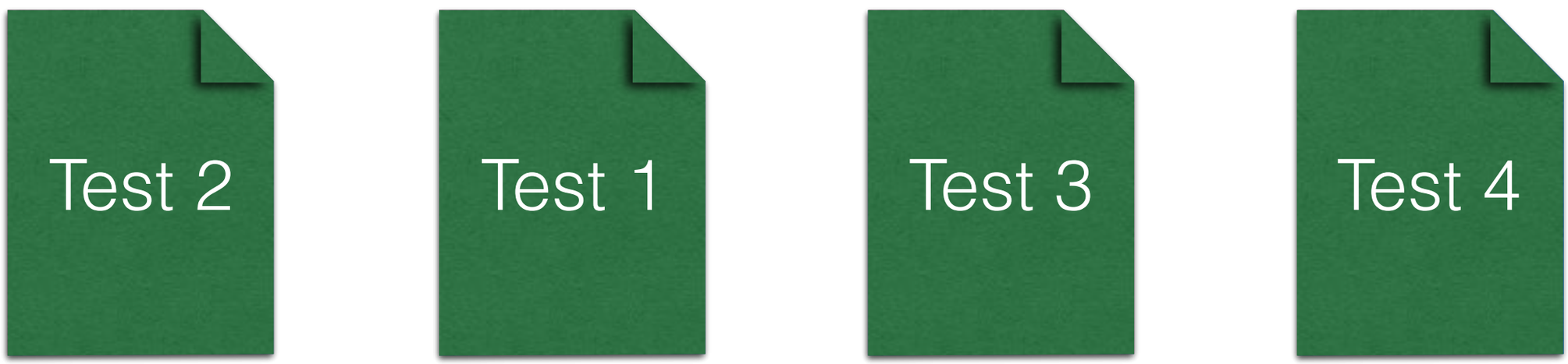
Combinatorial Dependency Detection



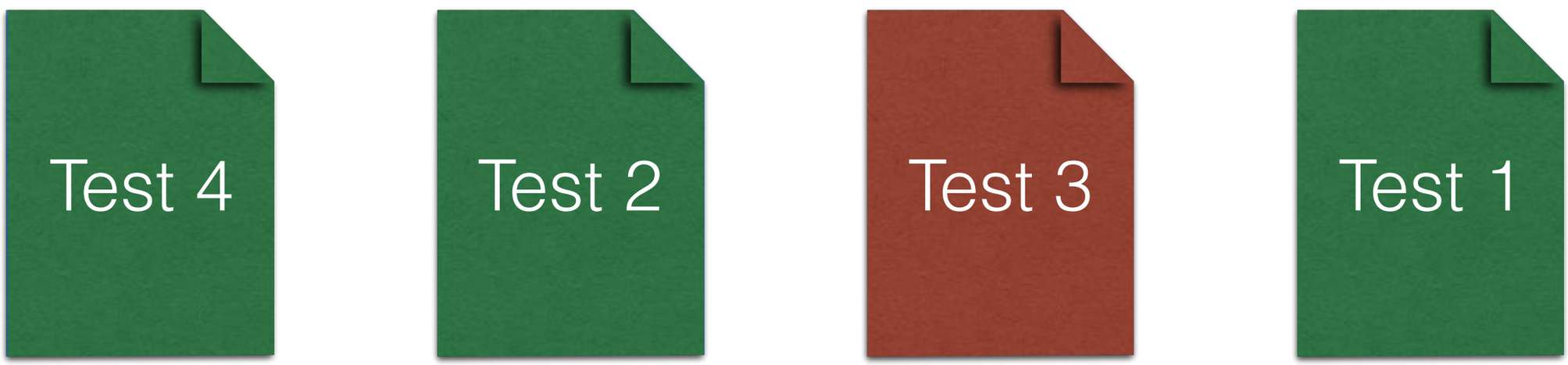
Combinatorial Dependency Detection



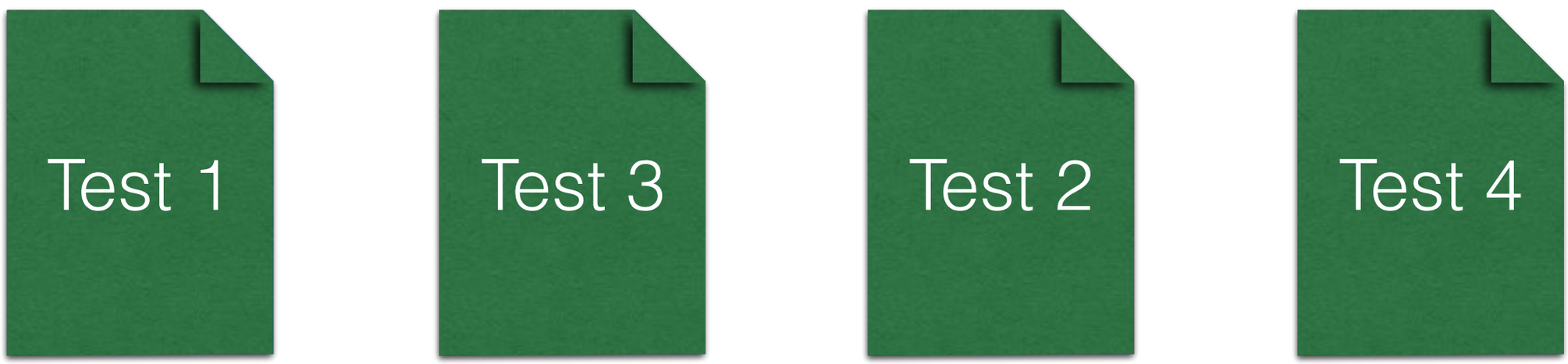
Combinatorial Dependency Detection



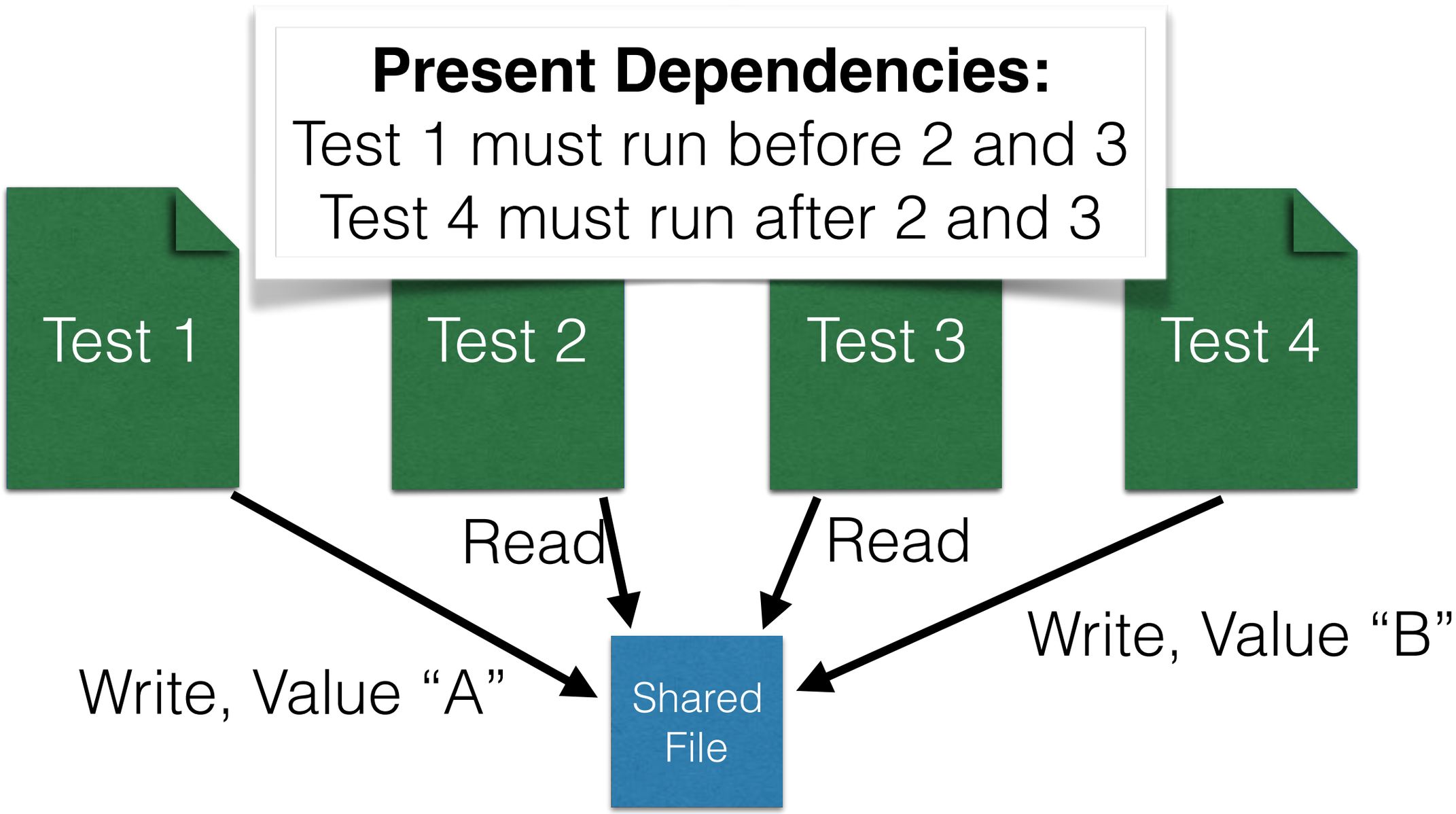
Combinatorial Dependency Detection



Combinatorial Dependency Detection



Data Dependencies



Sample Data Dependencies

```
int x = readSharedData();
assertEquals(6,x);
```

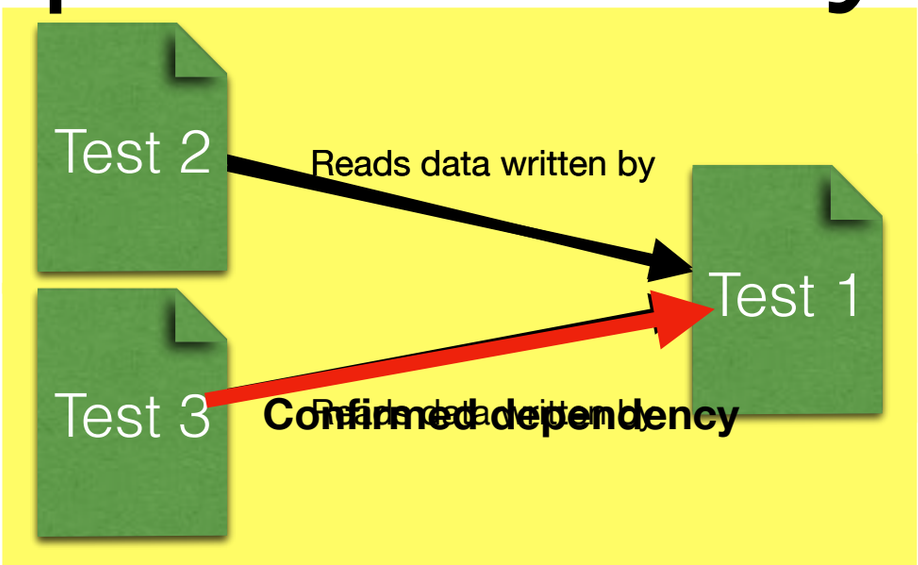
```
getSharedLogger().logVerbose("Log Ran");
```

Practical Test Order Dependency Detection

- PraDeT's two phase approach:
  - 1: Gather data dependencies
  - 2: Use dependency information to guide systematic exploration of dependencies

Dependency Refinement

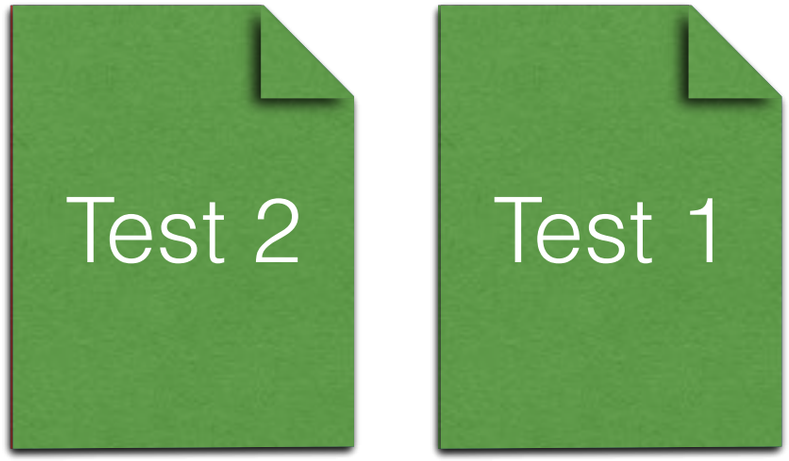
Data dependency graph:



At end of refinement, only true test order dependencies remain

Execution sequence:

Currently checking 3 depending on 1

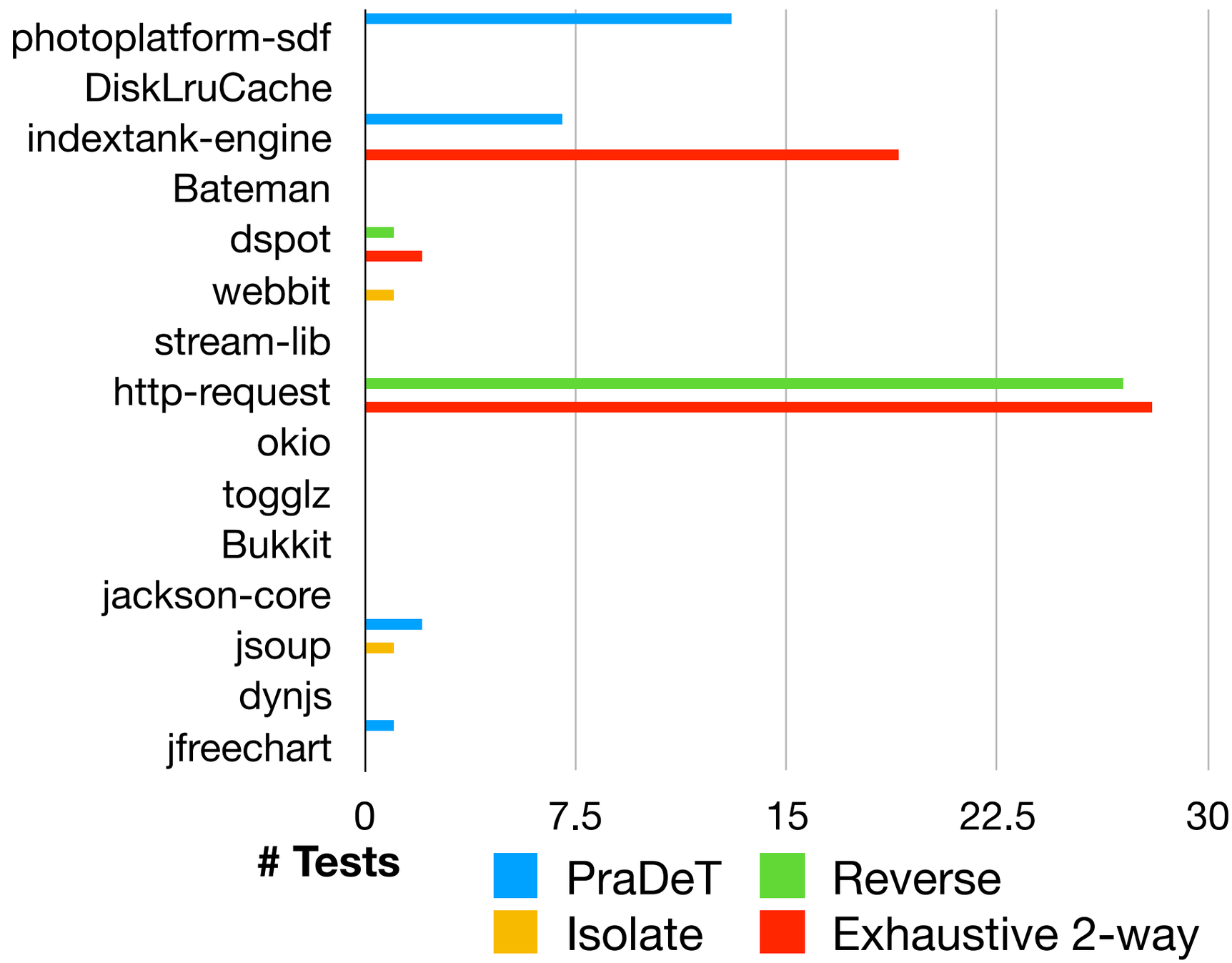


Evaluation

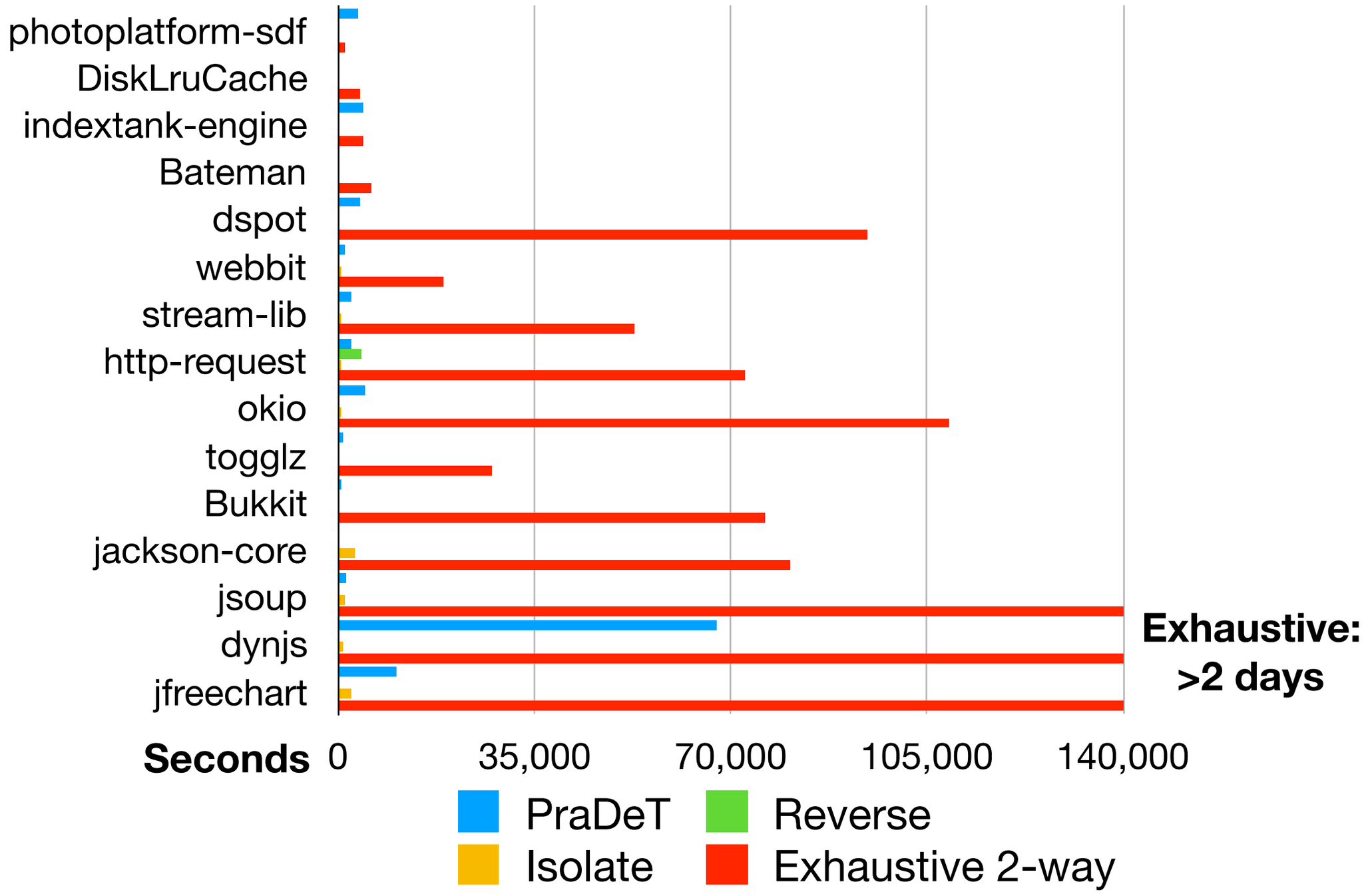
- How many test dependencies does PraDeT detect in comparison to prior approaches?
- How long does PraDeT take to run?
- When should developers run PraDeT?

PraDeT: Evaluation

PraDeT reliably finds test order dependencies

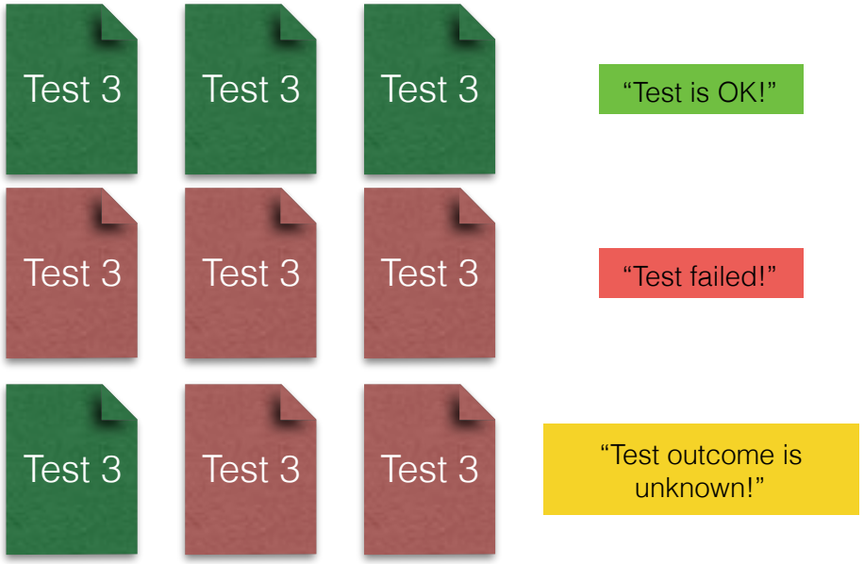


Evaluation: Performance



Flaky Tests

- What about tests that are flaky for other reasons? Do we still need to rerun them?
- What happened to *accelerating* testing?
- Now tests need to be run three times!
- Can we identify with certainty that a test is a false alarm without re-running?



Fork me on Github

DeFlaker: Automatically Detecting Flaky Tests

Jonathan Bell, Owolabi Legunsen, Michael Hilton,
Lamyaa Eloussi, Tifany Yung and Darko Marinov
George Mason University, University of Illinois at Urbana-Champaign
and Carnegie Mellon University

[To appear at ICSE 2018 in Gothenburg, May 31, 2018]



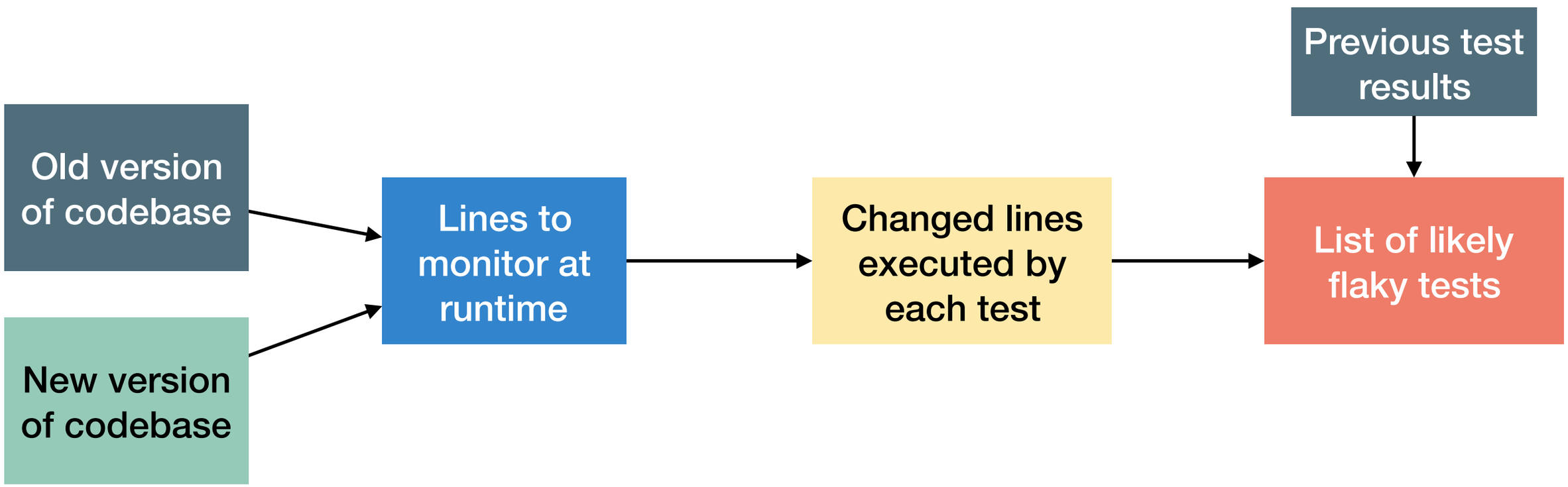


Flaky Tests

- Our key insight: there is lightweight information we can track while a test runs
- “Did this test run any code that changed?”
- Tracking coverage can be slow though! (40-50% overhead!)
  - …and we want to make things faster

DeFlaker's Differential Coverage

DeFlaker tracks *differential coverage* — only tracking code that changed since the last execution of the tool



Differential Coverage

Just **syntactic** diff (e.g. from git) is insufficient to notice coverage of all kinds of changes!

```
public class SuperOld {
   public void magic() {
   }
}
public class SuperNew extends SuperOld {
   public void magic() {
      assert(false); // causes test to fail
   }
}

public class App extends ~~SuperOld~~ SuperNew {
}
public class TestApp {
   @Test public void testApp() {
      new App().magic();
   }
}
```

Now calls SuperNew.magic!

DeFlaker

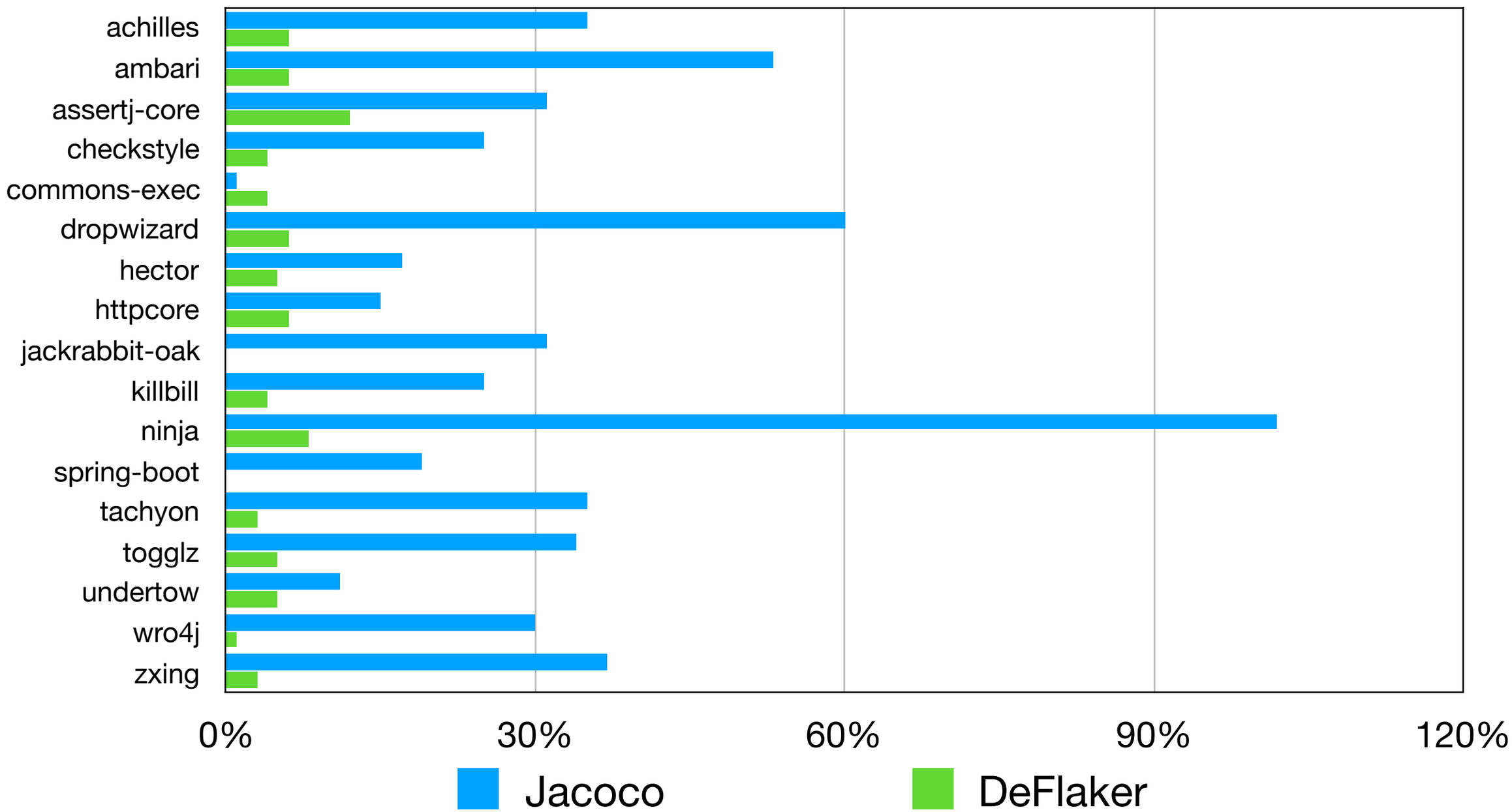
- Tracks line coverage of all changed statements (in both tests and SUT)
- Identifies non-statement changes in classes by parsing them, tracks with class-level coverage
- Detects flaky test failures “just-in-time” when they fail
- Implemented as a maven extension (3-line addition to pom.xml)

Evaluation

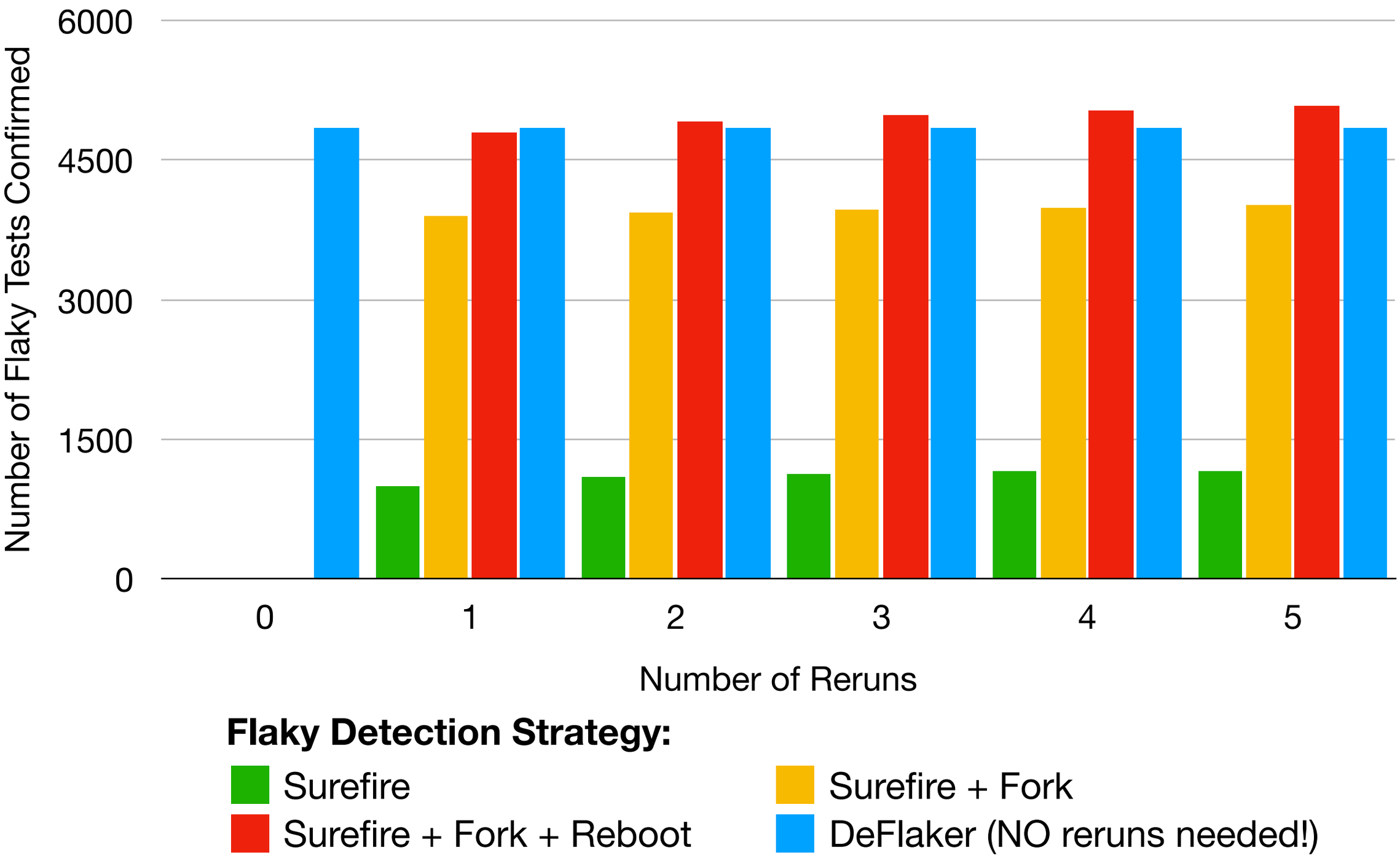
- What is the performance overhead of running DeFlaker?
- How many flaky tests does DeFlaker find in comparison to rerunning failed tests?

DeFlaker is Fast

Evaluation on 17 open source Java projects: average 5% overhead



DeFlaker Finds Flaky Tests



DeFlaker Findings

- HOW you re-run flaky tests matters much more than how many times you rerun them
- DeFlaker is extremely low overhead and can immediately identify flaky tests
- Also deployed shadowing live executions on TravisCI, found 87 new flaky tests and reported to developers, many now fixed
- Differential coverage may have many other useful applications as well
- Try it out! http://deflaker.org/

Further Reading on Flaky Tests

DeFlaker project site (ICSE 2018)
Jonathan Bell, Owolabi Legunsen, Michael Hilton, Lamyaa Eloussi, Tifany Yung and Darko Marinov
Includes a preprint of the paper and information on the tool

Measuring the cost of regression testing in practice: a study of Java projects using continuous integration (FSE 2017)
Adriaan Labuschagne, Laura Inozemtseva and Reid Holmes
A study of test suite executions on TravisCI that investigated the number of flaky test failures.

Flaky Tests at Google and How We Mitigate Them (Google Testing Blog, 2016)
John Micco
A summary of Flaky tests at Google and (as of 2016) the strategies used to manage them.

An Empirical Analysis of Flaky Tests (FSE 2014)
Qingzhou Luo, Farah Hariri, Lamyaa Eloussi, and Darko Marinov
A study of the various factors that might cause tests to behave erratically, and what developers do about them.

Chromium Project's Flaky Test Dashboard
A description of how the Chromium and WebKit teams triage and manage their flaky test failures.

Detecting and Debugging Flaky Tests

Jonathan Bell

bellj@gmu.edu
http://jonbell.net/
@_jon_bell_

Fork me on Github