Detecting Test Flakiness Without Rerunning Tests

Dissertation Proposal

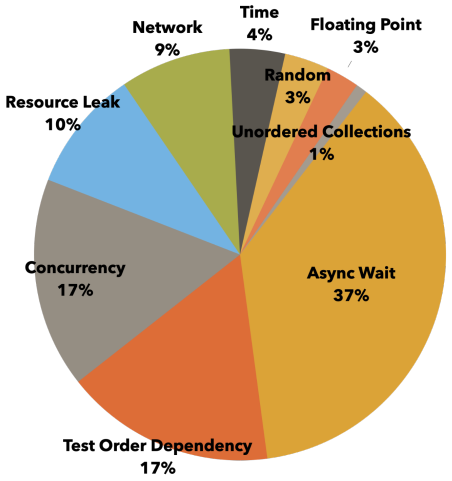
Abdulrahman Alshammari

Overview

- **Continuous Integration** (CI) is a software development practice
  - Run tests to check changed code
- Failed tests indicate the presence of faults
  - Assumption: Tests are deterministic
- Some tests exhibit non-deterministic behavior
  - known as **Flaky Tests**
- **Flaky Tests** affect the efficiency of CI

What Causes Tests to Be Flaky?

A flaky test is a test that passes and fails when executed many times on the same code.



Flaky Tests Causes [29]

```
1 @Test
2 public void testRsReportsWrongServerName() throws Exception {
3   MiniHBaseCluster cluster = TEST_UTIL.getHBaseCluster();
4   MiniHBaseClusterRegionServer firstServer =
5     (MiniHBaseClusterRegionServer)cluster.getRegionServer(0);
6   HServerInfo hsi = firstServer.getServerInfo();
7   firstServer.setHServerInfo(...);
8
9   // Sleep while the region server pings back
10  Thread.sleep(2000);
11  assertTrue(firstServer.isOnline());
12  assertEquals(2,cluster.getLiveRegionServerThreads().size());
13  ... // similarly for secondServer
14 }
```

An example of Async Wait flaky test from the HBase project [29]

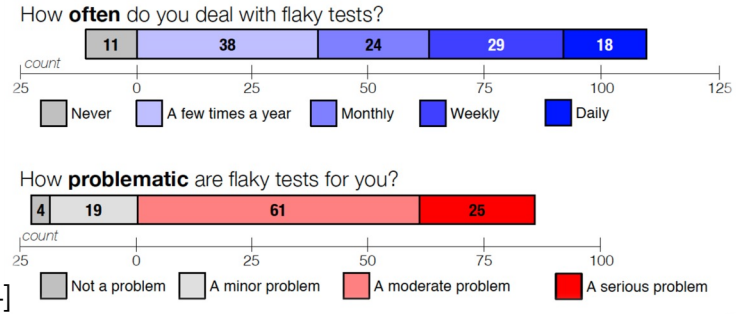
[29] [Luo et al, FSE 2014 “An empirical analysis of flaky tests”]

Why Do We Care about Test Flakiness?

Flaky tests are problematic

- **Repeatedly occurs:** At least weekly basis [11] [14]
- **Large scale:** 16 % of Google tests show some flakiness [14]
- **Delay release:** Hinder developers from merging pull requests [14]
- **Wasting developer time:** The most severe consequence of test flakiness [14]

A survey about flaky tests from Developer's perspective [11]



[11] [Eck et al, FSE 2019 "Understanding Flaky Tests: The Developer's Perspective "]
[14] Gruber and Fraser, ICST2022 "A Survey on How Test Flakiness Affects Developers and What Support They Need To Address It"]

What Developers Do Right Now With Flaky Tests

Detecting **Flaky Tests** Using Re-Run

- **Limitation**: Expensive – e.g. 2-16% of testing budget in Google just for Re-Run

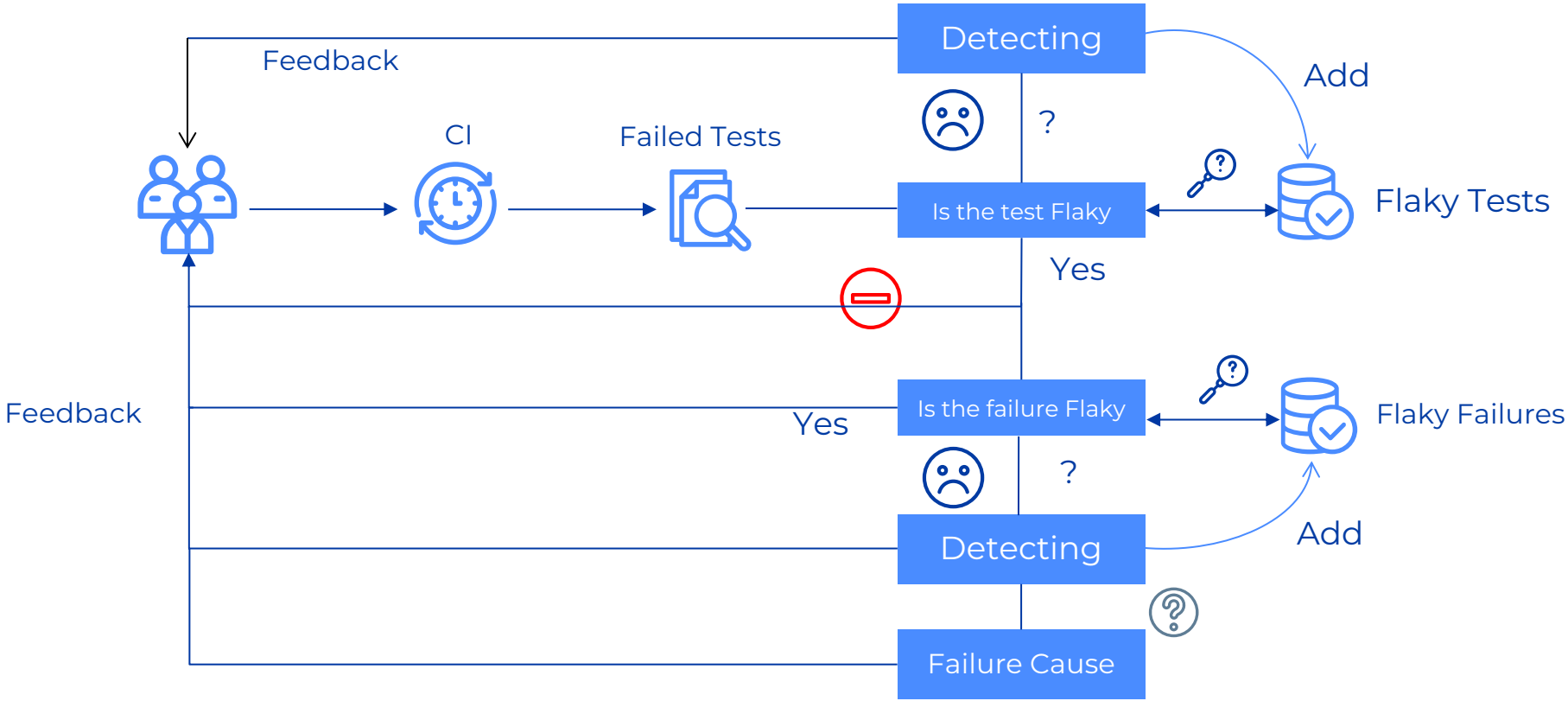
Research helps in detecting **Flaky Tests**

- Automated Detection Tools – e.g. DeFlaker [6], iDFlakies [7]
- **Overall**: Run-based- techniques and expected overhead costs

Developers need to detect flaky tests without overhead cost

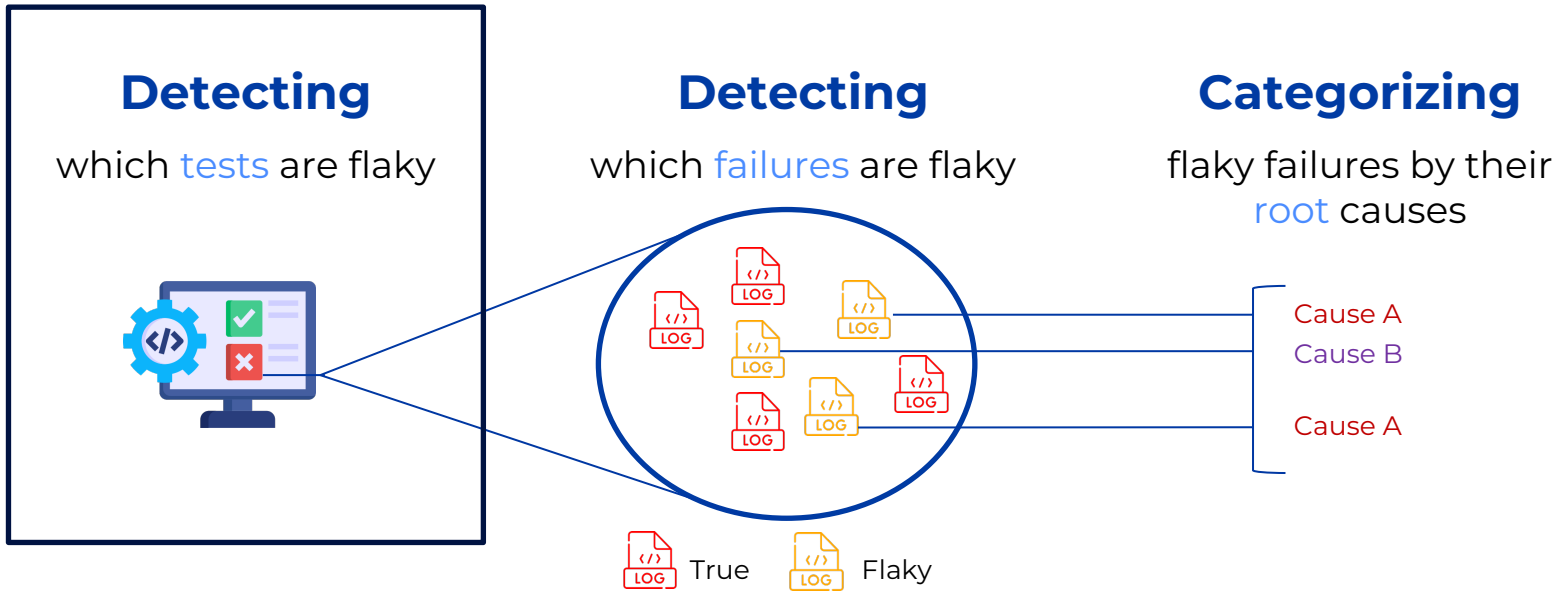
[6] [Bell et al, ICSE2018 “Deflaker: Automatically detecting flaky tests”]
[7] [Lam et al, ICST2019 “iDFlakies: A framework for detecting and partially classifying flaky tests”]

Current Challenges

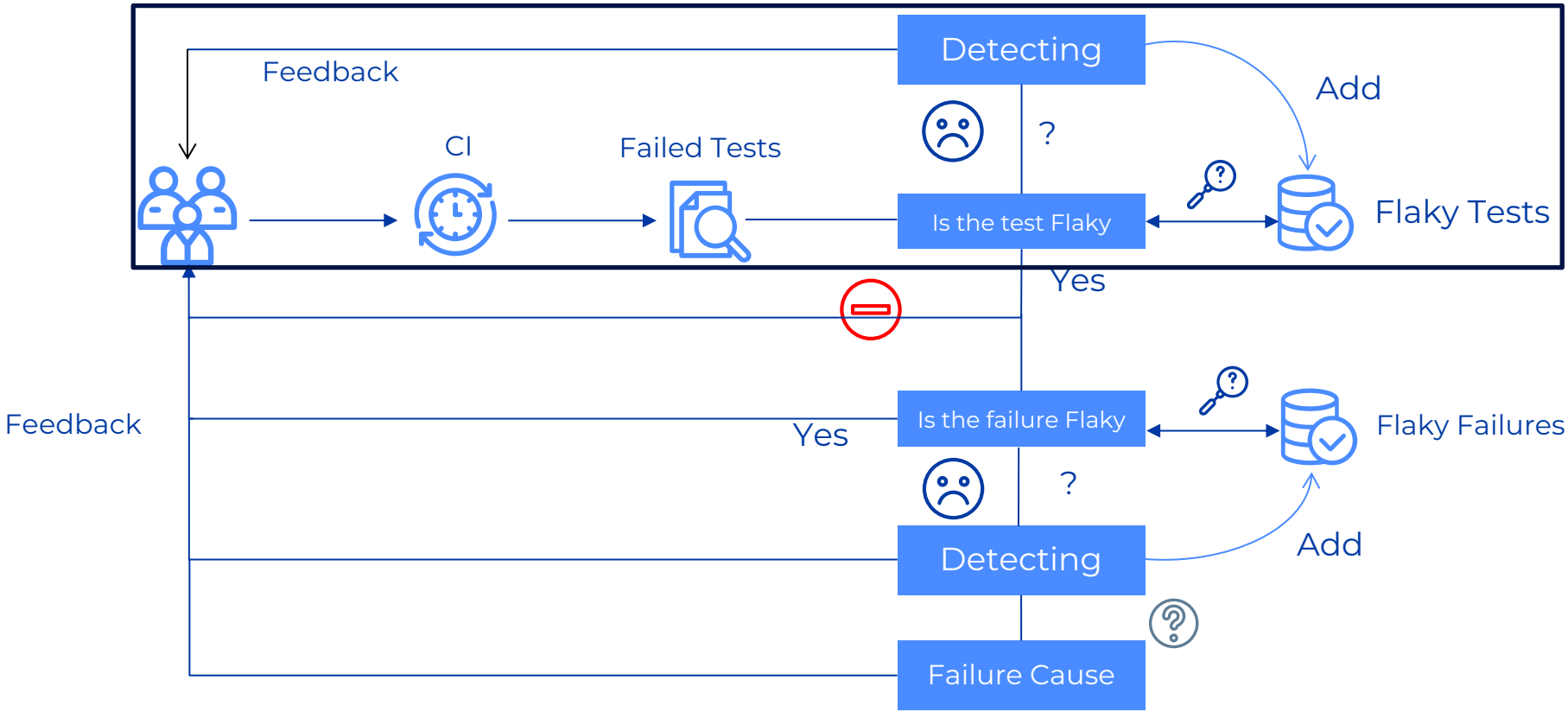


Proposal Statement

Machine learning and data science can address better the problem of test flakiness in terms of:



Current Challenges

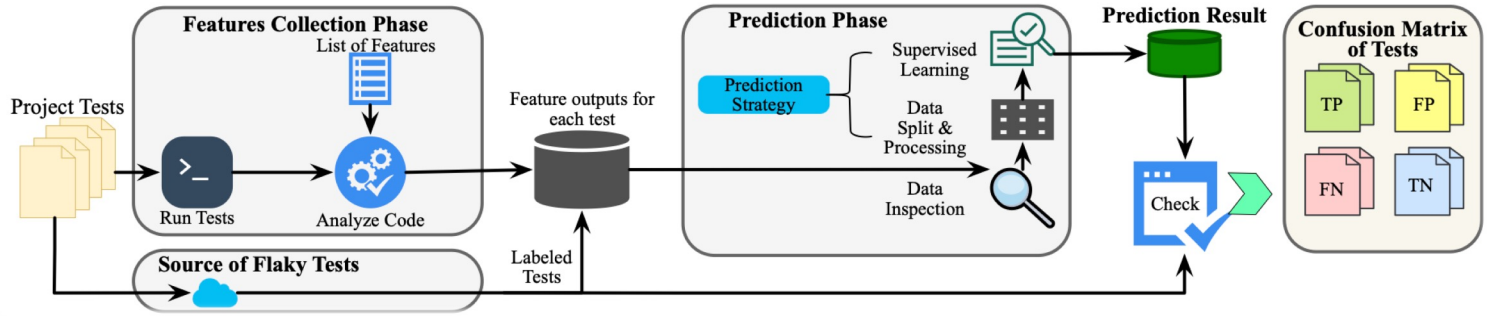


Detecting Which *Tests* Are Flaky

FlakeFlagger (ICSE2021)

Goal: using the machine learning to predict if a test is flaky or not leveraging the similarity of other already known flaky and non-flaky tests [4]

- **Source of Flaky Tests**: A set of flaky and non-flaky Tests
- **Feature Collection Phase**: List of features that may be predictive of flakiness
- **Prediction Phase**: Process the data and train the classifier



[4] [Alshammari et al, ICSE 2021 "FlakeFlagger: Predicting flakiness without rerunning tests"]

Detecting Which *Tests* Are Flaky

FlakeFlagger (ICSE2021)

Main Contributions [4]:

- **Dataset**: Running 24 projects test suites 10,000 times to collect flaky and non-flaky tests
  - Impacts: [Qin et al. ICSME 2022] [Dell'Anna et al. ESE 2023] [Pontillo et al. ESE 2022]
- **Test Detector**: Hybrid static/dynamic approach to collect behavioral features of tests
- **FlakeFlagger**: Machine learning classifier to predict test flakiness

Research Questions [4]:

- **RQ1**: How many flaky tests can be found by rerunning tests given different rerun budgets?
- **RQ2**: How hard is it to reproduce a flaky test failure?
- **RQ3**: How effective is FlakeFlagger at predicting flaky tests?
- **RQ4**: How helpful FlakeFlagger's features in distinguishing between flaky and non flaky?

[4] [Alshammari et al, ICSE 2021 "FlakeFlagger: Predicting flakiness without rerunning tests"]

- **Reproducibility**: Fail to reproduce all flaky tests by [6] [7]
- **Number of runs**: Hard to determine run limits

Project	Tests	Flaky by Reruns	DeFlaker [6] Shared	DeFlaker [6] Total	iDFlakies [7] Shared	iDFlakies [7] Total	% of all Flaky Tests Detectable at: 10 Reruns	100 Reruns	1,000 Reruns	Distribution of Failure Frequencies, as % of Tests Failing (0,10] (10, 100] (100, 1,000] (1,000, 10,000] runs of 10,000
spring-boot	2,128	163	0	5			71%	71%	77%	
hbase	431	145	0	1			52%	59%	75%	
alluxio	187	116	2	2			0%	91%	100%	
okhttp	810	100					8%	12%	15%	
ambari	324	52	1	1			0%	2%	94%	
hector	142	33	1	1			3%	3%	100%	
activiti	2,044	32					0%	3%	44%	
java-websocket	145	23			22	52	0%	26%	87%	
wildfly	1,238	23					0%	0%	4%	
httpcore	712	22	1	1			0%	9%	9%	
logback	842	22					5%	9%	41%	
incubator-dubbo	2,177	19			5	12	5%	11%	26%	
http-request	163	18					0%	83%	83%	
wro4j	1,145	16	1	1			44%	50%	81%	
orbit	86	7	0	1			14%	43%	86%	
undertow	183	7	0	3			0%	0%	29%	
achilles	1,317	4					0%	25%	75%	
elastic-job-lite	558	3			1	6	0%	0%	0%	
zxing	345	2	2	2			0%	100%	100%	
assertj-core	6,267	1	1	1			0%	100%	100%	
commons-exec	55	1					0%	0%	100%	
handlebars.java	428	1					0%	100%	100%	
ninja	306	1	1	1			0%	100%	100%	
jimfs	212	0								No flaky tests observed
Total	22,245	811	10	20	28	70	26%	45%	67%	

Dataset:

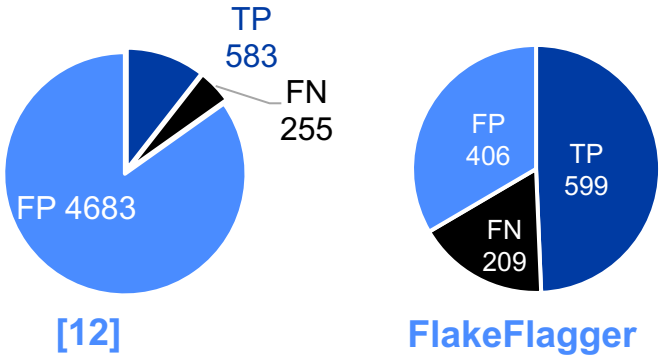
811 flaky tests were detected after running 24 projects 10,000 times.

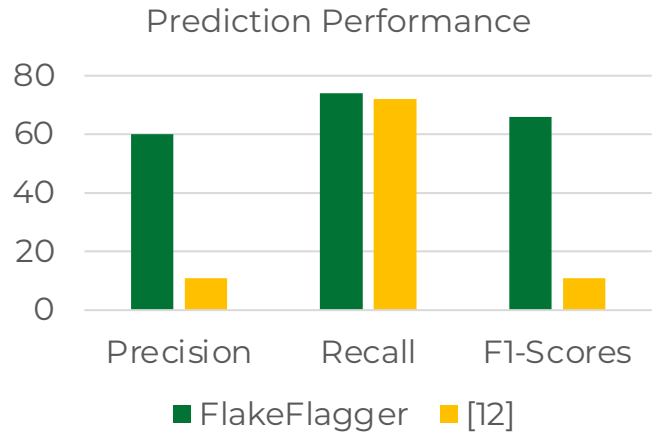
[6] [Bell et al, ICSE2018 “Deflaker: Automatically detecting flaky tests”]
[7] [Lam et al, ICST2019 “iDFlakies: A framework for detecting and partially classifying flaky tests”]

RQ3: How effective is FlakeFlagger at predicting flaky tests?

FlakeFlagger VS the state-of-the-art flaky test classifier [12]:

- **[12]**: Static flaky tests classifier using tests code as a bag-of-words
- FlakeFlagger outperformed **[12]**
  - By **F1** scores on **16** projects and tied on **4** projects
- FlakeFlagger's performance varied across projects
- FlakeFlagger as a pre-step of Re-Run

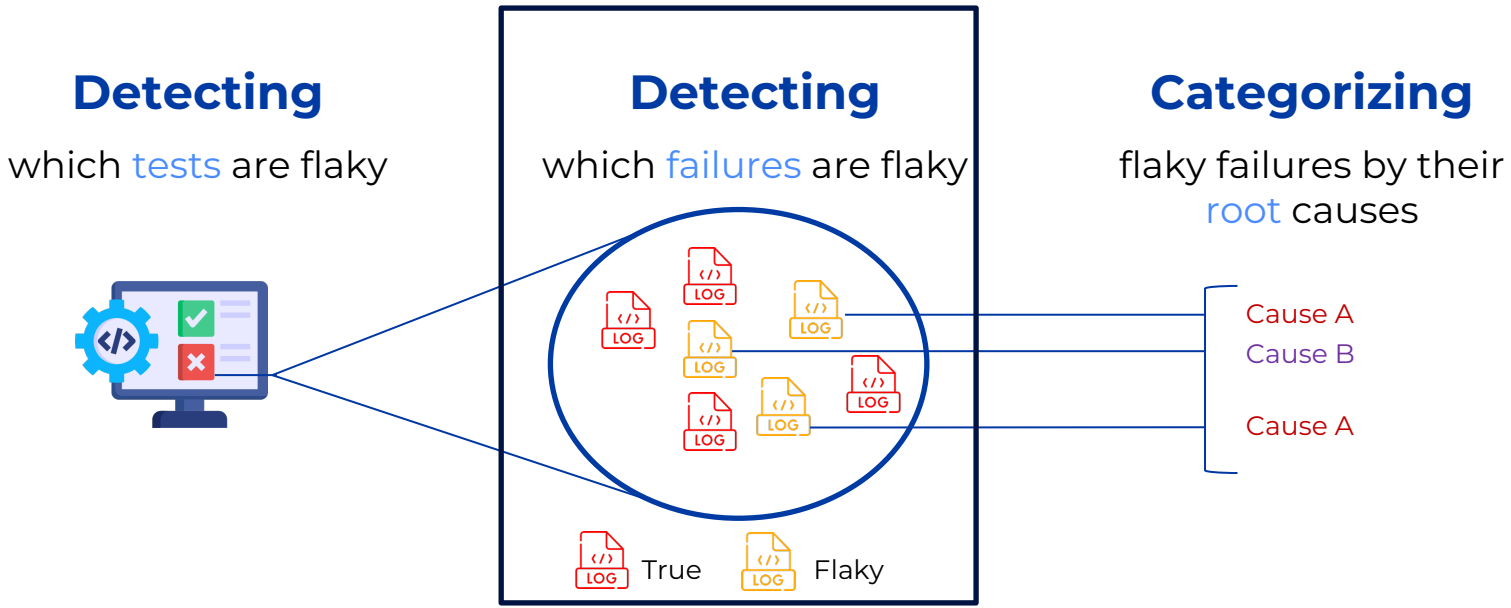




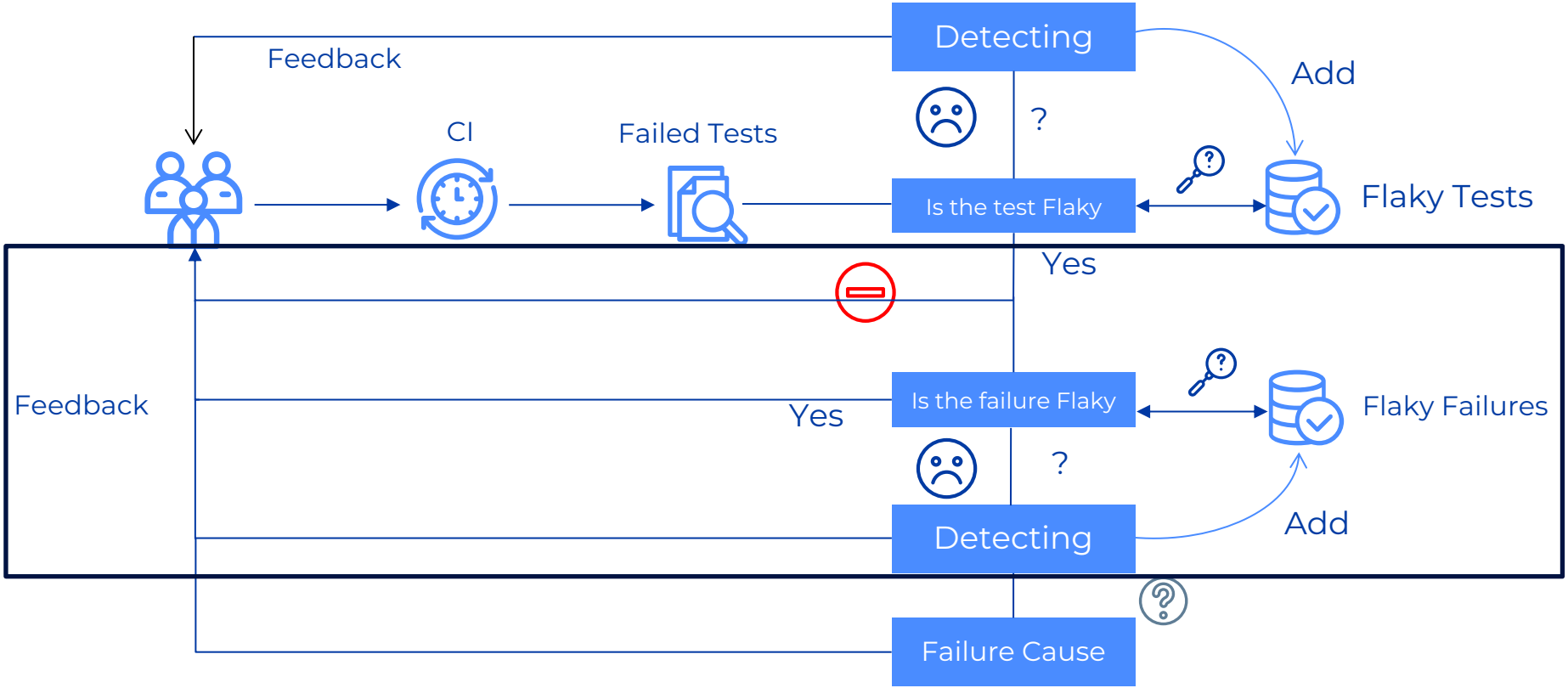
[12] [Pinto et al, MSR2020 "What is the vocabulary of flaky tests?"]

Proposal Statement

Machine learning and data science can address better the problem of test flakiness in terms of:

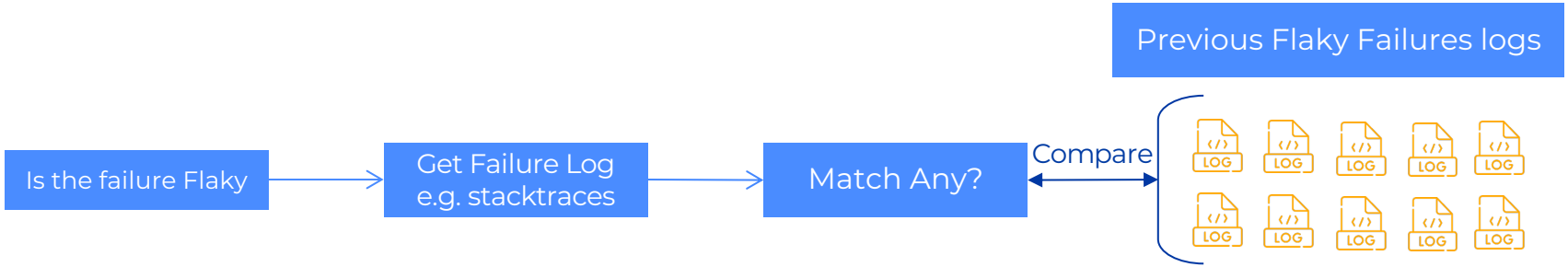


Current Challenges



Detecting Which *Failures* Are Flaky

Submitted



- Failure de-duplication [16]**:** matching new failure with previous failures
  - No previous failures available (e.g. new tests)?

Use Case: Developer has an existing history of test failures

- Task: determine if a new failure is flaky or true failure.

[16] Jiang et al. ICSE2017 [“What causes my test alarm? automatic cause analysis for test alarms in system and integration testing”]

Detecting Which *Failures* Are Flaky

Submitted

Main Contributions

- **Approach**: A de-duplication based approach using the failure messages and stacktraces
- **Failure Log Classifier**: Machine learning classifiers to predict if a failure is flaky or not

Research Questions:

- **RQ1**: How often are flaky failures repetitive?
- **RQ2**: With prior flaky and true failures, is it feasible to use the **failure de-duplication** to tell if a failure is flaky or true one?
- **RQ3**: How far utilizing machine learning being helpful in finding the differences between flaky and true failures?

RQ1 : How often are flaky failures repetitive?

Submitted

Main Findings:

- **99,488** repetitive failures (out of **99,622**)
- **134** non-repetitive (**95** from *single* failed tests)
- Frequently failed flaky tests **VS** repetitive failures
- Failures repetitive across tests
- Number of failed tests in a test suite

Takeaway:

Previous flaky failures can be a reference to check if a newly encountered failure is familiar.

REPETITIVE FLAKY FAILURES WITHIN AND ACROSS TESTS PER PROJECT. Failures column shows the number of flaky failures and the different failures (Set). The columns (1:n) and [1] refer to flaky failures that are and are not repetitives, repectively. Per Test refers to matching the failures within the same test. Across Tests refers matching all flaky failures from all tests.

Projects	Tests	Failures		Per Test		Across Tests	
		Flaky	Set	[1]	(1:n)	[1]	(1:n)
Alluxio-alluxio	114	16,858	310	11	16,847	5	16,853
square-okhttp	100	28,264	121	40	28,224	17	28,247
apache-hbase	62	19,822	100	14	19,808	5	19,817
apache-ambari	51	4,063	54	0	4,063	0	4,063
hector-client-hector	33	6,529	33	0	6,529	0	6,529
activiti-activiti	31	1,378	32	13	1,365	6	1,372
tootallnate-java-websocket	22	2,095	43	2	2,093	0	2,095
apache-httpcore	22	354	22	9	345	2	352
qos-ch-logback	20	438	21	8	430	4	434
kevinsawicki-http-request	18	3,501	18	3	3,498	0	3,501
wildfly-wildfly	18	50	18	12	38	4	46
wro4j-wro4j	14	10,833	21	3	10,830	2	10,831
spring-projects-spring-boot	12	14	13	12	2	5	9
orbit-orbit	7	2,943	7	0	2,943	0	2,943
undertow-io-undertow	7	92	12	3	89	1	91
doanduyhai-Achilles	4	165	5	1	164	1	164
elasticjob-elastic-job-lite	3	7	4	3	4	0	7
assertj-core	1	974	1	0	974	0	974
ninja-ninja	1	476	1	0	476	0	476
handlebars.java	1	411	1	0	411	0	411
apache-commons-exec	1	33	1	0	33	0	33
zxing-zxing	1	322	1	0	322	0	322
Total	543	99,622	839	134	99,488	52	99,570

RQ2 : With prior flaky and true failures, is it feasible to use the failure de-duplication to tell if a failure is flaky or true one?

Submitted

- **Matching** struggle in logs with assertion exceptions
- A test could have a match for one failure but not for another
- Flaky and true failures rates VS matching result

	Total Tests and Failures			Set of Failures		Confusion Matrix and Evaluation By Failures							# of Tests in	
Project	**Test**	**True**	**Flaky**	**True**	**Flaky**	**TP**	**FN**	**FP**	**TN**	**P**	**R**	**SP**	**TP**	**FN**
Alluxio-alluxio	114	32,608	16,858	6,491	310	9,615	7,243	1,694	30,914	85%	57%	94%	114	102
square-okhttp	100	34,266	28,264	18,609	121	16,517	11,747	114	34,152	99%	58%	99%	58	53
apache-hbase	62	11,324	19,822	811	100	18,496	1,326	1,198	10,126	93%	93%	89%	58	14
apache-ambari	51	11,049	4,063	4,563	54	4,003	60	5	11,044	99%	98%	99%	50	2
hector-client-hector	33	3,604	6,529	1,769	33	1,382	5,147	12	3,592	99%	21%	99%	32	1
activiti-activiti	31	46,100	1,378	16,018	32	932	446	2,609	43,491	26%	67%	94%	1	30
tootallnate-java-websocket	22	1,299	2,095	330	43	591	1,504	816	483	42%	28%	37%	19	22
apache-httpcore	22	8,333	354	663	22	0	354	2,117	6,216	0%	0%	74%	0	22
qos-ch-logback	20	2,614	438	903	21	56	382	368	2,246	13%	12%	85%	3	17
wildfly-wildfly	18	4,364	50	1,497	18	38	12	0	4,364	100%	76%	100%	6	12
kevinsawicki-http-request	18	387	3,501	229	18	981	2,520	40	347	96%	28%	89%	4	14
wro4j-wro4j	14	540	10,833	90	21	800	10,033	29	511	96%	7%	94%	9	11
spring-projects-spring-boot	12	2,150	14	244	13	2	12	0	2,150	100%	14%	100%	1	12
undertow-io-undertow	7	2,304	92	236	12	8	84	940	1,364	0%	8%	59%	2	6
orbit-orbit	7	822	2,943	302	7	87	2,856	57	765	60%	2%	93%	2	5
doanduyhai-Achilles	4	442	165	245	5	120	45	46	396	72%	72%	89%	1	3
elasticjob-elastic-job-lite	3	111	7	68	4	4	3	0	111	100%	57%	100%	1	3
apache-commons-exec	1	59	33	13	1	0	33	2	57	0%	0%	96%	0	1
assertj-core	1	17	974	9	1	974	0	0	17	100%	100%	100%	1	0
handlebars.java	1	147	411	61	1	0	411	16	131	0%	0%	89%	0	1
zxing-zxing	1	76	322	37	1	322	0	0	76	100%	100%	100%	1	0
ninja-ninja	1	209	476	6	1	0	476	90	119	0%	0%	56%	0	1
22 Projects Total	543	162,825	99,622	53,194	839	54,928	44,694	10,153	152,672				363	332

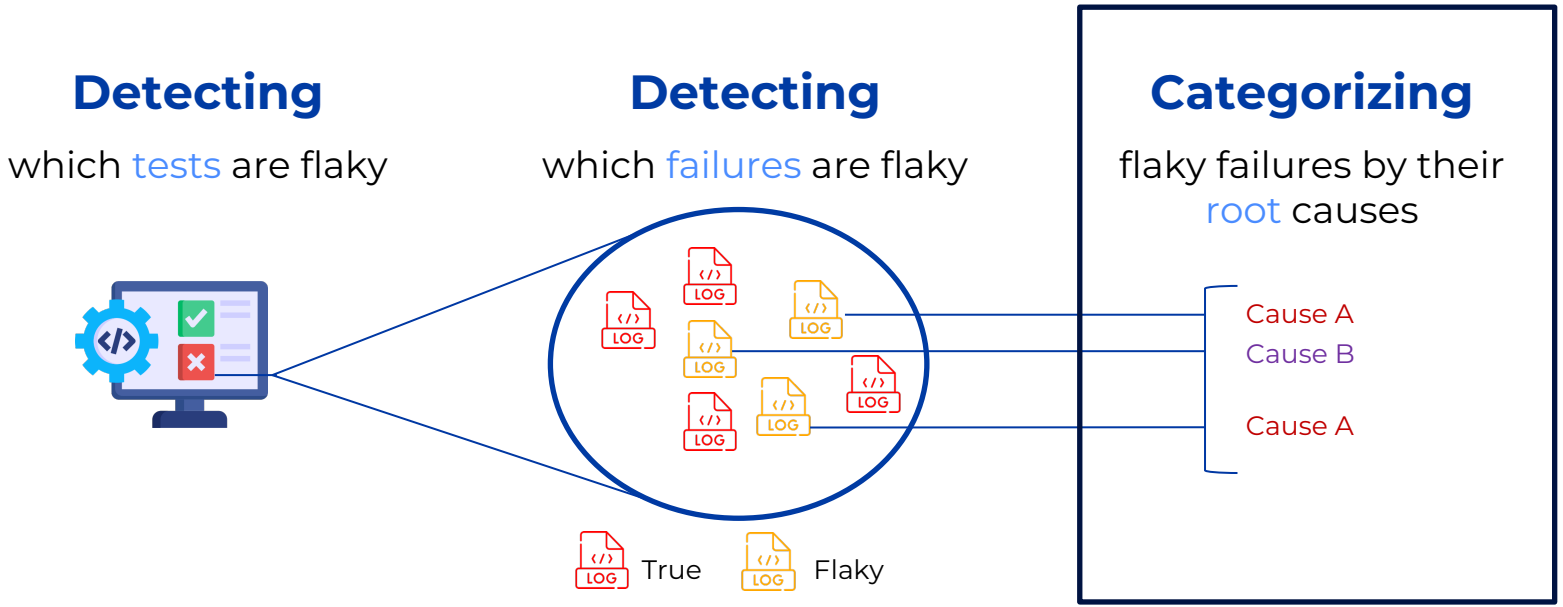
Failure de-duplication evaluation

Failure Label	Failure	Match	
		Flaky	True
TP	Flaky	✓	
FN	Flaky		✓
FP	True	✓	
TN	True		✓

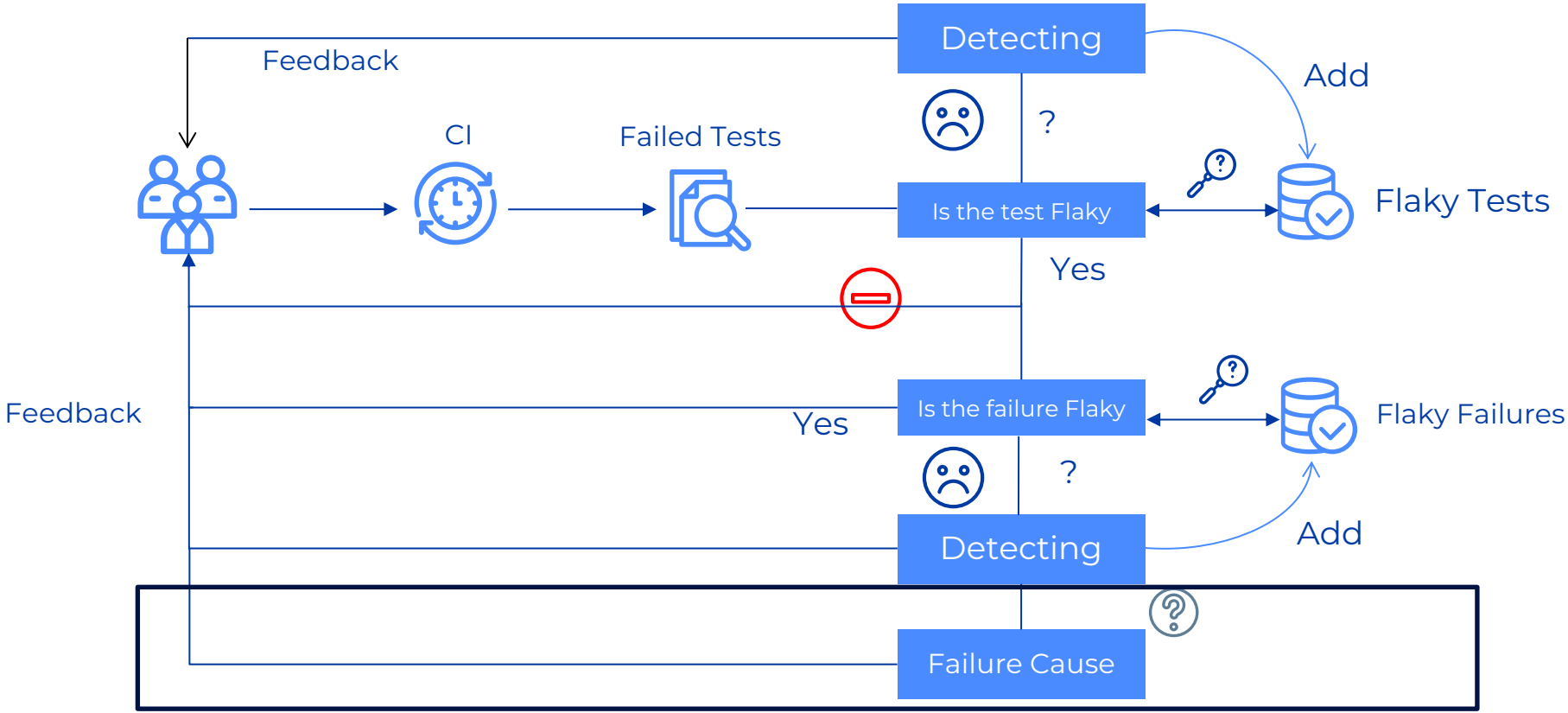
Precision (P), Recall (R), Specificity (SP)

Proposal Statement

Machine learning and data science can address better the problem of test flakiness in terms of:



Current Challenges



Categorizing flaky failures by their root causes

Current

- Goal: Finding the root causes of flaky tests
  - Some rely on human feedback
- Idea: Finding the root causes of flaky **failures**
  - Assumption: A test is triggered by multiple flakiness causes
- Challenging: Flaky Failures dataset with the root causes.

Initial Research Questions:

- **RQ1**: Is it possible for a flaky test to be triggered by multiple flakiness root causes?
- **RQ2**: Can failure logs associate flaky failures with their root causes using machine learning?

Alternative Plan: **Flakiness in mutation testing**

- Goal: Detect flaky mutants without rerun them

Summary

Proposal Statement: Machine learning and data science can address better the problem of test flakiness in terms of:

Detecting	**Detecting**	**Categorizing**
which tests are flaky	which failures are flaky	flaky failures by their root causes
FlakeFlagger (ICSE2021)	Submitted	Current Work

Thank You