



What is the Vocabulary of Flaky Tests?

 @gustavopinto



Gustavo
Pinto



Breno
Miranda



Supun
Dissanayake



Marcelo
d'Amorim



Christoph
Treude



Antonia
Bertolino



What is a
flaky test



```

public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}

```

```

public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}

```

Runs: 1/1

 Errors: 0

 Failures: 0

```

public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}

```

Runs: 1/1

 Errors: 1

 Failures: 0

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```



```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

What do we know about flaky tests?



Luo @ FSE 2014



Thorve @ ICSME 2018

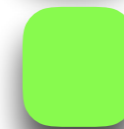


Empirical-based

What do we know about flaky tests?



Empirical-based



Dynamic detection of flakiness



Luo @ FSE 2014



Thorve @ ICSME 2018



Bell @ ICSE 2018



Lam @ ICST 2019

What do we know about flaky tests?



Luo @ FSE 2014



Thorve @ ICSME 2018



Bell @ ICSE 2018



Lam @ ICST 2019



Empirical-based



Dynamic detection of flakiness



Static detection of flakiness



How to
statically detect
if a test code is
flaky





Flaky tests



Tokenized Flaky test



**Machine Learning
algorithms**



**Ranking of code identifiers
associated with flakiness**



**Vocabulary of
Flaky tests**



Flaky tests



Tokenized Flaky test



Machine Learning algorithms



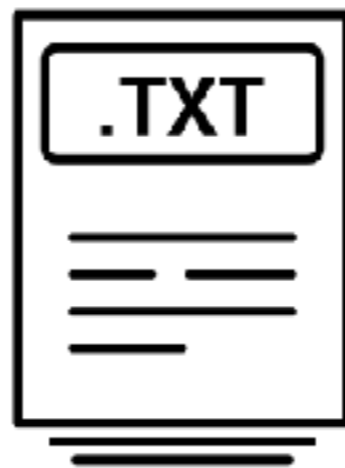
Ranking of code identifiers associated with flakiness



Vocabulary of Flaky tests



Flaky tests



Tokenized Flaky test



Machine Learning algorithms



Ranking of code identifiers associated with flakiness



Vocabulary of Flaky tests



Flaky tests



Tokenized Flaky test



Machine Learning algorithms



Ranking of code identifiers associated with flakiness



Vocabulary of Flaky tests



Flaky tests



Tokenized Flaky test



Machine Learning algorithms



Ranking of code identifiers associated with flakiness



Vocabulary of Flaky tests



Flaky tests



Tokenized Flaky test



Machine Learning algorithms



Ranking of code identifiers associated with flakiness



Vocabulary of Flaky tests

Is this test **flaky**?

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

Is this test **flaky**?

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

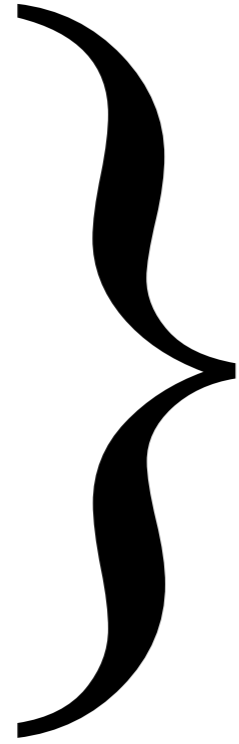
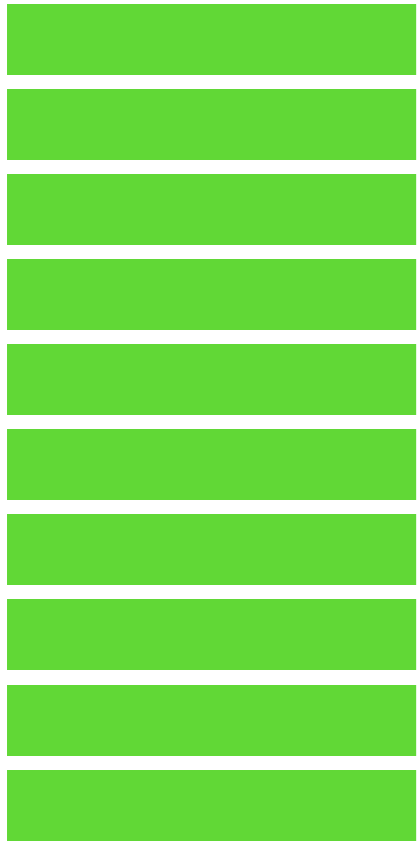
        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

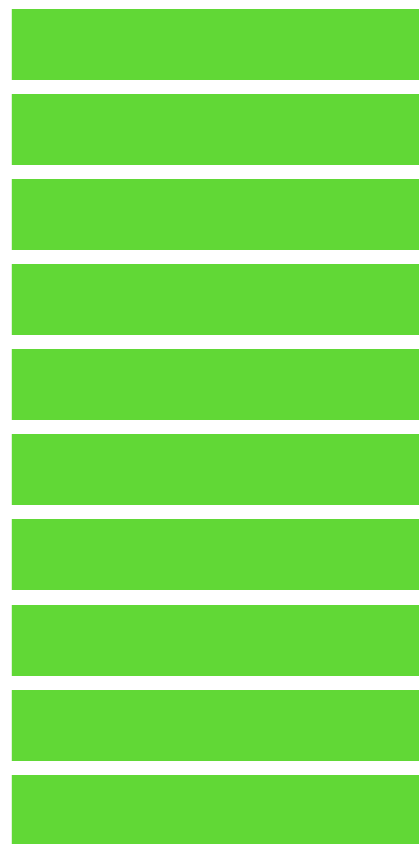
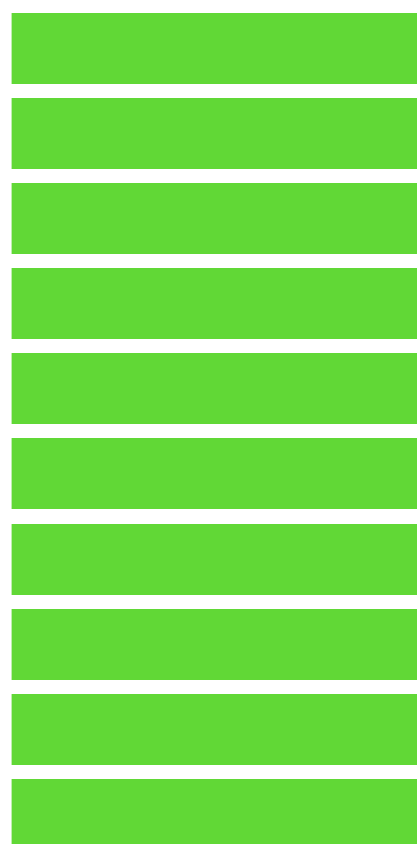
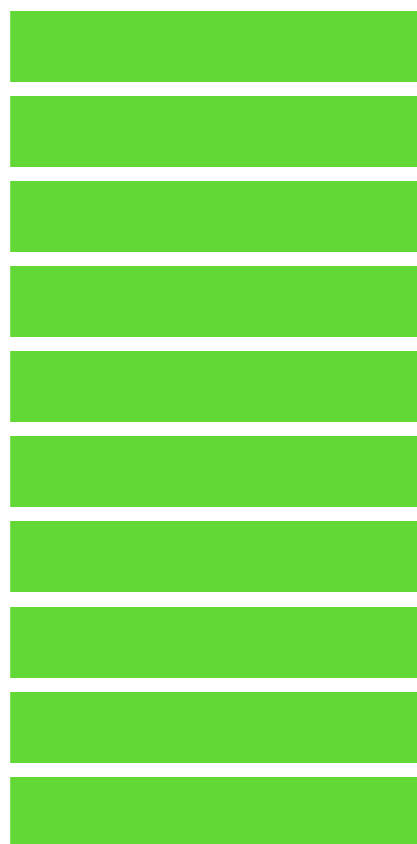
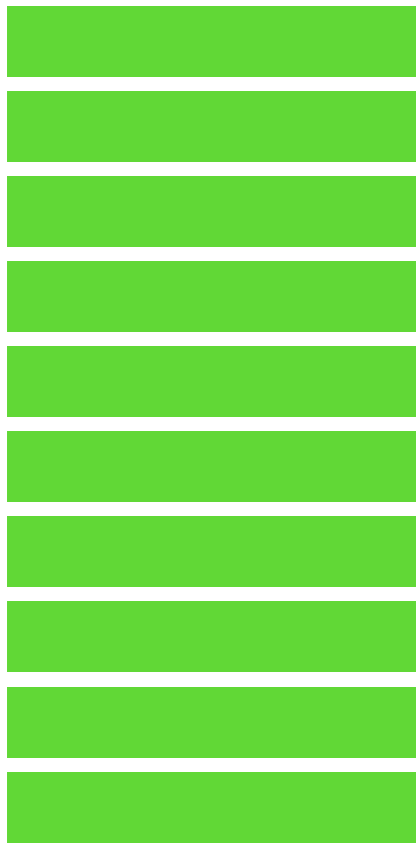
Runs: 1/1

 Errors: 0

 Failures: 0



10 executions





100 executions



GitHub Project

achilles

alluxio

ambari

assertj-core

checkstyle

commons-exec

dropwizard

hadoop

handlebars

hbase

hector

httpcore

jackrabbit-oak

jimfs

logback

ninja

okhttp

oozie

orbit

oryx

spring-boot

togglz

undertow

wro4j

zxing

GitHub Project

achilles

alluxio

ambari

assertj-core

checkstyle

commons-exec

dropwizard

hadoop

handlebars

hbase

hector

httpcore

jackrabbit-oak

jimfs

logback

ninja

okhttp

oozie

orbit

oryx

spring-boot

togglz

undertow

wro4j

zxing

The same studied here



DEFLAKER: Automatically Detecting Flaky Tests

Jonathan Bell¹, Owolabi Ogunsan², Michael Hilton¹,

Linyuan Euzou¹, Tiffany Yung¹, and Duke Mannov¹

¹George Mason University, Fairfax, VA, USA

²University of Illinois at Urbana-Champaign, Urbana, IL, USA

³Carnegie Mellon University, Pittsburgh, PA, USA

bellj@gmu.edu, ogunsan@uiuc.edu, ogunsan@illinois.edu, mhilton@gmu.edu

ABSTRACT

Developers often write tests to check that their later changes to a code repository do not break any previously working functionality. While any new test failures could be due to a regression caused by the later changes, however, some test failures may not be due to the later changes but due to test conditions that do not exist in the real world. The typical cause of such test failures is test flakiness, which is the property of a test that it fails on some runs but not on others. Unfortunately, recognizing flaky tests can be costly and can slow down the development cycle.

We present the first extensive evaluation of recognizing flaky tests and propose a new technique, called DEFLAKER, that detects test failures that are likely to be caused by test conditions that vary over runtime execution. DEFLAKER monitors the coverage of test cases, compares test results to a baseline, and identifies test failures that occur only in the presence of the test conditions. DEFLAKER performs a preliminary analysis on the 12 of those projects. We also ran experiments on project test suites where DEFLAKER detected which flaky tests that were failures with a low false alarm rate (FAR). DEFLAKER had a higher recall (90.5%) of covered flaky tests than the current default flaky test detectors.

CCS CONCEPTS

Software and its engineering — Software testing and debugging

KEYWORDS

Software testing, flaky tests, code coverage

ACM Reference format

Jonathan Bell, Owolabi Ogunsan, Michael Hilton, Linyuan Euzou, Tiffany Yung, and Duke Mannov. 2018. DEFLAKER: Automatically Detecting Flaky Tests. In Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, June 17–19, 2018, Houston, Texas, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3192311>

Permission is granted to reproduce and distribute reprints for non-commercial use, provided the original work is properly cited. Copyright © 2018, Jonathan Bell, Owolabi Ogunsan, Michael Hilton, Linyuan Euzou, Tiffany Yung, and Duke Mannov. This is an open access article distributed under the terms of the [Creative Commons Attribution License \(CC BY 4.0\)](https://creativecommons.org/licenses/by/4.0/). This work is licensed under a Creative Commons Attribution 4.0 International License. For more information, see <https://creativecommons.org/licenses/by/4.0/>. Copyright held by the author(s). All rights reserved. ACM Reference format: Jonathan Bell, Owolabi Ogunsan, Michael Hilton, Linyuan Euzou, Tiffany Yung, and Duke Mannov. 2018. DEFLAKER: Automatically Detecting Flaky Tests. In Proceedings of ACM SIGPLAN Conference on Programming Language Design and Implementation, June 17–19, 2018, Houston, Texas, USA. ACM, New York, NY, USA, 11 pages. <https://doi.org/10.1145/3192311>

1 INTRODUCTION

Automated regression testing is widely used in modern software development. Whenever a developer makes some changes to a repository, tests are run to check whether the changes broke some functionality. Ideally, every new test failure would be due to the latest changes that the developer made and the developer could then investigate these failures. Unfortunately, some failures are not due to the latest changes but due to test conditions. Such problems occur, we define a flaky test as a test that can sometimes fail and sometimes pass on the same version of the code.

Flaky tests are frequent in many large software and create problems in development, as described by many researchers and practitioners [1, 2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30]. For example, according to Wang and Nagappan [23], the Microsoft Windows and Dynamics product teams estimate that approximately 10% of their test failures are due to flaky tests [23]. In an experiment [24], we report that 13.6% of tests in a Firefox driver failed because of flaky tests, and Lee et al. [12] reported that flaky tests occurred for 25% of the 1,000 tests. Many test failures in the Google TAP system are reported as flaky.

When a test fails, developers spend a lot of time to investigate that failure. It is often difficult to determine whether the failure is due to a flaky test or to a real problem in the code [20, 21]. The most widely used technique to identify flaky test failures, SURE [3], is to run each flaky test multiple times and identify the test as flaky if some runs passed, the test is still failing flaky, but if all runs fail, the status is unknown. This is supported by several testing frameworks, e.g., JUnit4 [21], JUnit5 [22], TestNG [7], Spring [16], and the Google TAP system [23, 24]. These techniques are generally costly to filter test failures because they repeatedly run flaky tests, which is not a good idea for developers.

There is little empirical guidance on how to detect flaky tests in order to reduce the likelihood of re-running the test [23]. Some might want to be alerted to allow the cause of the failure to be investigated to get to the root of it. We by test case were determined by definition, so there is no guidance that recognizing flaky test failures can reduce the number of flaky tests. It is well known that flaky tests can cause a lot of time, potentially by re-executing a flaky test over and over again. Recognizing a flaky test can reduce the number of test cases that are re-executed. In this paper, we present a technique to identify flaky tests. We evaluate our technique on 12 of those projects and report that DEFLAKER had a higher recall (90.5%) of covered flaky tests than the current default flaky test detectors [24].

Bell @ ICSE 2018

GitHub Project

achilles

alluxio

ambari

assertj-core

checkstyle

commons-exec

dropwizard

hadoop

handlebars

hbase

hector

httpcore

jackrabbit-oak

jimfs

logback

ninja

okhttp

oozie

orbit

oryx

spring-boot

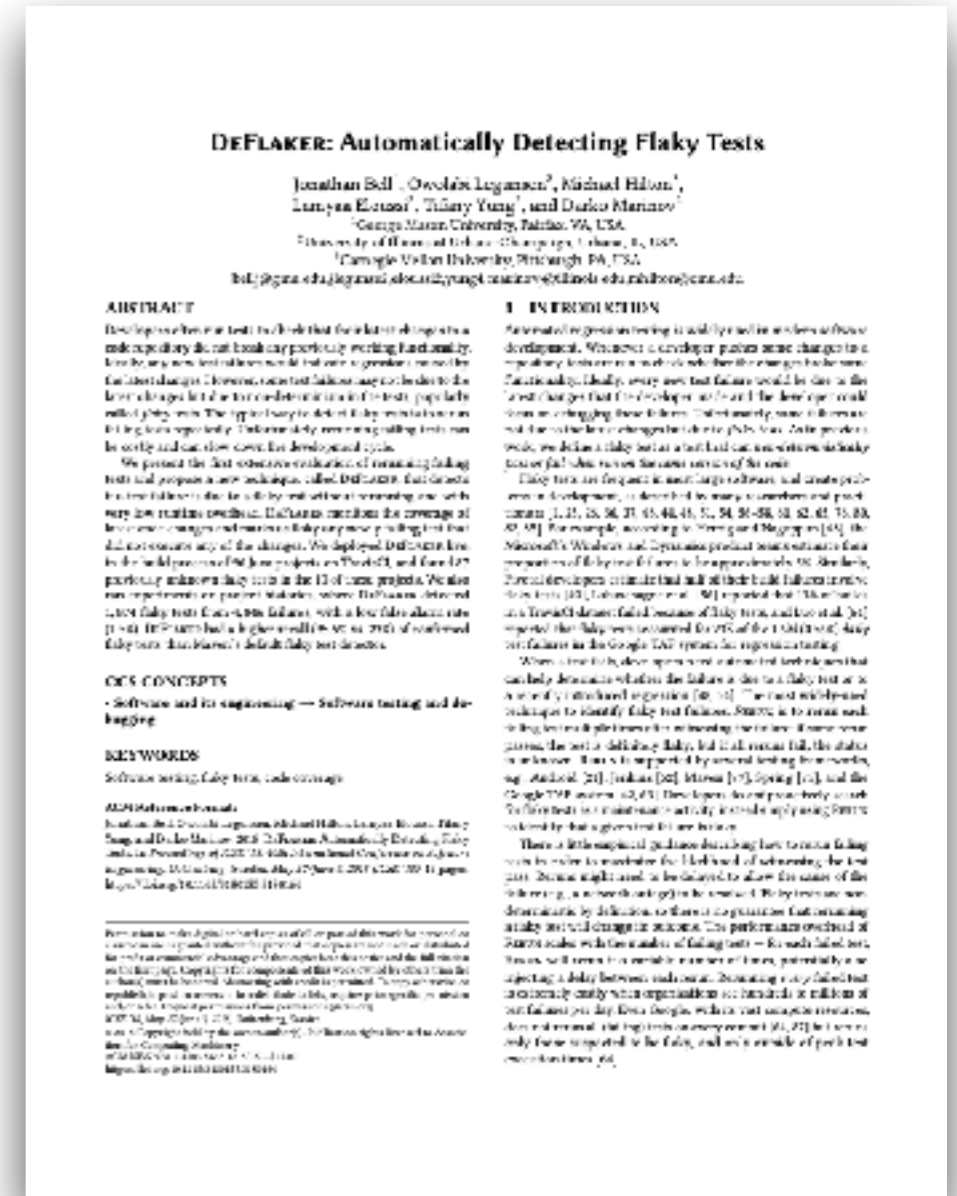
togglz

undertow

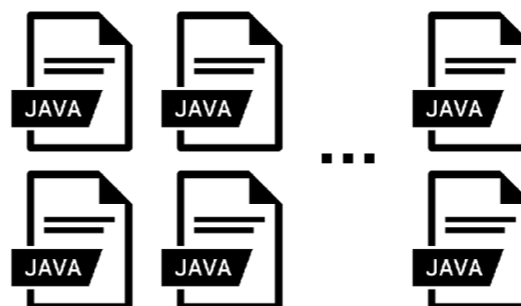
wro4j

zxing

The same studied here



Bell @ ICSE 2018





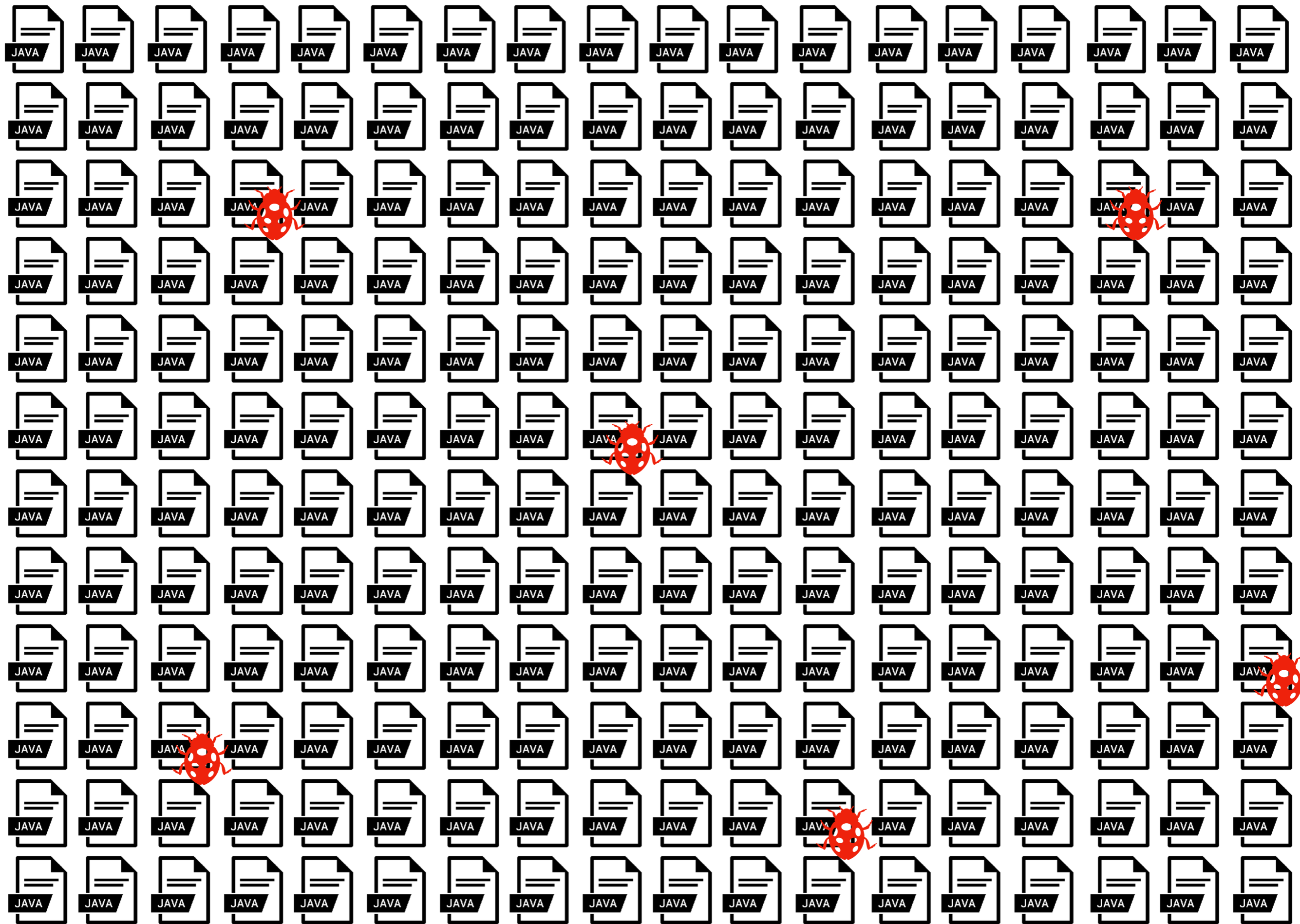
64k test files

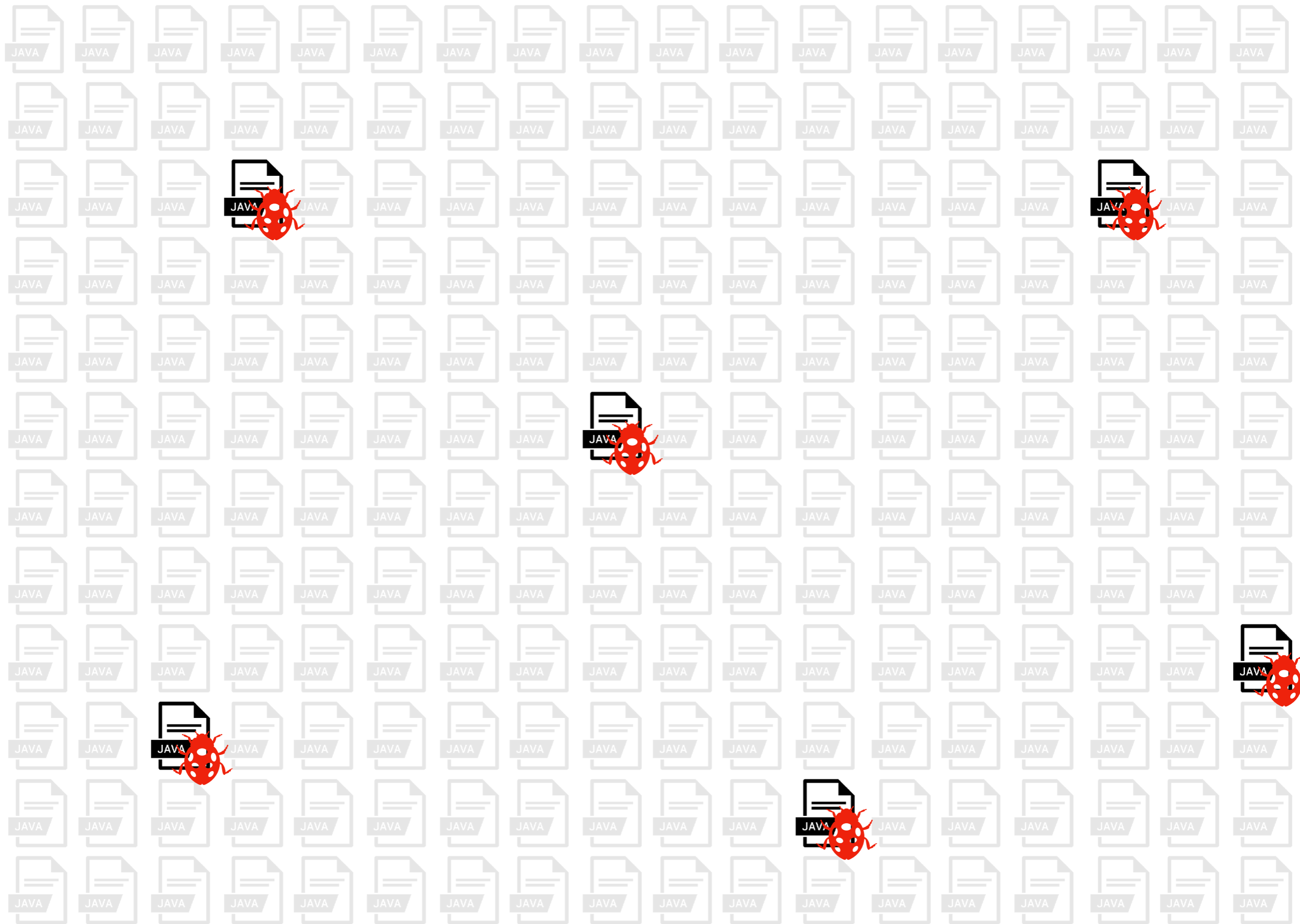


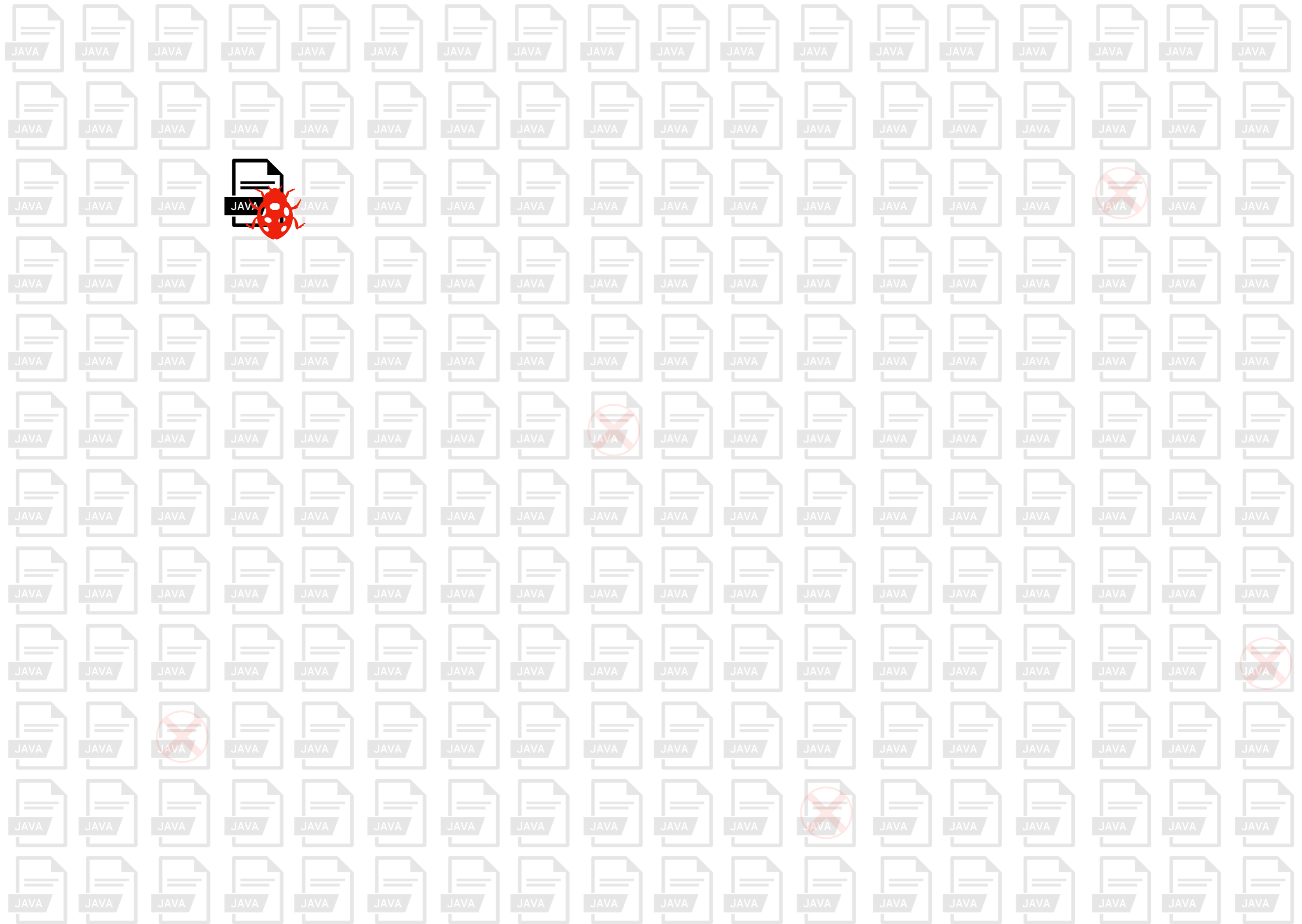


$64k * 100 =$
6.4 mi executions











```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

```
public class TestIdentifyEncoder {
    @Test
    public void testCodingEmptySrcBuffer() throws Exception {
        final WritableByteChannelMock channel = new WritableByteChannelMock(64);
        final SessionOutputBuffer outbuf = new SessionOutputBufferImpl(1024, 128);
        final HttpTransportMetricsImpl metrics = new HttpTransportMetricsImpl();

        final IdentityEncoder encoder = new IdentityEncoder(channel, outbuf, metrics);
        encoder.write(CodecTestUtils.wrap("stuff"));

        final ByteBuffer empty = ByteBuffer.allocate(100);
        empty.flip();
        encoder.write(empty);
        encoder.write(null);
        encoder.complete();

        outbuf.flush(channel);
        final String s = channel.dump(Constants.ASCII);

        Assert.assertTrue(encoder.isCompleted());
        Assert.assertEquals("stuff", s);
    }
}
```

```

public class test identify encoder {
    @Test
    public void test coding empty src buffer() throws exception {
        final writable byte channel mock channel = new writable byte channel mock(64);
        final session output buffer outbuf = new session output buffer impl(1024, 128);
        final http transport metrics impl metrics = new http transport metrics impl();

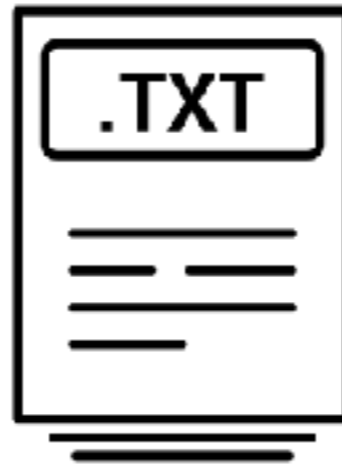
        final identity encoder encoder = new identity encoder channel outbuf metrics
encoder.write(codec test utils wrap("stuff"));

        final byte buffer empty = byte buffer.allocate(100);
empty.flip();
encoder.write(empty);
encoder.write(null);
encoder.complete();

outbuf.flush(channel);
final string s = channel.dump(consts.ascii);

assert.assertTrue(encoder.is completed());
assert.assertEquals("stuff", s);
    }
}

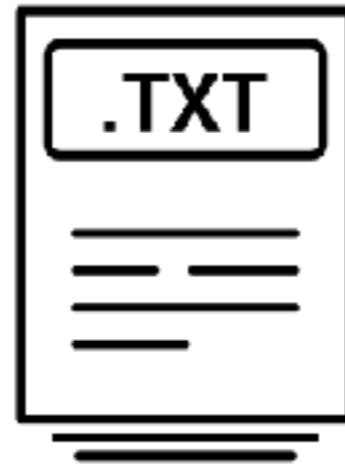
```



```
test identify encoder coding empty src buffer  
    byte channel mock writable session  
output outbuf session http transport metrics  
impl identity codec utils wrap buffer allocate  
flip write complete flush dump consts ascii is  
    completed assert equals
```



Known Flaky test



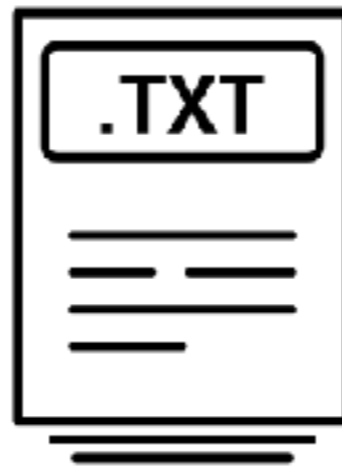
Tokenized Flaky test



**Machine Learning
algorithms**



Known Flaky test



Tokenized Flaky test



Machine Learning algorithms



Weka

Nearest Neighbour

Support Vector
Machine

Decision Tree

Naive Bayes

Random Forest

Implementation available at:
<https://github.com/damorimRG/msr4flakiness/>

 @gustavopinto

RQ1:

How prevalent
are flaky tests?

RQ1: How prevalent are flaky tests?

GitHub Project

- achilles
- aluxio
- ambari
- asserj-core
- chevy-style
- commons-iced
- dropwizard
- hadoop
- lwdelivers
- noase
- honor
- Introscope
- jackrabbit-oak
- jinfa
- logback
- mya
- olshrp
- oculie
- rebit
- onyx
- spring-bact
- testkit
- wtocj
- zong

The same studied here →

Bell @ ICSE 2018

@gustavo

64k * 100 =
6.4 mi executions

@gustavopinto

@gustavopinto

RQ1: How **prevalent** are flaky tests?

Project	# Test	# Flaky	% Flaky
alluxio	3,034	12	0.4
hector	322	40	12.4
jackrabbit-oak	13,193	2	2
okhttp	1,682	19	19
undertow	609	2	2
wro4j	1,158	11	11

25% of the projects
have at least one flaky test

We found
86 flaky tests

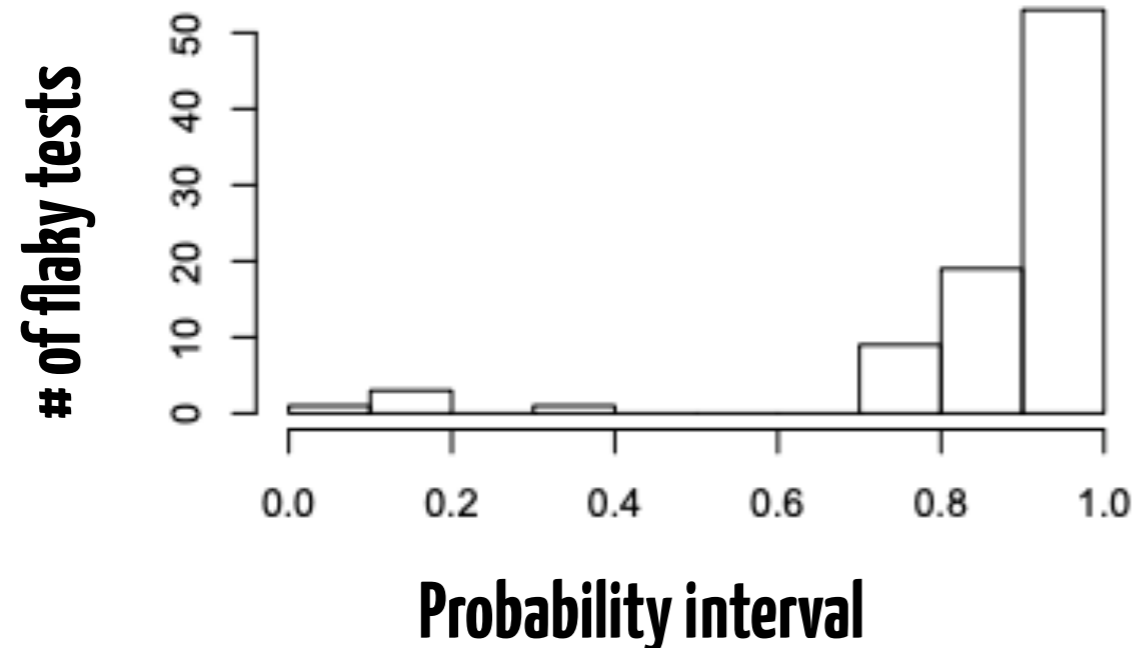
RQ1: How **prevalent** are flaky tests?

Project	# Test	# Flaky	% Flaky
alluxio	3,034	12	0.4
hector	322	40	12.4
jackrabbit-oak	13,193	2	2
okhttp	1,682	19	19
undertow	609	2	2
wro4j	1,158	11	11

25% of the projects
have at least one flaky test

We found
86 flaky tests

70% (61 out of the 86)
passed more than 90%



RQ2:

How accurately can
we **predict** test
flakiness?

RQ2: Can we **predict** flakiness?

ML	Precision	Recall	F1	MCC	AUC
Random Forest	0.99	0.91	0.95	0.90	0.98
Decision Tree	0.98	0.88	0.89	0.77	0.91
Naive Bayes	0.93	0.80	0.86	0.74	0.93
Support Vector	0.93	0.92	0.93	0.85	0.92
Nearest Neighbour	0.97	0.88	0.92	0.85	0.93

Random Forest
achieved best precision

Tuning (e.g., # of trees)
**Had no performance
impact**

RQ3:

What value do

different features add

to the classifier?

RQ3: What's the value of different features

Random Forest

Features	Precision	Recall	F1	MCC	AUC
All features	0.99	0.91	0.95	0.90	0.98
No stemming	0.99	0.91	0.95	0.90	0.98
No Stop W. removal	0.99	0.91	0.95	0.90	0.98
No Lowercasing	0.98	0.91	0.94	0.89	0.98
No Java Keywords	0.99	0.90	0.94	0.89	0.98

RQ3: What's the value of different features

Random Forest

Features	Precision	Recall	F1	MCC	AUC
All features	0.99	0.91	0.95	0.90	0.98
No stemming	0.99	0.91	0.95	0.90	0.98
No Stop W. removal	0.99	0.91	0.95	0.90	0.98
No Lowercasing	0.98	0.91	0.94	0.89	0.98
No Java Keywords	0.99	0.90	0.94	0.89	0.98

Support Vector Machine

Features	Precision	Recall	F1	MCC	AUC
All features	0.93	0.92	0.93	0.85	0.93
No stemming	0.93	0.92	0.93	0.85	0.93
No Stop W. removal	0.93	0.92	0.93	0.85	0.93
No Lowercasing	0.91	0.93	0.92	0.84	0.92
No Java Keywords	0.93	0.92	0.93	0.85	0.93

RQ3: What's the value of different features

Random Forest

Features	Precision	Recall	F1	MCC	AUC
All features	0.99	0.91	0.95	0.90	0.98
No stemming	0.99	0.91	0.95	0.90	0.98
No Stop W. removal	0.99	0.91	0.95	0.90	0.98
No Lowercasing	0.98	0.91	0.94	0.89	0.98
No Java Keywords	0.99	0.90	0.94	0.89	0.98

Support Vector Machine

Features	Precision	Recall	F1	MCC	AUC
All features	0.93	0.92	0.93	0.85	0.93
No stemming	0.93	0.92	0.93	0.85	0.93
No Stop W. removal	0.93	0.92	0.93	0.85	0.93
No Lowercasing	0.91	0.93	0.92	0.84	0.92
No Java Keywords	0.93	0.92	0.93	0.85	0.93

**No performance
impact**

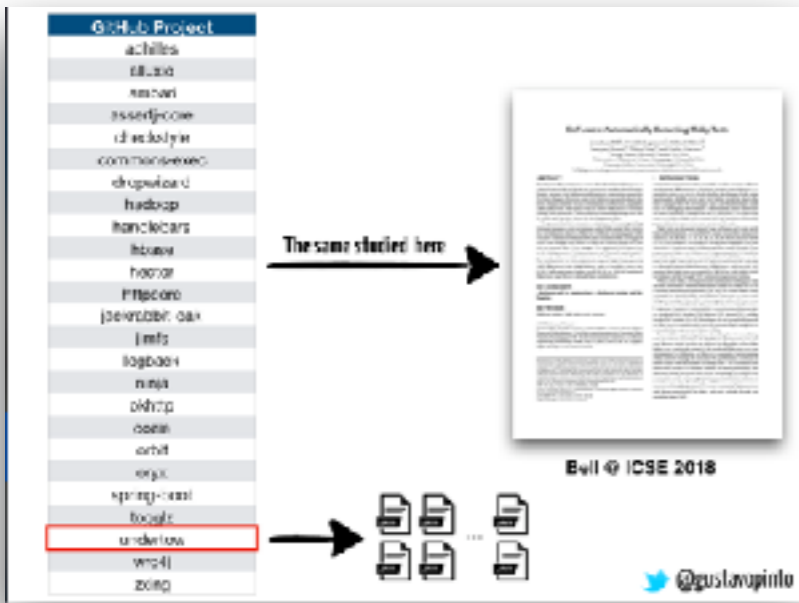
RQ4:

Which **test code identifiers** are strongly associated with test flakiness?

RQ4: What's the **vocabulary**?

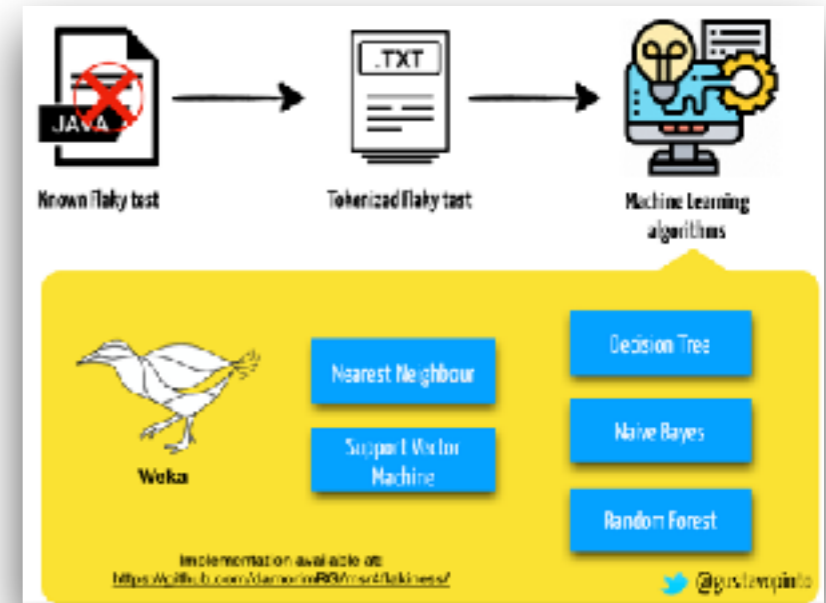
Features	inf. gain	Flaky		Non-Flaky	
		# test	# projects	# test	# projects
job	0.20	524	2	4	1
table	0.14	406	4	8	2
id	0.14	552	9	52	4
action	0.13	387	3	8	2
oozie	0.13	274	1	0	0
services	0.13	371	2	7	1
coord	0.11	307	1	0	0
getid	0.11	287	4	1	1
coordinator	0.10	258	1	0	0
xml	0.10	147	2	6	2
workflow	0.09	207	1	0	0
getstatus	0.08	246	2	2	2
record	0.08	296	2	18	1
jpa	0.07	207	2	0	0
jpaservice	0.07	200	1	0	0
service	0.07	367	4	67	3

Many of them associated with
remote tasks
and/or
queue events



64k * 100 =
6.4 mi executions

@gustavopinto



RQ1: How prevalent are flaky tests?

Project	# Test	# Flaky	% Flaky
akka	3,034	12	0.4
hadoop	127	40	31.4
jackson-core	13,101	2	0.015
okhttp	1,842	19	1.0
undertow	99	2	2.0
wro4j	1,158	1	0.086

25% of the projects have at least one flaky test

We found 86 flaky tests

70% of our tests passed more than 90%

RQ3: What's the value of different features?

Random Forest

Features	Precision	Recall	F1	MCC	AUC
All features	0.99	0.91	0.95	0.90	0.98
No stemming	0.99	0.91	0.95	0.90	0.98
No Stop-Word removal	0.99	0.91	0.95	0.90	0.98
No Lowercasing	0.98	0.91	0.94	0.89	0.98
No Java Keywords	0.99	0.90	0.94	0.89	0.98

Support Vector Machine

Features	Precision	Recall	F1	MCC	AUC
All features	0.90	0.82	0.86	0.75	0.93
No stemming	0.90	0.82	0.86	0.75	0.93
No Stop-Word removal	0.90	0.82	0.86	0.75	0.93
No Lowercasing	0.91	0.81	0.86	0.74	0.92
No Java Keywords	0.90	0.82	0.86	0.75	0.93

No performance impact

RQ2: Can we predict flakiness?

ML	Precision	Recall	F1	MCC	AUC
Random Forest	0.99	0.91	0.95	0.90	0.98
Decision Tree	0.98	0.88	0.93	0.77	0.91
Naive Bayes	0.93	0.87	0.90	0.74	0.93
Support Vector	0.93	0.82	0.87	0.75	0.92
Nearest Neighbour	0.97	0.88	0.92	0.85	0.93

Random Forest achieved best precision

Tuning (e.g. # of trees) had no performance impact

@gustavopinto

RQ4: What's the vocabulary?

Features	inf. gain	Flaky		Non-Flaky	
		# test	# projects	# test	# projects
job	0.20	524	2	4	1
table	0.14	405	4	8	2
id	0.14	552	5	32	4
action	0.13	387	5	8	2
code	0.13	274	1	0	0
variables	0.13	371	2	7	1
method	0.11	367	1	0	0
grep	0.11	387	1	1	1
coordinator	0.10	256	1	0	0
url	0.10	547	2	6	2
workflow	0.09	307	1	0	0
github.com	0.08	246	2	2	2
record	0.08	286	2	18	1
ip	0.07	207	2	0	0
password	0.07	200	1	0	0
docker	0.07	367	4	67	2

Common words: job, table, and action

Many of them associated with remote tasks and/or queue events

@gustavopinto