



Lightweight Assessment of Test-Case Effectiveness Using Source-Code-Quality Indicators

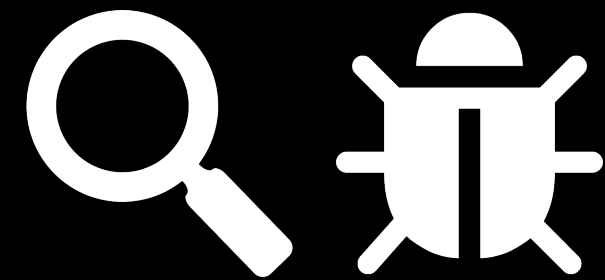
Giovanni Grano, Fabio Palomba, Harald C. Gall



University of
Zurich^{UZH}

<http://tiny.uzh.ch/Zx>

effectiveness



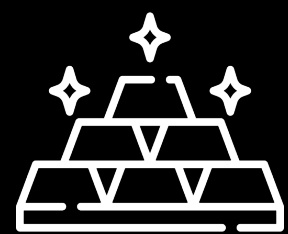
detect bugs



how do you measure it?

real-faults

code-coverage

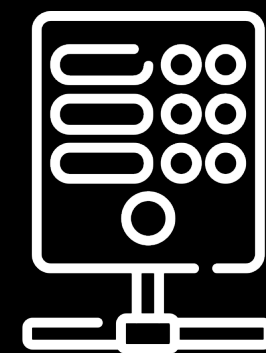


ideal

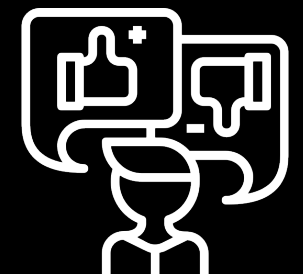


unknown

proxy



coverage 100%

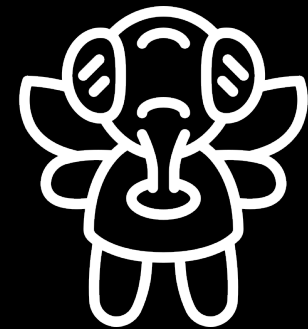


behavior

measuring effectiveness

mutation testing

mutants



mutation score

$$\frac{\text{detected (killed) mutants}}{\text{generated mutants}}$$

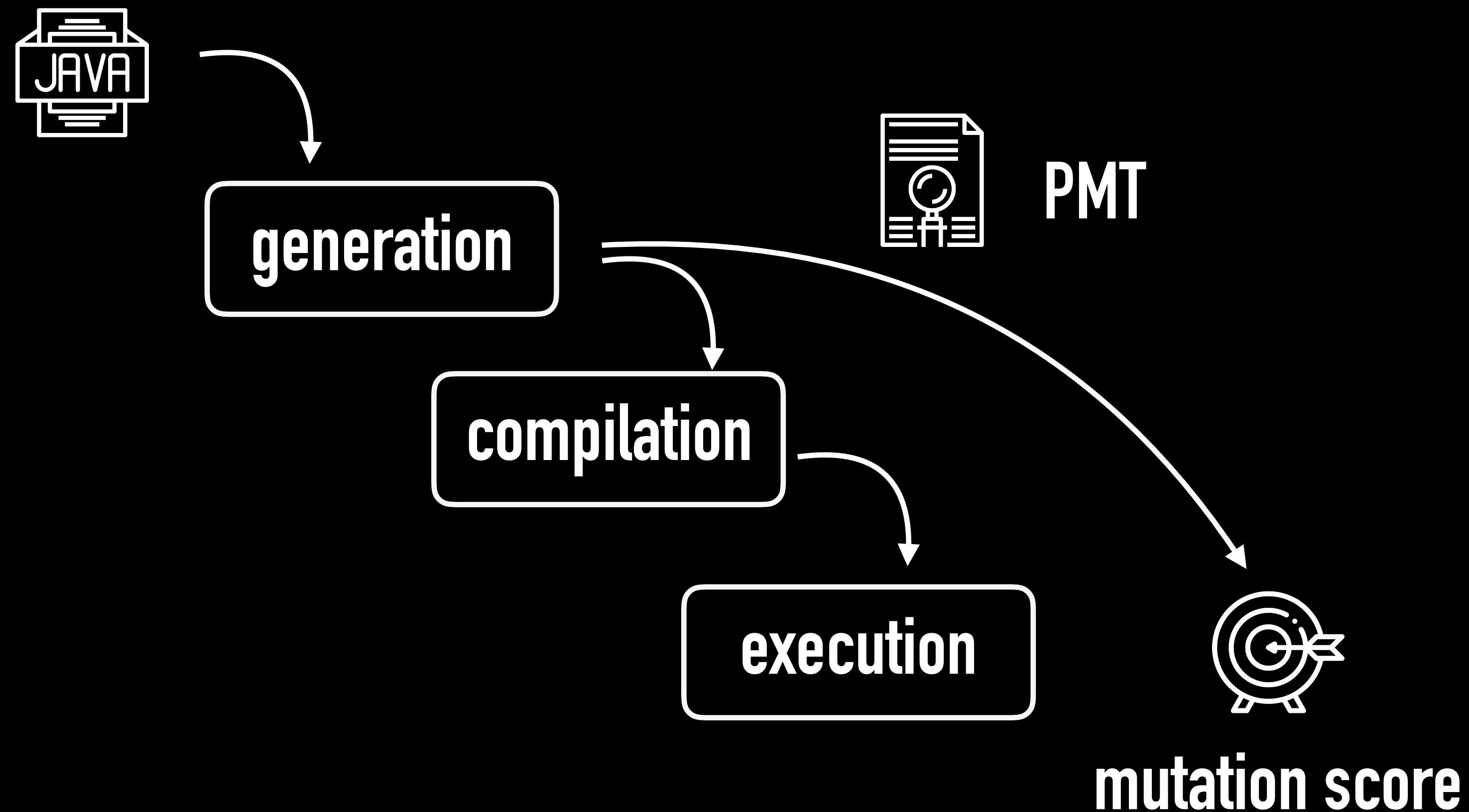
```
if (a && b)
    c = c + 1;
else
    c = 0;
```



```
if (a || b)
    c = c + 1;
else
    c = 0;
```

powerful...

...but expensive



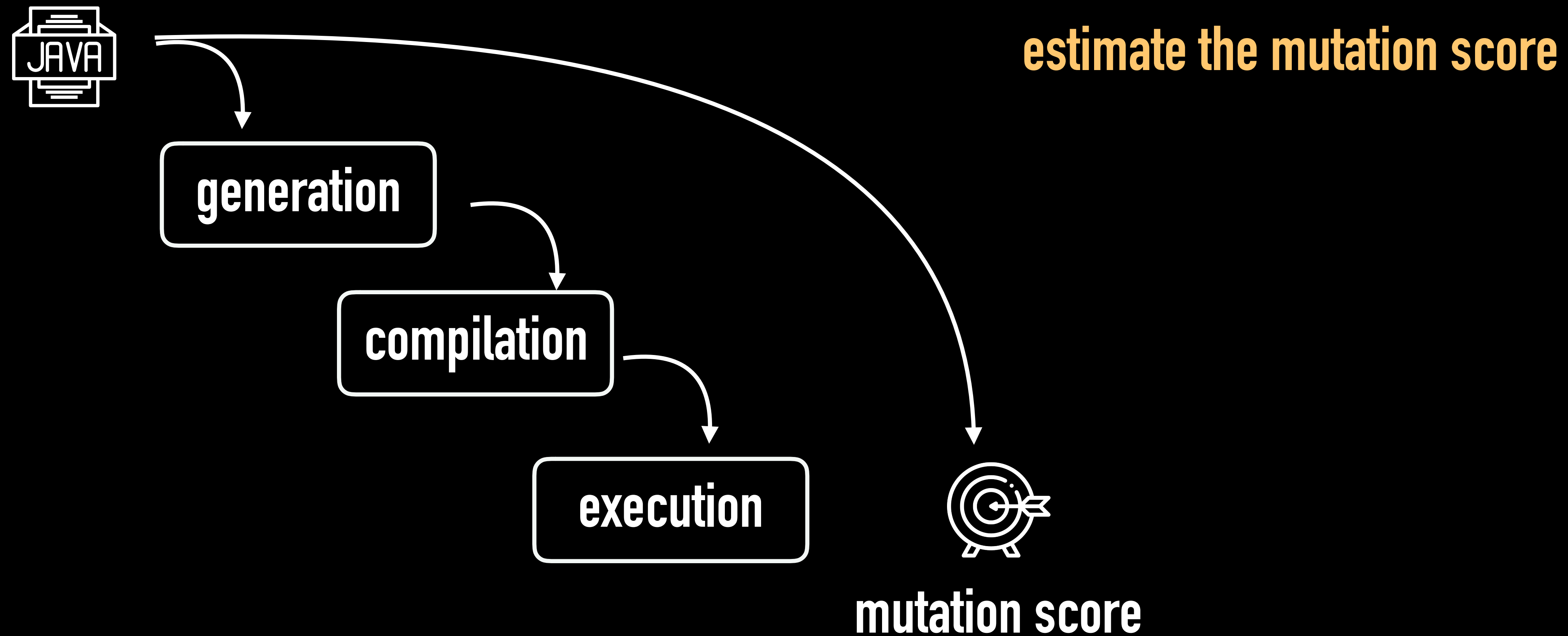
reduce the cost

do fewer

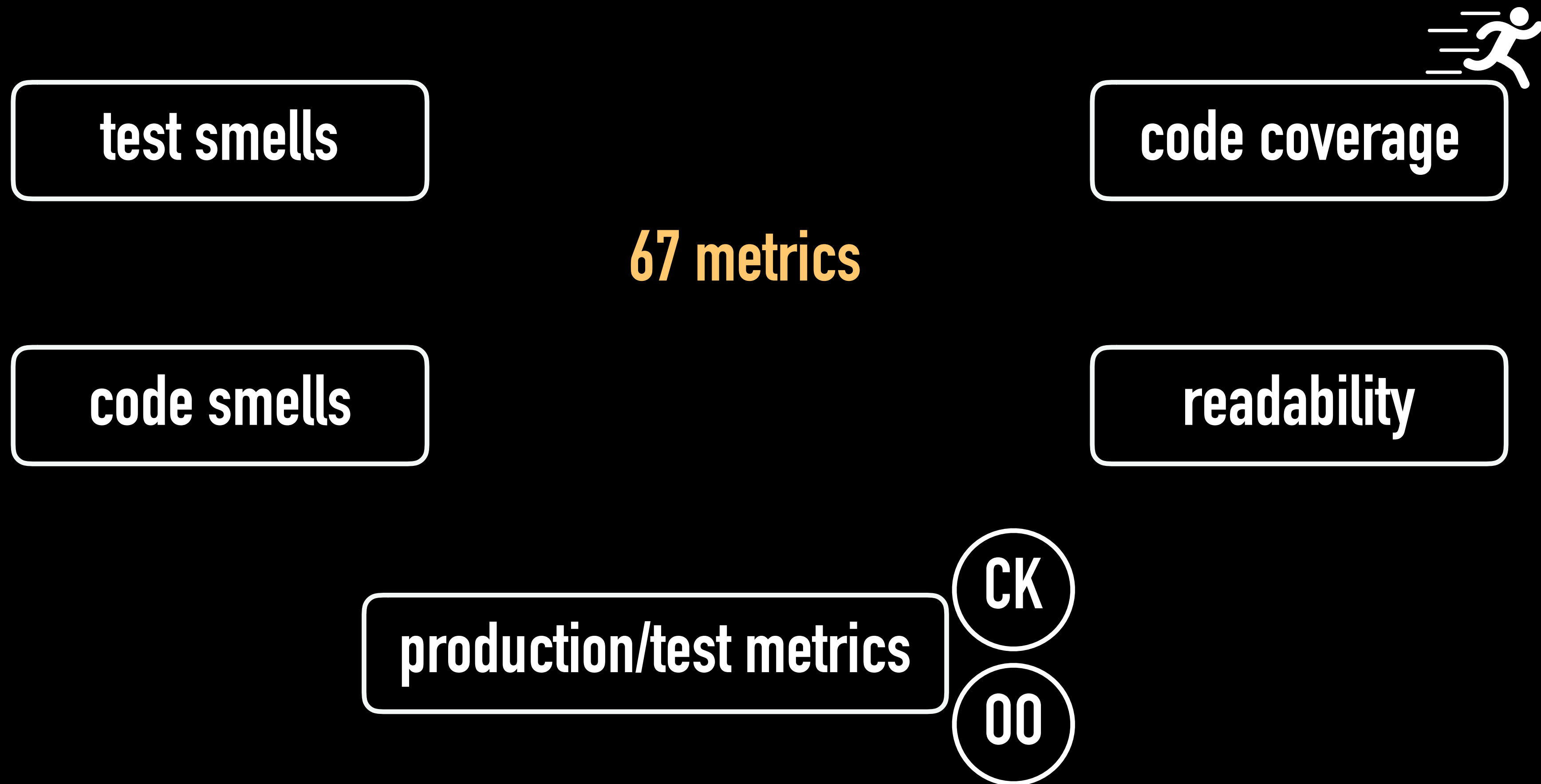
do smarter

do faster

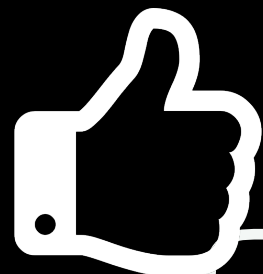
our approach



source-code metrics



feasibility study



high mutation
score



low mutation
score

distribution

41 metrics

DIMENSION	METRICS	REL	D-VALUE
Coverage	statement coverage	+	0.84 (large)
Test Smells	Eager Test	-	0.31 (small)
CUT's Code Metrics	LOC	-	0.43 (medium)
	HALSTEAD	-	0.40 (medium)
	RFC	-	0.62 (large)
	CBO	-	0.38 (medium)
	MPC	-	0.58 (large)
	IFC	-	0.29 (small)
	DAC	-	0.35 (medium)
	DAC2	-	0.34 (medium)
	LCOM1	-	0.60 (large)
	LCOM2	-	0.49 (large)
	LCOM3	-	0.38 (medium)
	LCOM4	-	0.49 (large)
	CONNECTIVITY	-	0.15 (small)
	LCOM5	-	0.39 (medium)
	COH	-	0.37 (medium)
	TCC	-	0.33 (medium)
	LCC	-	0.39 (medium)
	ICH	-	0.36 (medium)
	WMC	-	0.61 (large)
	NOA	-	0.35 (medium)
Test Code Metrics	NOPA	-	0.23 (small)
	NOP	-	0.44 (medium)
	McCABE	-	0.62 (large)
	LOC	+	0.22 (small)
	HALSTEAD	+	0.17 (small)
	RFC	+	0.37 (medium)
	MPC	+	0.34 (medium)
	LCOM1	+	0.44 (medium)
	LCOM2	+	0.40 (medium)
	LCOM4	+	0.34 (medium)
	CONNECTIVITY	+	0.25 (small)
	LCC	+	0.35 (medium)
Code Smell	ICH	+	0.19 (small)
	WMC	+	0.45 (medium)
Readability	McCABE	+	0.40 (medium)
	MC	+	0.33 (medium)
	FE	-	0.31 (small)
	production	-	0.19 (small)
	test	-	0.18 (small)

machine learning problem

binary classification

**0 low score
1 high score**

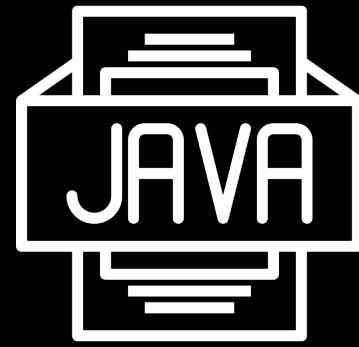
3 ML algorithms

**random forest
k-neighbors
support vector machine**

2 models

**all features
only static**

approach



pairs

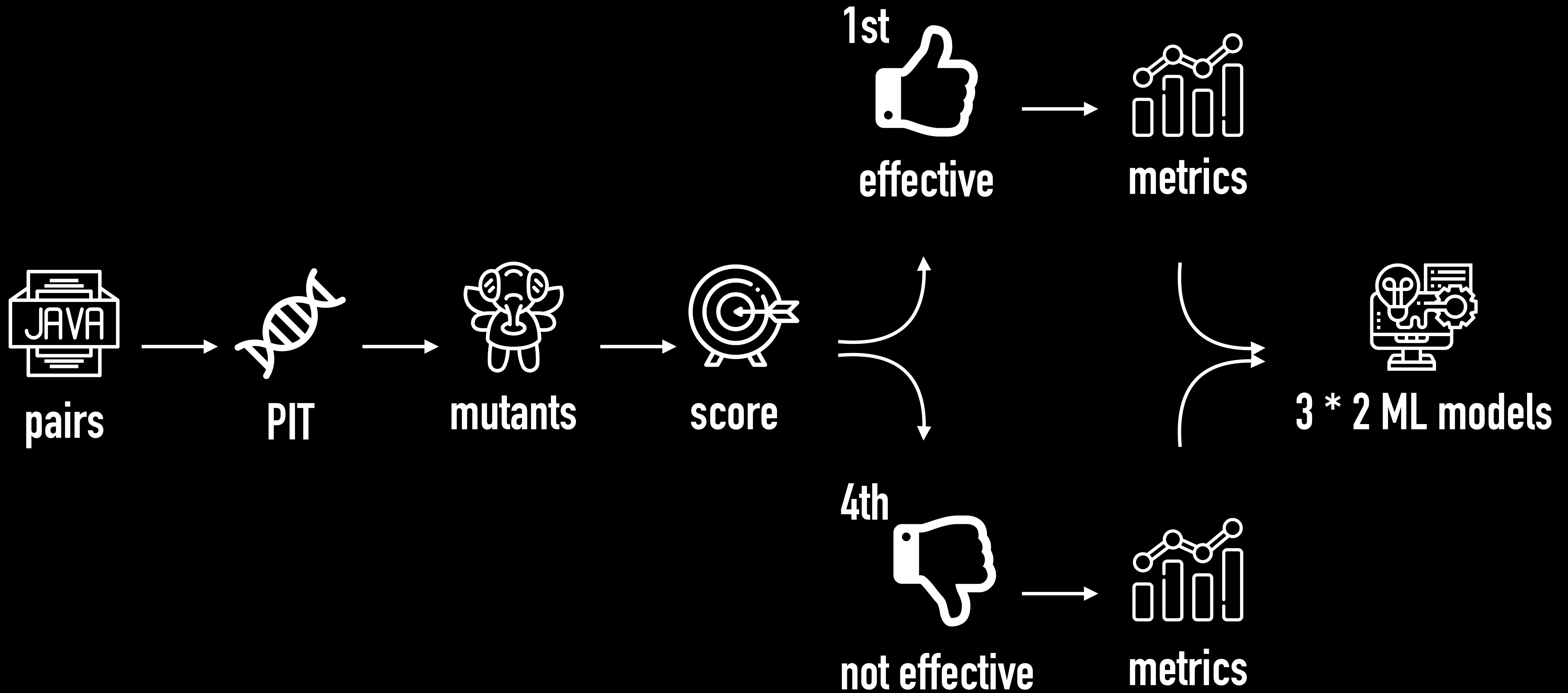
```
<build>
  <sourceDirectory>/src/main/java</sourceDirectory>
  <testSourceDirectory>/src/test/java</testSourceDirectory>
  ...
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
      <includes>
        <include>**/*TestCase.java</include>
      </includes>
      <excludes>
        <exclude>**/*MemoryTestCase.java</exclude>
      </excludes>
    </configuration>
  </plugin>
</build>
```



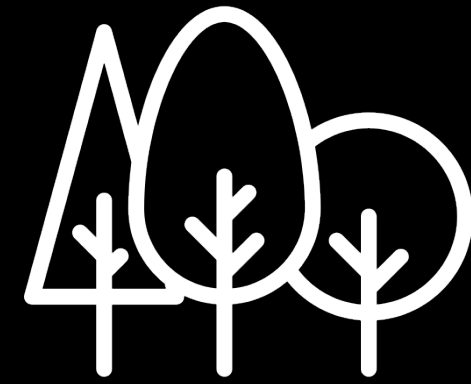
```
<build>
  <sourceDirectory>/src/main/java</sourceDirectory>
  <testSourceDirectory>/src/test/java</testSourceDirectory>
  ...
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
      <includes>
        <include>**/*TestCase.java</include>
      </includes>
      <excludes>
        <exclude>**/*MemoryTestCase.java</exclude>
      </excludes>
    </configuration>
  </plugin>
</build>
```

```
<build>
  <sourceDirectory>/src/main/java</sourceDirectory>
  <testSourceDirectory>/src/test/java</testSourceDirectory>
  ...
  <plugin>
    <groupId>org.apache.maven.plugins</groupId>
    <artifactId>maven-surefire-plugin</artifactId>
    <configuration>
      <includes>
        <include>**/*TestCase.java</include>
      </includes>
      <excludes>
        <exclude>**/*MemoryTestCase.java</exclude>
      </excludes>
    </configuration>
  </plugin>
</build>
```


approach



results



random forest

dynamic model

0.949 AUC

static model

0.864 AUC

estimation of the effectiveness without actually run any test

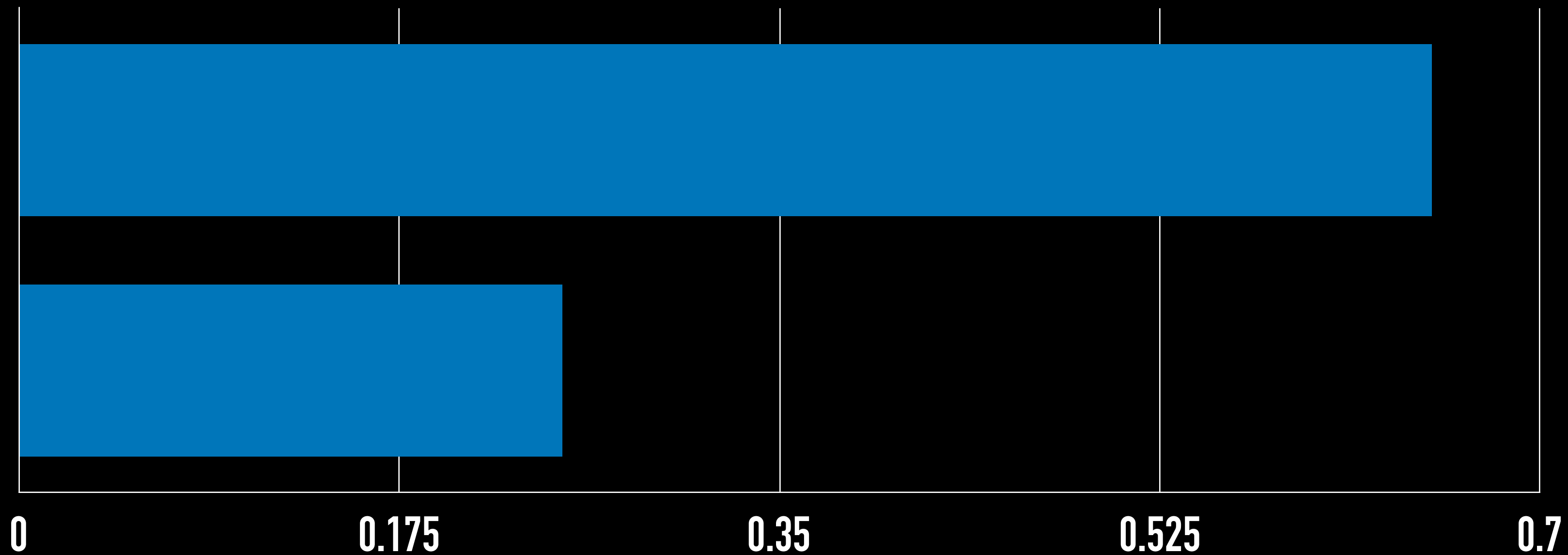
important factors

mean decrease in impurity

dynamic model

line coverage

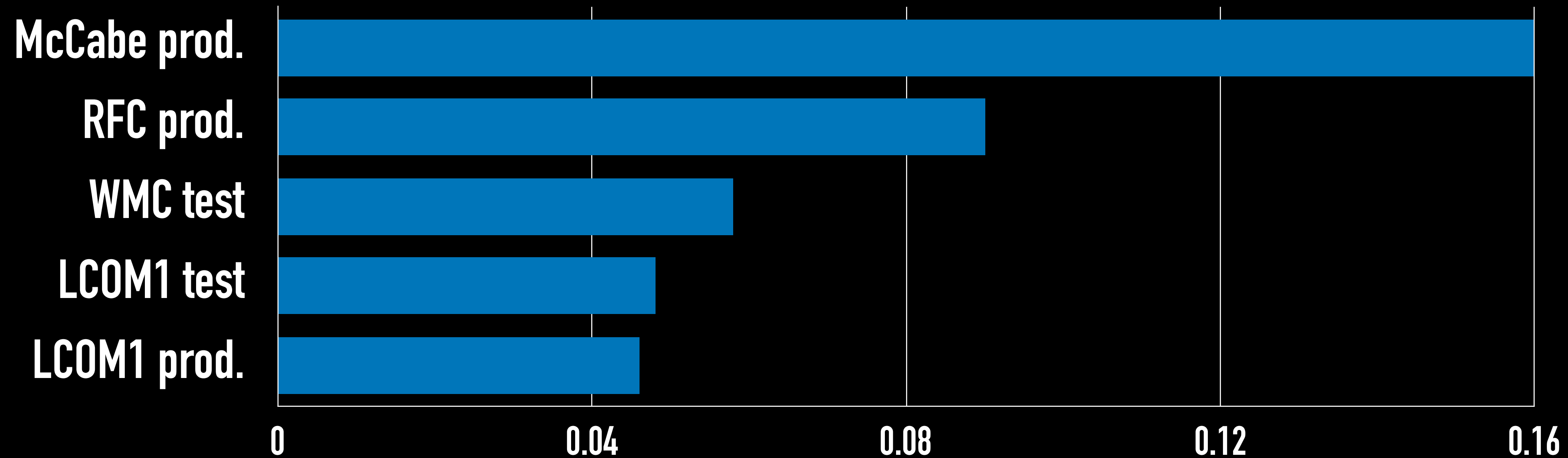
others



important factors

mean decrease in impurity

static model



practical usage

analytics dashboards



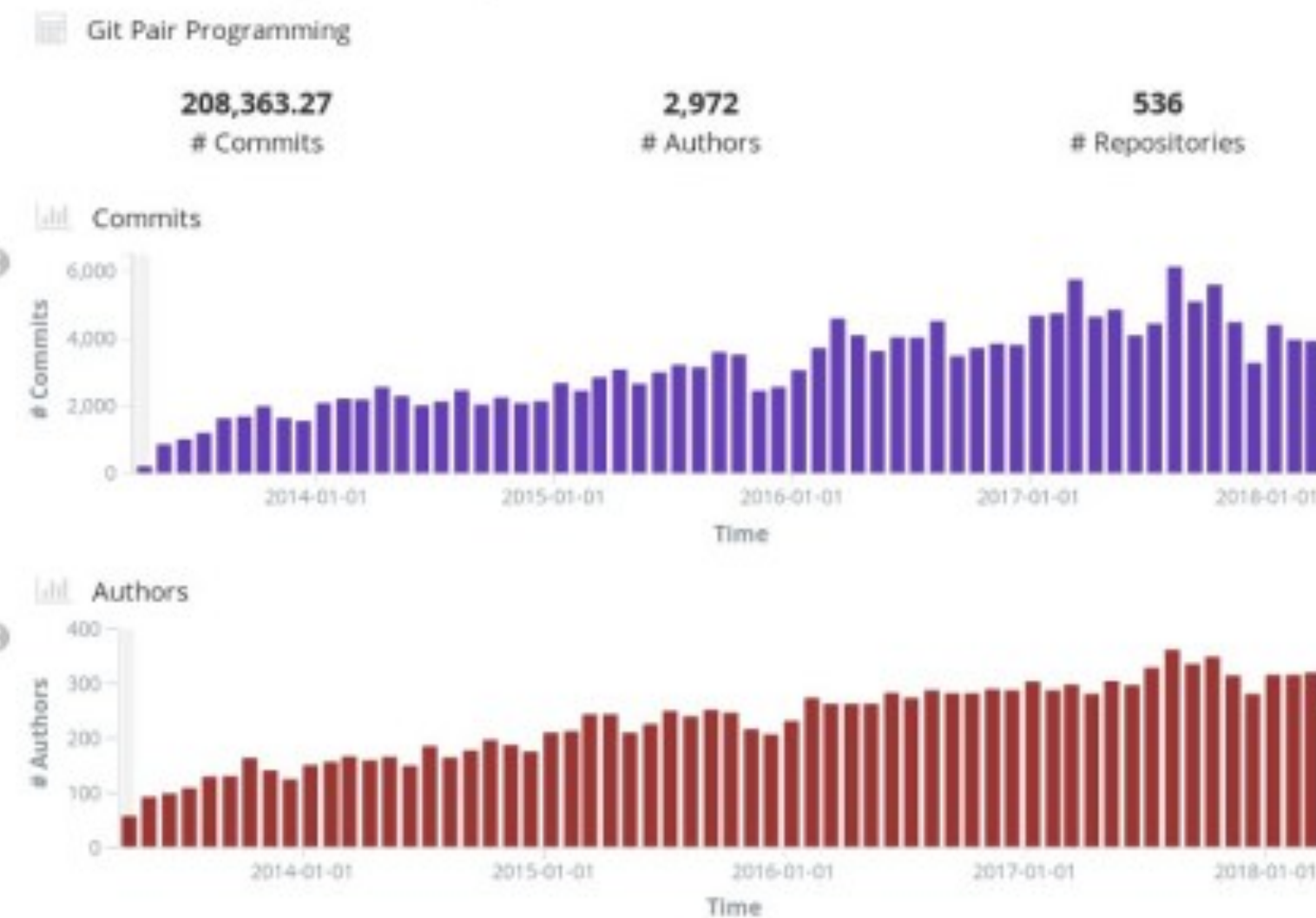
Cloud Foundry

- Overview
- Git
- GitHub PRs
- GitHub Issues
- Mailing Lists
- Slack
- Meetup
- Data Status
- About

GitHub Issues



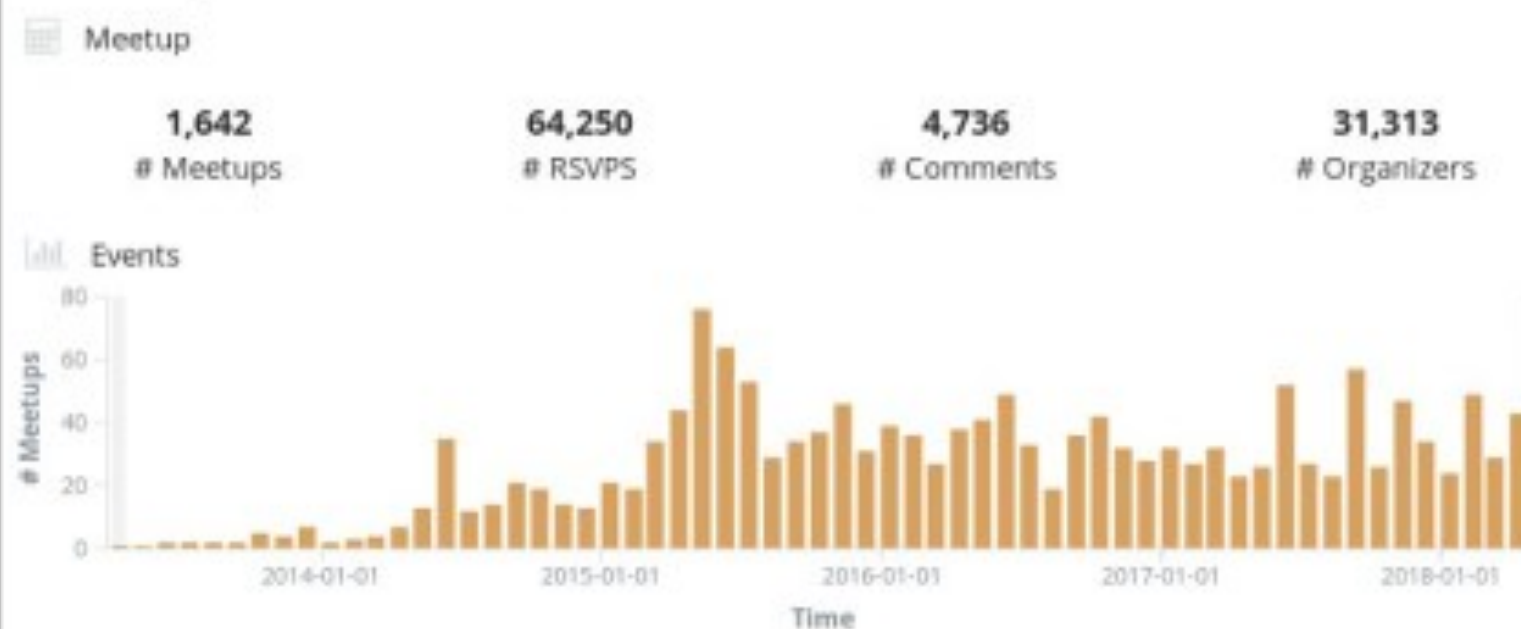
Git Pair Programming



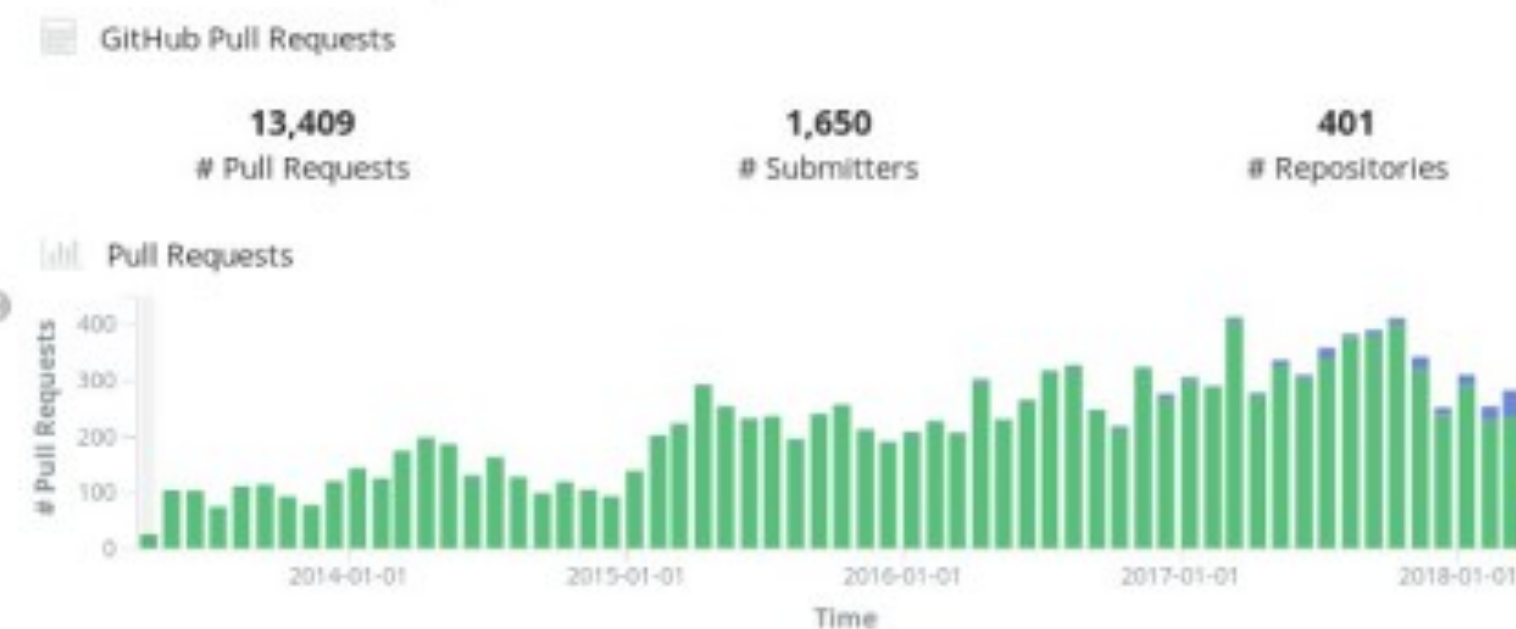
Mailing Lists



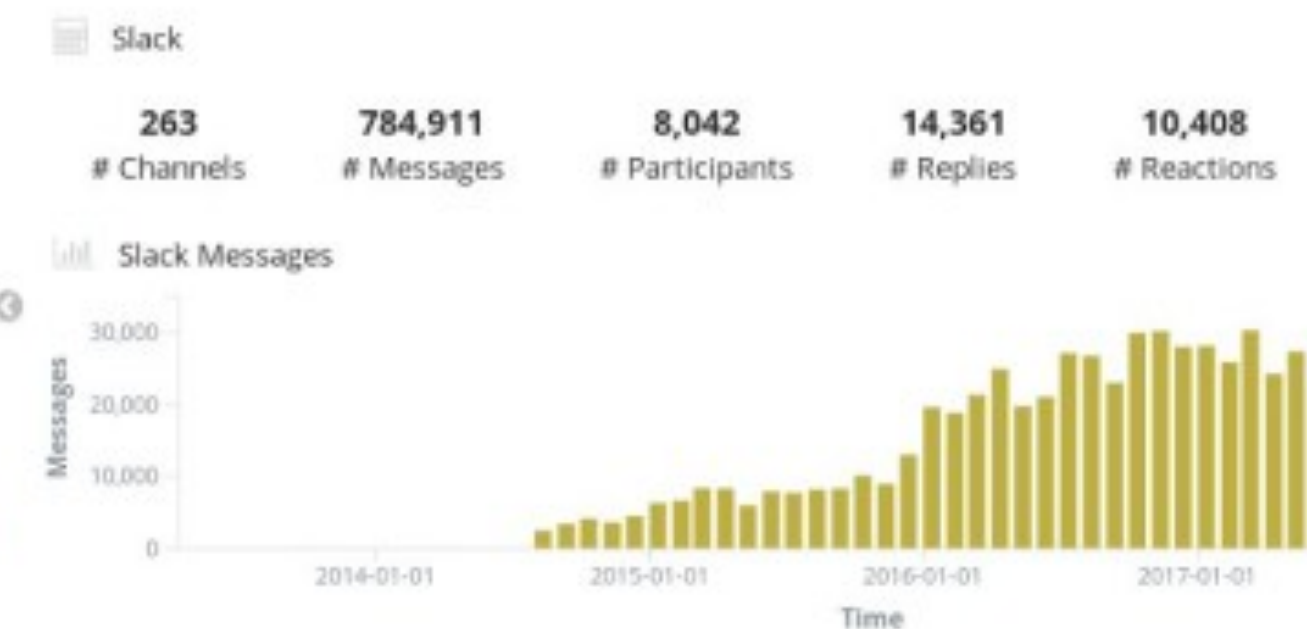
Meetup



GitHub Pull Requests

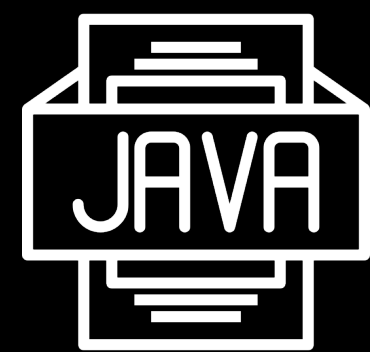


Slack



practical usage

complementarity



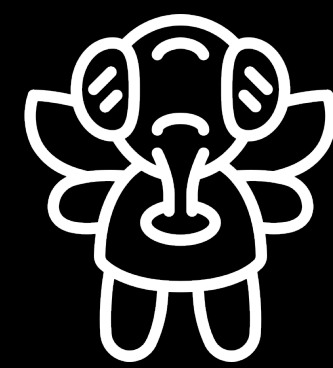
pairs



prediction



oops...not effective



fine-grained
investigation

measuring effectiveness

mutation testing

mutants



mutation score

$\frac{\text{detected (killed) mutants}}{\text{generated mutants}}$

```
if (a && b)
  c = c + 1;
else
  c = 0;
```

```
if (a || b)
  c = c + 1;
else
  c = 0;
```

our approach



generation

compilation

execution

estimate the mutation score



mutation score

results



random forest

dynamic model

0.949 AUC

static model

0.864 AUC

estimation of the effectiveness without actually run any test



University of Zurich^{UZH}

<http://tiny.uzh.ch/Zx>