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

Giovanni Grano, Fabio Palomba, Harald C. Gall

@giograno90

University of Zurich[UZH]

http://tiny.uzh.ch/Zx

34th IEEE/ACM International Conference on Automated Software Engineering 2019 (San Diego)

effectiveness

detect bugs

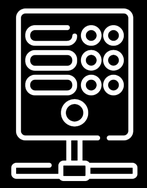
how do you measure it?

real-faults → code-coverage

proxy

ideal

unknown

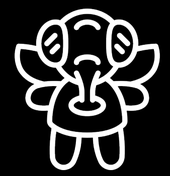


behavior

measuring effectiveness

mutation testing

mutants



mutation score

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

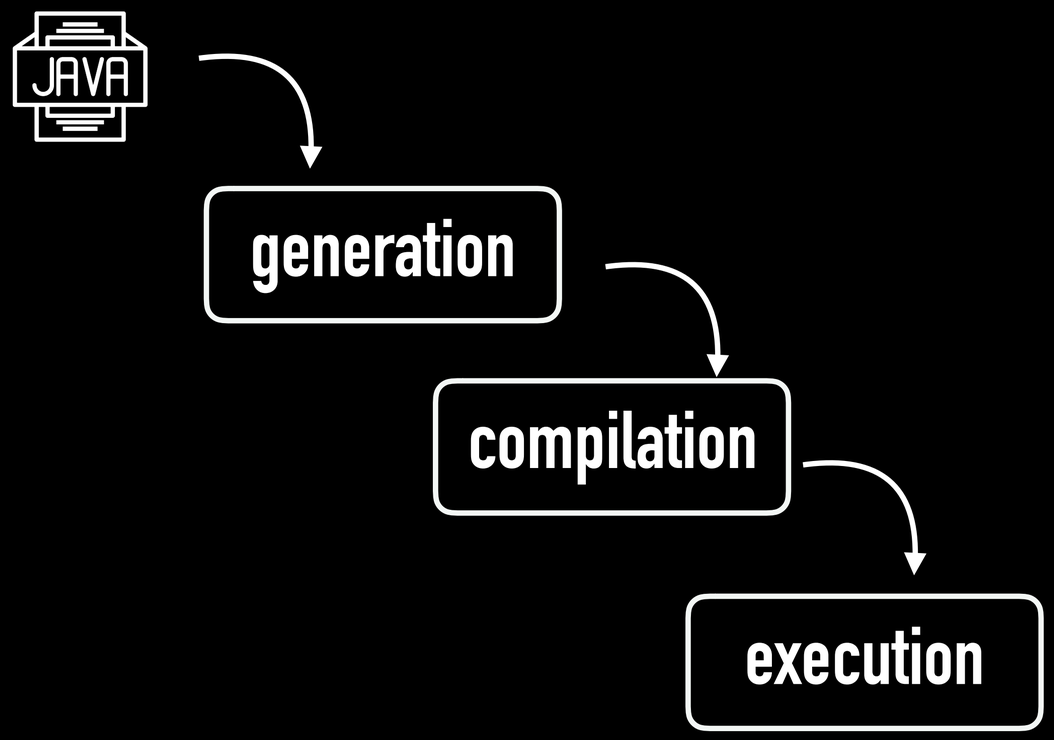
```
if (a && b)
    c = c + 1;
else
    c = 0;
```



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

powerful. . .

. . . but expensive

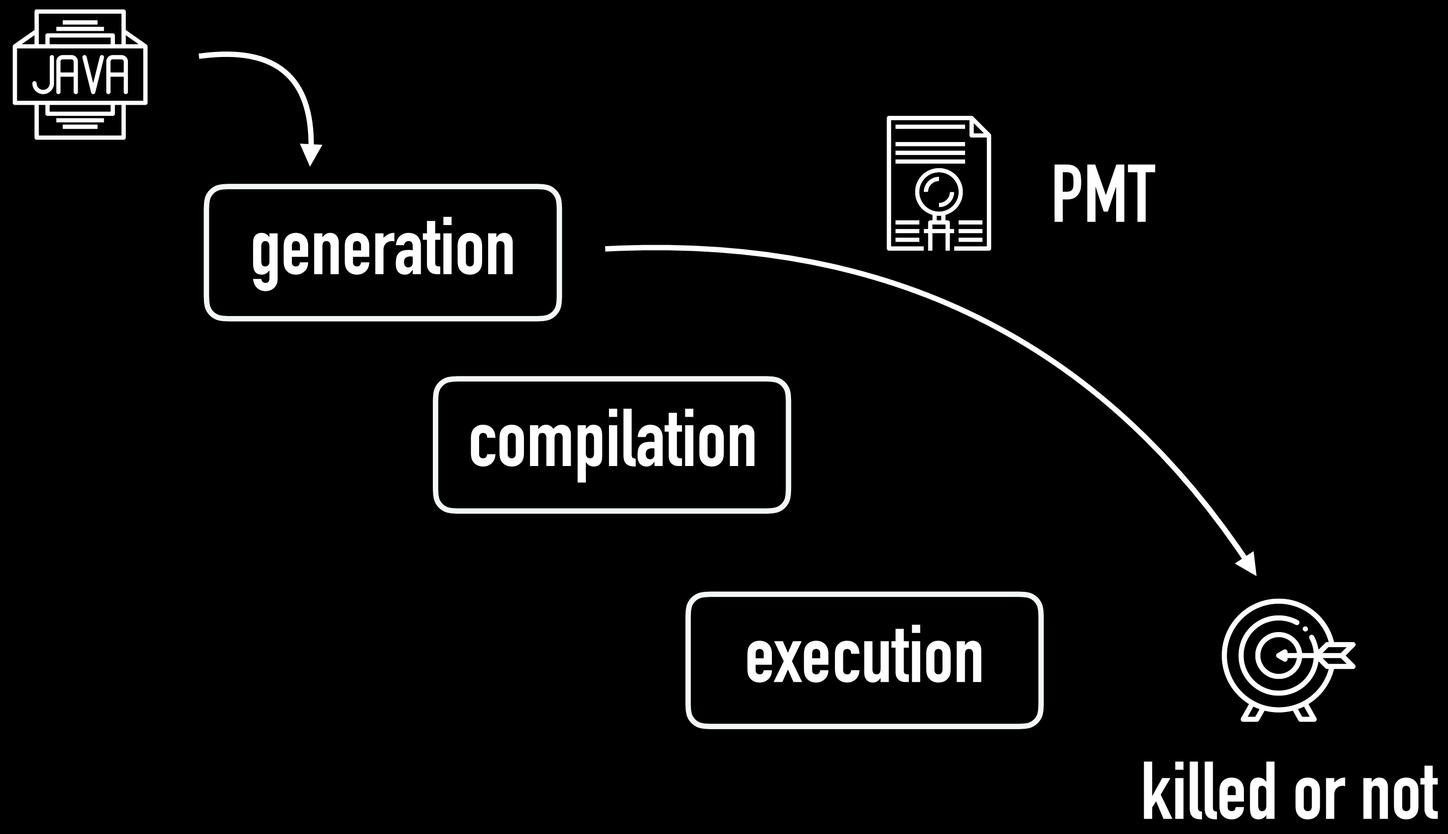


reduce the cost

- do fewer
- do smarter
- do faster

powerful. . .

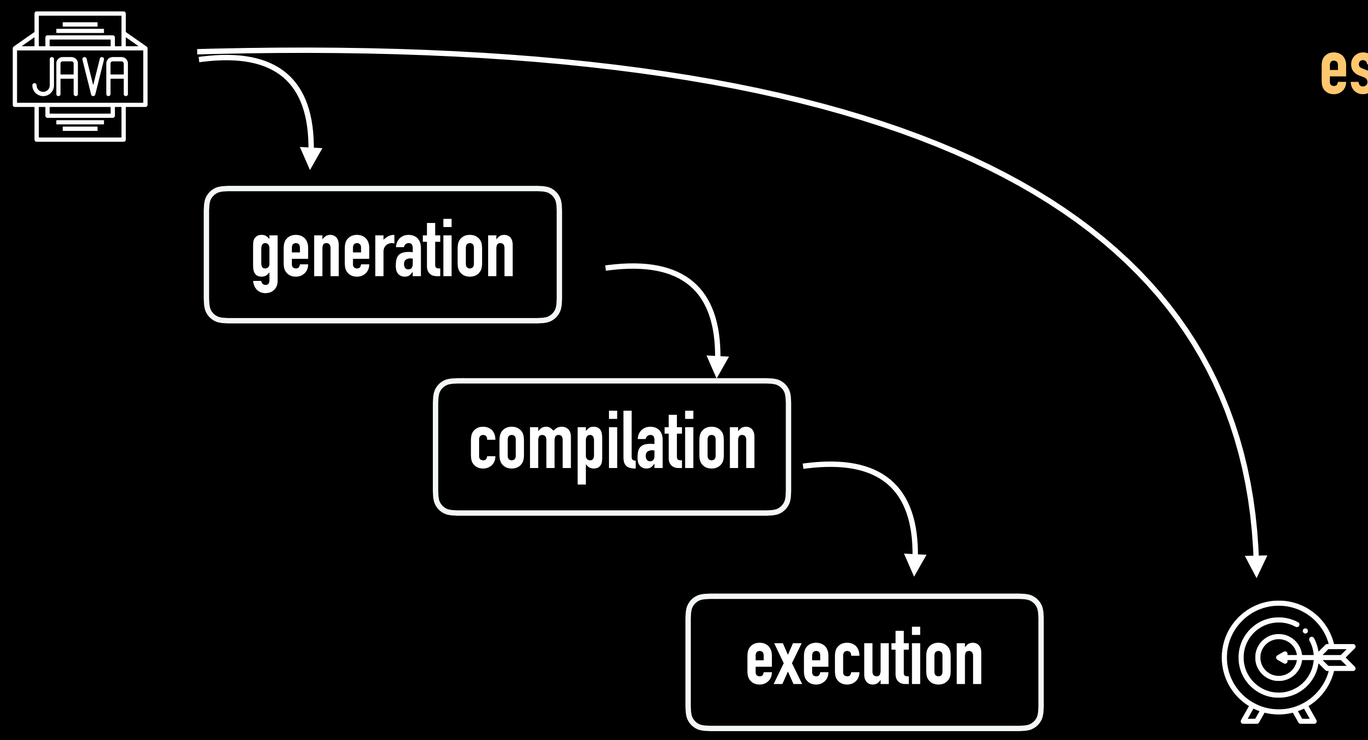
. . . but expensive



reduce the cost

- do fewer
- do smarter
- do faster

our approach



estimate the mutation score

source-code metrics

test smells

code coverage

67 metrics

code smells

readability

production/test metrics

CK

OO

relationship with fault-proneness of production code

feasibility study

high mutation score

distribution

41 metrics

low mutation score

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)**
	NOPA	-	0.23 (small)
	NOP	**-**	**0.44 (medium)**
	McCABE	**-**	**0.62 (large)**
Test Code Metrics	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)**
	ICH	+	0.19 (small)
	WMC	**+**	**0.45 (medium)**
	McCABE	**+**	**0.40 (medium)**
Code Smell	**MC**	**+**	**0.33 (medium)**
	FE	-	0.31 (small)
Readability	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

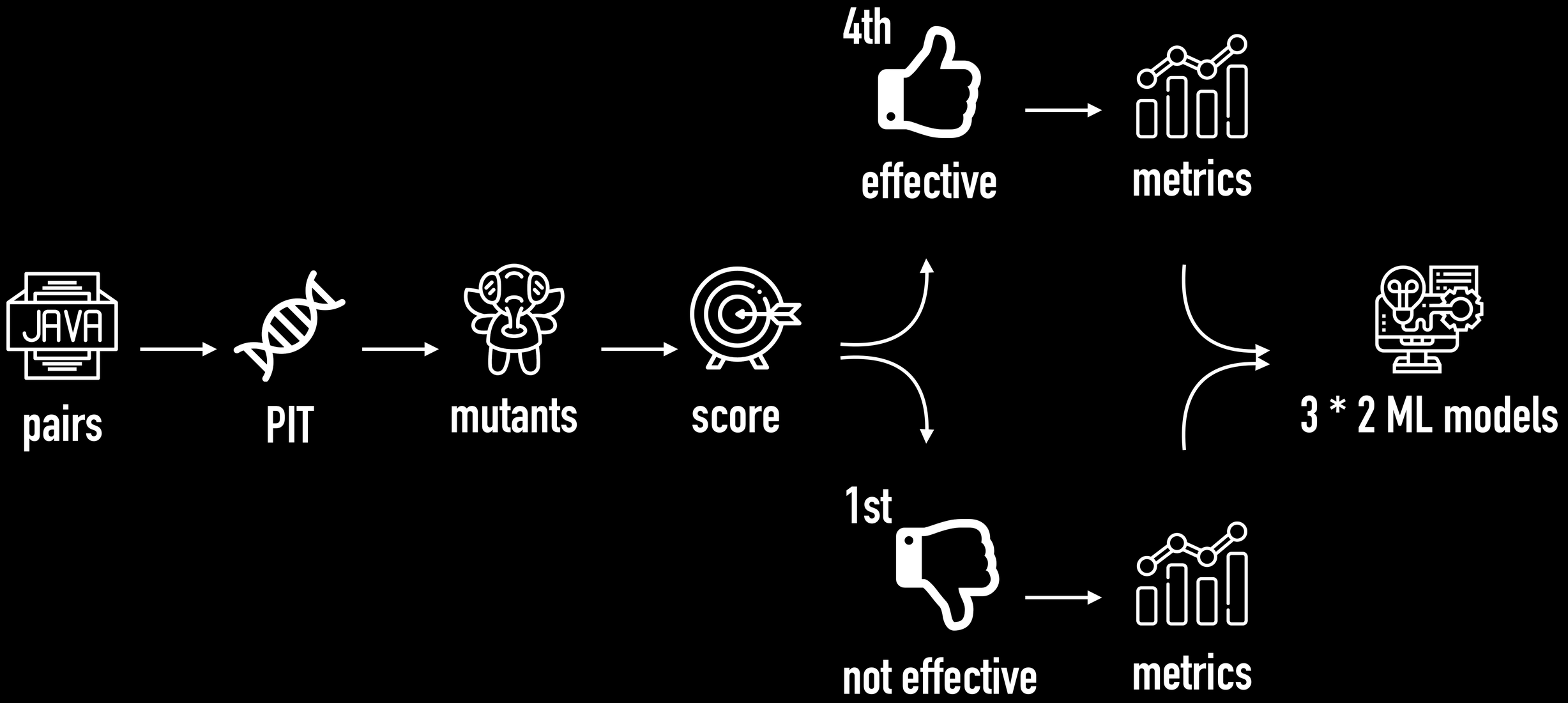
pairs

```
<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>
```

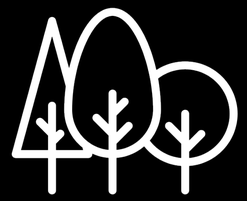
```
<build>
    <sourceDirectory>/src/main/java</sourceDirectory>
    <testSourceDirectory>/src/test/java</testSourceDirectory>
    ...
    <plugin>
       <groupId>org.apache.maven.plugins</groupId>
       <artifactId>maven-surefire-plugin</artifactId>
       <configuration>
          <includes>
             <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


4th
effective
metrics
JAVA
pairs
PIT
mutants
score
3 * 2 ML models
1st
not effective
metrics

results



random forest

dynamic model

0.949 AUC

static model

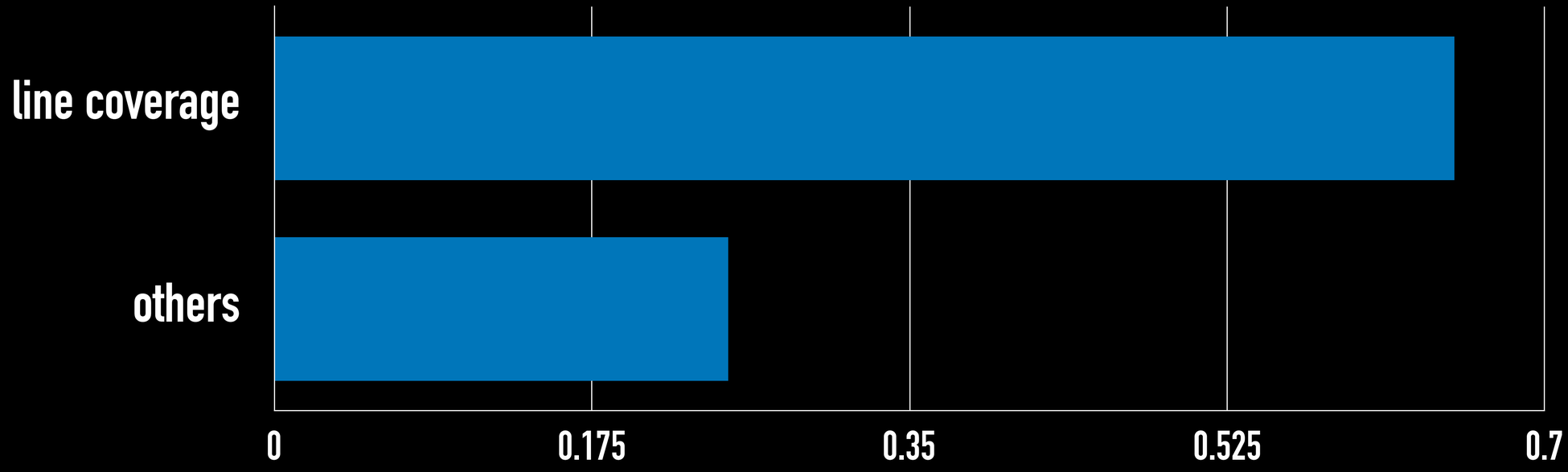
0.864 AUC

estimation of the effectiveness without actually run any test

important factors

mean decrease in impurity

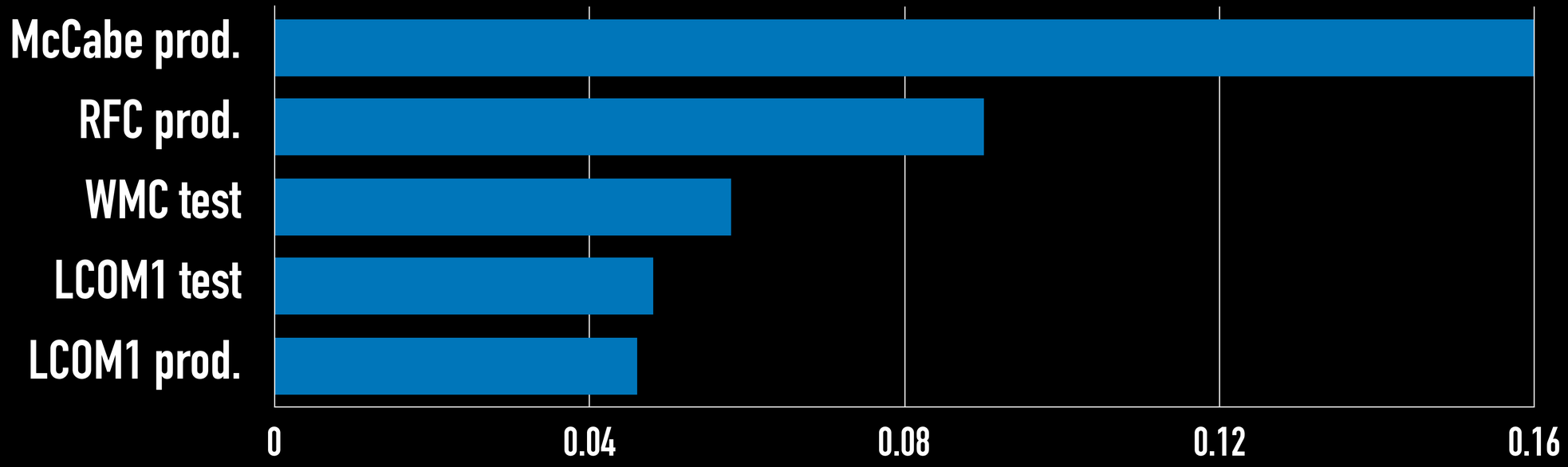
dynamic model



important factors

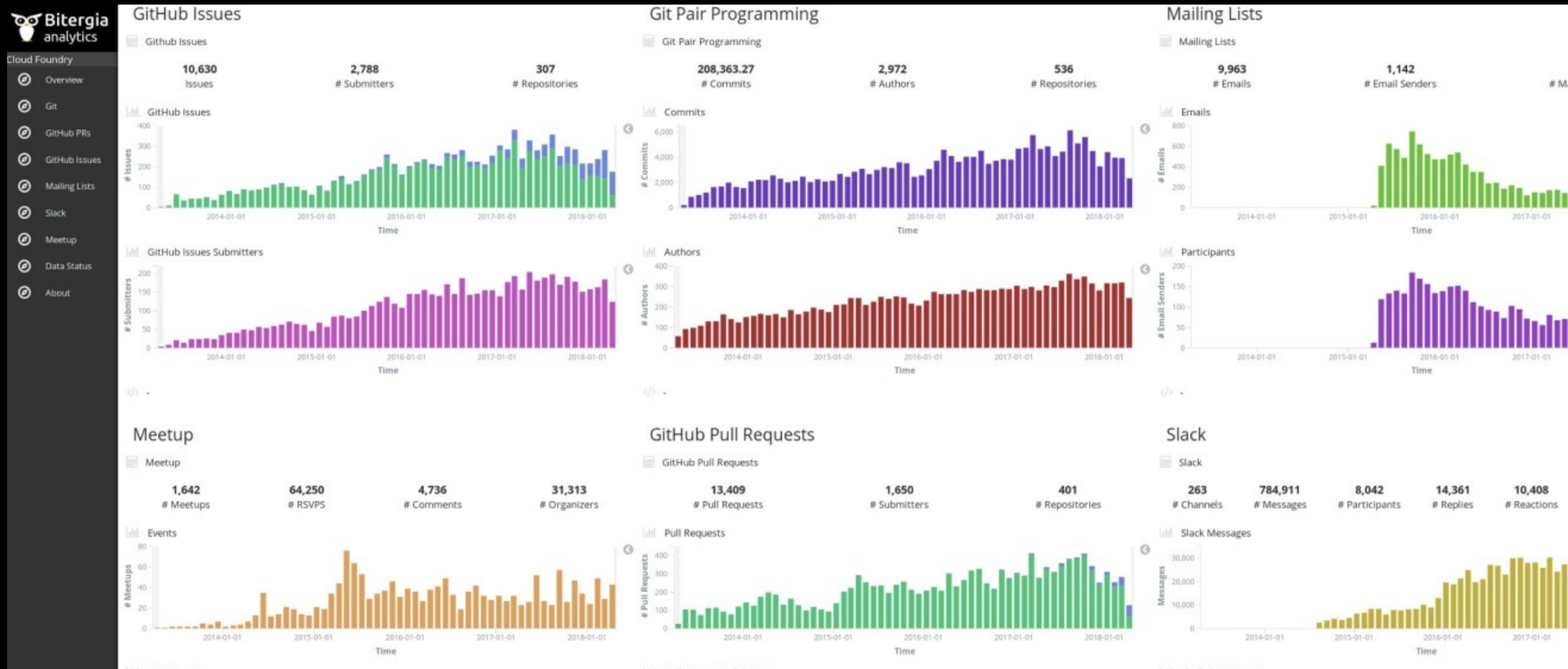
mean decrease in impurity

static model



practical usage

analytics dashboards



practical usage

complementarity

pairs → prediction → oops...not effective → fine-grained investigation

measuring effectiveness

mutation testing

mutants

mutation score

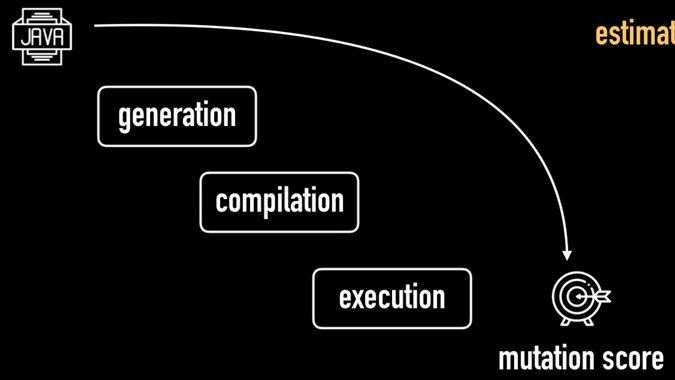
$$\frac{\text{detected (killed) mutants}}{\text{generated mutants}}$$

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

↓

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

our approach



results

random forest

dynamic model	static model
0.949 AUC	0.864 AUC

estimation of the effectivess without actually run any test

s.e.a.l.
software evolution & architecture lab

University of Zurich[UZH]

http://tiny.uzh.ch/Zx