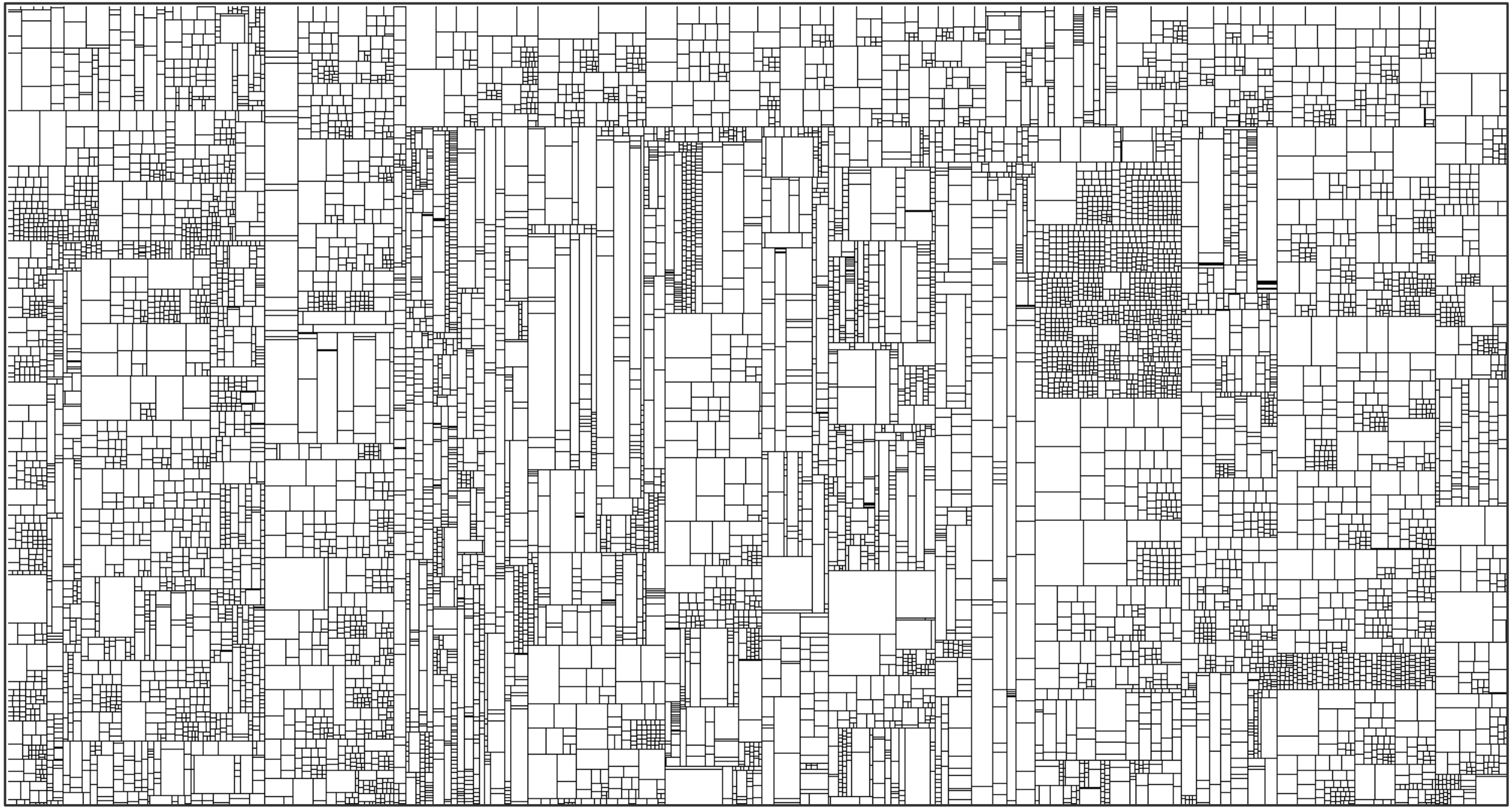
# How can I test, how good my tests are?

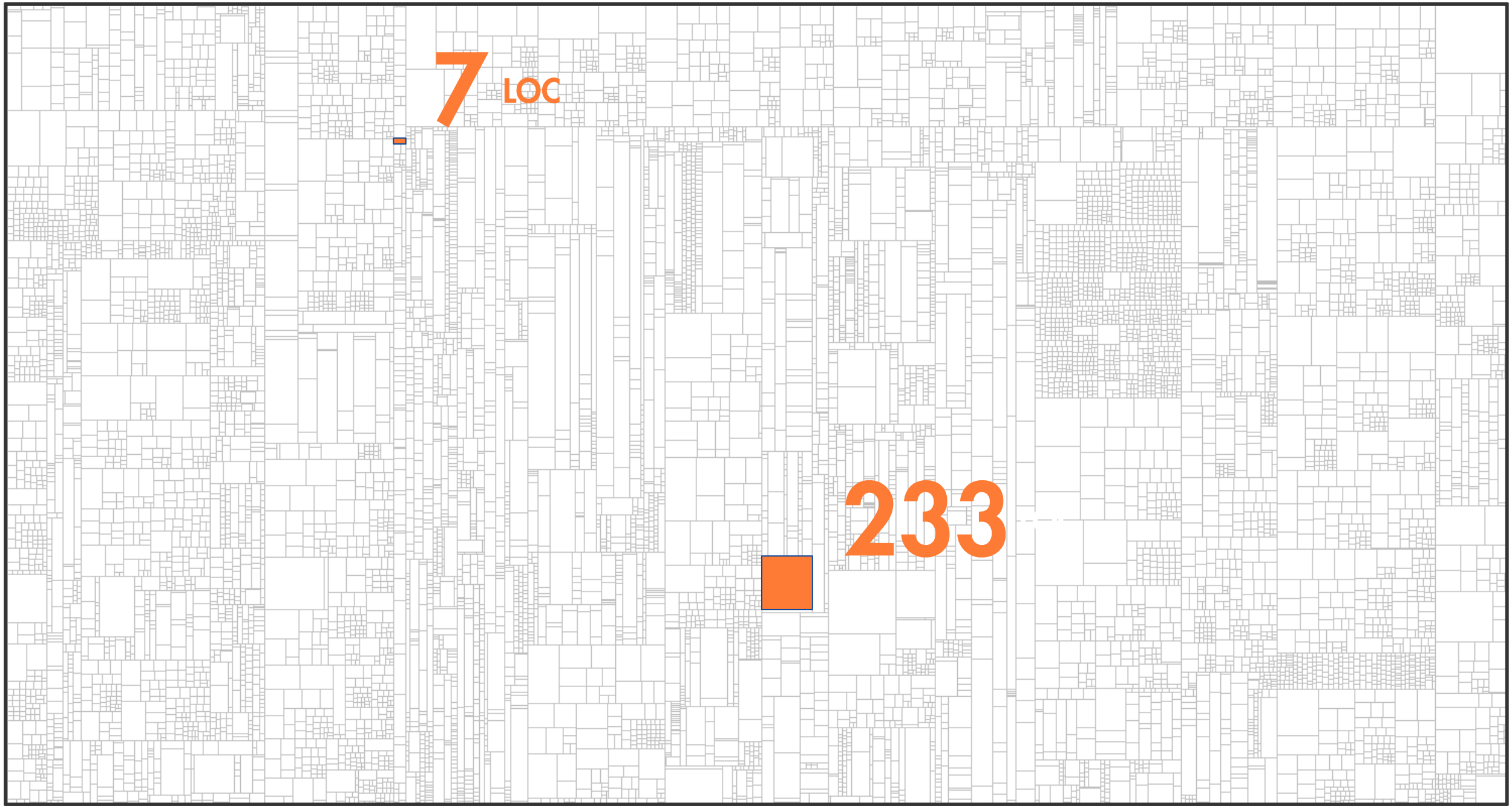
Dr. Elmar Juergens

CQSE

Will my tests tell me, if I break working code?
Where won‘t they?



Biojava system




Biojava system

# Code Coverage

- Measures what portion of the application code is executed by test cases
- Can be measured fully automatically, often tools are available for free (e.g. JaCoCo)

Commonly used in practice to assess test suites

```
protected void calculateIndirectAmbiguities() {
    Map<NucleotideCompound, List<NucleotideCompound>> equivalentsMap = new HashMap<>();

    List<NucleotideCompound> ambiguousCompounds = new ArrayList<NucleotideCompound>();
    for (NucleotideCompound compound : getAllCompounds()) {
        if (!compound.isAmbiguous()) {
            continue;
        }
        ambiguousCompounds.add(compound);
    }

    for (NucleotideCompound sourceCompound : ambiguousCompounds) {
        Set<NucleotideCompound> compoundConstituents = sourceCompound.getConstituents();
        for (NucleotideCompound targetCompound : ambiguousCompounds) {
            Set<NucleotideCompound> targetConstituents = targetCompound.getConstituents();
            if (targetConstituents.containsAll(compoundConstituents)) {
                NucleotideCompound lcSourceCompound = toLowerCase(sourceCompound);
                NucleotideCompound lcTargetCompound = toLowerCase(targetCompound);
                checkAdd(equivalentsMap, sourceCompound, targetCompound);
                checkAdd(equivalentsMap, sourceCompound, lcTargetCompound);
                checkAdd(equivalentsMap, targetCompound, sourceCompound);
                checkAdd(equivalentsMap, lcTargetCompound, sourceCompound);
                checkAdd(equivalentsMap, lcSourceCompound, targetCompound);
                checkAdd(equivalentsMap, lcSourceCompound, lcTargetCompound);
            }
        }
    }

    for (NucleotideCompound key : equivalentsMap.keySet()) {
        List<NucleotideCompound> vals = equivalentsMap.get(key);
        for (NucleotideCompound value : vals) {
            addEquivalent((C) key, (C) value);
            addEquivalent((C) value, (C) key);
        }
    }
}
```

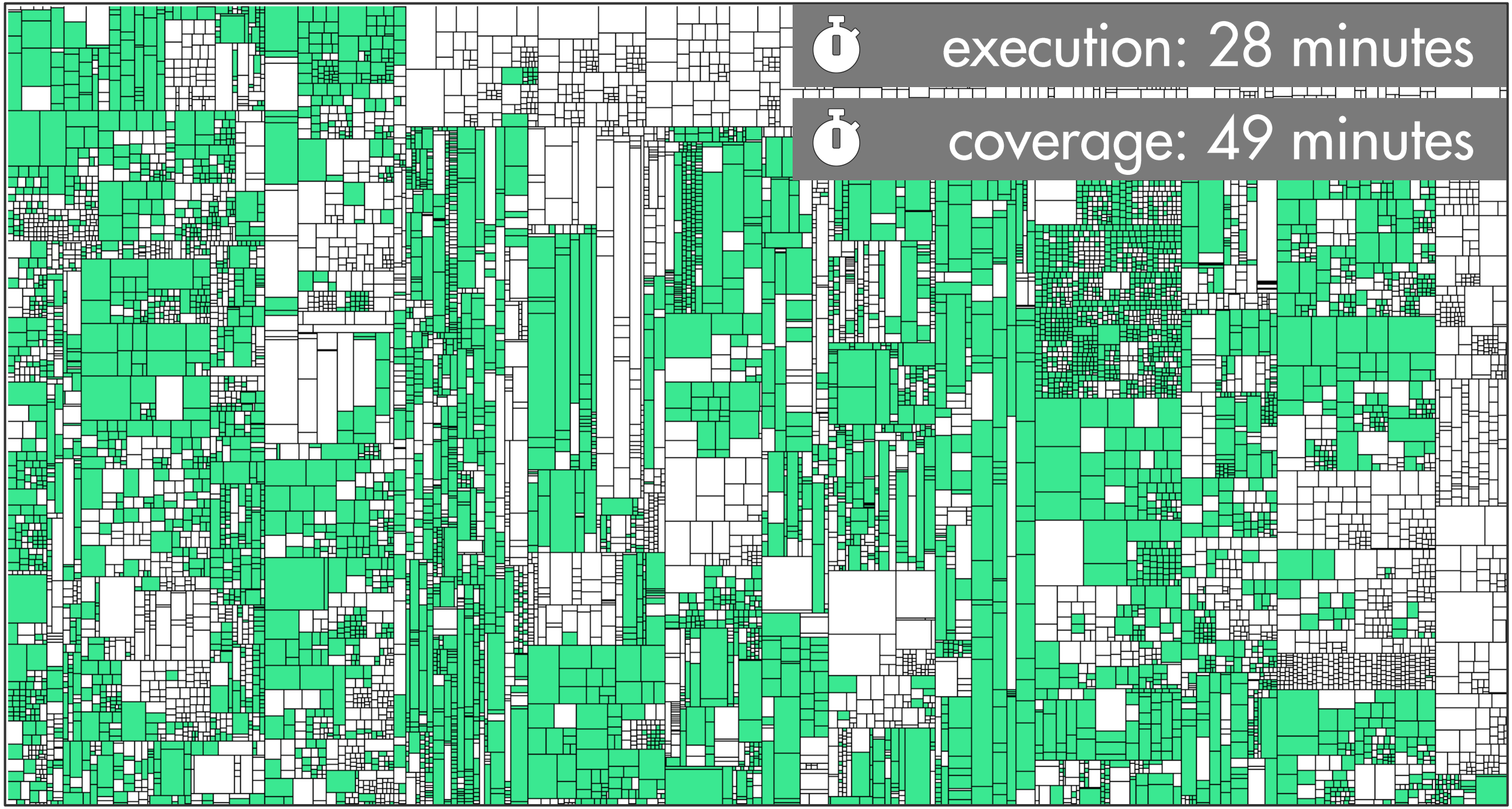
```
    protected void calculateIndirectAmbiguities() {
        Map<NucleotideCompound, List<NucleotideCompound>> equivalentsMap = new HashMap<>>();

        List<NucleotideCompound> ambiguousCompounds = new ArrayList<NucleotideCompound>();
        for (NucleotideCompound compound : getAllCompounds()) {
            if (!compound.isAmbiguous()) {
                continue;
            }
            ambiguousCompounds.add(compound);
        }

        for (NucleotideCompound sourceCompound : ambiguousCompounds) {
            Set<NucleotideCompound> compoundConstituents = sourceCompound.getConstituents();
            for (NucleotideCompound targetCompound : ambiguousCompounds) {
                Set<NucleotideCompound> targetConstituents = targetCompound.getConstituents();
                if (targetConstituents.containsAll(compoundConstituents)) {
                    NucleotideCompound lcSourceCompound = toLowerCase(sourceCompound);
                    NucleotideCompound lcTargetCompound = toLowerCase(targetCompound);
                    checkAdd(equivalentsMap, sourceCompound, targetCompound);
                    checkAdd(equivalentsMap, sourceCompound, lcTargetCompound);
                    checkAdd(equivalentsMap, targetCompound, sourceCompound);
                    checkAdd(equivalentsMap, lcTargetCompound, sourceCompound);
                    checkAdd(equivalentsMap, lcSourceCompound, targetCompound);
                    checkAdd(equivalentsMap, lcSourceCompound, lcTargetCompound);
                }
            }
        }

        for (NucleotideCompound key : equivalentsMap.keySet()) {
            List<NucleotideCompound> vals = equivalentsMap.get(key);
            for (NucleotideCompound value : vals) {
                addEquivalent((C) key, (C) value);
                addEquivalent((C) value, (C) key);
            }
        }
    }
```

```
protected void calculateIndirectAmbiguities() {
    Map<NucleotideCompound, List<NucleotideCompound>> equivalentsMap = new HashMap<>();

    List<NucleotideCompound> ambiguousCompounds = new ArrayList<NucleotideCompound>();
    for (NucleotideCompound compound : getAllCompounds()) {
        if (!compound.isAmbiguous()) {
            continue;
        }
        ambiguousCompounds.add(compound);
    }

    for (NucleotideCompound sourceCompound : ambiguousCompounds) {
        Set<NucleotideCompound> compoundConstituents = sourceCompound.getConstituents();
        for (NucleotideCompound targetCompound : ambiguousCompounds) {
            Set<NucleotideCompound> targetConstituents = targetCompound.getConstituents();
            if (targetConstituents.containsAll(compoundConstituents)) {
                NucleotideCompound lcSourceCompound = toLowerCase(sourceCompound);
                NucleotideCompound lcTargetCompound = toLowerCase(targetCompound);
                checkAdd(equivalentsMap, sourceCompound, targetCompound);
                checkAdd(equivalentsMap, sourceCompound, lcTargetCompound);
                checkAdd(equivalentsMap, targetCompound, sourceCompound);
                checkAdd(equivalentsMap, lcTargetCompound, sourceCompound);
                checkAdd(equivalentsMap, lcSourceCompound, targetCompound);
                checkAdd(equivalentsMap, lcSourceCompound, lcTargetCompound);
            }
        }
    }

    for (NucleotideCompound key : equivalentsMap.keySet()) {
        List<NucleotideCompound> vals = equivalentsMap.get(key);
        for (NucleotideCompound value : vals) {
            addEquivalent((C) key, (C) value);
            addEquivalent((C) value, (C) key);
        }
    }
}
```

**26** test cases

**96%** line coverage

**86%** branch coverage


execution: 28 minutes
coverage: 49 minutes
COVERED

Will my tests tell me, if I break working code?

No, for the 55% of methods that are not covered by tests.
What about the 45% covered methods?

```
1   protected void calculateIndirectAmbiguities() {
2
3
4
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
31
32
33
34
35
36  }
```

26 test cases

0 failing test cases

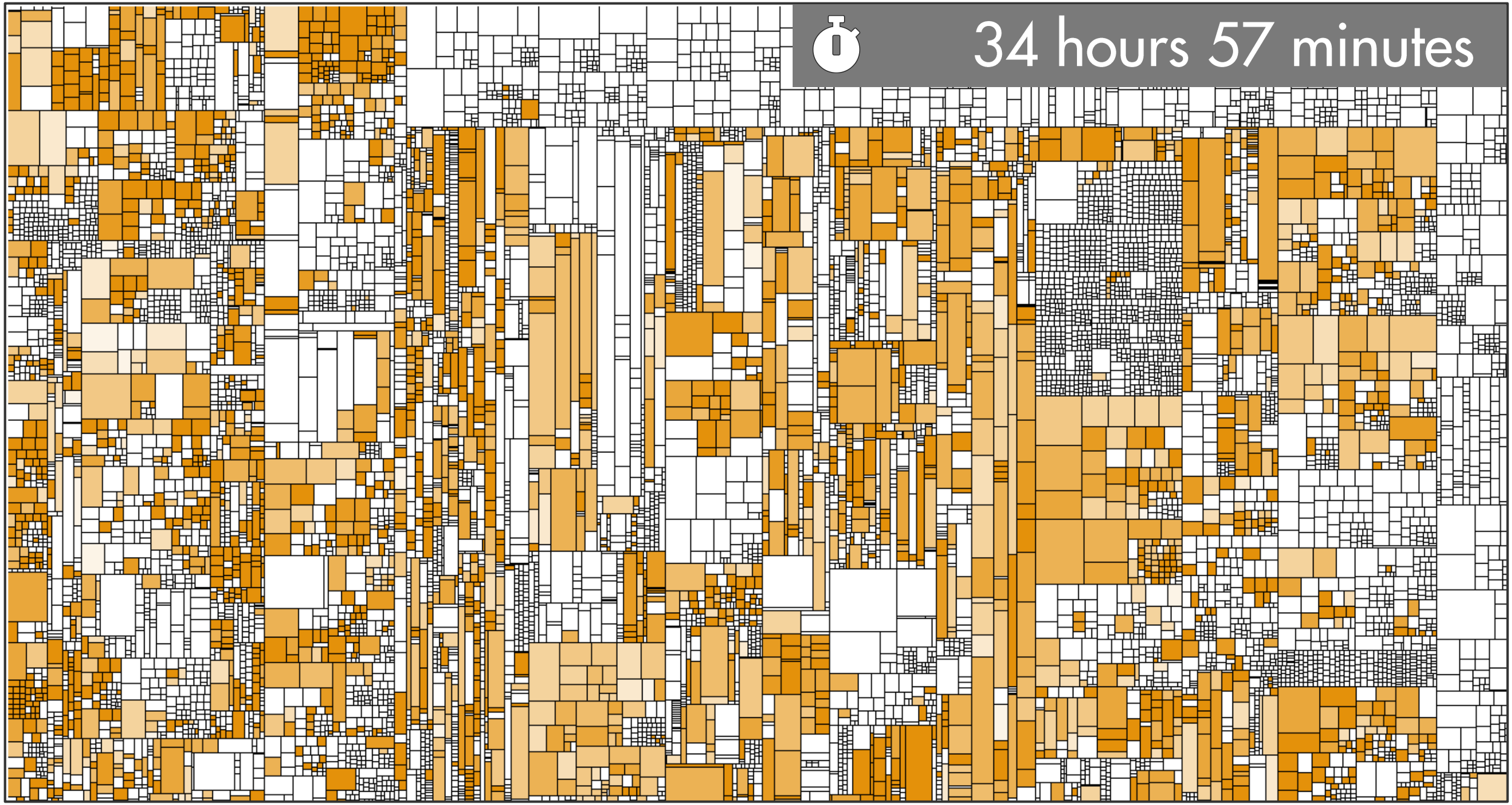
# Mutation Testing

- Inject a fault into the application code
- Execute all relevant test cases
- Check if at least one test fails

Commonly proposed by reseachers to assess test suites

```
public int computeSum(int[] values) {
      int sum = 0;

      for (int i = 0; i < values.length; i++) {
-           sum += values[i];
+           sum -= values[i];
      }

      return sum;
}
```


COVERED


34 hours 57 minutes
MUTATION SCORE

WTF?

# Problems with Mutation Testing in Practice

- Takes long to execute mutated tests, often too long
- Equivalent mutants cause false positives
- Interpretation unclear: How should I act on these results as a developer? What does mutation score really mean for my project?

=> No widespread use (I personally know not a single team that uses mutation testing on a continuous basis)

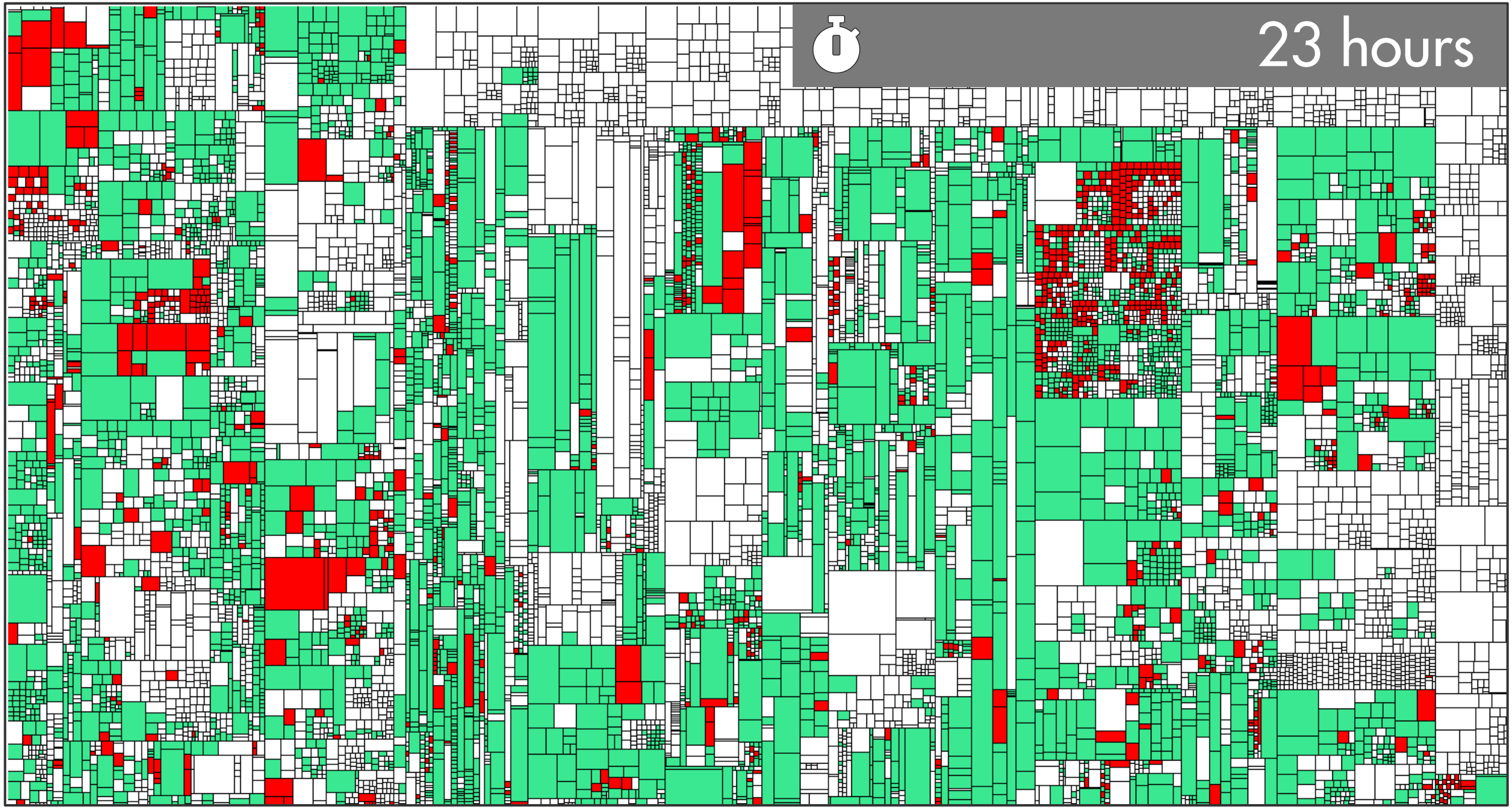
```
1   protected void calculateIndirectAmbiguities() {
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36  }
```


26 test cases

0 failing test cases

# Detection of Pseudo-Tested Methods

- Faster than mutation testing
- Hardly any equivalent mutants (since mutations are much more extreme)
- Interpretation clear: There is no test for this method


23 hours
COVERED
COVERED AND PSEUDO-TESTED

| Name ↓ | Purpose | LOC | # Tests | % pseudo-tested methods |
|---|---|---|---|---|
| Apache Commons Collections | data structure framework | 109.4 k | 4,372 | |
| Apache Commons Lang | utility classes for Java | 100.4 k | 1,996 | |
| Apache Commons Math | mathematics library | 275.6 k | 3,427 | |
| Apache Commons Net | network protocol implementations | 53.6 k | 163 | |
| Asterisk | internet telephony toolkit | 76.1 k | 217 | |
| ConQAT Engine Core | code analysis platform base | 26.4 k | 143 | |
| ConQAT Lib Commons | common utilities | 39.7 k | 611 | |
| JabRef | BibTeX manager | 124.1 k | 1,561 | |
| JFreeChart | chart library | 234.0 k | 2,219 | |
| JSONDoc | REST API documentation generator | 4.3 k | 26 | |
| Twitter Graphjet | real-time graph processing library | 15.6 k | 81 | |
| Urban-Airship | Java API for Airship | 27.8 k | 575 | |
| ConQAT dotnet | processors to analyze .NET code | 8.0 k | 20 | |
| Daisydiff | visual comparator of HTML | 11.3 k | 247 | |
| Histone | HTML template engine | 244.0 k | 89 | |
| LittleProxy | high performance HTTP proxy | 7.3 k | 18 | |
| Predictor | genetic algorithms framework | 7.7 k | 21 | |
| Symja | algebra system with plot functionality | 443.1 k | 445 | |
| Tspmccabe | code metrics analyzer | 45.0 k | 10 | |

**«Will My Tests Tell Me If I Break This Code?»**
**R. Niedermayr, E. Juergens, S. Wagner. CSED 2016.**

# Descartes Mutation Engine for PIT

https://github.com/STAMP-project/pitest-descartes

A mutation engine plugin for PIT which implements some of the mutation operators proposed in Will my tests tell me if I break this code?.

### Mutation Testing

Unit test suites need to be verified to see if they can detect possible bugs. Mutation testing does it by introducing small changes or faults into the original program. These modified versions are called **mutants**. A good test suite should able to **kill** or detect a mutant. Read more.

### PIT

PIT is a mutation testing system for Java. It is actively developed, scalable and targets real world projects. It also provides a framework to extend its core functionality using plugins. PIT integrates with majors test and build tools such as Maven, Ant and Gradle. A list of built-in mutation operators can be found in the tool's web page.

### Descartes

The authors of Will my tests tell me if I break this code? proposed an *extreme mutation* strategy in which the whole logic of a method under test is eliminated. All statements in a void method are removed. In other case the body is replaced by a return statement. With this approach, a smaller number of mutants is generated. Code from the authors can be found in their GitHub repository.

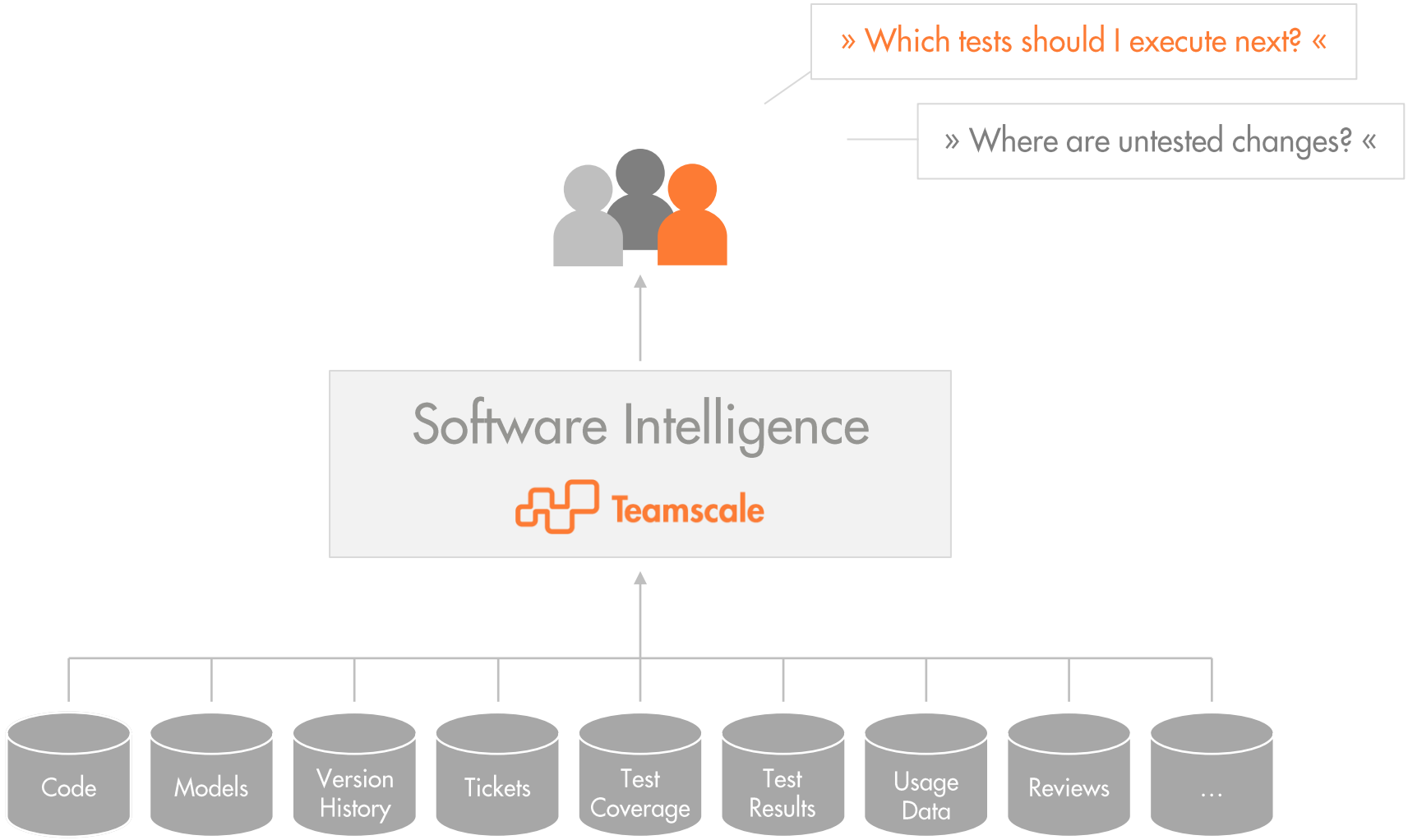
Will my tests tell me, if I break working code?

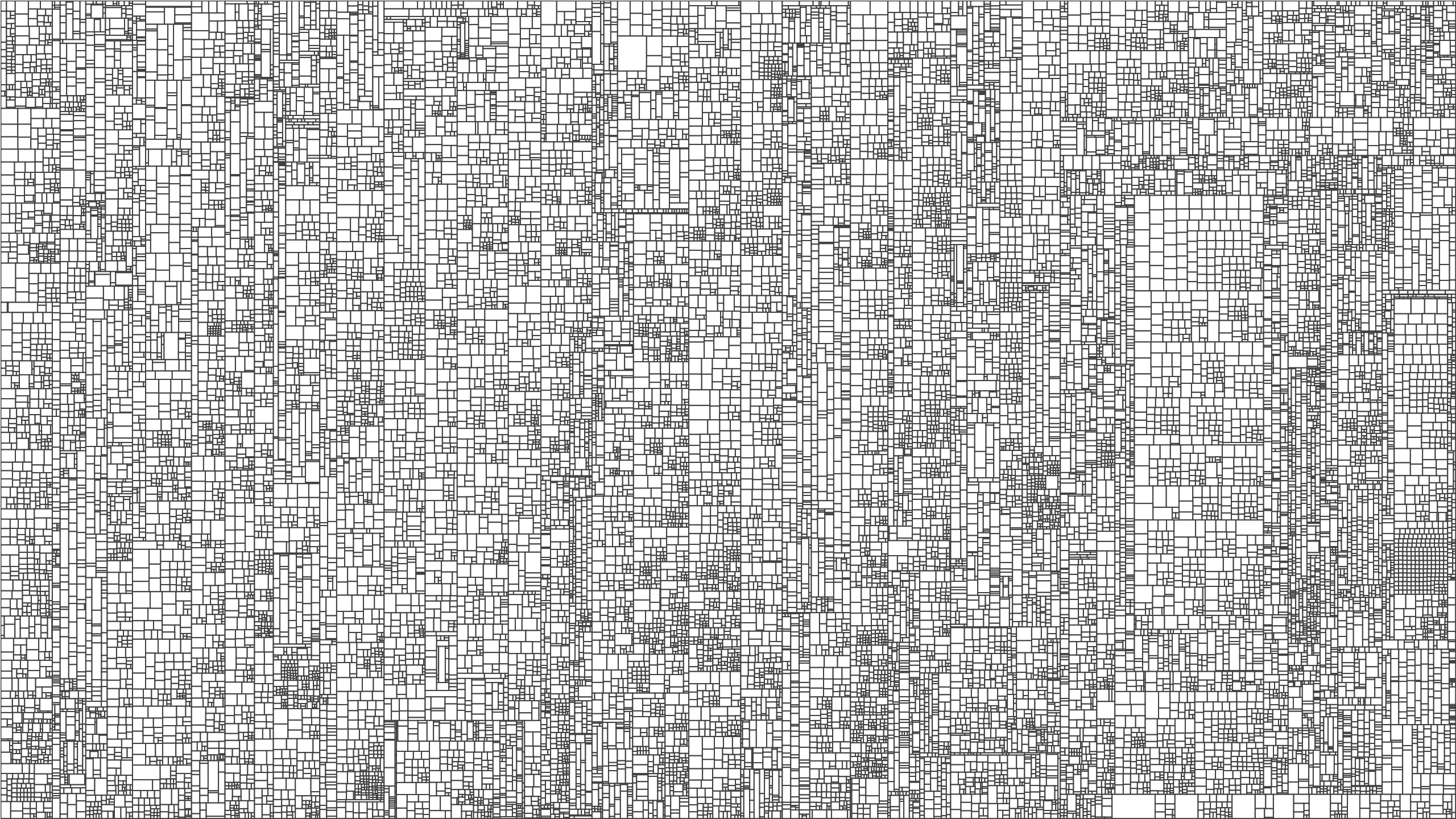
No, for the 55% of methods that are not covered by tests.
No, for the 15% pseudo-tested methods

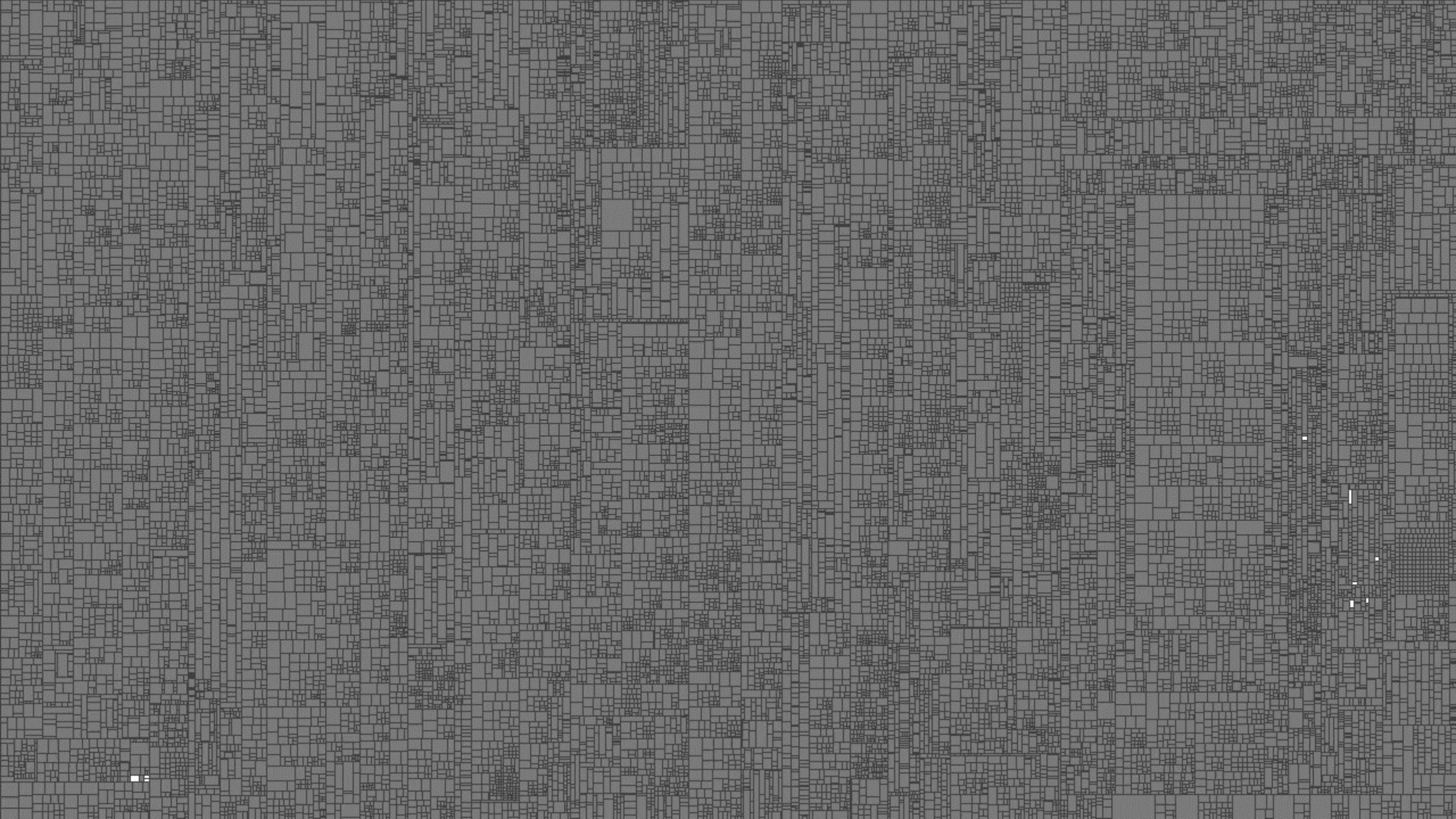
Will my tests tell me, if I break working code?

No, for the 55% of methods that are not covered by tests.
No, for the 15% pseudo-tested methods

The remaining 30%: We don‘t know for sure
(But 70% untested is probably the bigger problem…)


» Which tests should I execute next? «
» Where are untested changes? «
Software Intelligence
Teamscale
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…





Teamscale / TS-10829

# Data Flow Analysis can not handle java lambdas (logs many errors currently)

Edit | Comment | Assign | More | Back to New | Export

## Details

| | | | |
|---|---|---|---|
| Type: | Bug | Status: | DONE (View Workflow) |
| Priority: | High | Resolution: | Green |
| Component/s: | Backend | Fix Version/s: | Teamscale 4.3 |
| Labels: | dataflow java | | |
| Affected Version: | master | | |
| Merge Request: | https://git.cqse.eu/cqse/teamscale/merge_requests/2734 | | |
| PDash Task: | #4886 | | |

## Description

Analysis profile: Java
Repository: JabRef, start revision 9efd23b71871747fe5e18e915e637891d7e55b6d

```
ERROR : An error occurred while trying to construct a CFG for function 'null' in element src/main/java/net/sf/jabref/exporter/layout/format/DOI
[STATEMENT: lambda expression: null (lines 33-33)
]
Tokens: doi . getURL ( )
Occurred in src/main/java/net/sf/jabref/exporter/layout/format/DOICheck.java:32-32 (com.teamscale.index.dataflow.DataFlowFindingsSynchronizer.c
org.conqat.engine.core.core.ConQATException: Could not find any rule that applies to the following entity list:
[STATEMENT: lambda expression: null (lines 33-33)
]
Tokens: doi . getURL ( )
Occurred in src/main/java/net/sf/jabref/exporter/layout/format/DOICheck.java:32-32
        at org.conqat.engine.sourcecode.dataflow.heuristics.ControlFlowCreator.findApplicableRule(ControlFlowCreator.java:164)
        at org.conqat.engine.sourcecode.dataflow.heuristics.ControlFlowCreator.transformOneStep(ControlFlowCreator.java:139)
        at org.conqat.engine.sourcecode.dataflow.heuristics.ControlFlowCreator.transform(ControlFlowCreator.java:92)
[...]

ERROR : An error occurred while trying to construct a CFG for function 'null' in element src/main/java/net/sf/jabref/exporter/layout/format/DOI
[STATEMENT: lambda expression: null (lines 31-31)
]
Tokens: doi . getDOI ( )
Occurred in src/main/java/net/sf/jabref/exporter/layout/format/DOIStrip.java:30-30 (com.teamscale.index.dataflow.DataFlowFindingsSynchronizer.c
org.conqat.engine.core.core.ConQATException: Could not find any rule that applies to the following entity list:
```

## People

| | |
|---|---|
| Assignee: | Andreas Sewe |
| | Assign to me |
| Reporter: | Rainer Niedermayr |
| QA-Contact: | Alexander von Rhein |
| Votes: | 2 Vote for this issue |
| Watchers: | 3 Start watching this issue |

## Dates

| | |
|---|---|
| Created: | 24/Nov/16 8:35 AM |
| Updated: | 4 days ago |
| Resolved: | 08/May/18 12:42 PM |

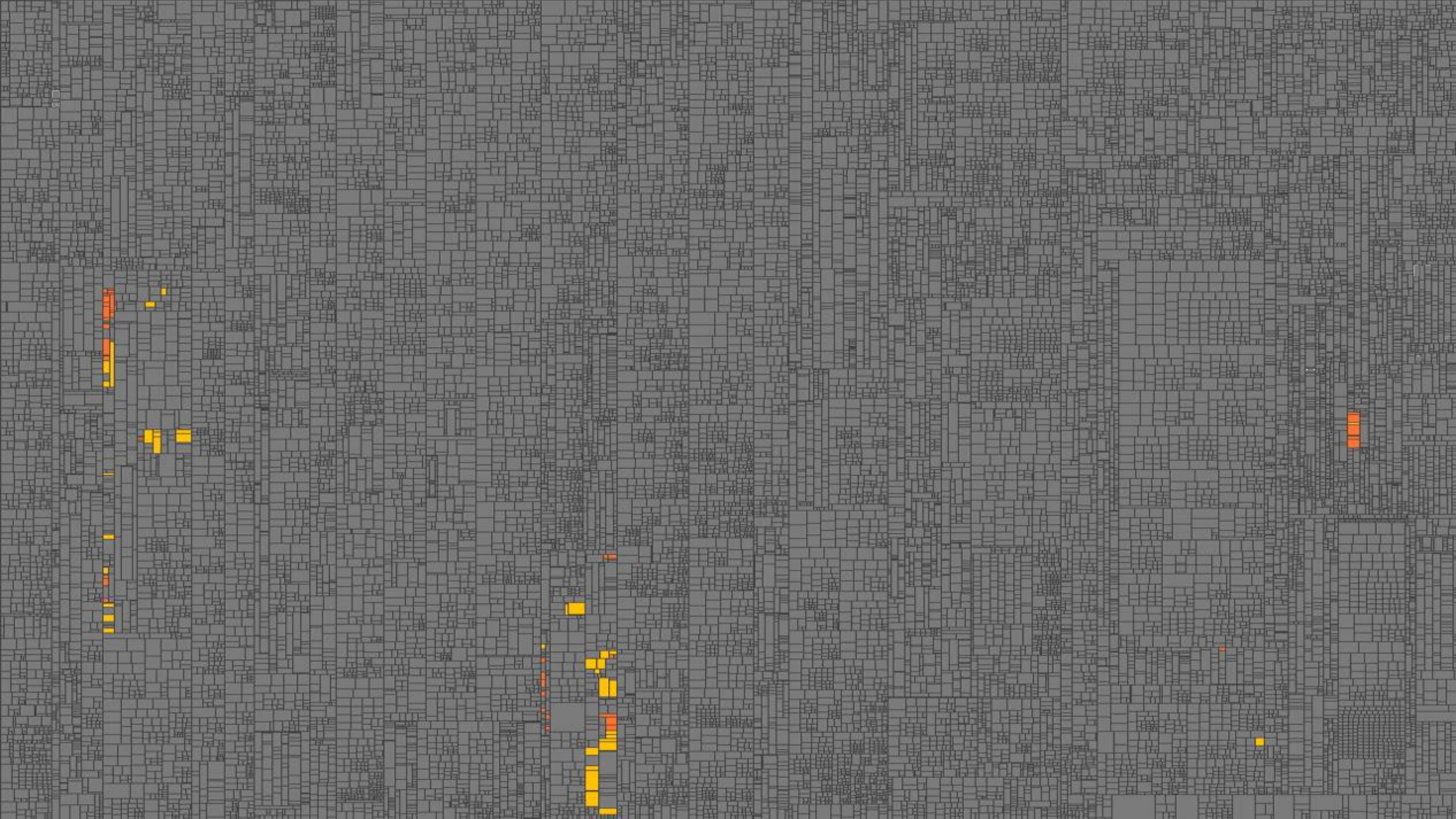
## Time Tracking

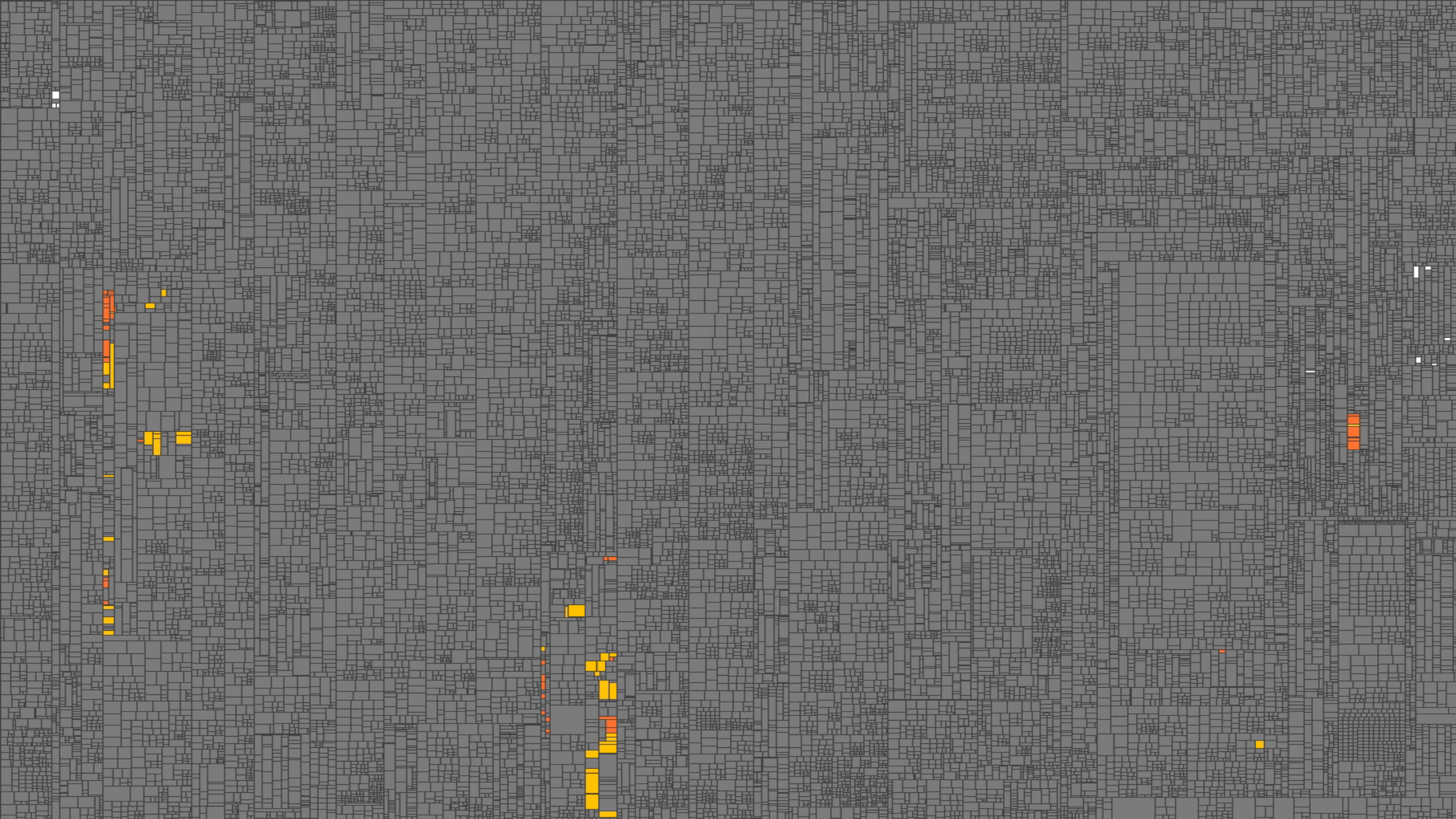
| | |
|---|---|
| Estimated: | Not Specified |
| Remaining: | 0m |
| Logged: | 4d 1h 57m |

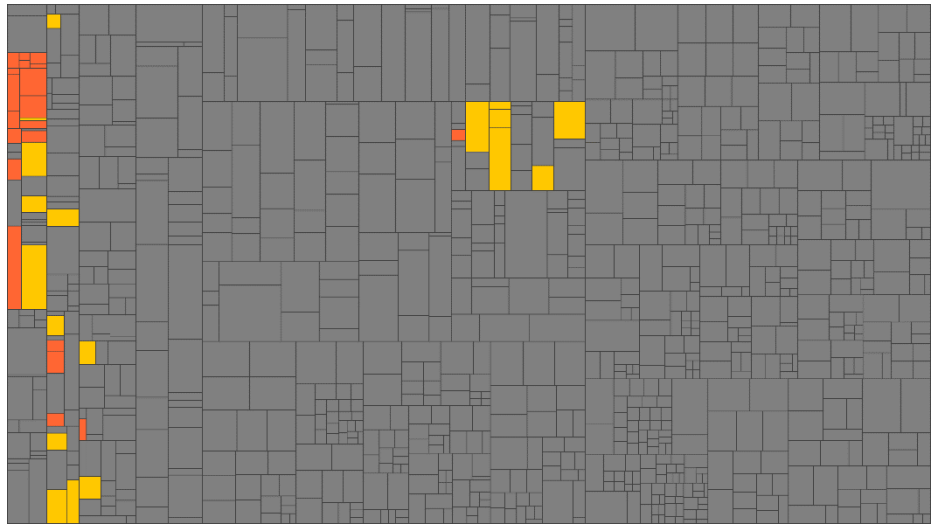
## Agile

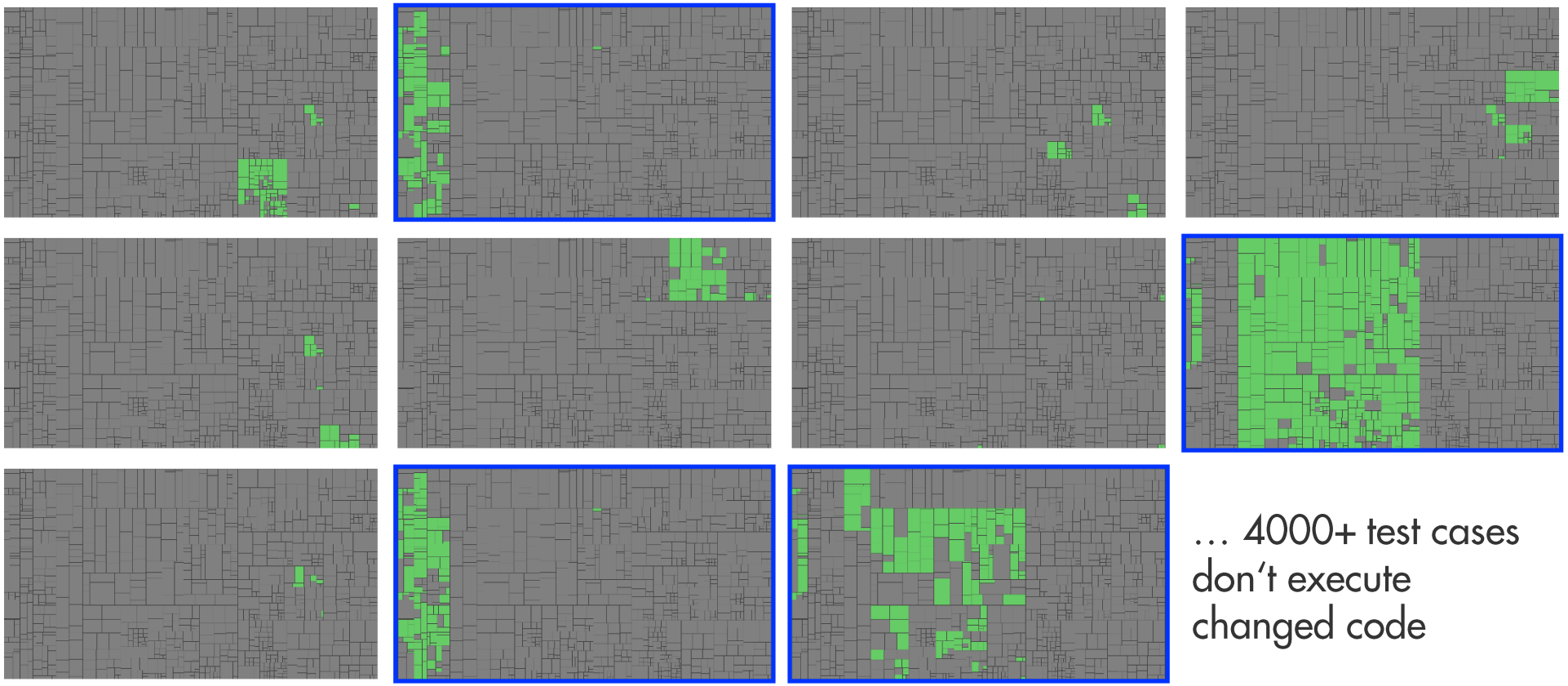
View on Board

## Gitlab CI

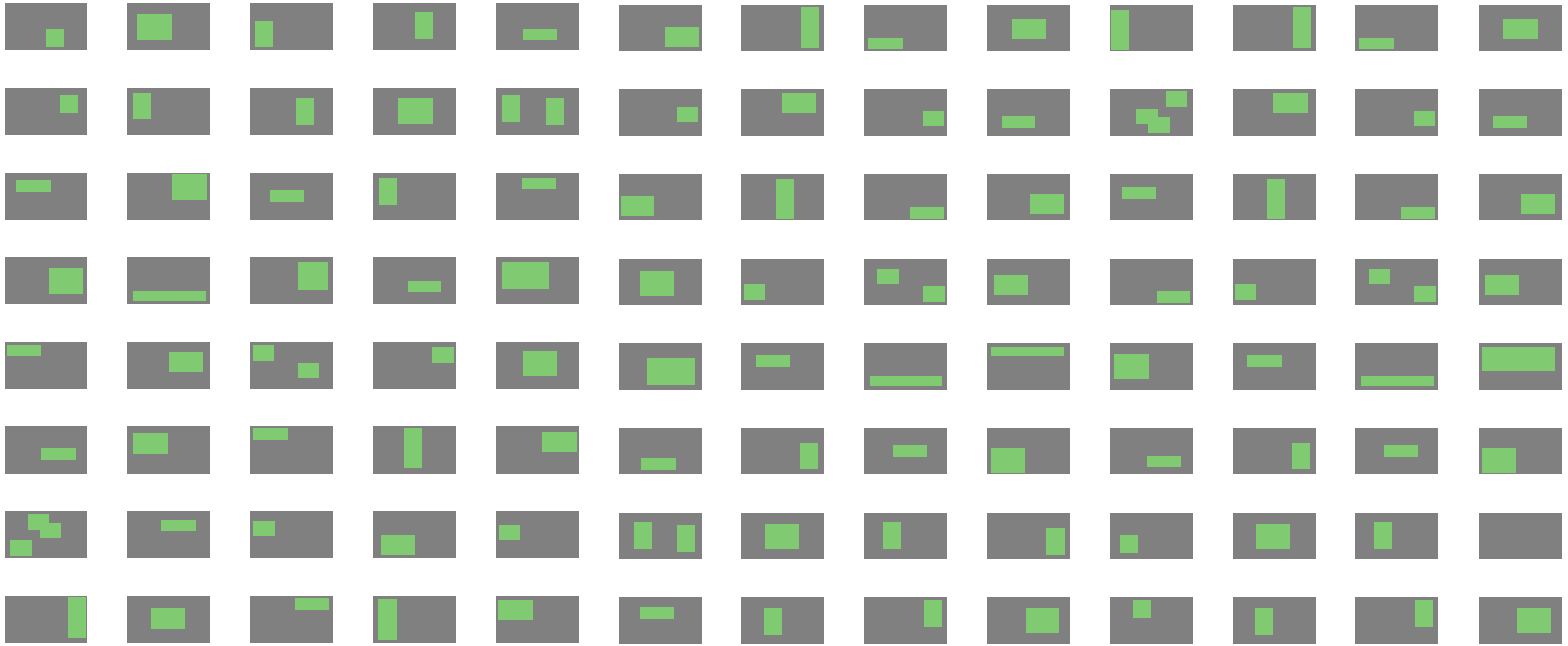




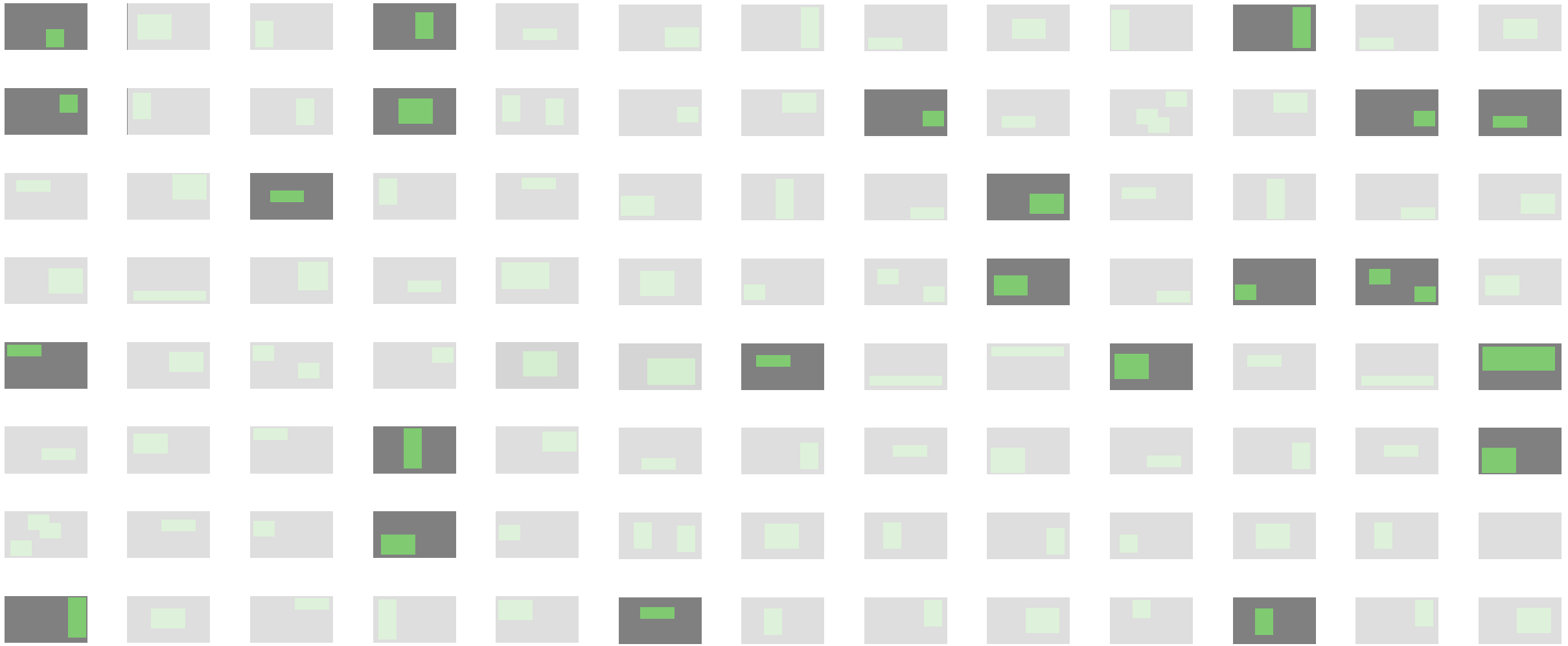




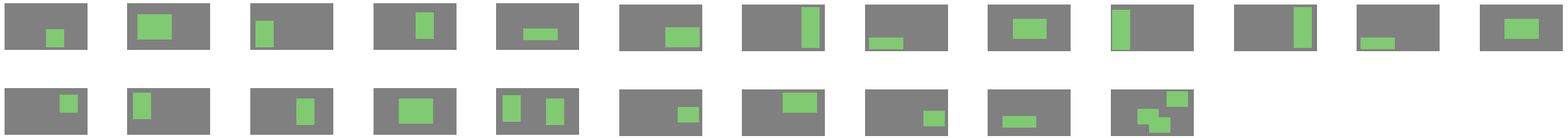
… 4000+ test cases
don't execute
changed code



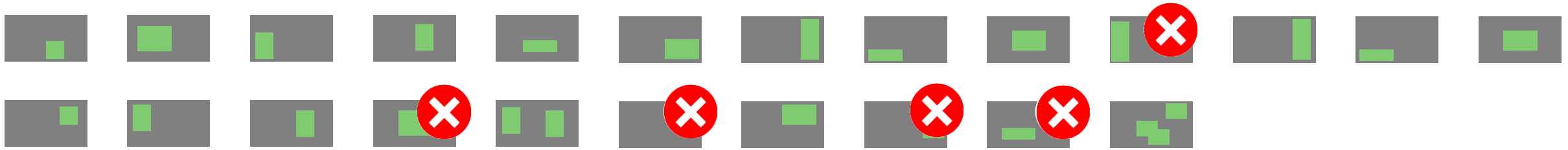
# Step 1: Select Impacted Test Cases



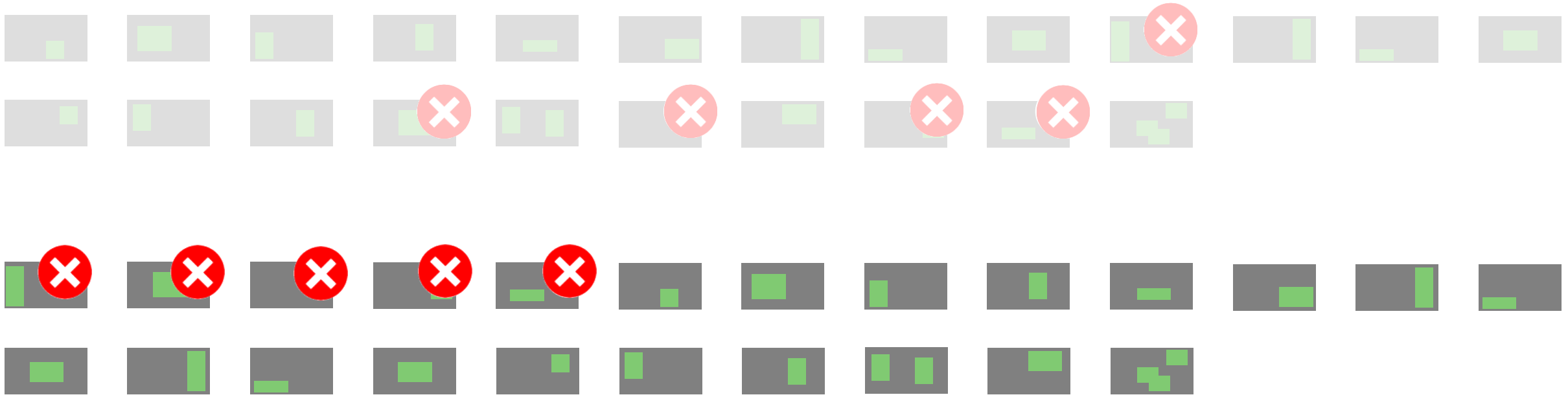
# Step 1: Select Impacted Test Cases



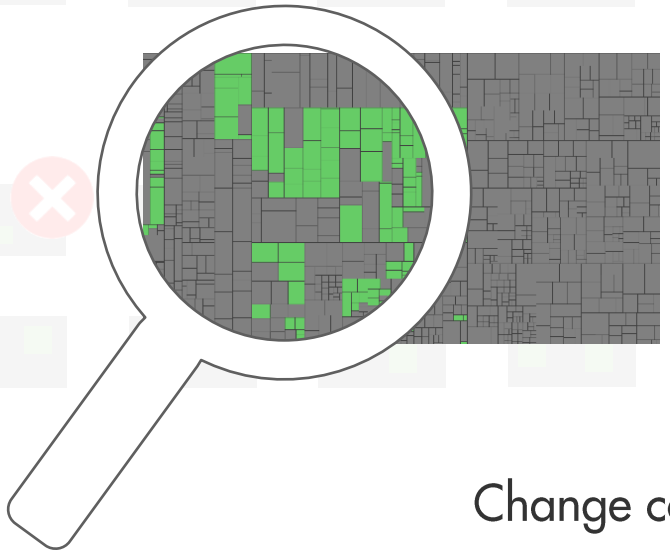
# Step 2: Prioritize Selected Test Cases



# Step 2: Prioritize Selected Test Cases



# Step 2: Prioritize Selected Test Cases



Change coverage

Execution time

# Step 2: Prioritize Selected Test Cases

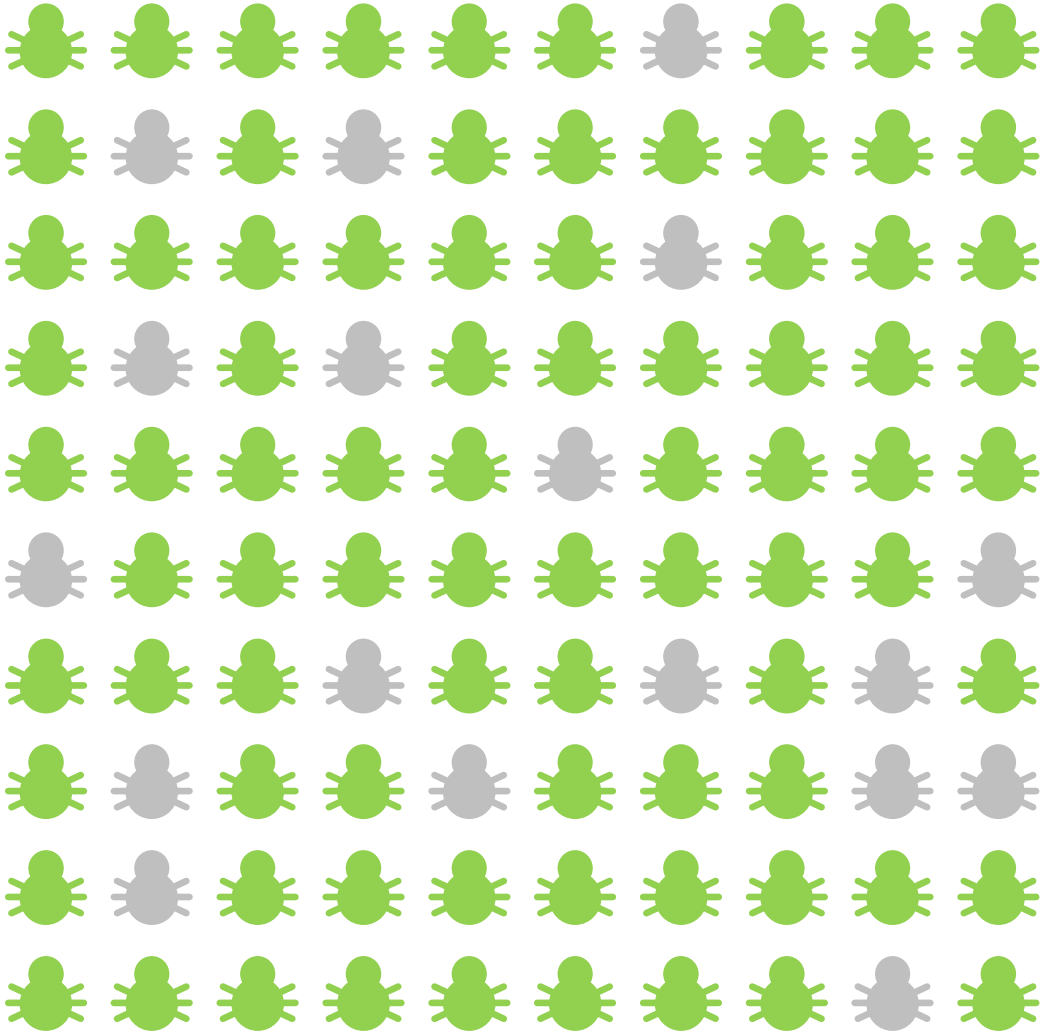
# Limitations of Test Impact Analysis

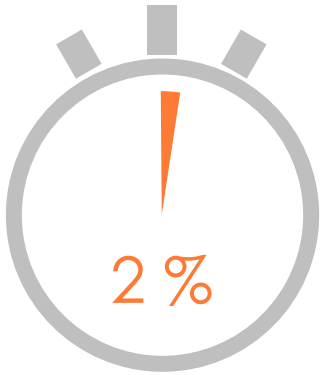
- Some changes affect every test
  - Changes in configuration files
  - Changes in test data
- When the tests themselves change, they must be executed again
- There is no guarantee that TIA will find all errors

⇒ Execute all tests in regular intervals

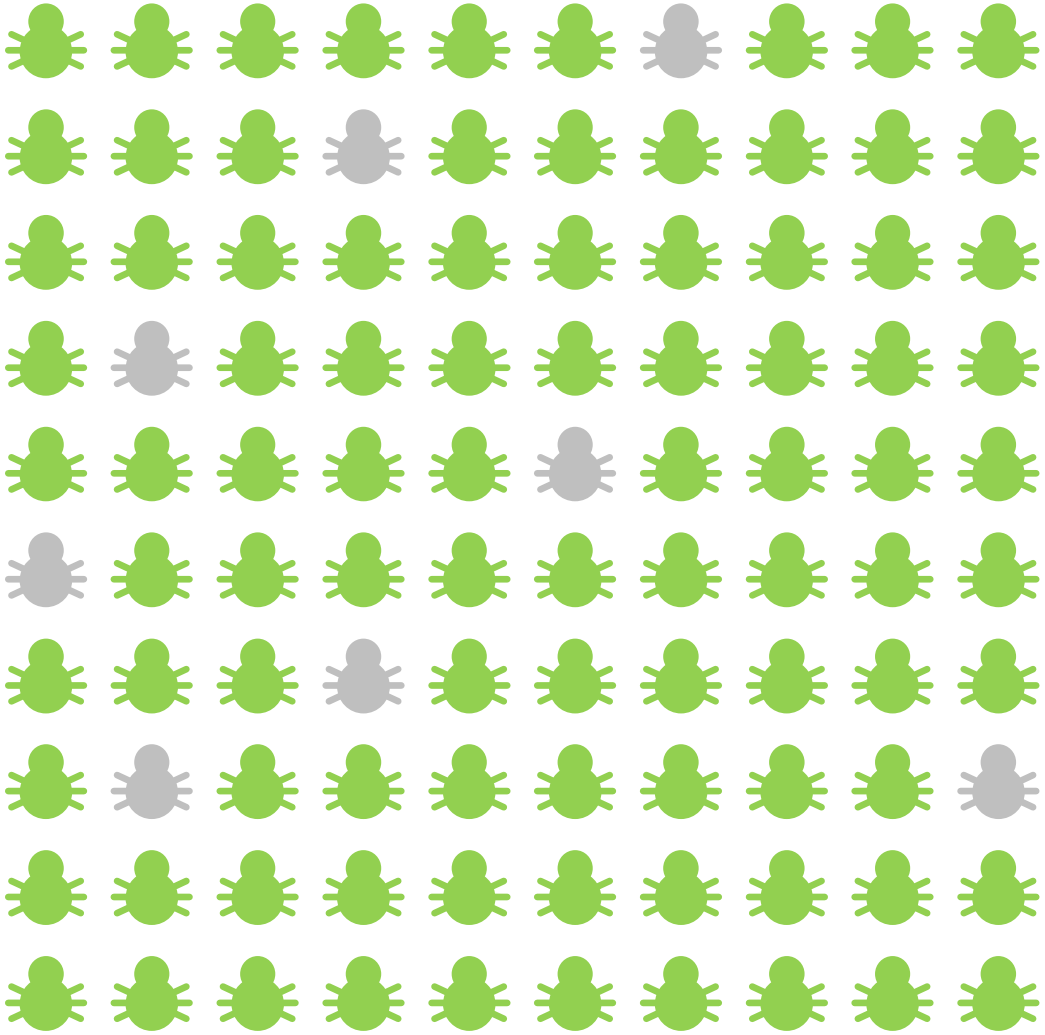
⇒ This also keeps the test impact data up-to-date

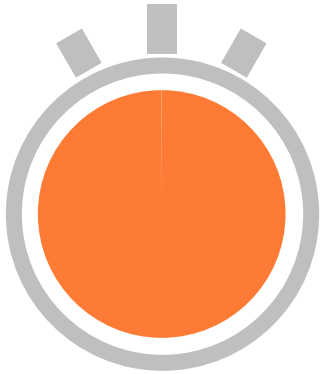

80 %

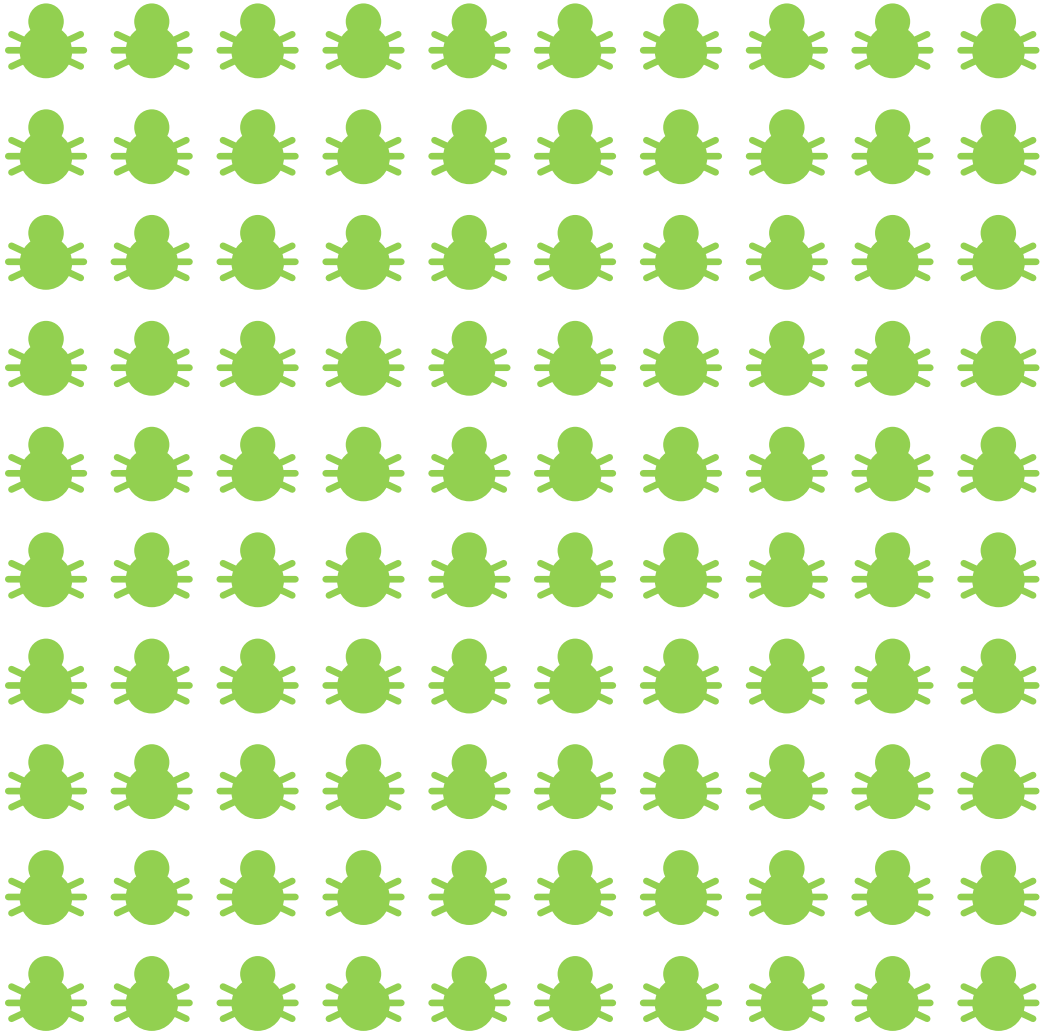


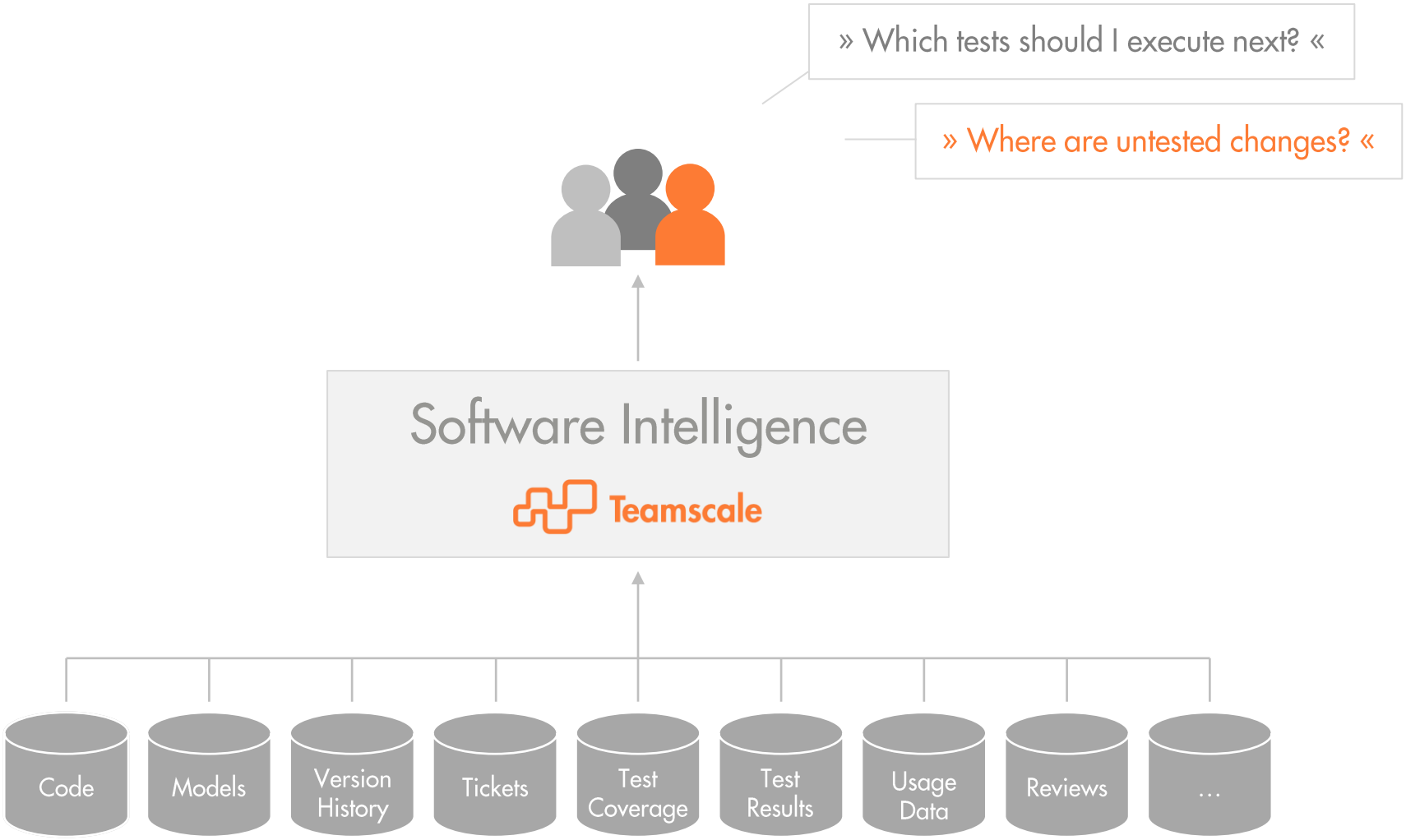


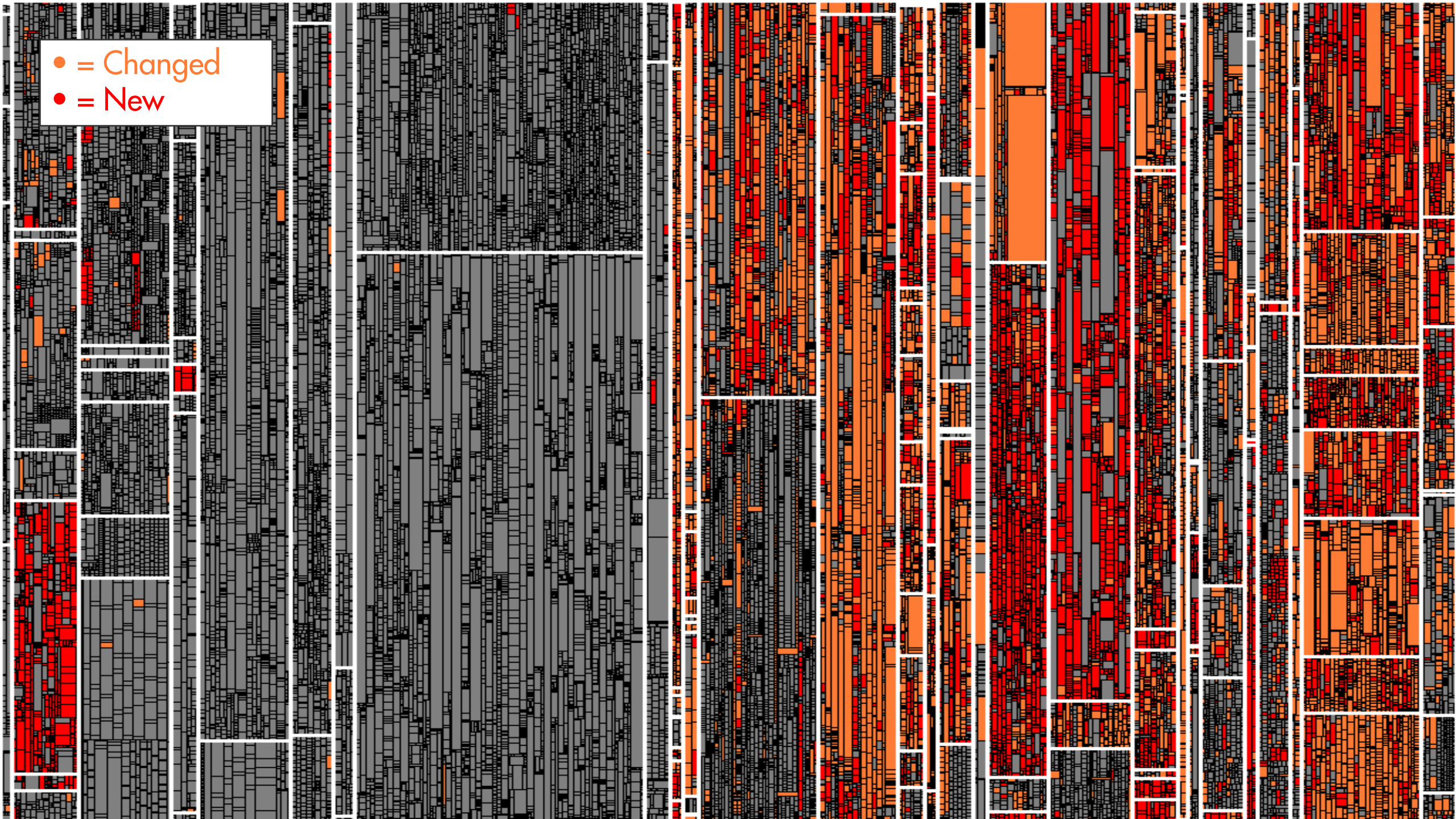

90 %

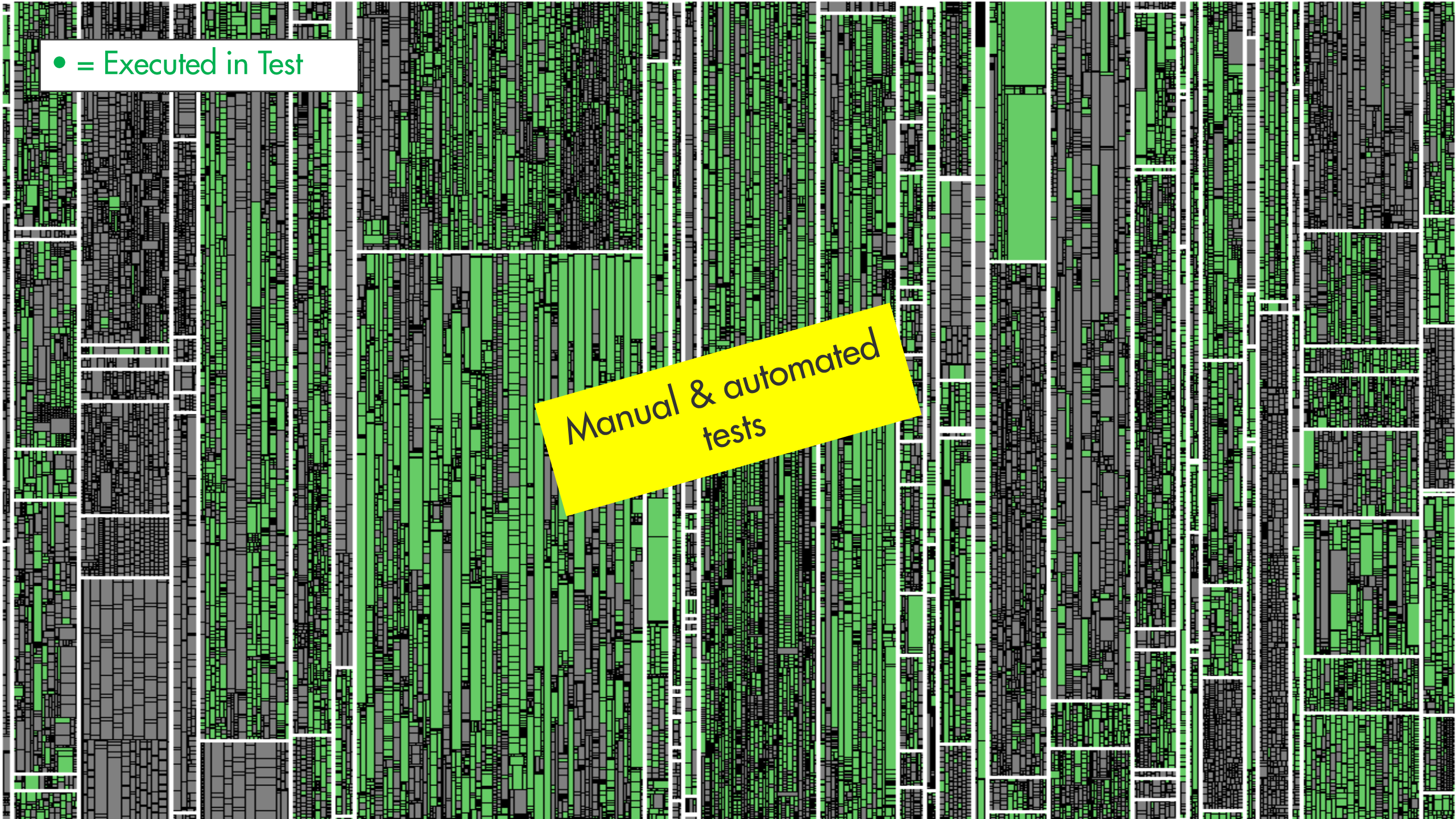


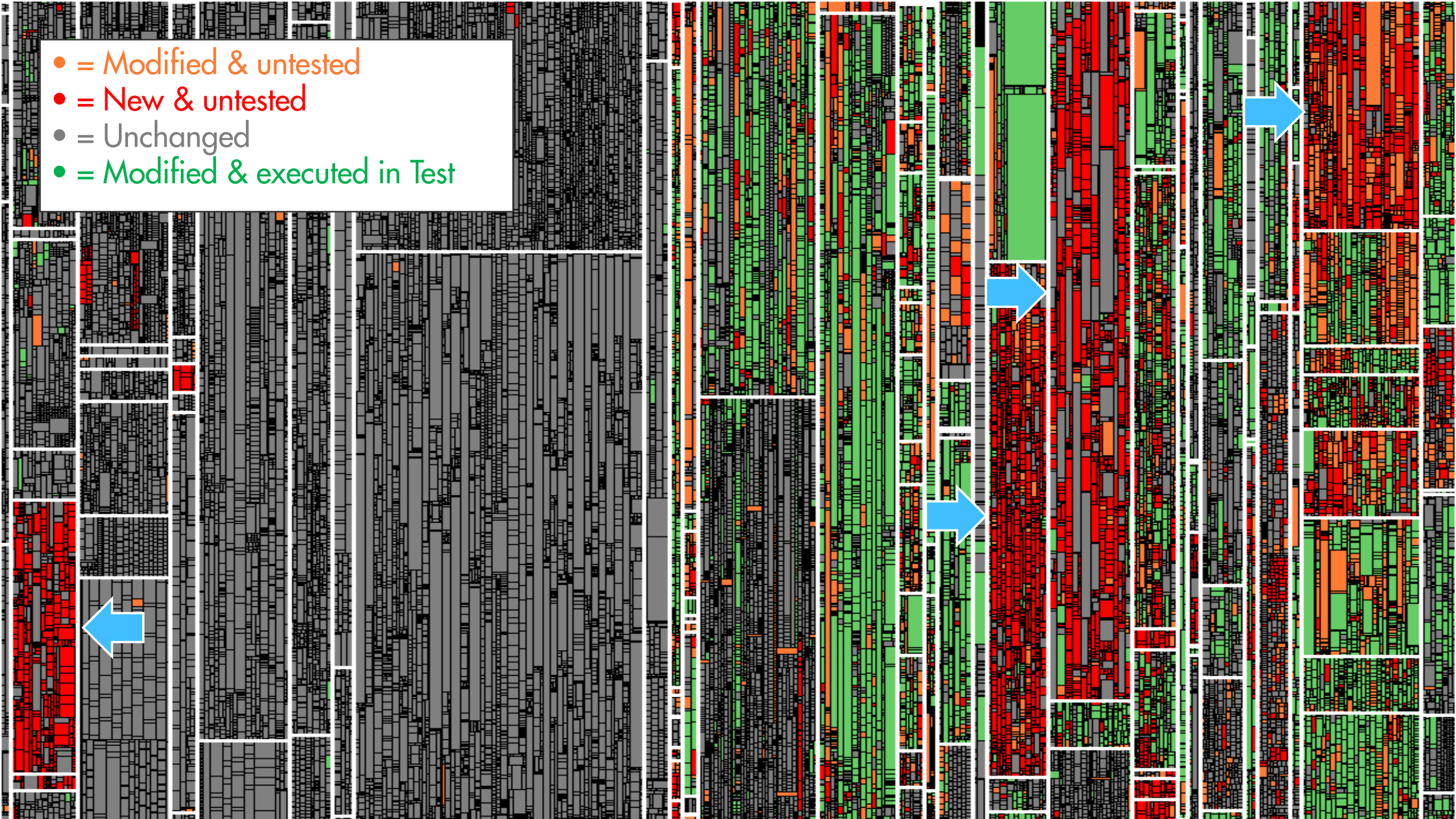





» Which tests should I execute next? «
» Where are untested changes? «
Software Intelligence
Teamscale
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…


• = Changed
• = New


• = Executed in Test
Manual & automated tests


• = Modified & untested
• = New & untested
• = Unchanged
• = Modified & executed in Test

| Issue # | Subject | | | Test Gap |
|---|---|---|---|---|
| TS-10549 | Undo/Redo for web-based architecture editor | Done |  | 0% |
| TS-10784 | Fix long method finding in TaintAnalysisRunner | Done |  | 0% |
| TS-10923 | Implement metric 'Nesting Depth' for Simulink | Done |  | 29% |
| TS-11364 | External findings are not registered during first upload | Done |  | 14% |
| TS-11942 | Manual test coverage upload during development | Done |  | 43% |
| TS-12050 | Tool for transferring findings blacklists and tasks | Done |  | 50% |
| TS-12262 | Cannot set or alter alias without reanalysis | Done |  | 0% |
| TS-13151 | Fetch parent relationship of TFS work items | Done |  | 0% |

| Issue # | Subject | | Test Gap |
|---|---|---|---|
| TS-14421 | Get rid of TestGapSynchronizer block | Done | 0% |
| TS-14733 | Remove Dataflow blocks | Done | 22% |

Done **Issue TS-14733 - Remove Dataflow blocks**

Creator: (on Apr 06 2018 19:44)

Last update: Aug 24 2018 09:32

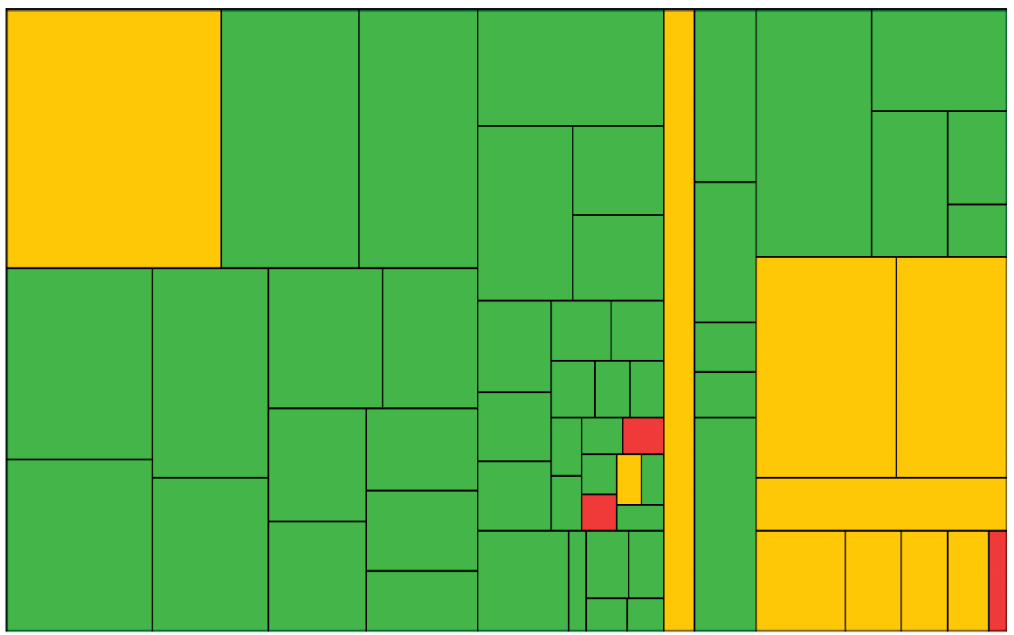
Assignee:

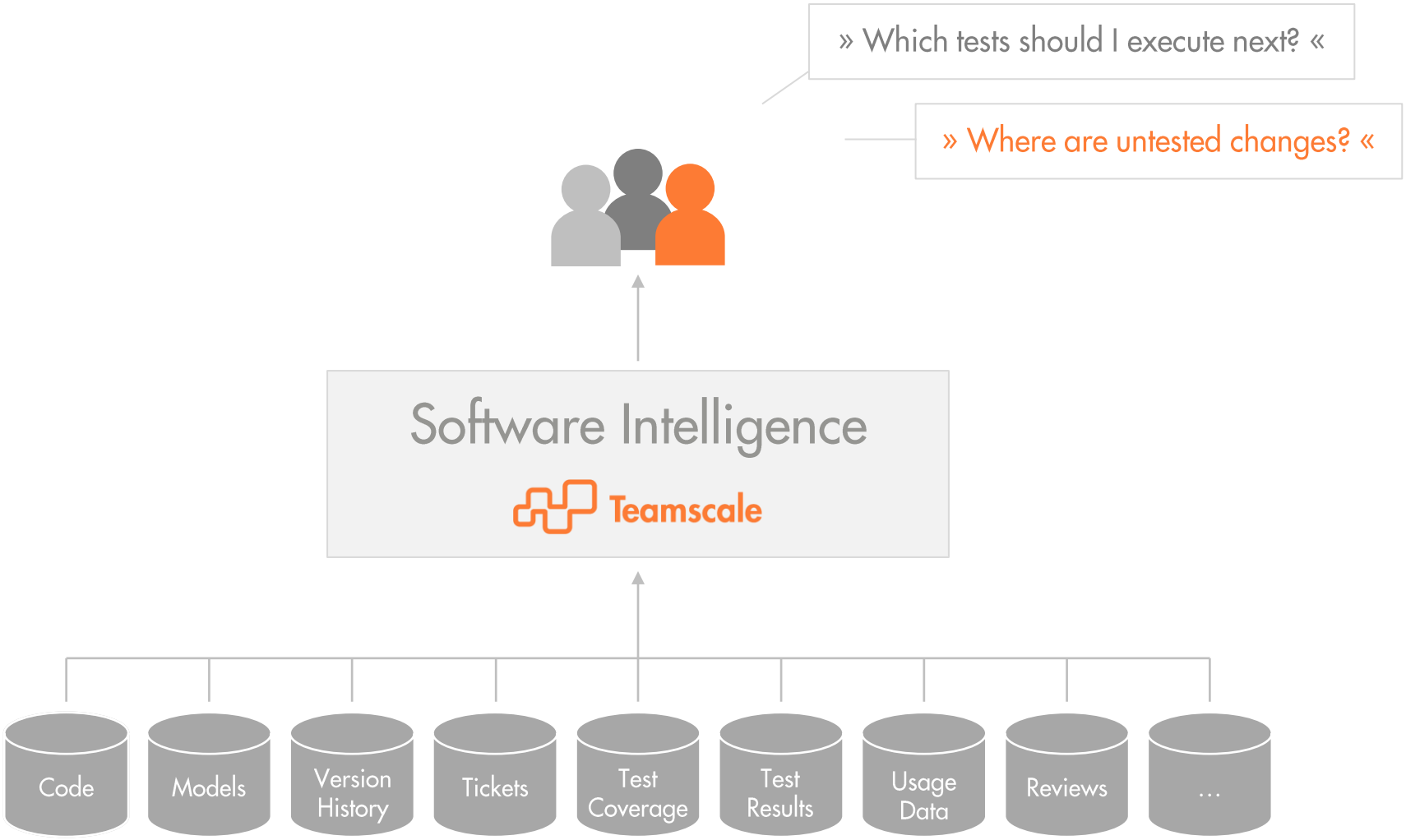
| Project | Type | Priority | Resolution | Fix Version |
|---|---|---|---|---|
| TS | Maintenance | Normal | Green | Teamscale 4.5 |

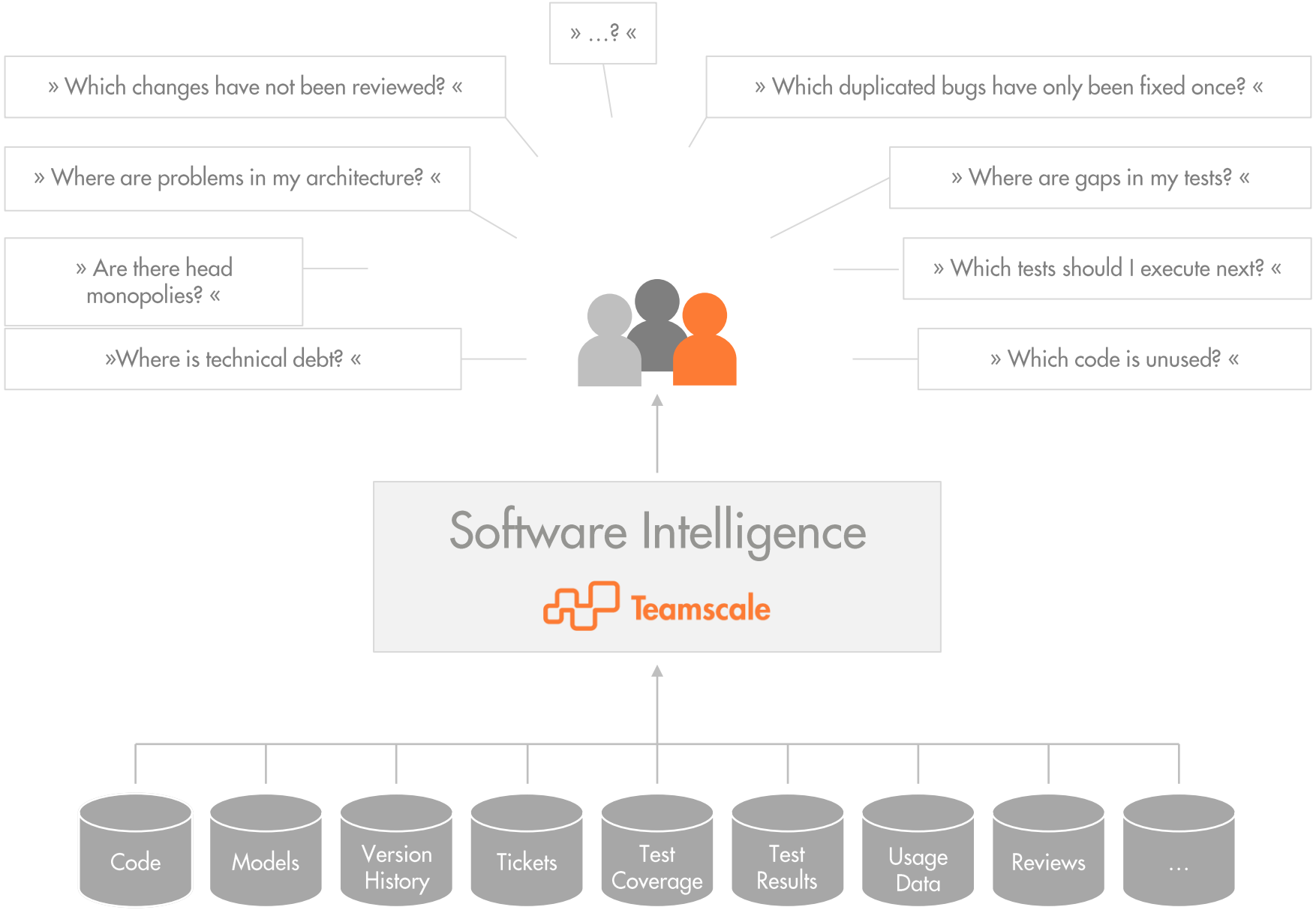
| Component | Labels | Affected Version | Customer | Customer Issue |
|---|---|---|---|---|
| Backend | Performance | | | |

| Epic Name | Freshdesk URL | Merge Request |
|---|---|---|
| | | https://git.cqse.eu/cqse/teamscale/3621 |

Aug 15 2018 12:37–Now | Test Gap: 22%




» Which tests should I execute next? «
» Where are untested changes? «
Software Intelligence
Teamscale
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…


» …? «
» Which changes have not been reviewed? «
» Which duplicated bugs have only been fixed once? «
» Where are problems in my architecture? «
» Where are gaps in my tests? «
» Are there head monopolies? «
» Which tests should I execute next? «
»Where is technical debt? «
» Which code is unused? «
Software Intelligence
Teamscale
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…

# Contact – Looking Forward to Discussions ☺

Dr. Elmar Jürgens juergens@cqse.eu +49 179 675 3863

CQSE GmbH
Centa-Hafenbrädl-Str 59
81249 München
www.cqse.eu