Test Intelligence

Explore Your Own Data From Software Tests to Find More Errors in Less Time

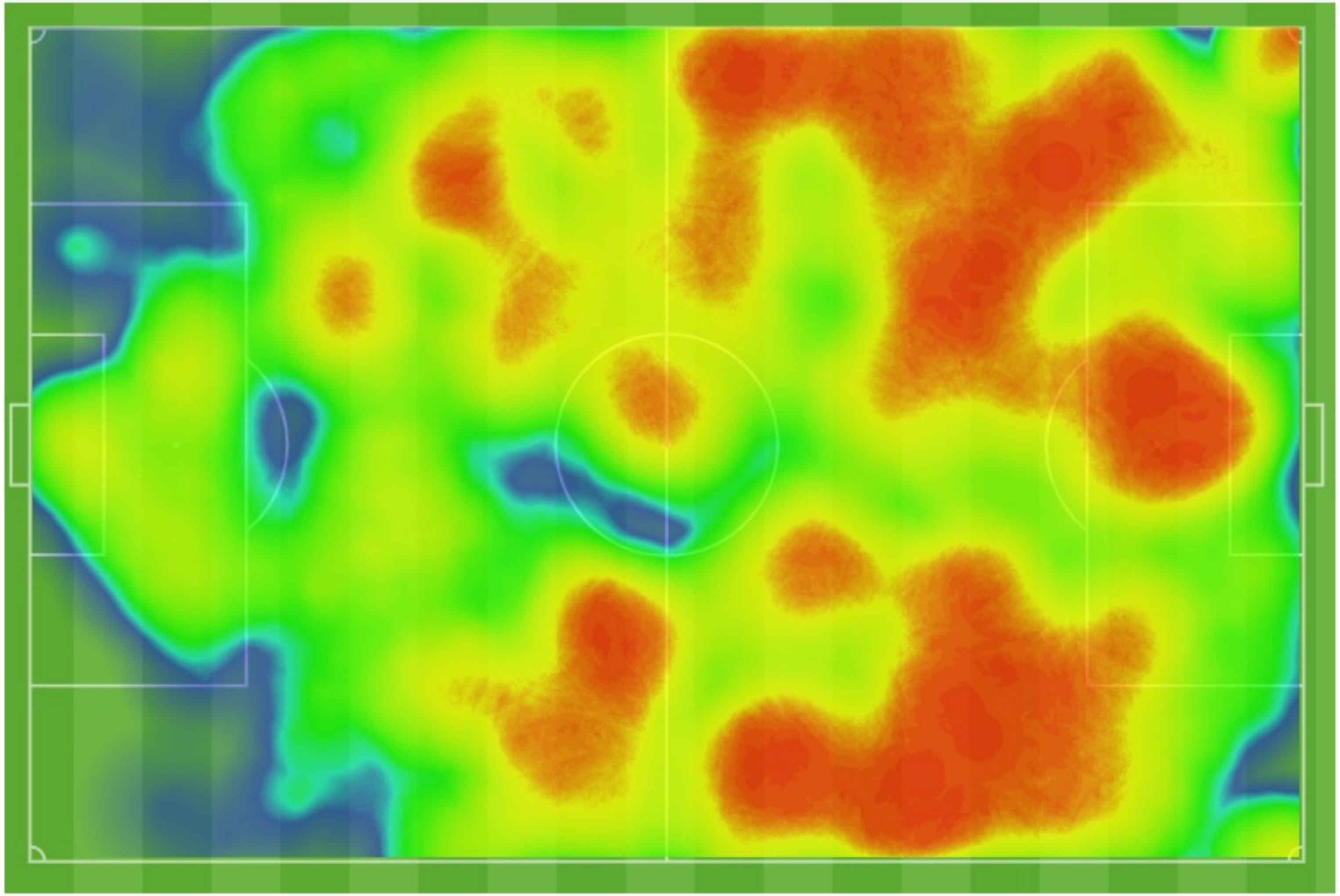
Jakob Rott

CQSE

FC Bayern München

- FC Bayern München
- TW 1 Manuel Neuer
- AW 21 Lucas Hernández
- AW 5 Benjamin Pavard
- AW 4 Niklas Süle
- AW 27 David Alaba
- MF 29 Kingsley Coman
- MF 10 Leroy Sané
- MF 25 Thomas Müller
- MF 6 Joshua Kimmich
- MF 18 Leon Goretzka
- ST 13 Eric Maxim Choupo-Moting
- Einwechselspieler
- 17 Jérôme Boateng
- 19 Alphonso Davies

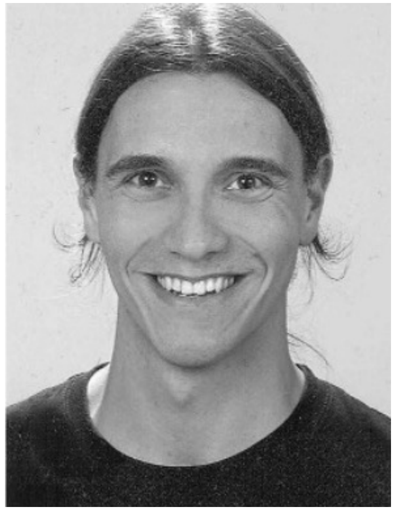
Heatmap | Touchmap



Paris Saint-Germain

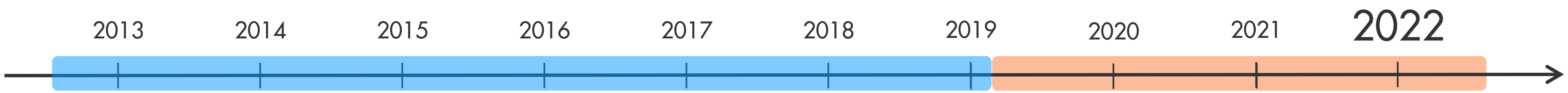
- Paris Saint-Germain
- TW 1 Keylor Navas
- AW 5 Marquinhos
- AW 31 Colin Dagba
- AW 22 Abdou Diallo
- AW 3 Presnel Kimpembe
- MF 23 Julian Draxler
- MF 27 Idrissa Gueye
- MF 10 Neymar
- MF 15 Danilo Pereira
- MF 11 Ángel Di María
- ST 7 Kylian Mbappé
- Einwechselspieler
- 25 Mitchel Bakker
- 18 Moise Kean
- 21 Ander Herrera
- 12 Rafinha

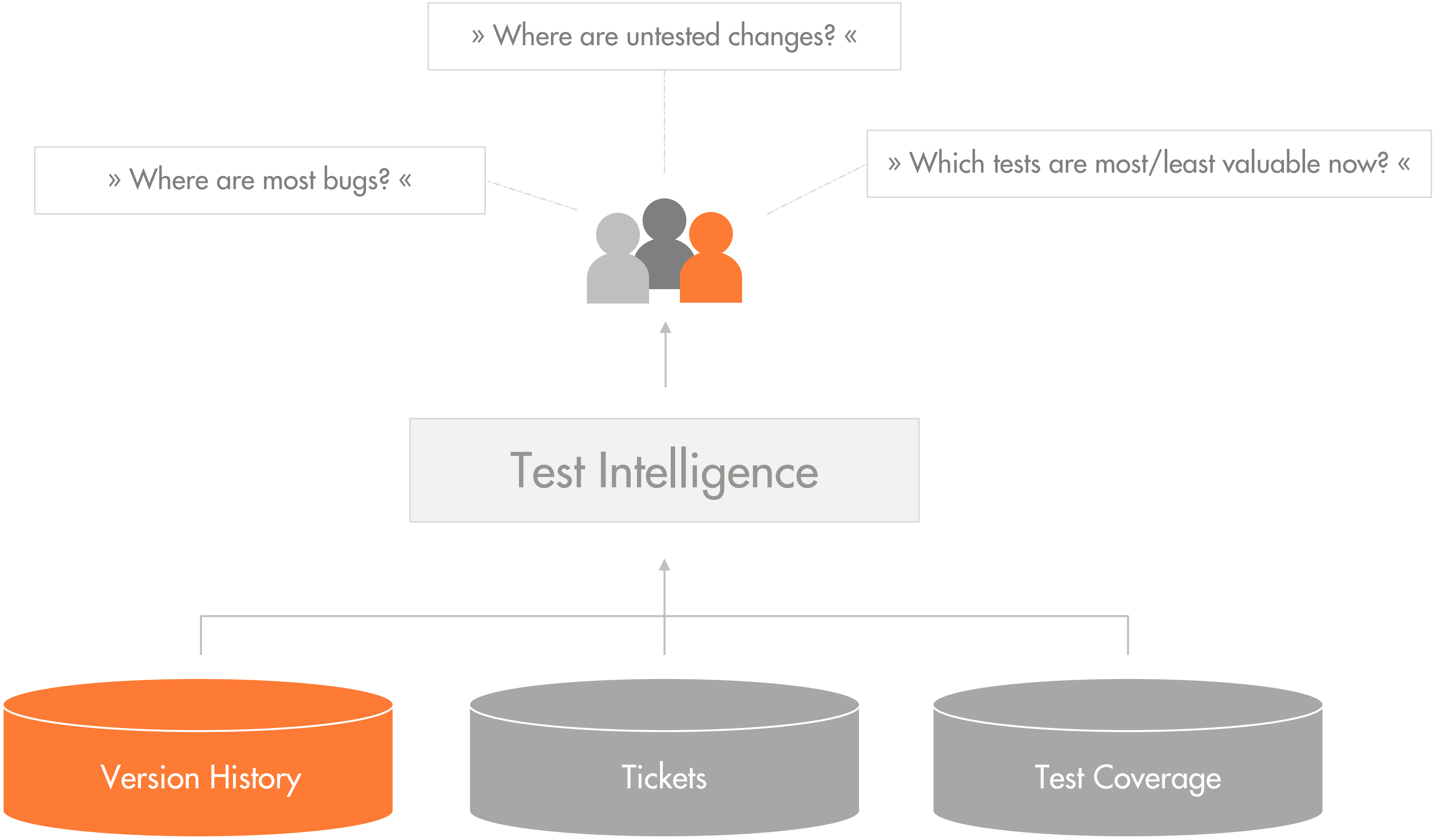
https://sportdaten.welt.de/fussball/champions-league/bayern-muenchen/paris-st-germain/2021-04-07/clvfh/2021/142784/heatmap



TUM

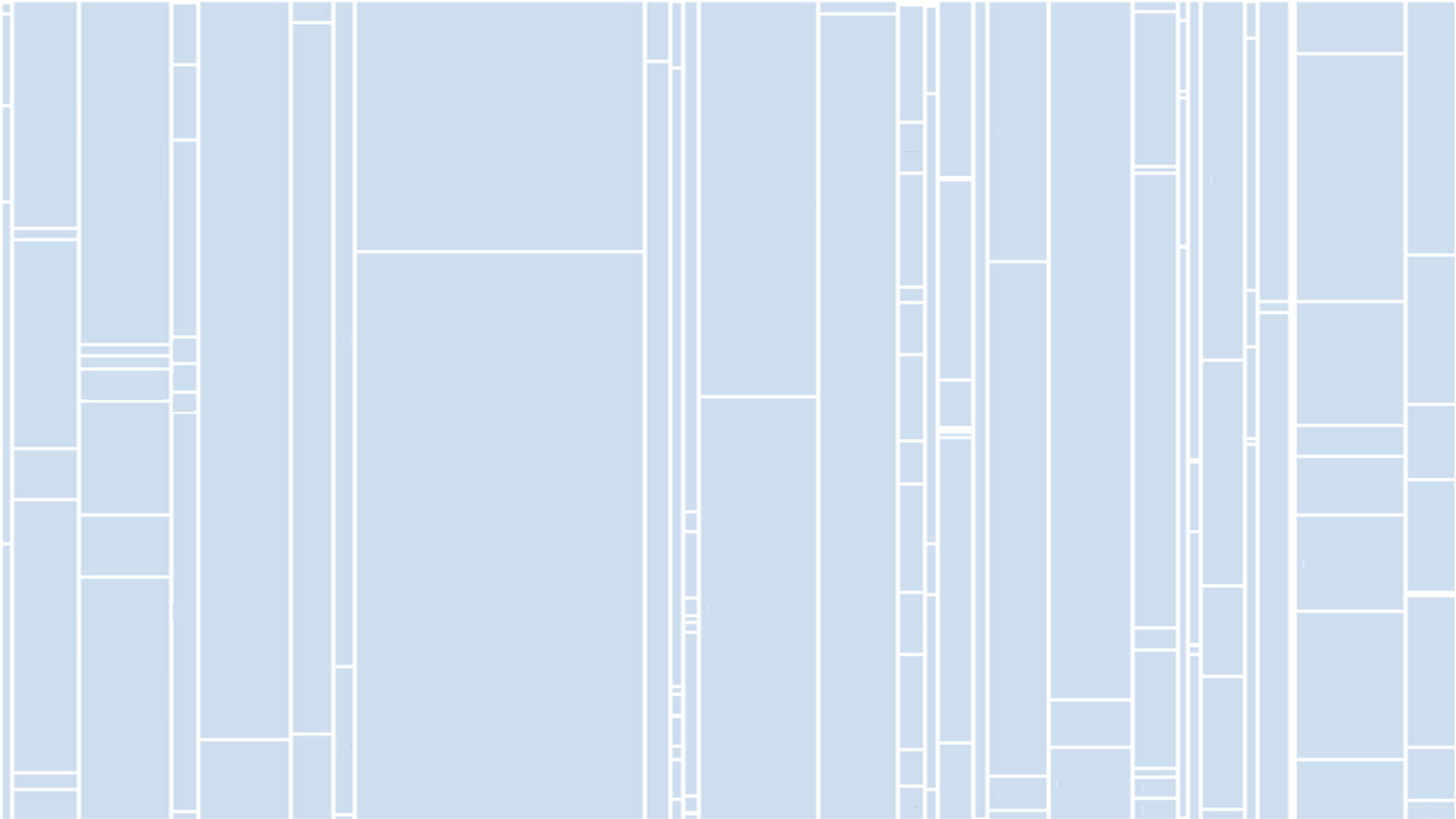
CQSE

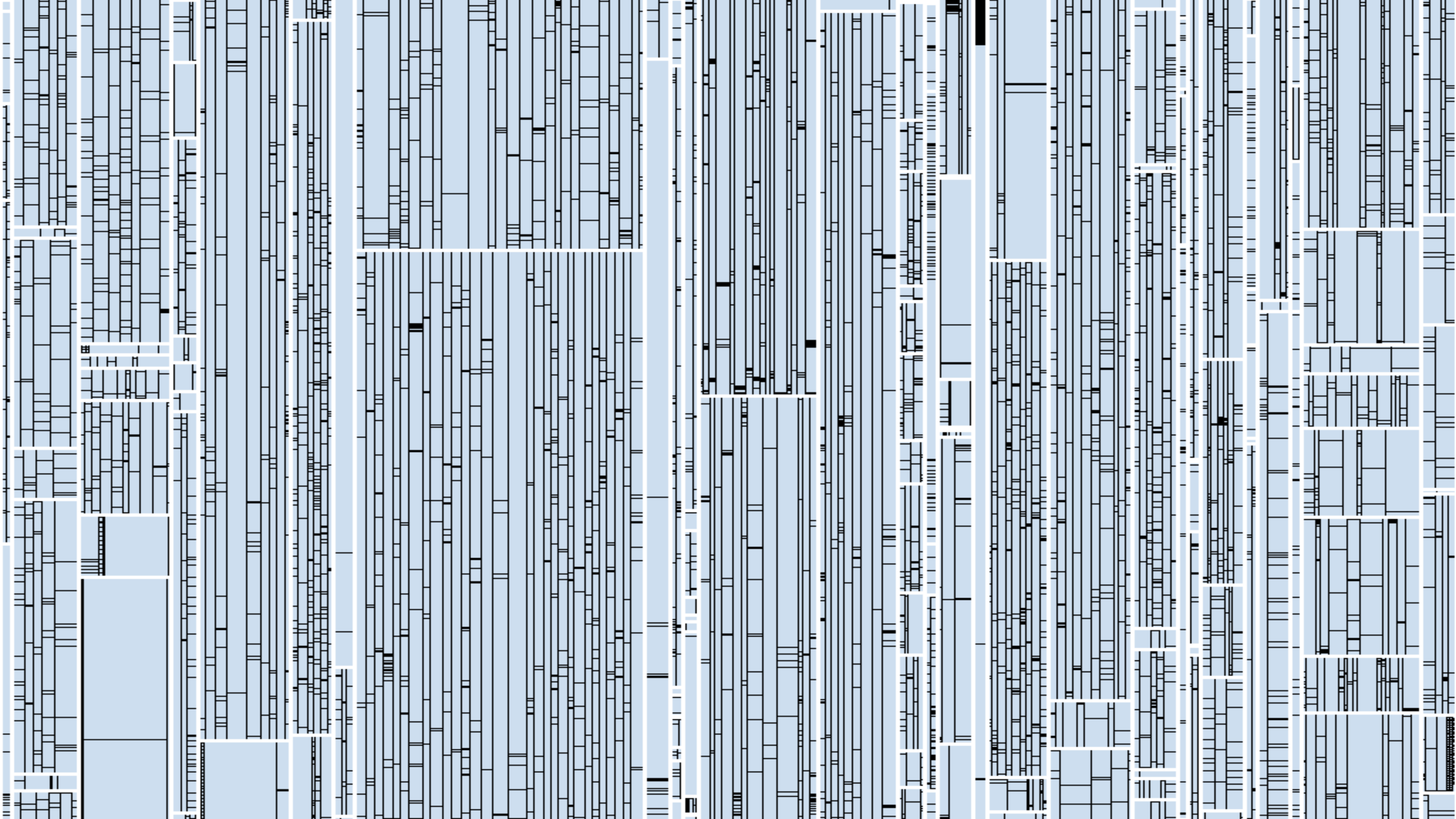


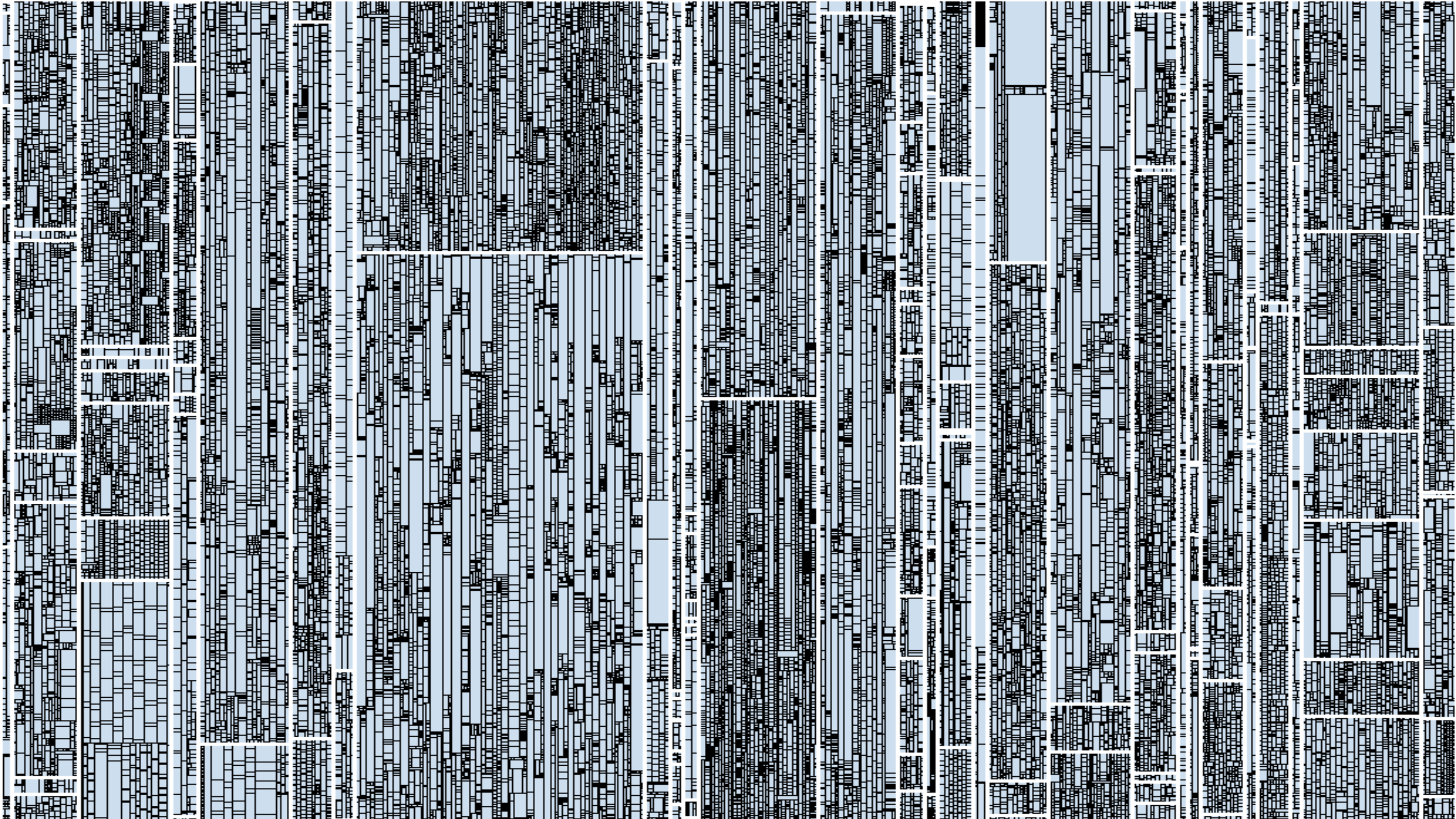

» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valuable now? «
Test Intelligence
Version History
Tickets
Test Coverage

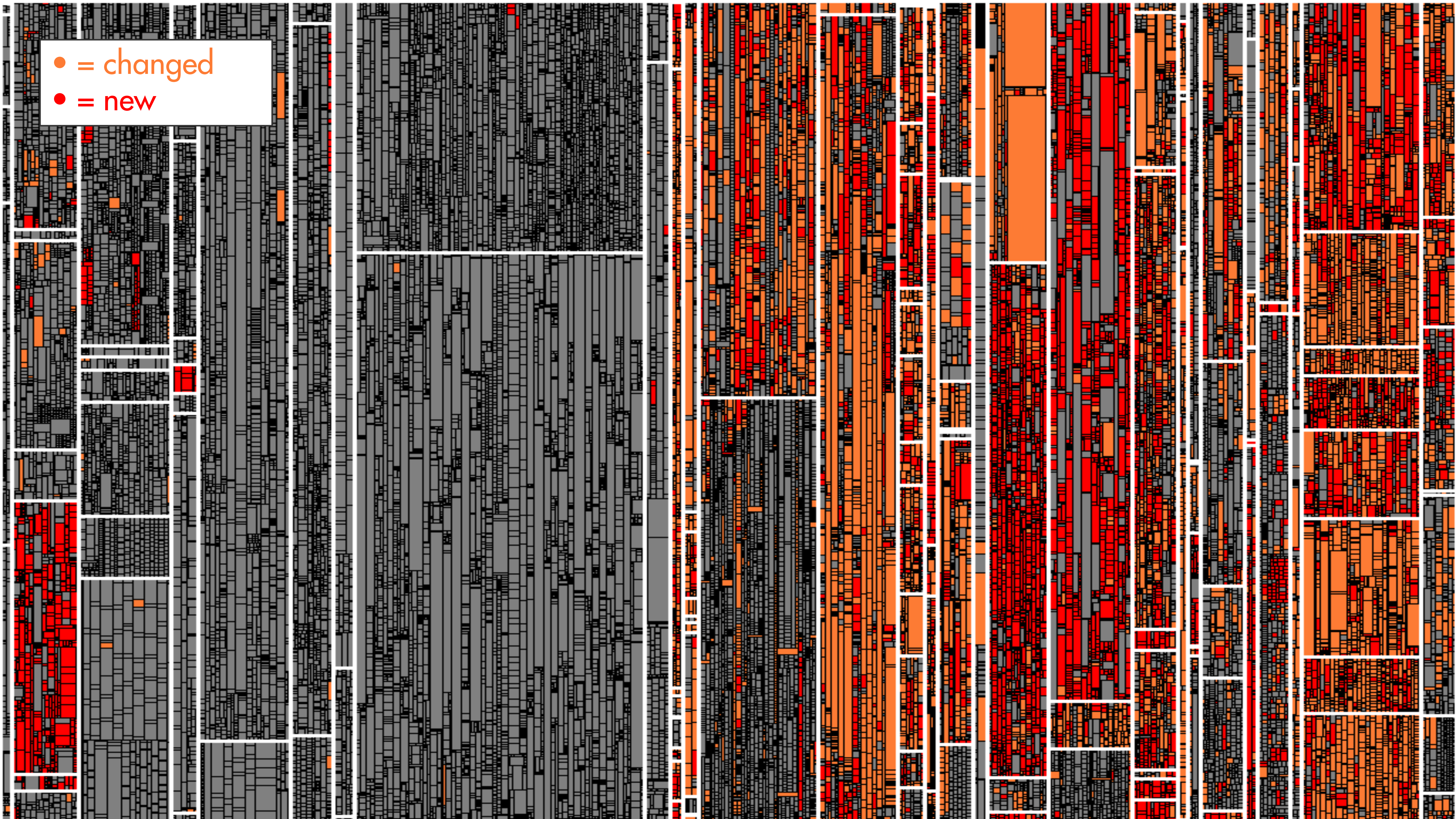
```java
/* package */class NamespaceRenames {

    /** Maps from old name fragment to new name fragment */
    public Set<ImmutablePair<String, String>> namespaceRenames = new HashSet<

    /**
     * Computes a rename rule from an old and a new namespace name based on a
     * type name correspondence. <code>
     * For name1 = a.oldnamespace.b.c.D and name2 = a.newnamespace.b.c.D, res
     * </code>
     * */
    public static ImmutablePair<String, String> computeRenameRule(
            String fqTypeName1, String fqTypeName2, String separator) {

        if (fqTypeName1.equals(fqTypeName2)) {
            return null;
        }

        String commonSuffix = StringUtils.longestCommonSuffix(fqTypeName1,
                fqTypeName2);
        if (StringUtils.isEmpty(commonSuffix)) {
            return null;
        }

        int separatorPosition = commonSuffix.indexOf(separator);
        if (separatorPosition != -1) {
            commonSuffix = commonSuffix.substring(separatorPosition);
        }

        String from = StringUtils.stripSuffix(commonSuffix, fqTypeName1);
        String to = StringUtils.stripSuffix(commonSuffix, fqTypeName2);

        return new ImmutablePair<String, String>(from, to);
    }
```

```java
/* package */class NamespaceRenames {

    /** Maps from old name fragment to new name fragment */
    public Set<ImmutablePair<String, String>> namespaceRenames = new Has

    // TODO (LH) Doc vs. method name : 'Compute' vs. 'find'
    /**
     * Computes a rename rule from an old and a new namespace name based
     * type name correspondence. <code>
     * For name1 = a.oldnamespace.b.c.D and name2 = a.newnamespace.b.c.D
     * </code>
     * */
    // TODO (LH) PLease reflect in identifier names that they refert to
    // names
    public static ImmutablePair<String, String> findRenameRule(String na
            String name2, String separator) {

        if (name1.equals(name2)) {
            return null;
        }

        // TODO (LH) Looks like StringUtils is missing a 'longestCommons
        String commonSuffix = reverse(StringUtils.longestCommonPrefix(
                reverse(name1), reverse(name2)));
        if (StringUtils.isEmpty(commonSuffix)) {
            return null;
        }

        int separatorPosition = commonSuffix.indexOf(separator);
        if (separatorPosition != -1) {
            commonSuffix = commonSuffix.substring(separatorPosition);
        }

        // TODO (LH) Please reflect in identifier names that these are
        // namespaces
        String from = StringUtils.stripSuffix(commonSuffix, name1);
        String to = StringUtils.stripSuffix(commonSuffix, name2);

        return new ImmutablePair<String, String>(from, to);
    }
```

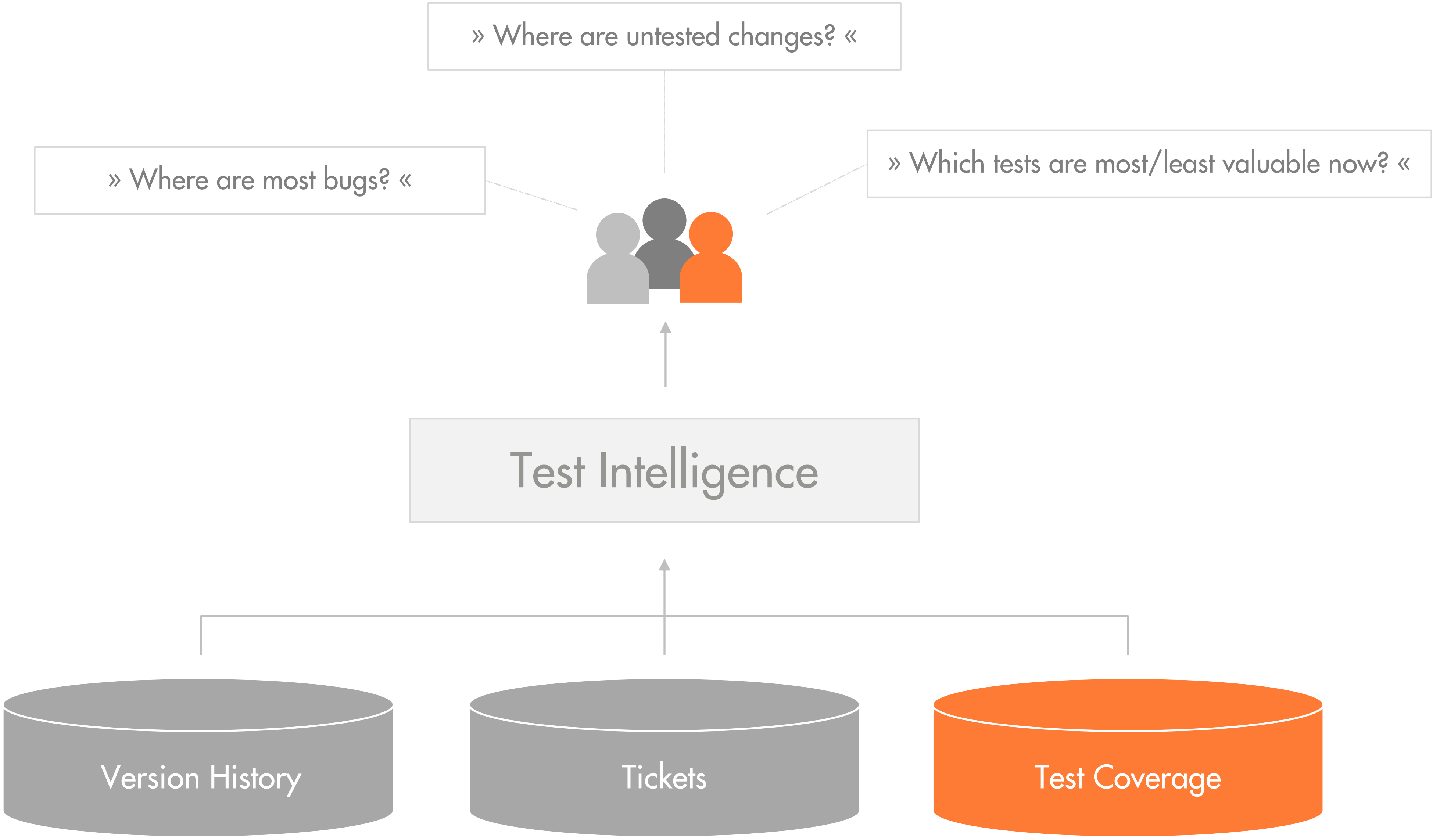




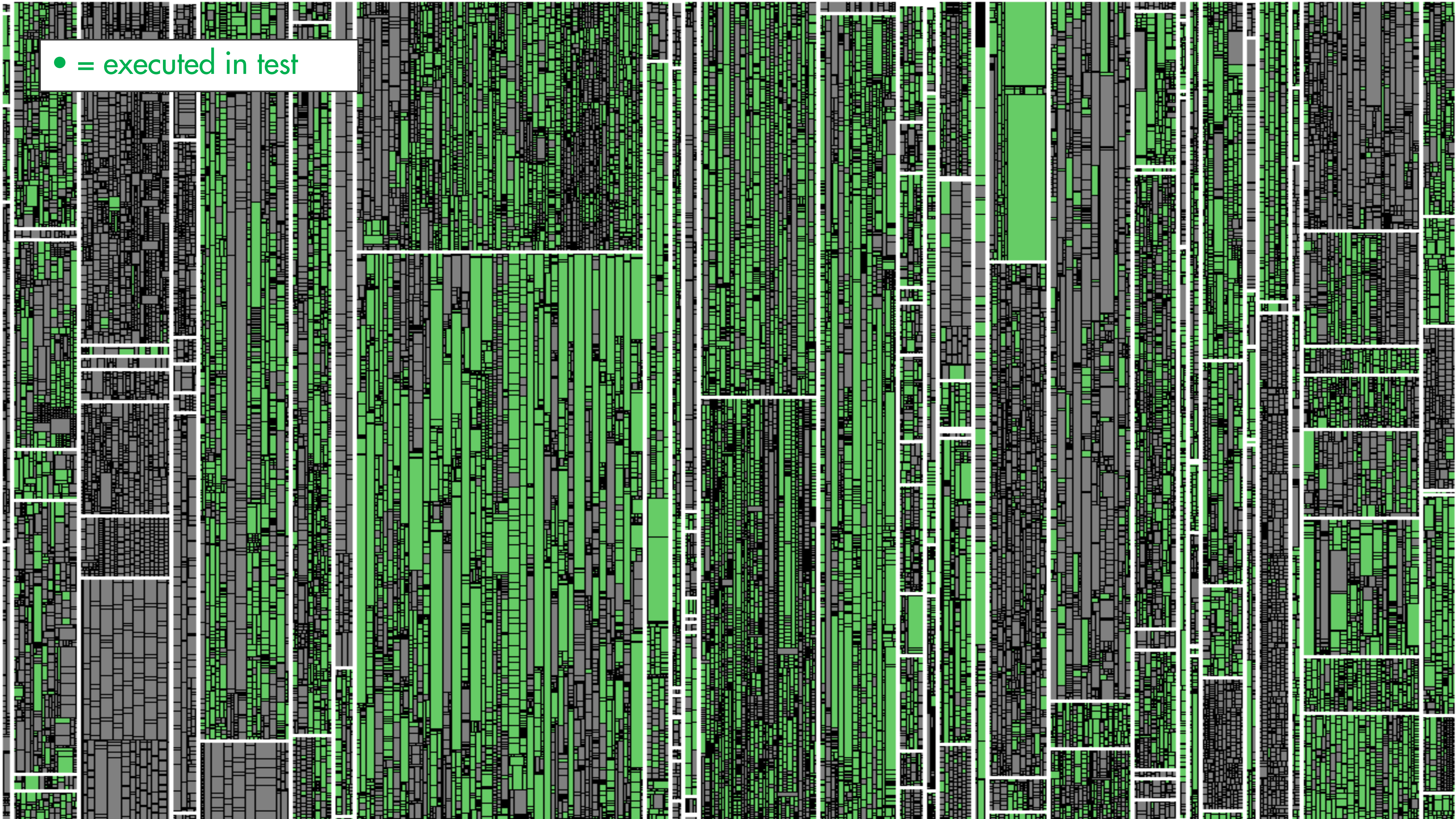


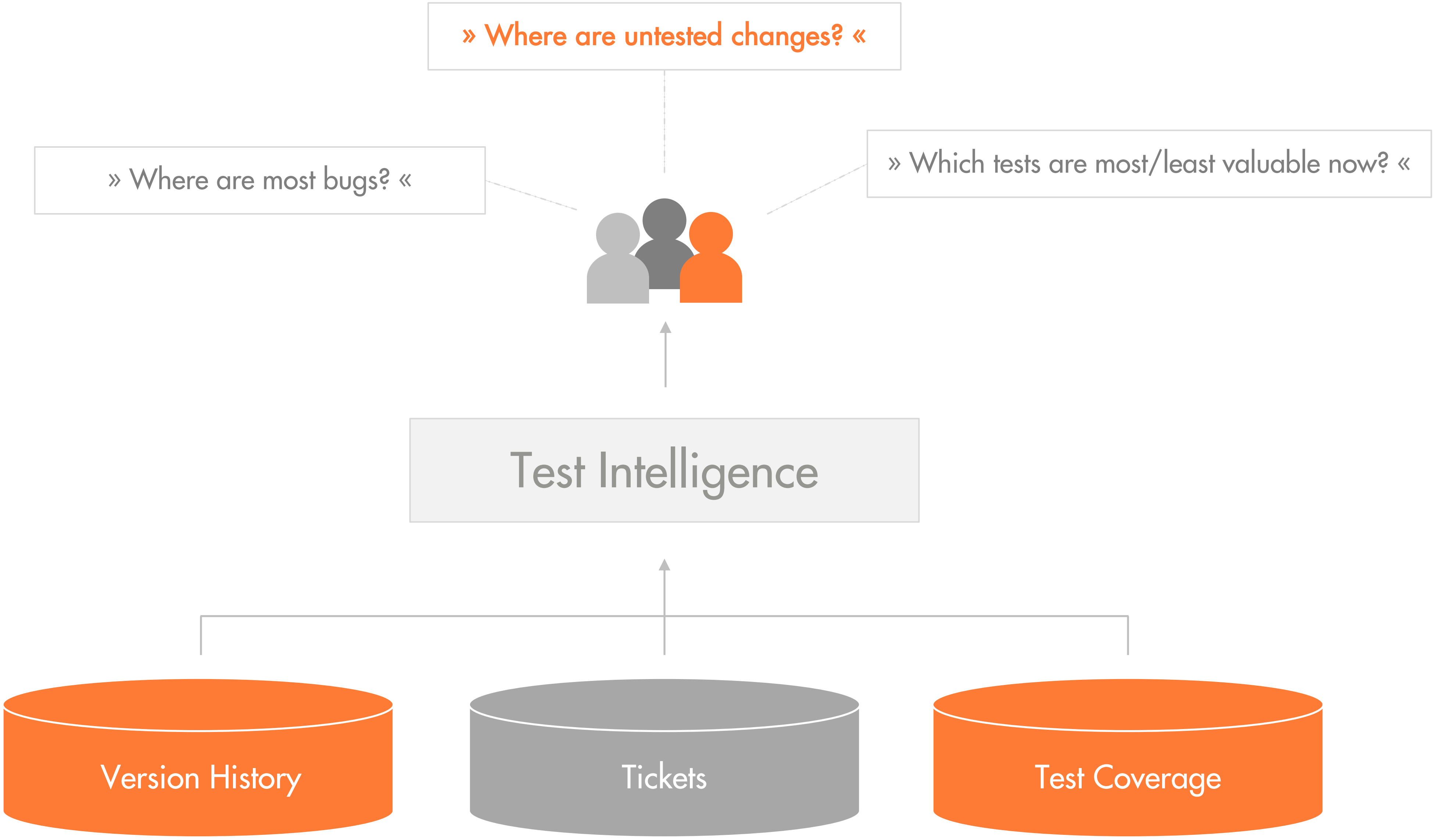


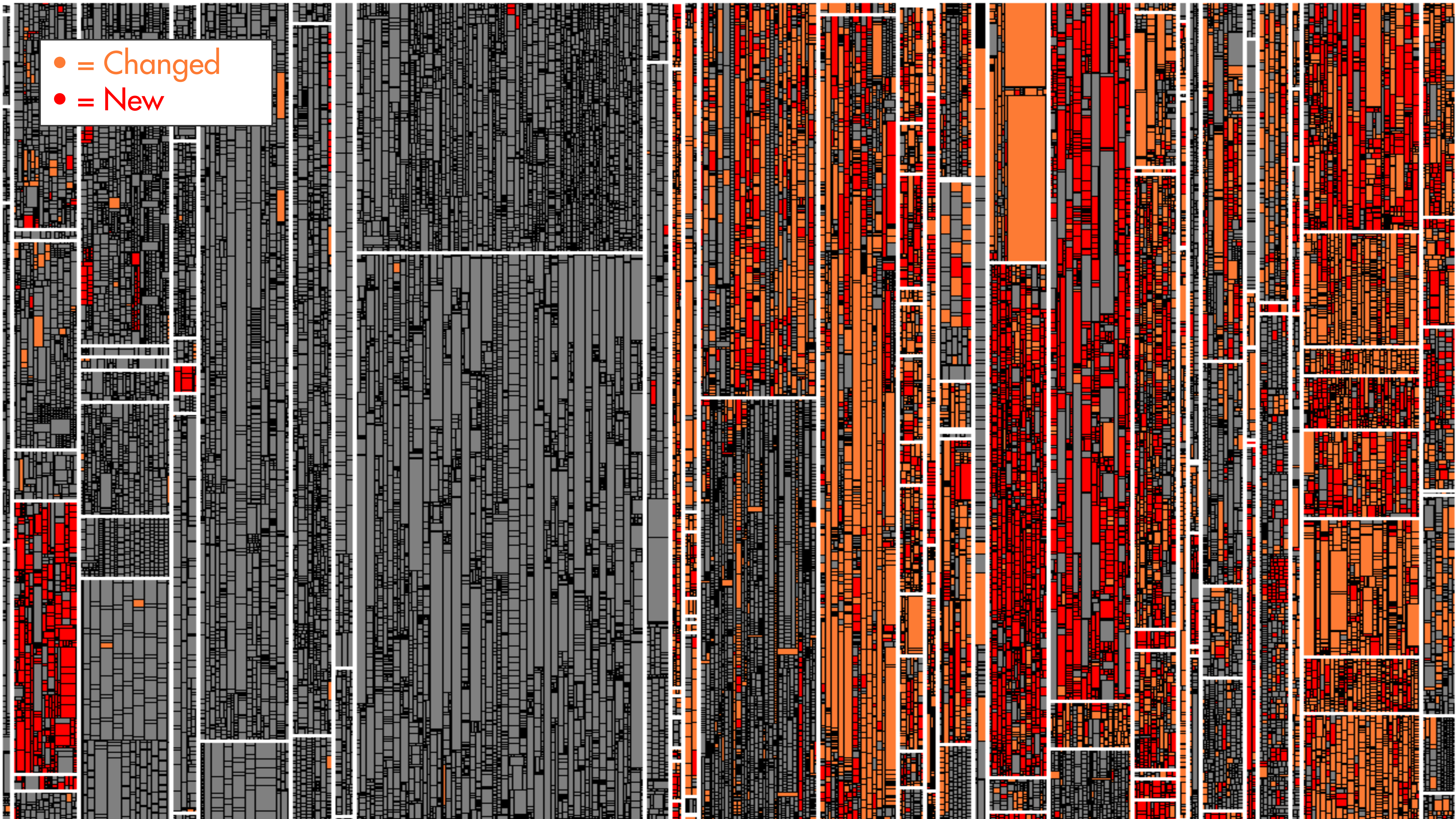
● = changed
● = new

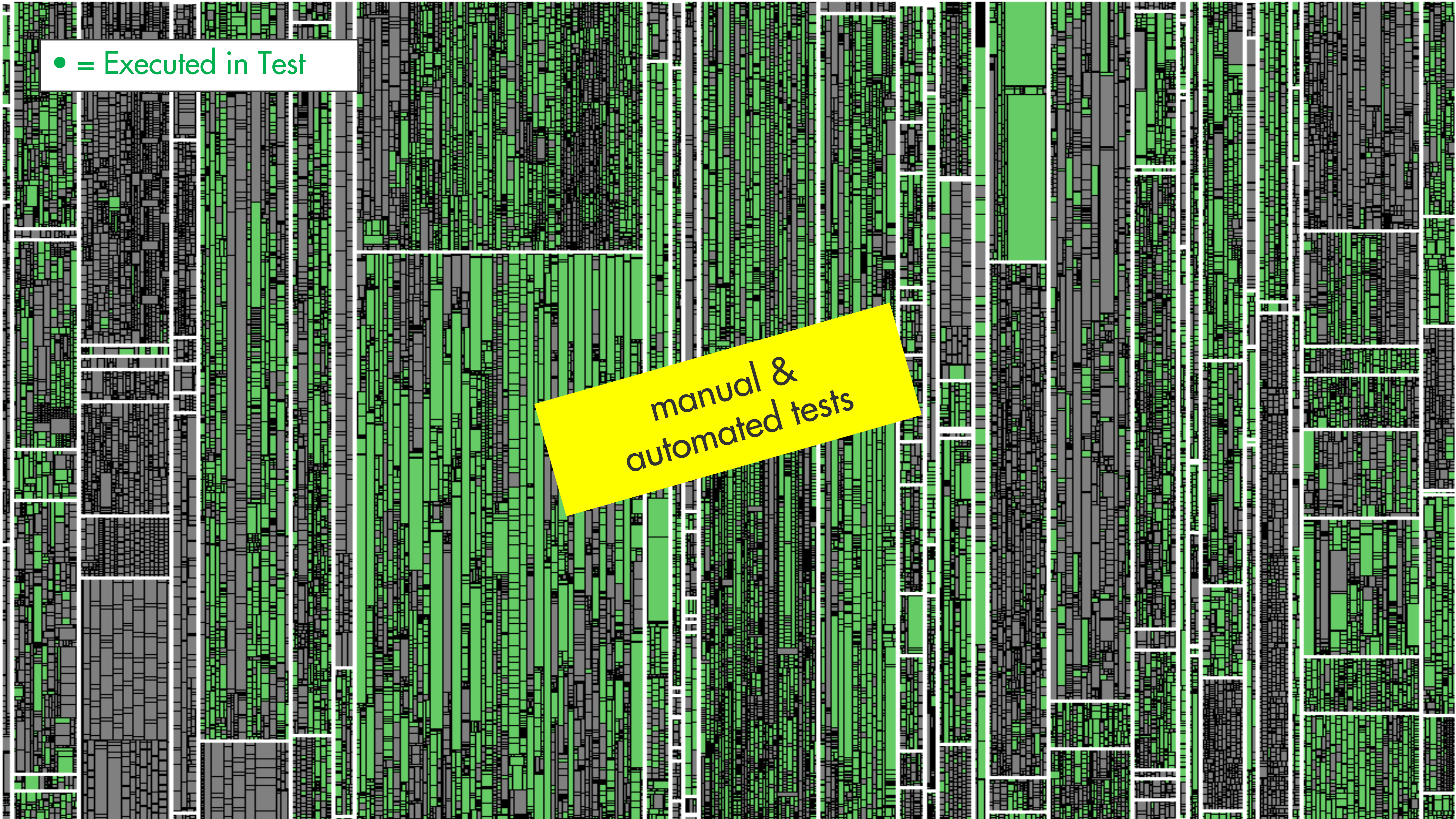

» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valuable now? «
Test Intelligence
Version History
Tickets
Test Coverage

```java
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);
        }
    }
}
```


● = executed in test


» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valuable now? «
Test Intelligence
Version History
Tickets
Test Coverage


= Changed
= New

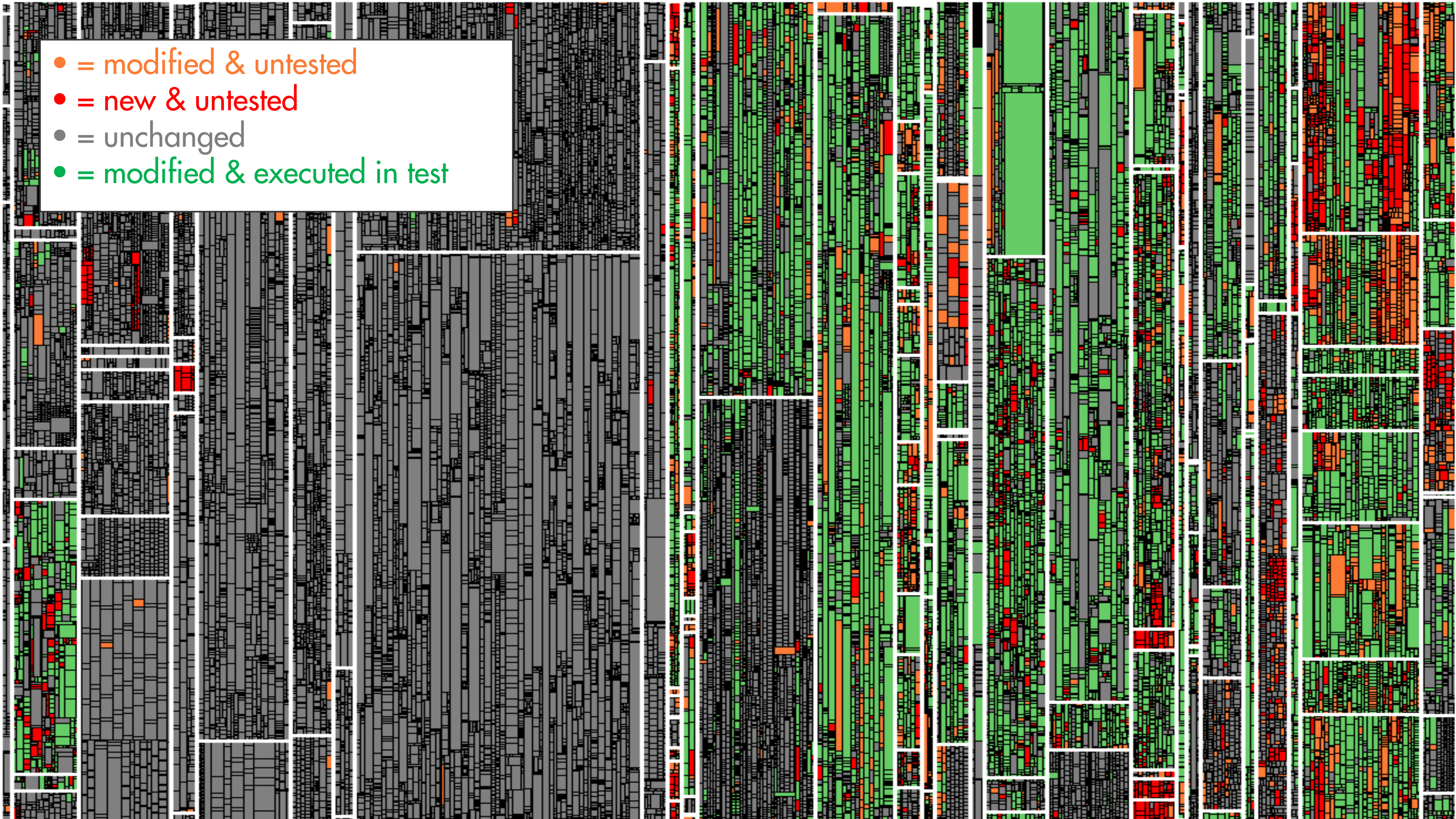

● = Executed in Test
manual &
automated tests


● = modified & untested
● = new & untested
● = unchanged
● = modified & executed in test

100% Change Coverage

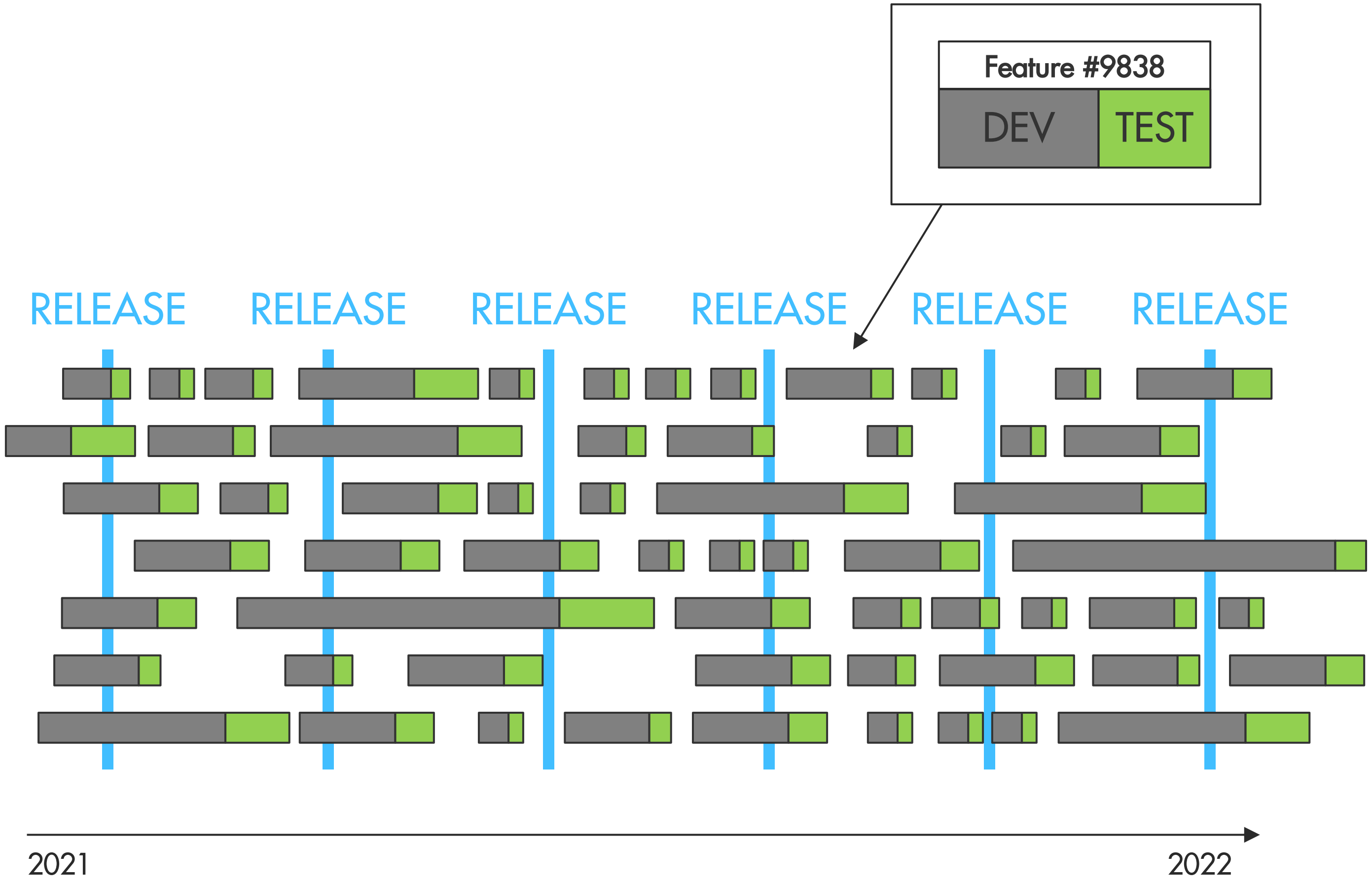
100% Change Coverage → 0 Bugs

● = modified & untested
● = new & untested
● = unchanged
● = modified & executed in test

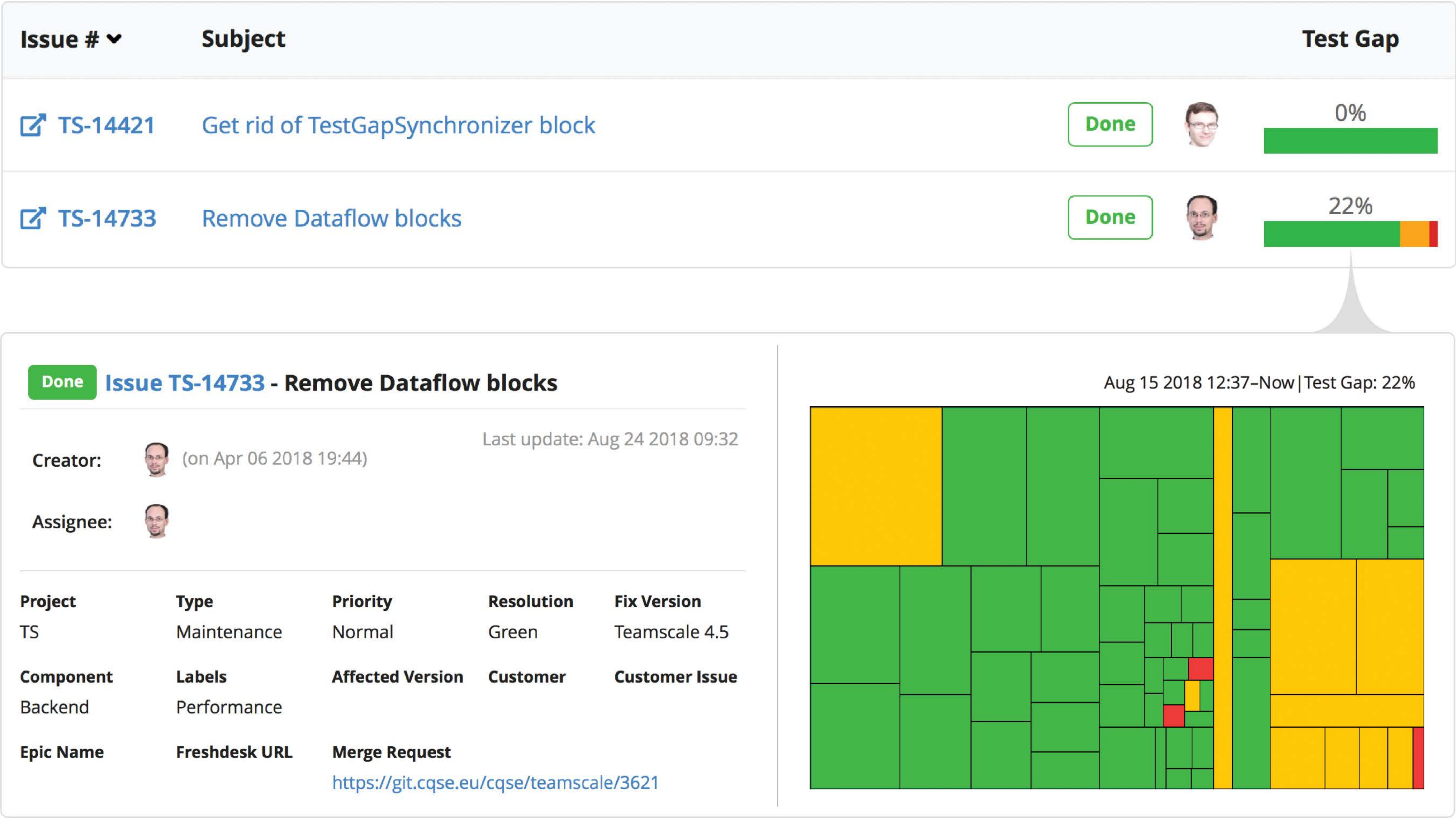


• = modified & untested
• = new & untested
• = unchanged
• = modified & executed in test

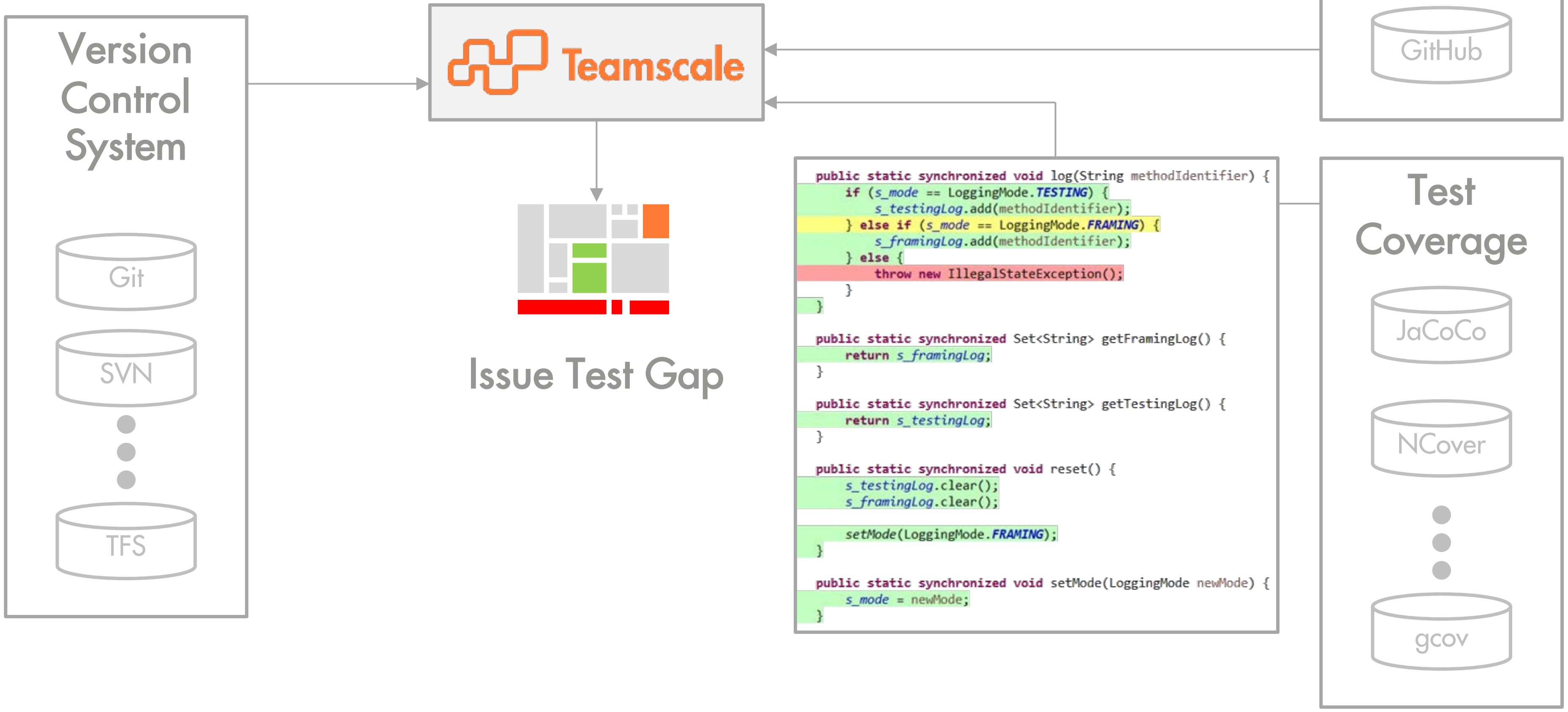
Continuous Testing



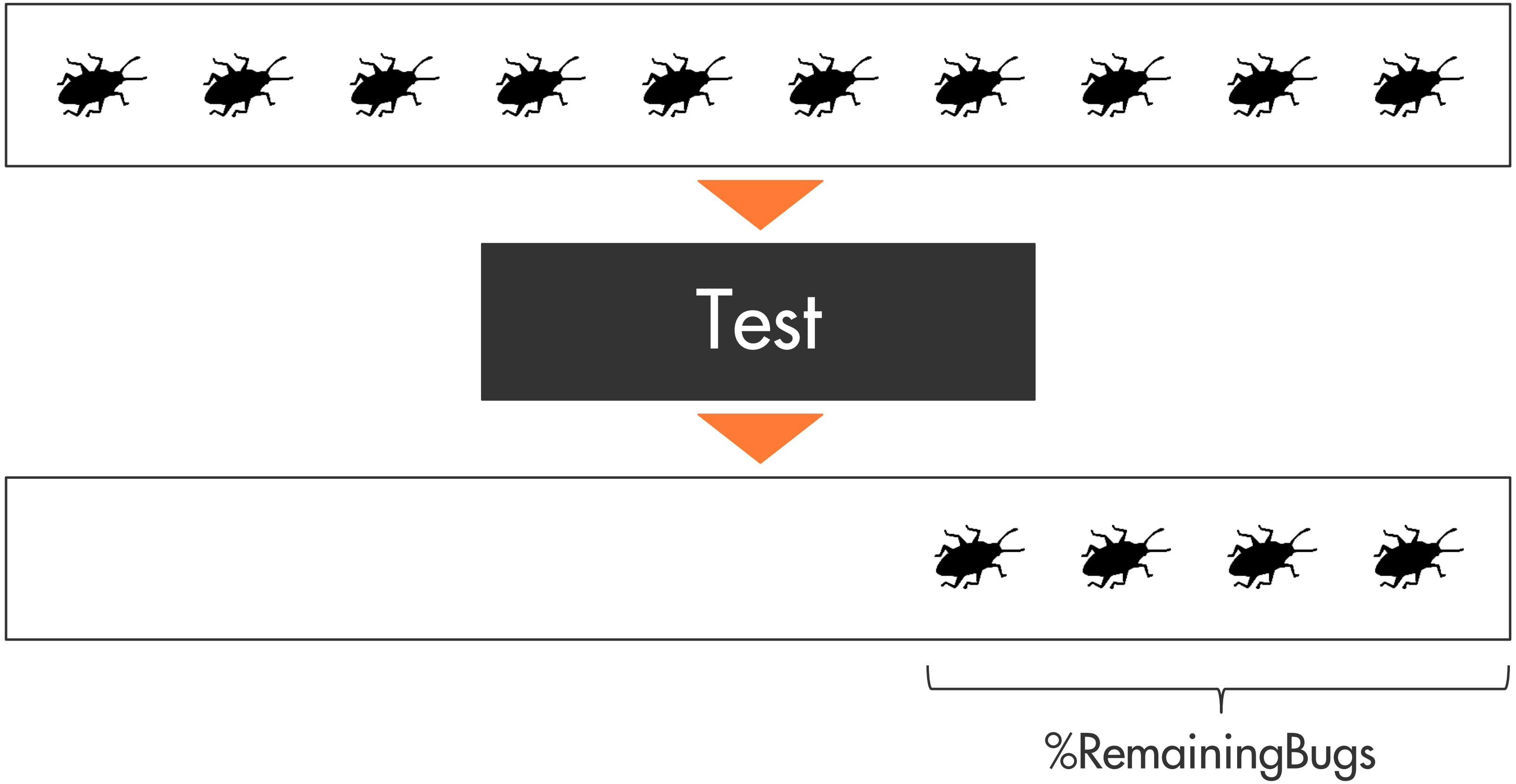
Issue #	Subject		Test Gap
TS-10549	Undo/Redo for web-based architecture editor	Done	0%
TS-10784	Fix long method finding in TaintAnalysisRunner	Done	0%
TS-10923	Implement metric 'Nesting Depth' for Simulink	Done	29%
TS-11364	External findings are not registered during first upload	Done	14%
TS-11942	Manual 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
Last update: Aug 24 2018 09:32
Creator:
(on Apr 06 2018 19:44)
Assignee:
Project
TS
Type
Maintenance
Priority
Normal
Resolution
Green
Fix Version
Teamscale 4.5
Component
Backend
Labels
Performance
Affected Version
Customer
Customer Issue
Epic Name
Freshdesk URL
Merge Request
https://git.cqse.eu/cqse/teamscale/3621
Aug 15 2018 12:37–Now | Test Gap: 22%

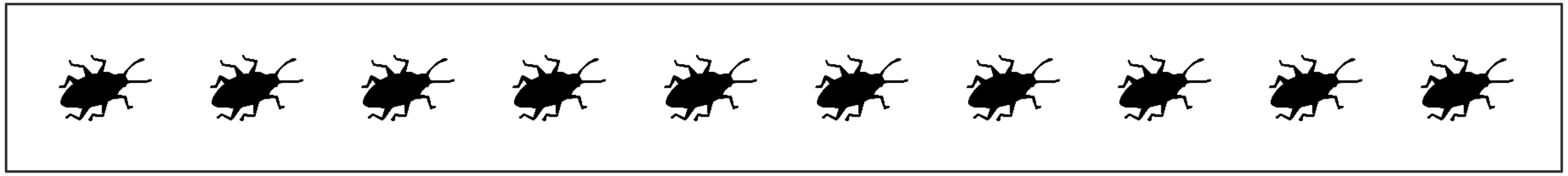
CR#9838: Added TODO	26.07.16 16:38
CR#9838: Adjust naming	26.07.16 15:33
CR#9533: RED	26.07.16 15:13
CR#9533: GREEN	26.07.16 15:12
CR#10181: Added new finding for deprecated classes, methods and fields	26.07.16 14:43
CR#10037: Moved ReviewMetricsSynchronizer to Crucible package and made some improvements to its internal structure	26.07.16 14:31
CR#10037: Updated aggregation strategy of open reviews so each review is only counted once, even over multiple files	26.07.16 13:04
CR#10203: Fixed "field could be made final" for Java interfaces	26.07.16 12:16
CR#10200: Rename pathRestriction -> subPath (1)	26.07.16 11:35
CR#10172: Removed unwanted colons from headers in the commit view of the activity perspective	26.07.16 11:20
CR#9838: Fix: only one color of a threshold is specified in a corridor	26.07.16 11:14
CR#0: Fix findings	26.07.16 11:01
CR#9838: minor improvement	26.07.16 10:56
CR#10199: Mail notifications do now support starTLS	26.07.16 10:52
CR#9533: working on developer feedback	26.07.16 09:50
CR#9838: Amend last commit	26.07.16 09:38
CR#9838: minor refactoring	26.07.16 09:05
CR#9838: Fix NPE	26.07.16 09:01



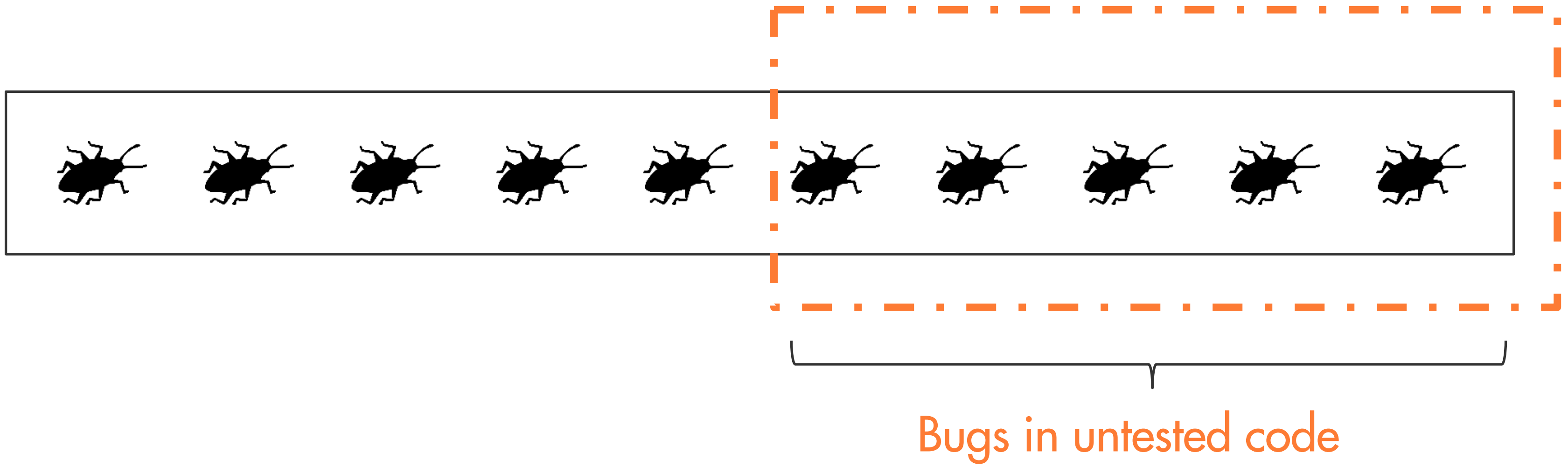
Benefit Analysis of TGA


Test
%RemainingBugs

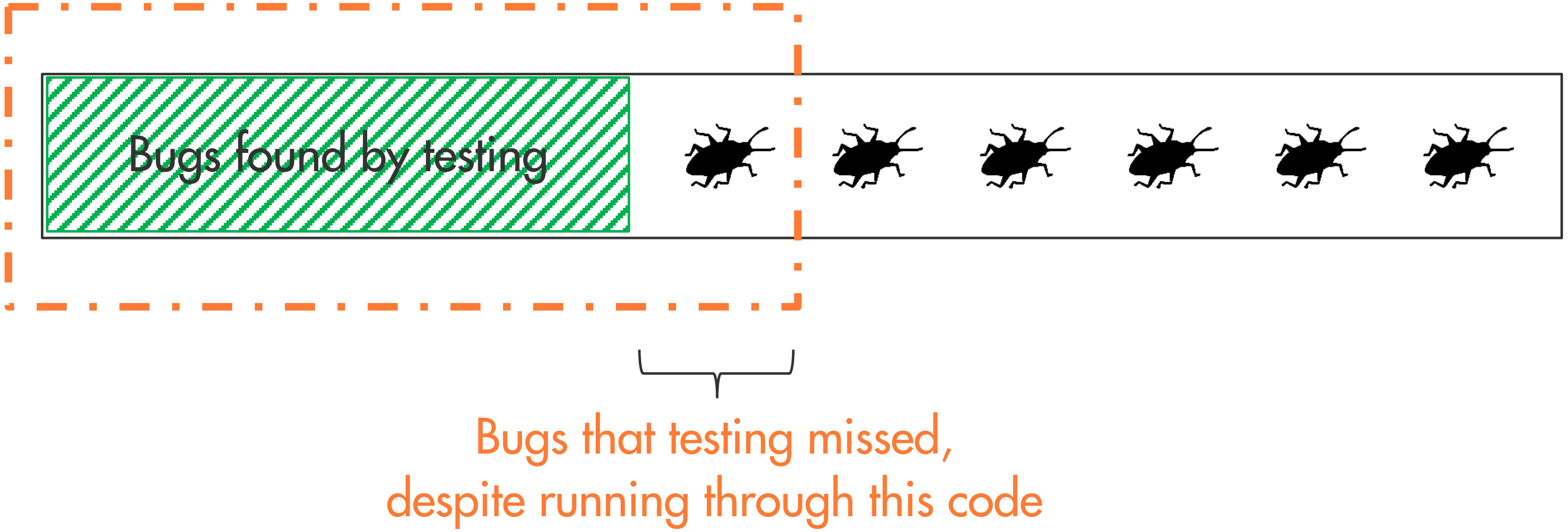
$$\%RemainingBugs = \%Tested * TestIneffectiveness + \%Testgap$$

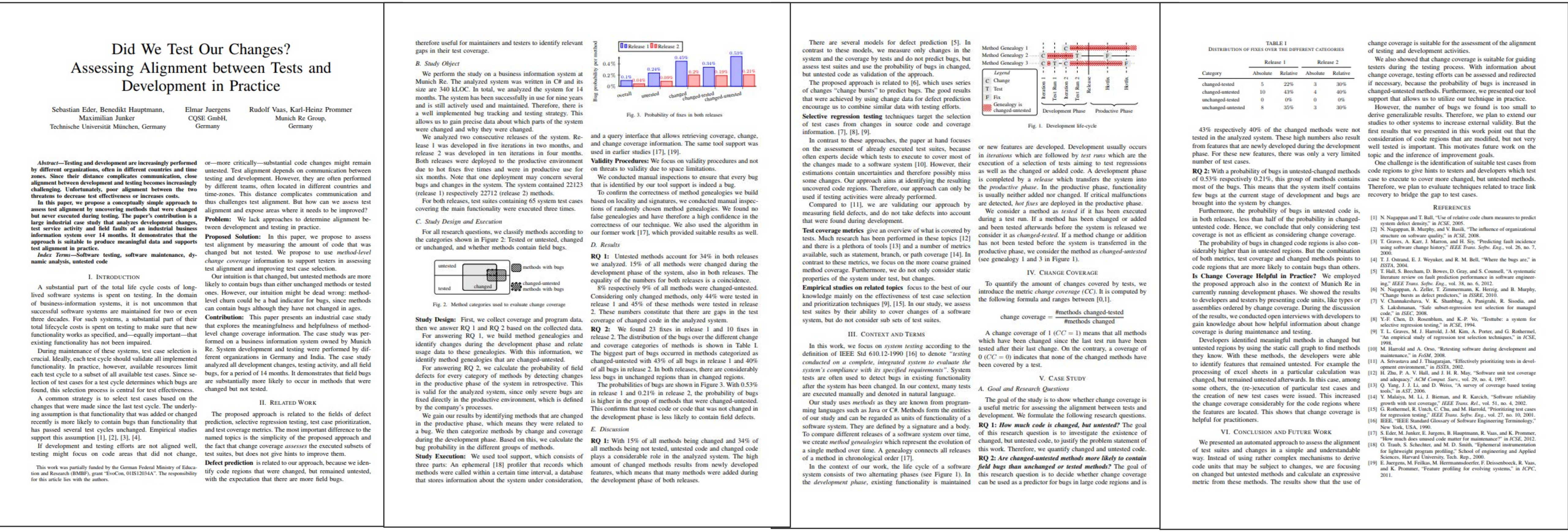


$$\%RemainingBugs = \%Tested * TestIneffectiveness + \%Testgap$$



$$\%RemainingBugs = \%Tested * TestIneffectiveness + \%Testgap$$





Did We Test Our Changes?
Assessing Alignment between Tests and Development in Practice

Sebastian Eder, Benedikt Hauptmann, Maximilian Junker, Technische Universität München, Germany
Elmar Juergens, CQSE GmbH, Germany
Rudolf Vaas, Karl-Heinz Prommer, Munich Re Group, Germany

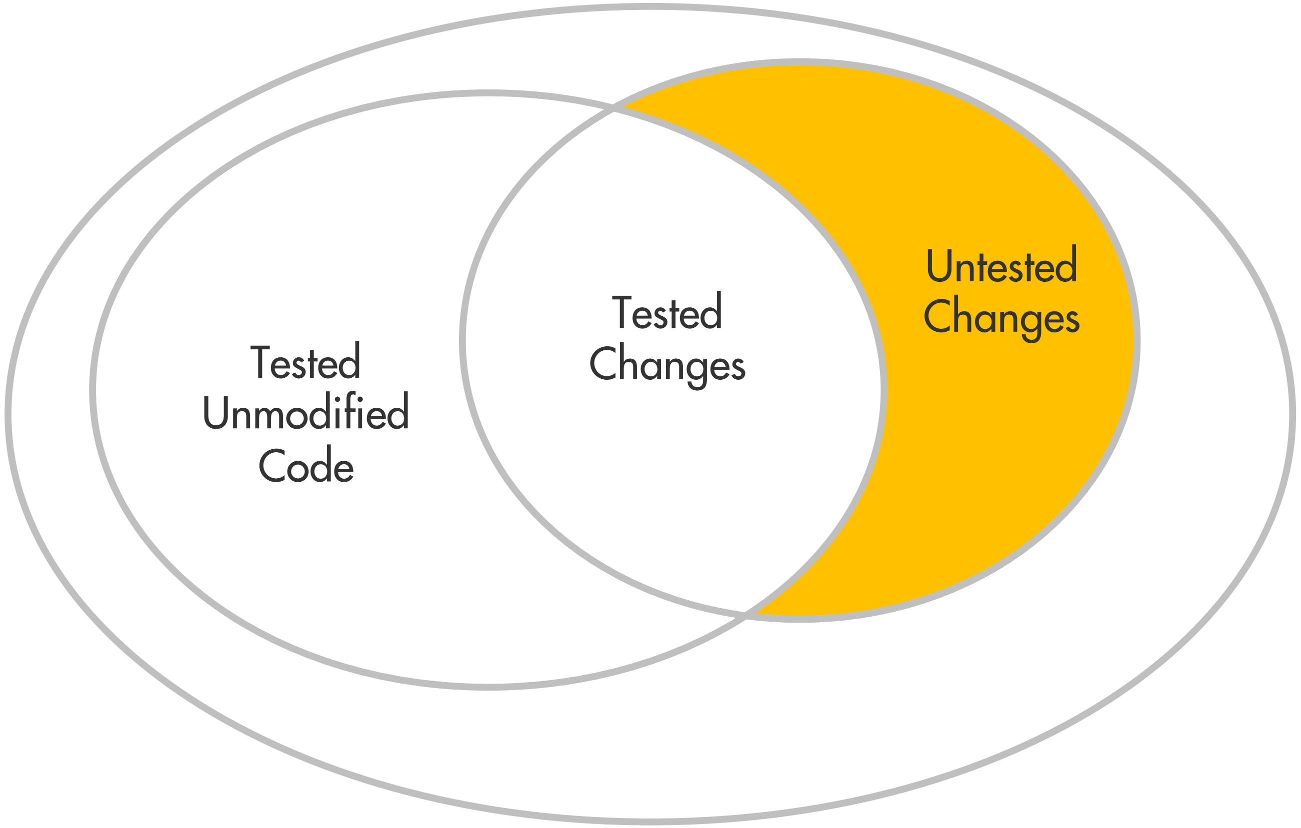
Paper published at Workshop on Automation of Software Test, 2013.

How Many Changes Remain Untested?

Study: C# System @ Munich Re

Release A:
15% Code new/modified,
>50% untested

Release B:
15% Code new/modified,
>60% untested

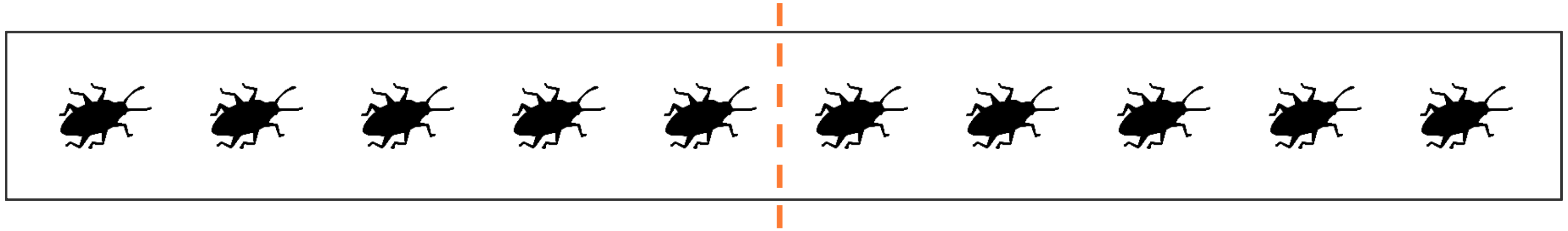


Probability of remaining field bugs is 5x higher in untested changes!

Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

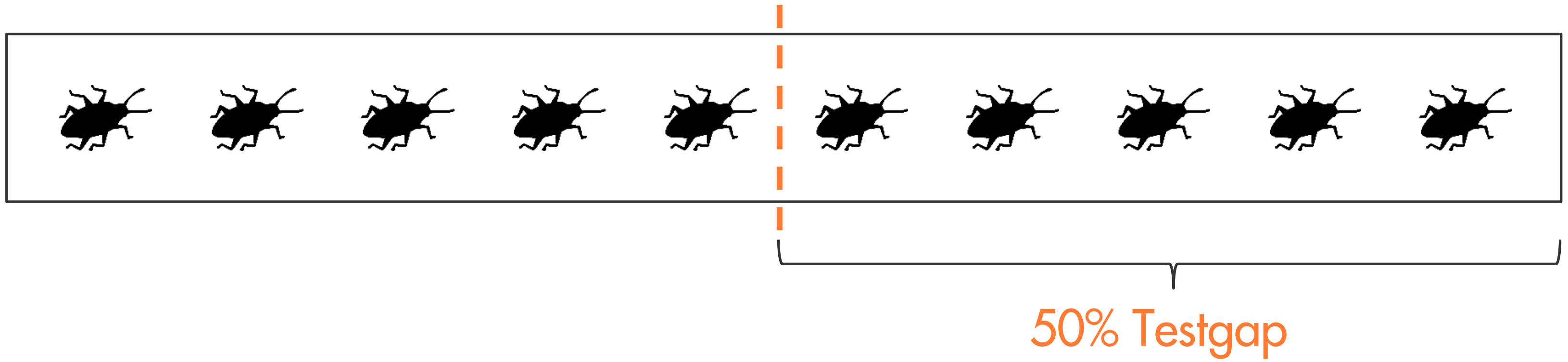
Without Test Gap Analysis

$$\%\text{RemainingBugs} = \%\text{Tested} * \text{TestIneffectiveness} + \%\text{Testgap}$$



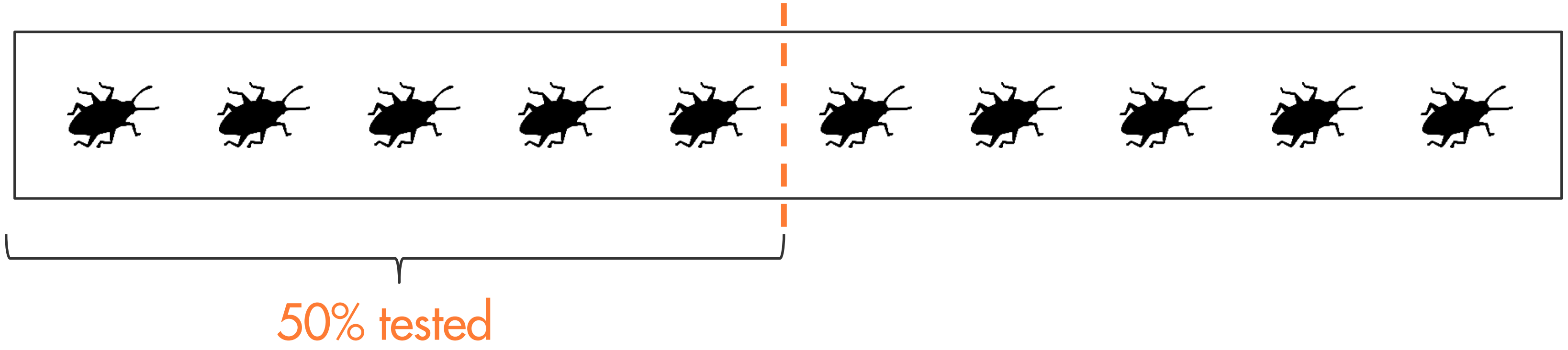
Without Test Gap Analysis

$$\%RemainingBugs = \%Tested * TestIneffectiveness + \%Testgap$$



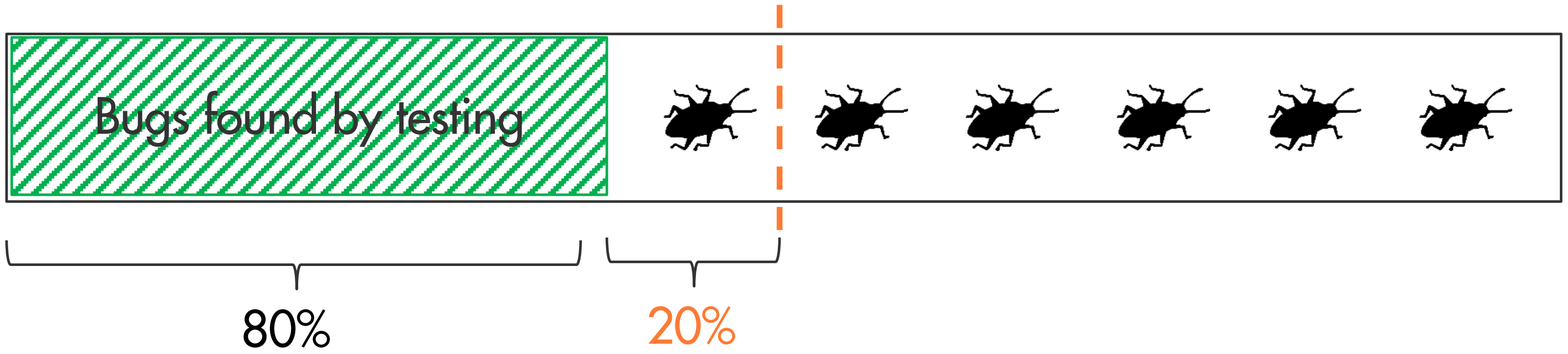
Without Test Gap Analysis

$$\%RemainingBugs = 50\% * TestIneffectiveness + \%Testgap$$



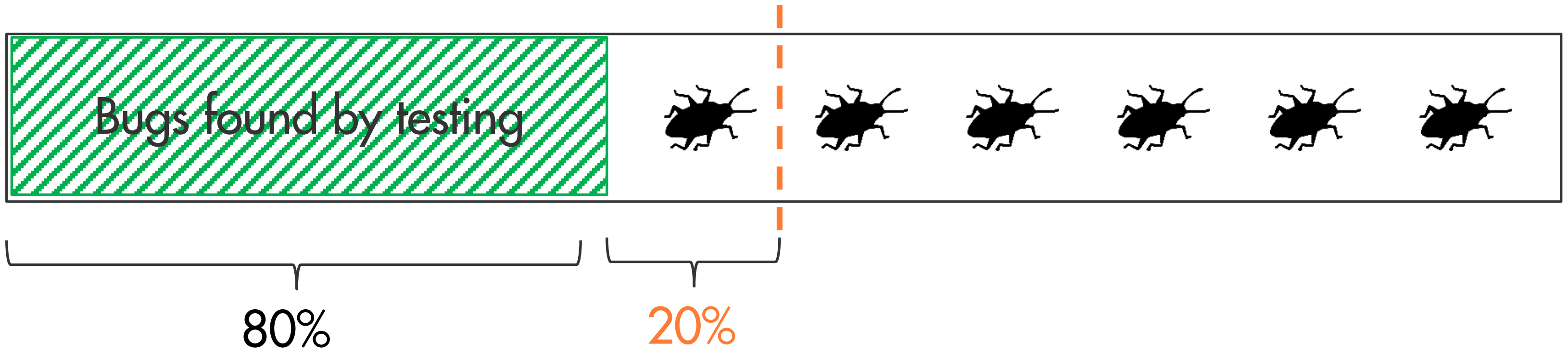
Without Test Gap Analysis

$$\%\mathrm{RemainingBugs} = 50\% * \mathrm{TestIneffectiveness} + 50\%$$



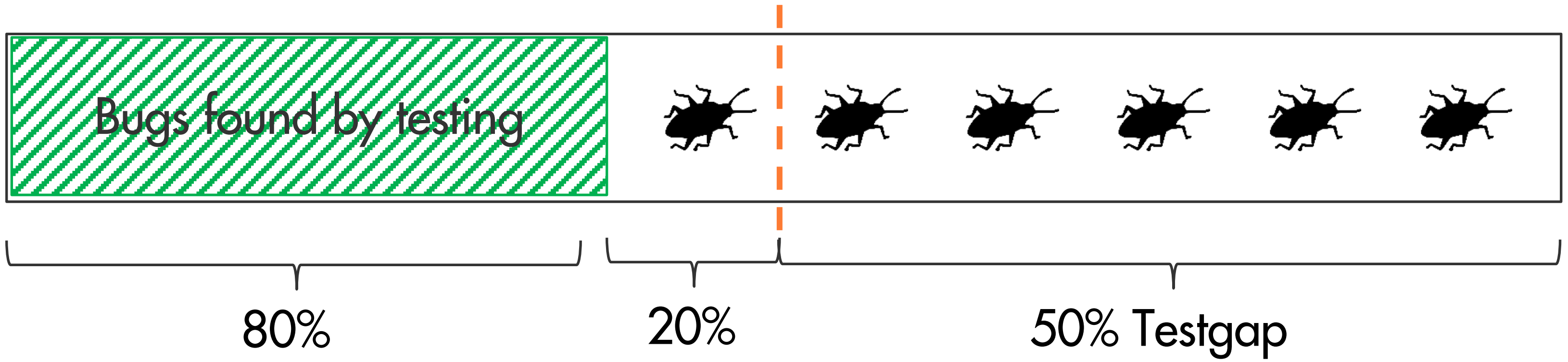
Without Test Gap Analysis

$$\%\text{RemainingBugs} = 50\% * 20\% + 50\%$$



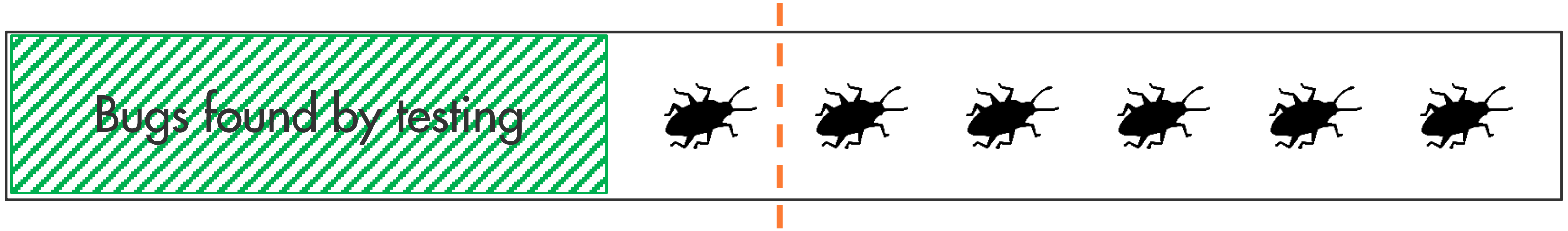
Without Test Gap Analysis

$$\%\text{RemainingBugs} = 10\% + 50\%$$



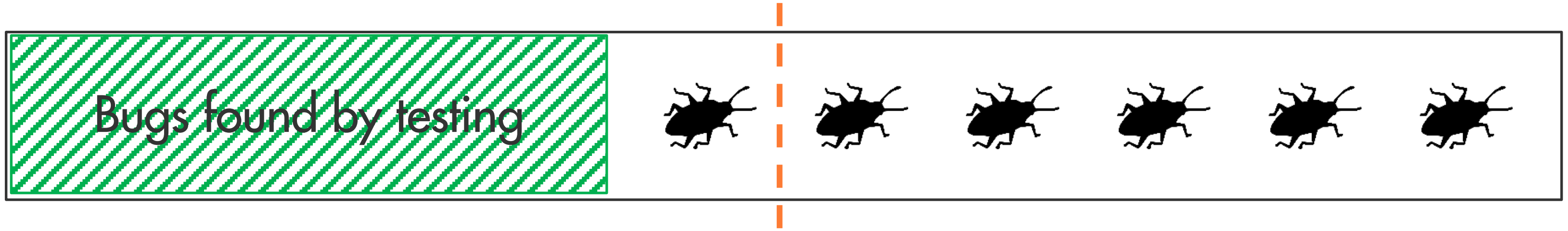
Without Test Gap Analysis

$$\%\mathrm{RemainingBugs} = 60\%$$



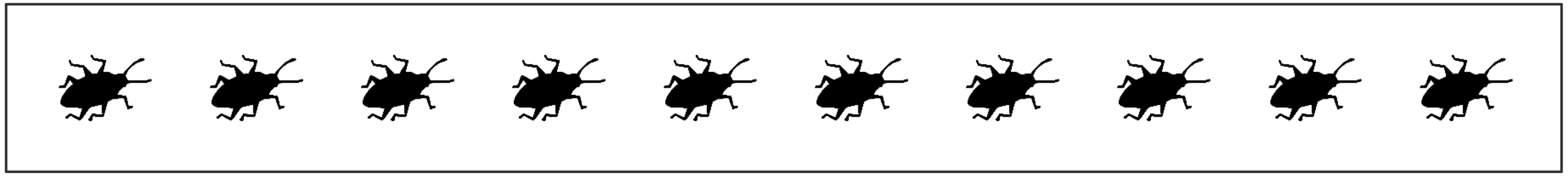
Without Test Gap Analysis

$$\%RemainingBugs = 60\%$$



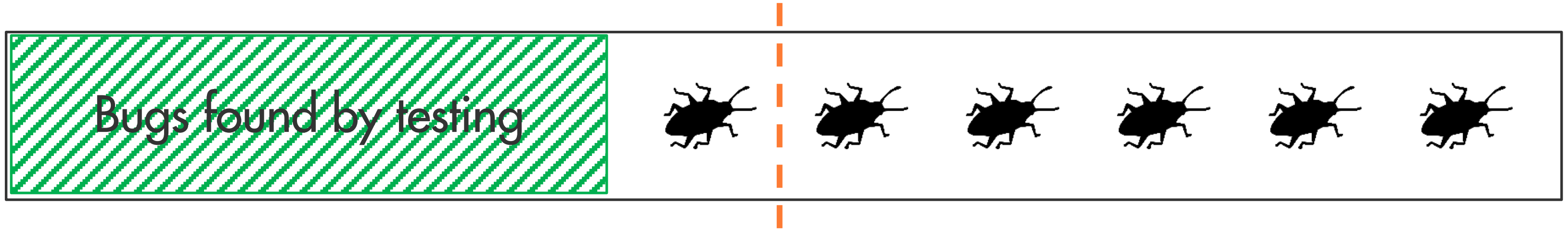
With Test Gap Analysis

$$\%RemainingBugs = \%Tested * TestIneffectiveness + \%Testgap$$



Without Test Gap Analysis

$$\%\text{RemainingBugs} = 60\%$$

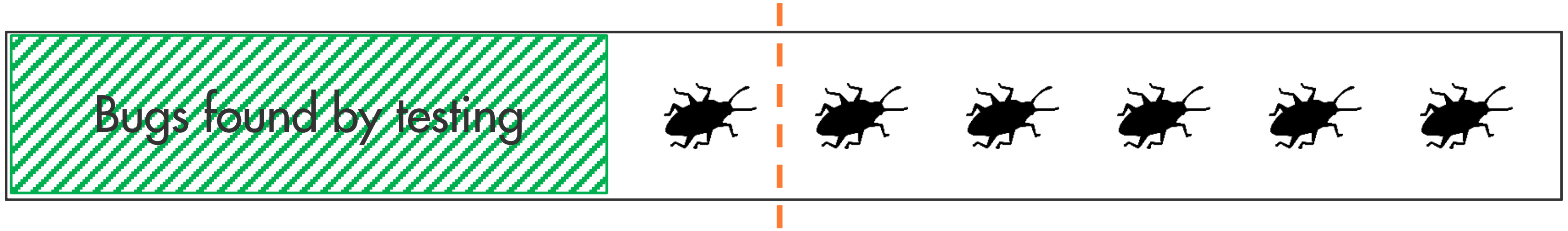


With Test Gap Analysis

$$\%\text{RemainingBugs} = 90\% * 20\% + 10\%$$

Without Test Gap Analysis

$$\%\text{RemainingBugs} = 60\%$$

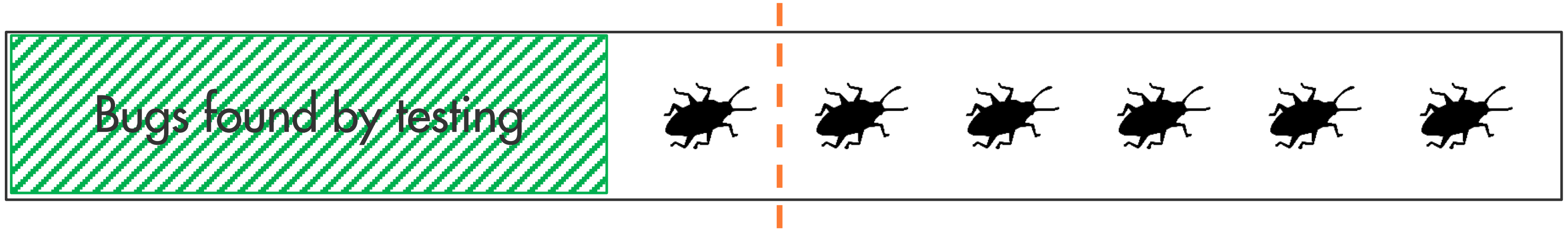


With Test Gap Analysis

$$\%\text{RemainingBugs} = 18\% + 10\%$$

Without Test Gap Analysis

$$\%\text{RemainingBugs} = 60\%$$

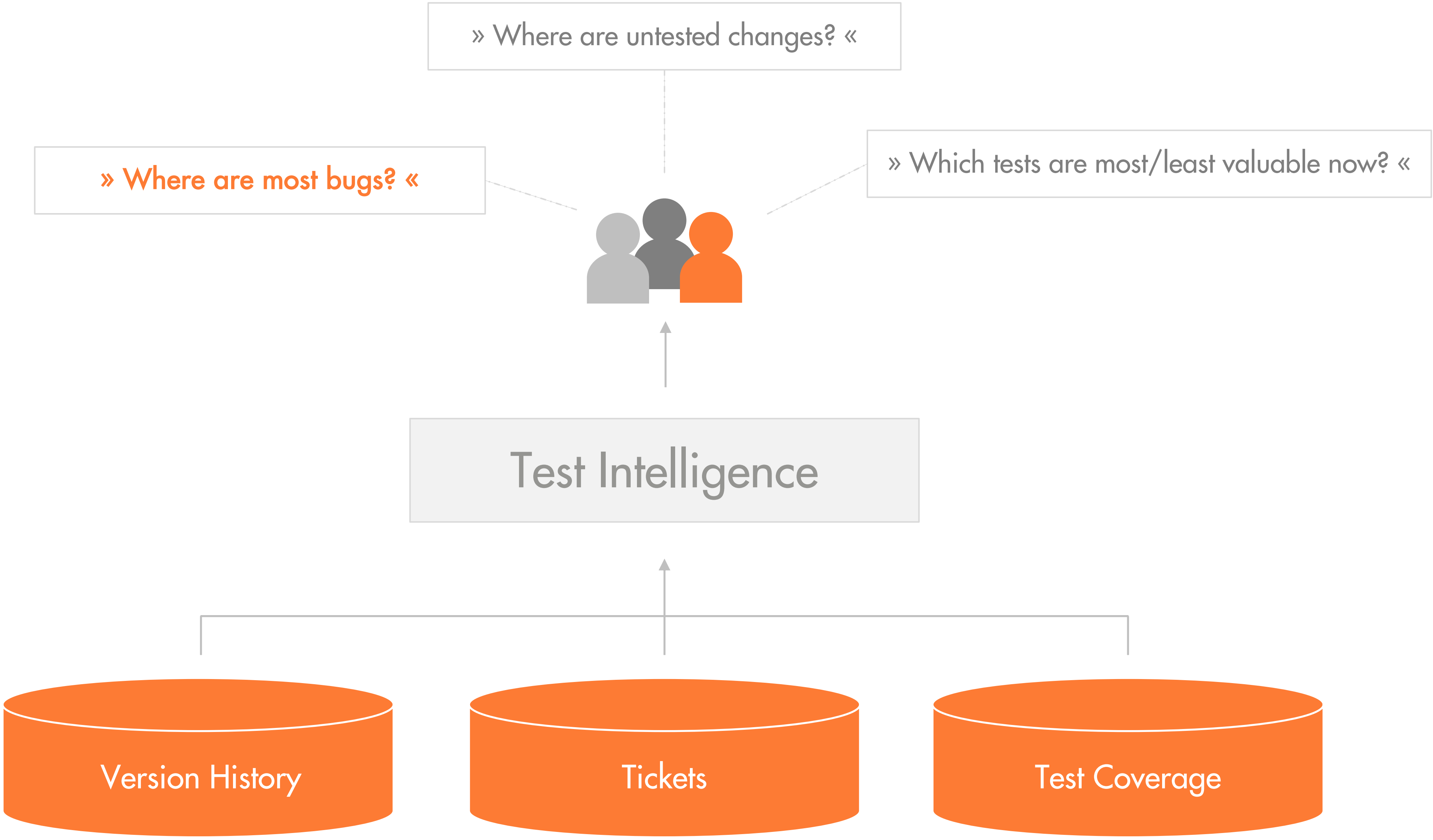


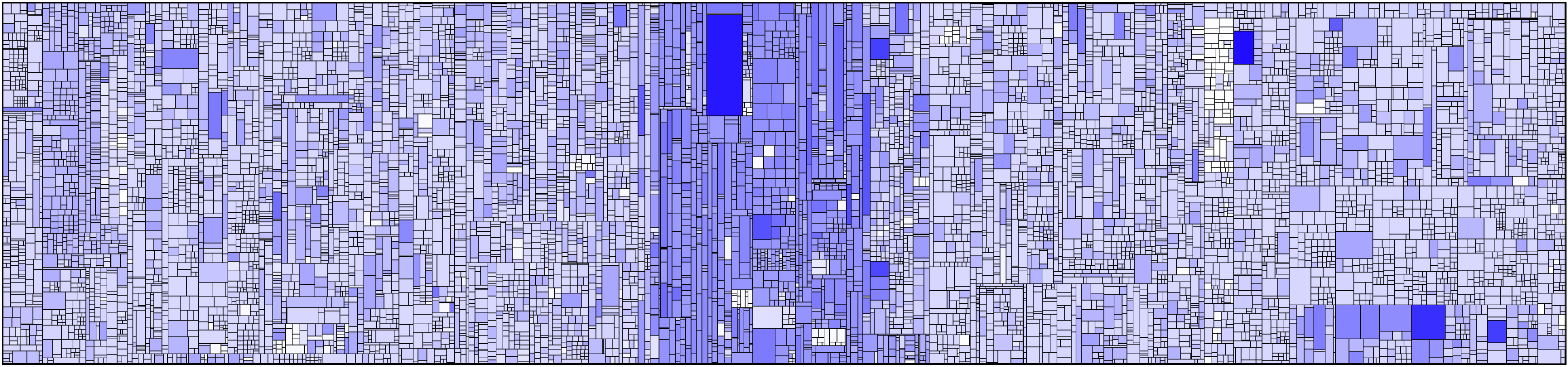
With Test Gap Analysis

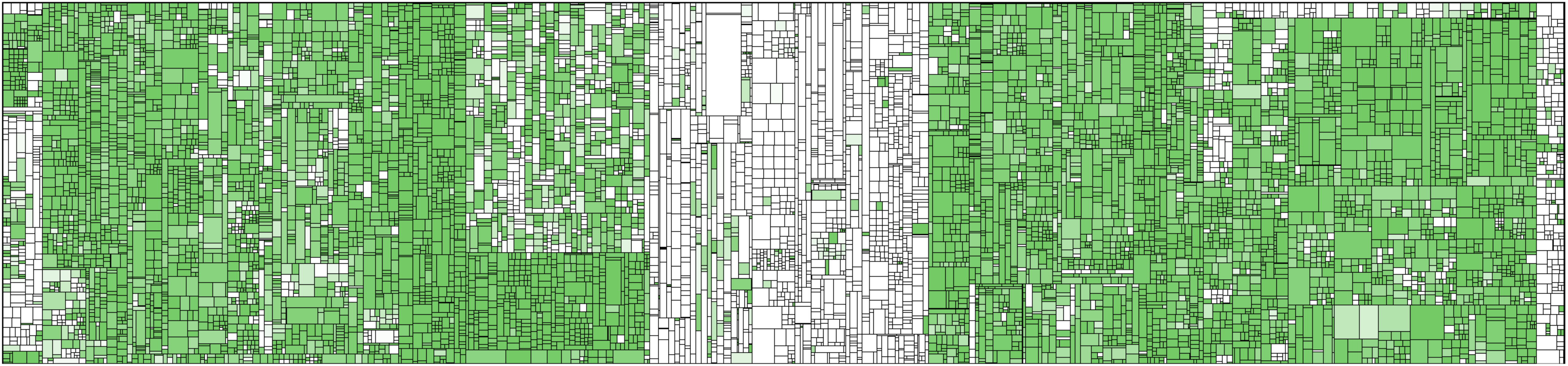
$$\%\text{RemainingBugs} = 28\%$$

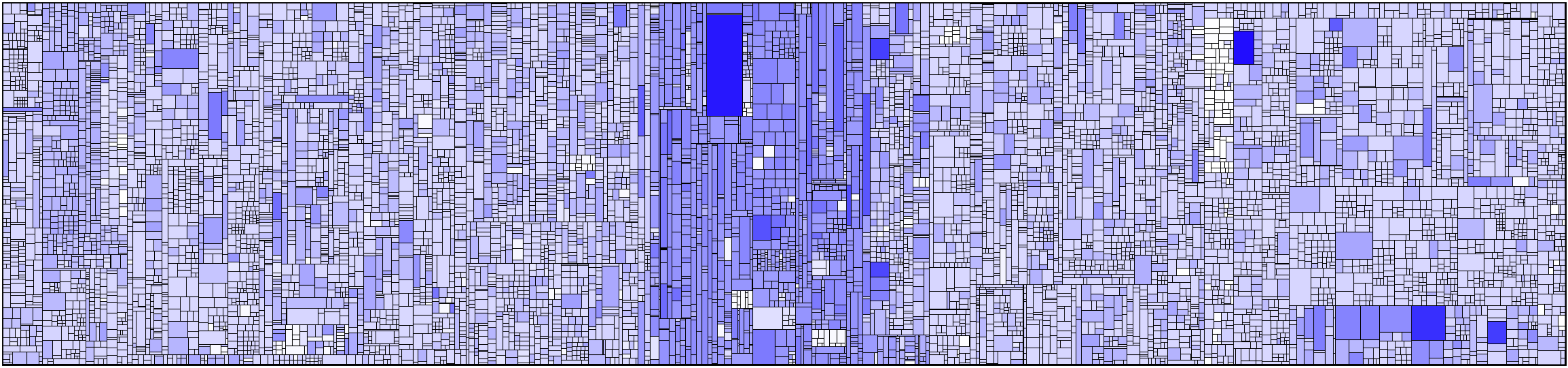
Remaining Field Bugs = **50%**

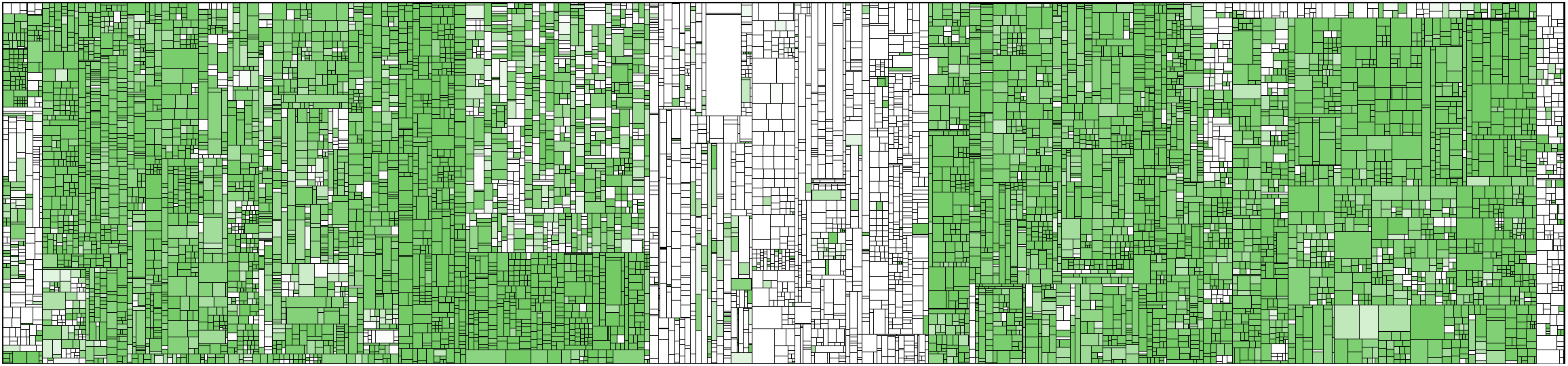
Test Gap Analysis reduced field bugs in applications of Munich Re by ½


» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valuable now? «
Test Intelligence
Version History
Tickets
Test Coverage

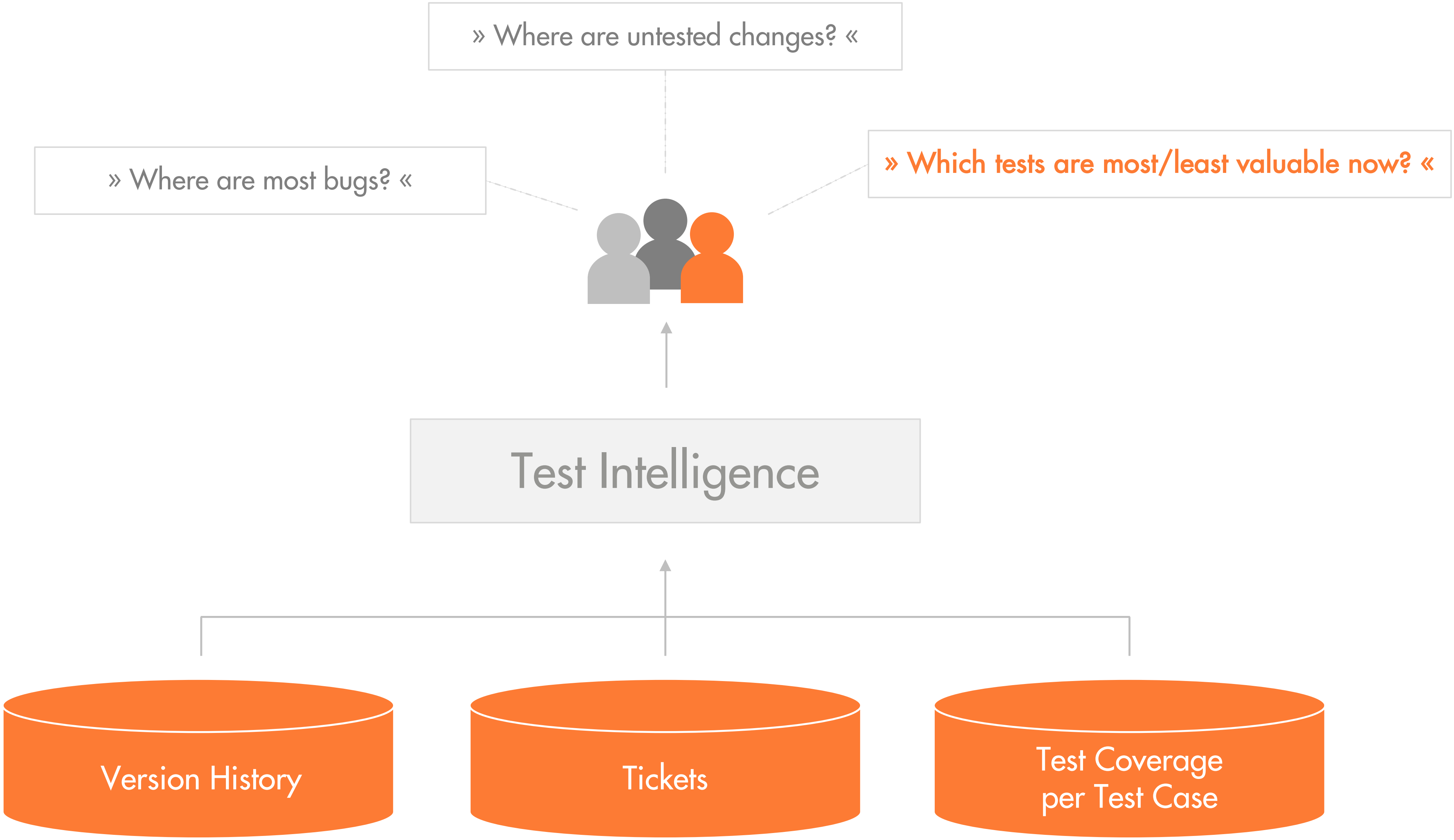


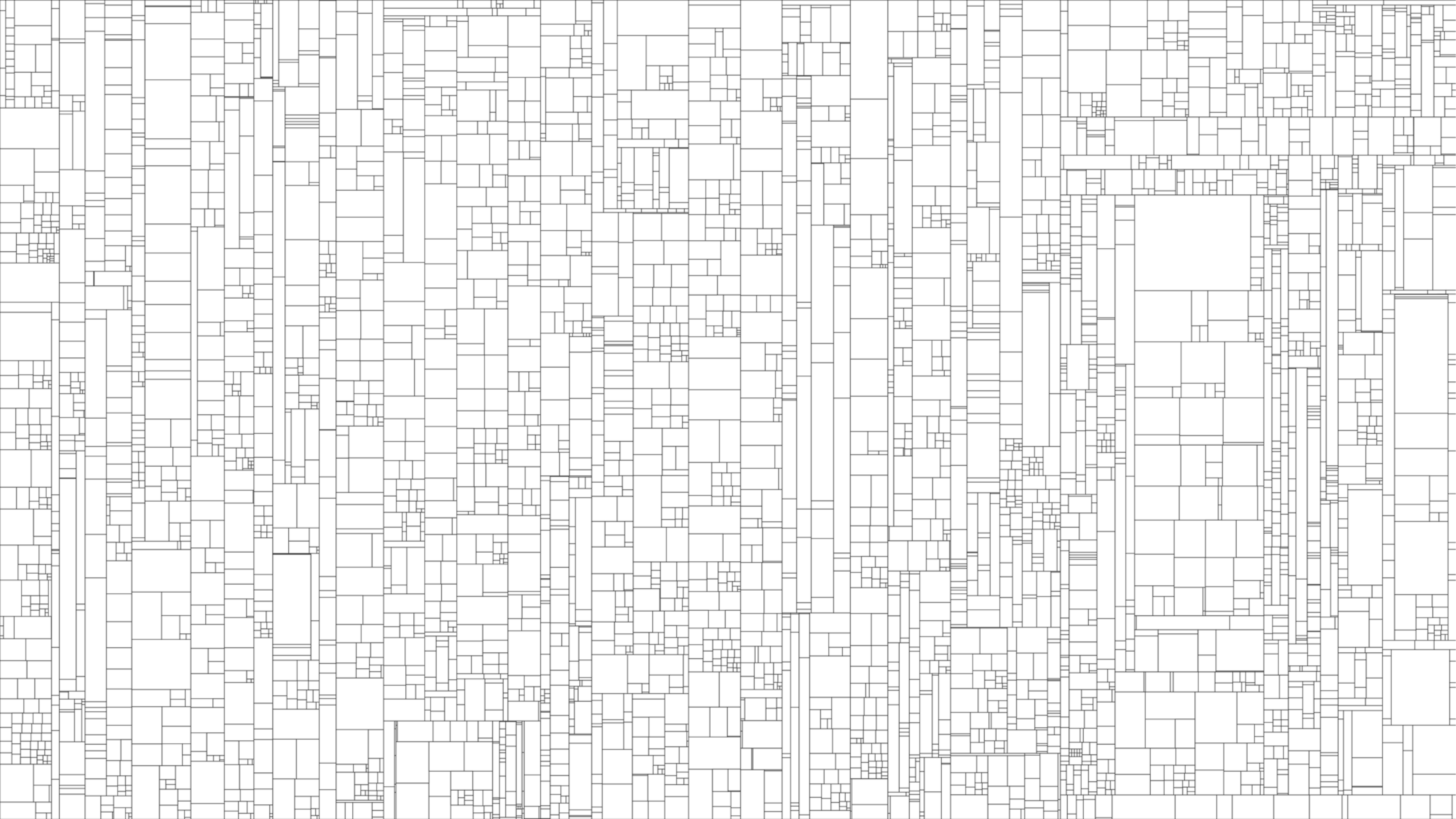


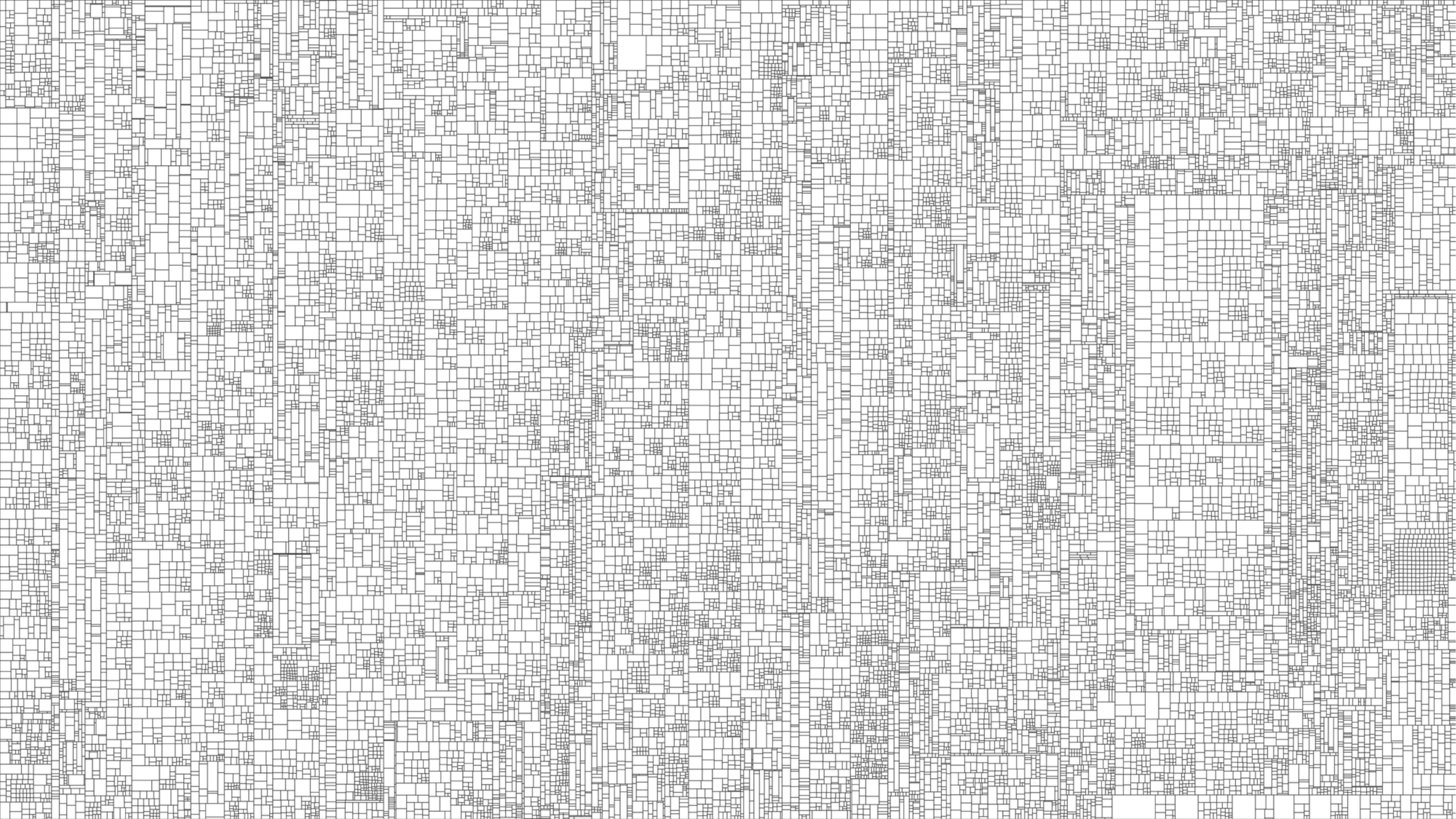




https://www.technica-engineering.de/produkte/bts-body-electronic-test-system/


» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valuable now? «
Test Intelligence
Version History
Tickets
Test Coverage
per Test Case





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

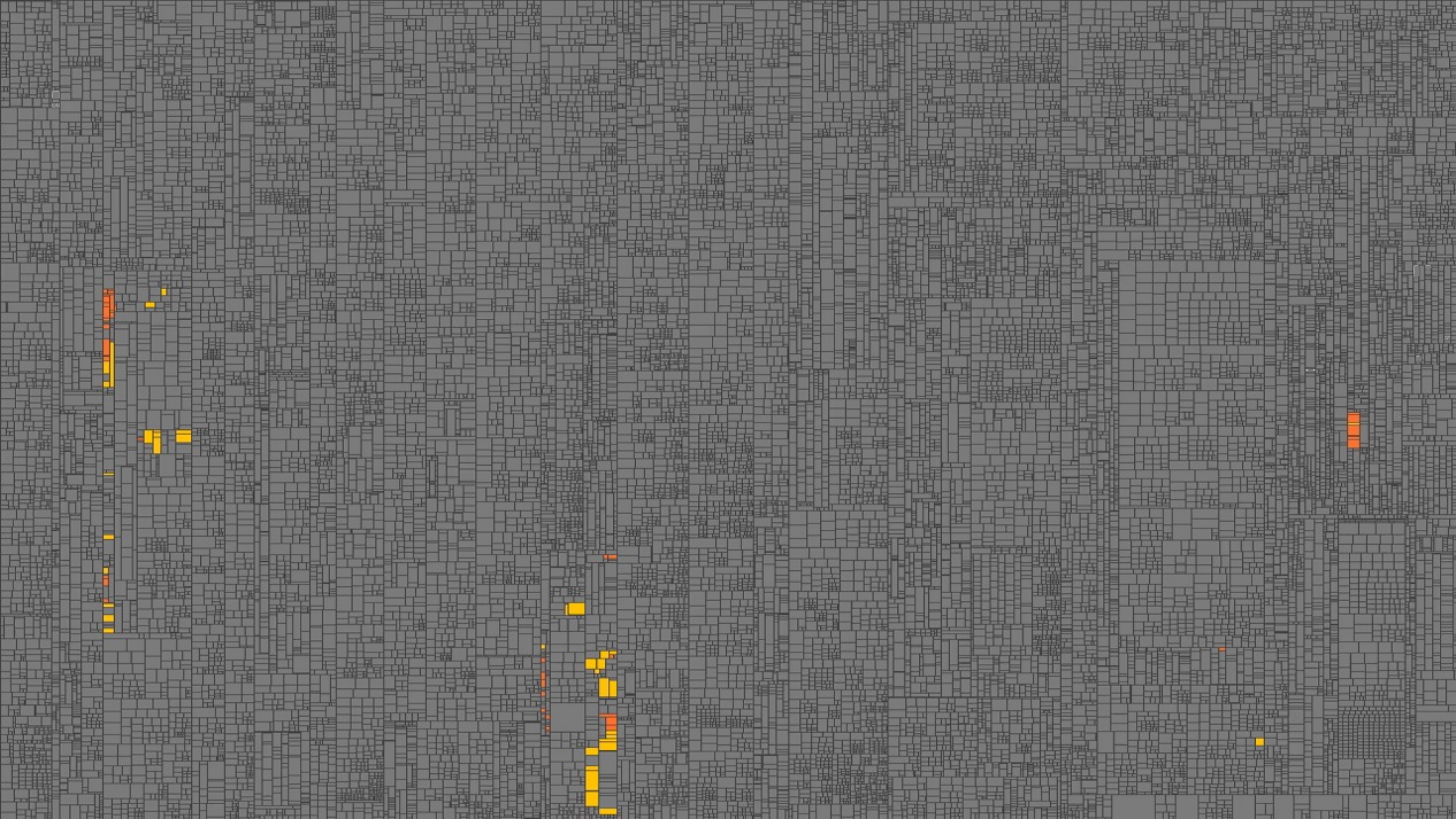
Estimated:	Not Specified
Remaining:	0m
Logged:	4d 1h 57m

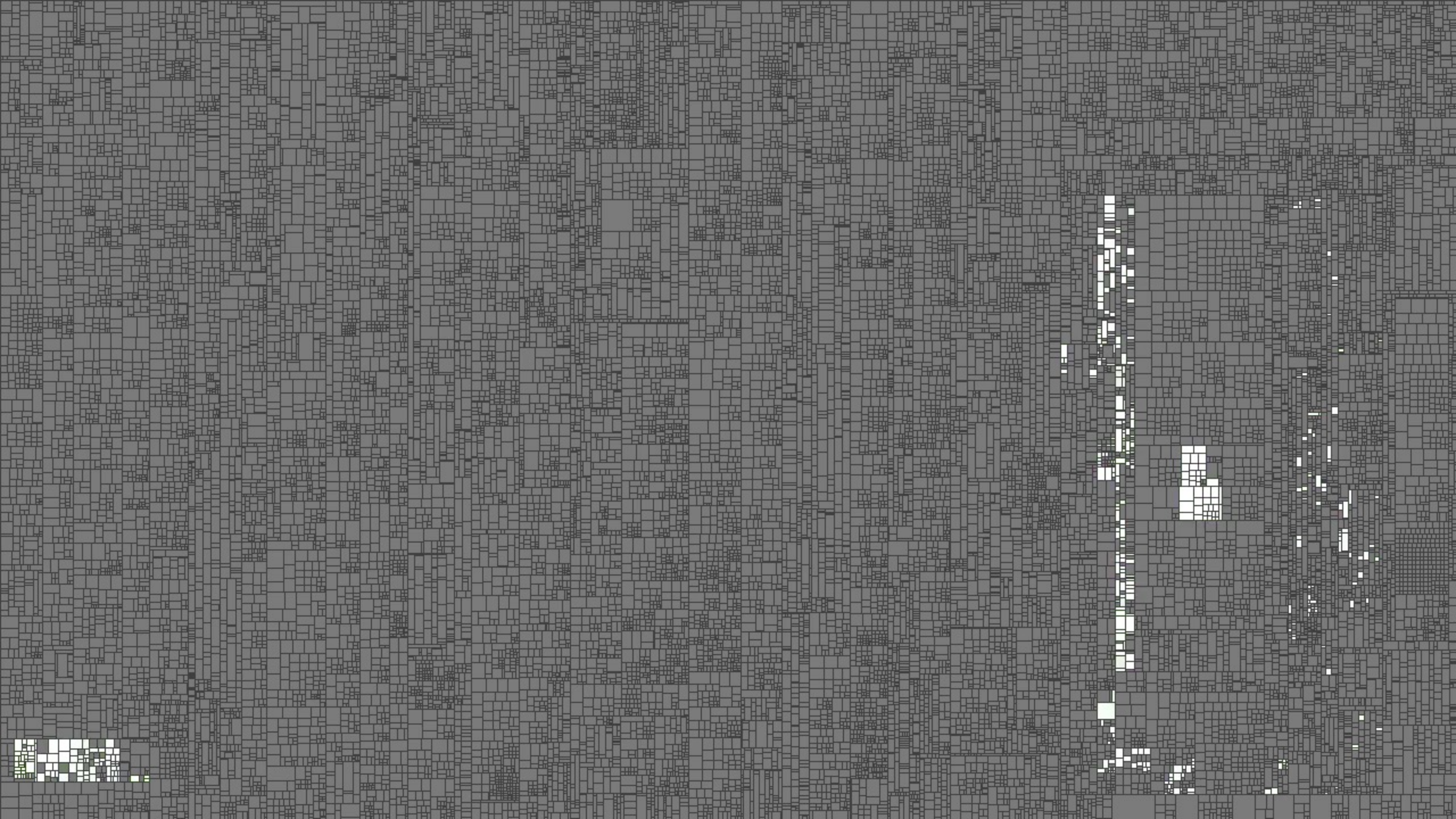
Agile

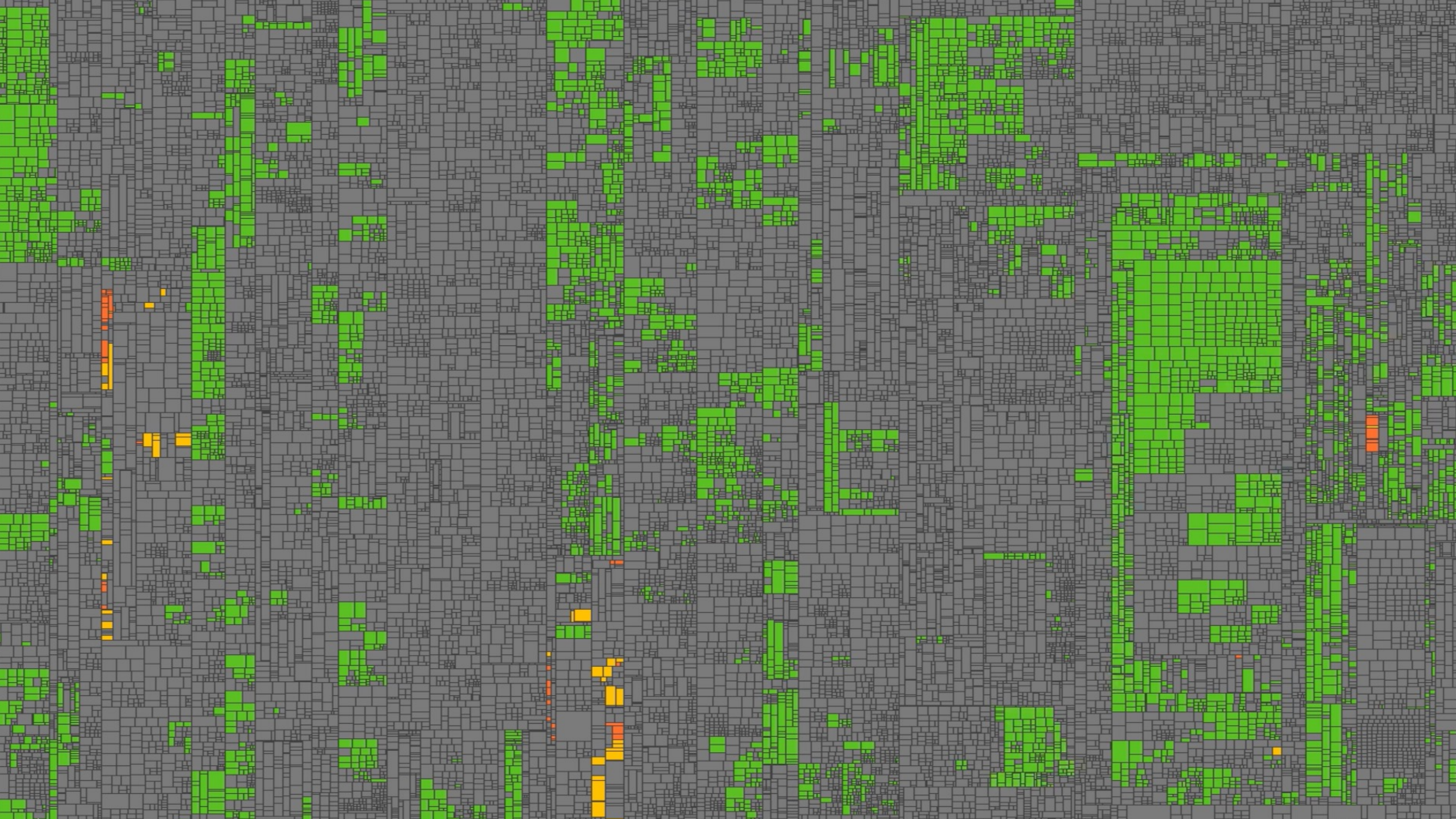
View on Board

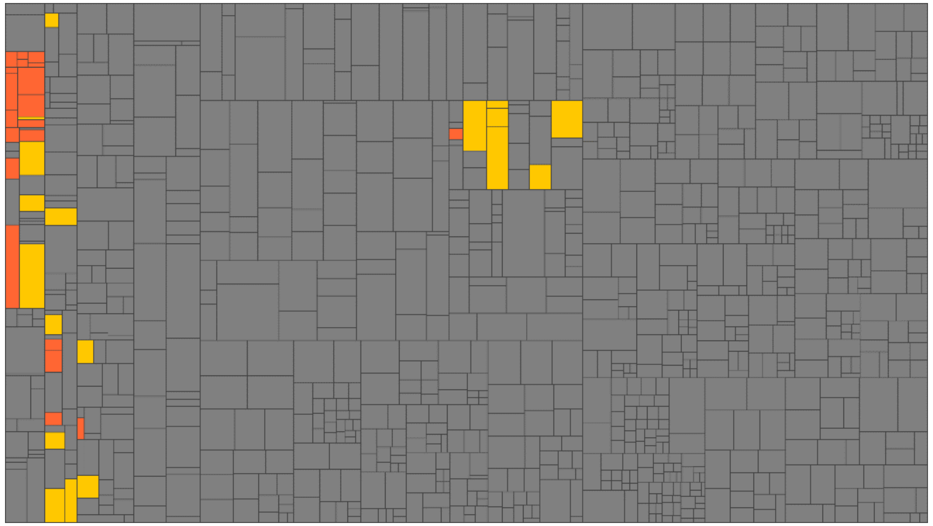
Gitlab CI

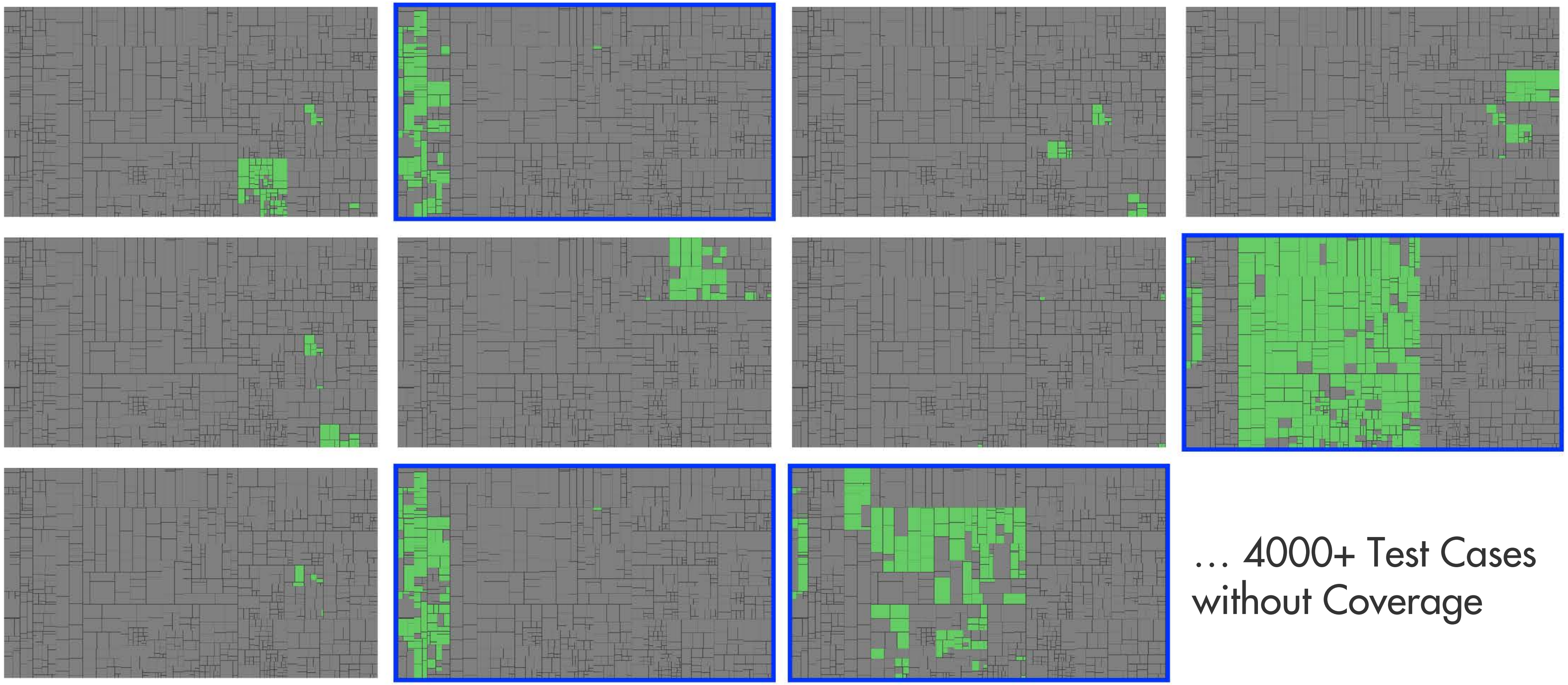
Projects More



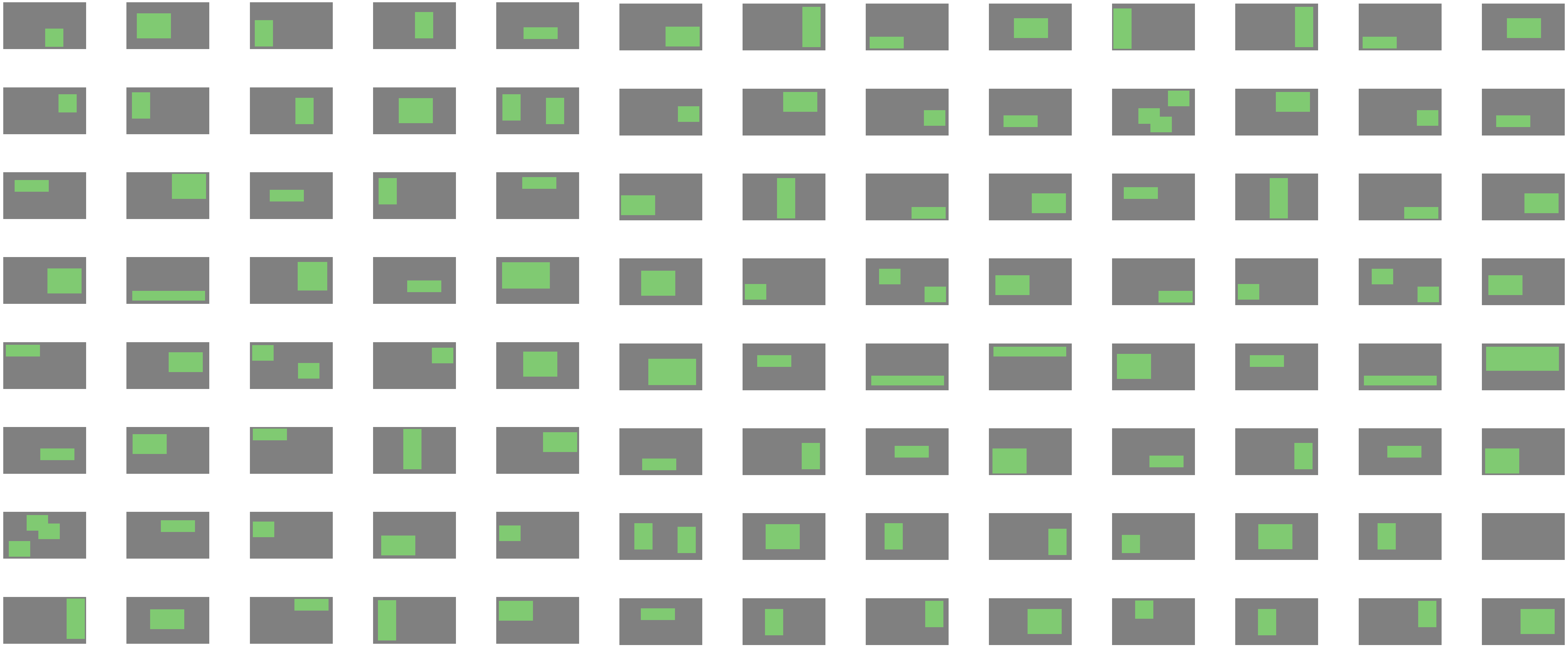




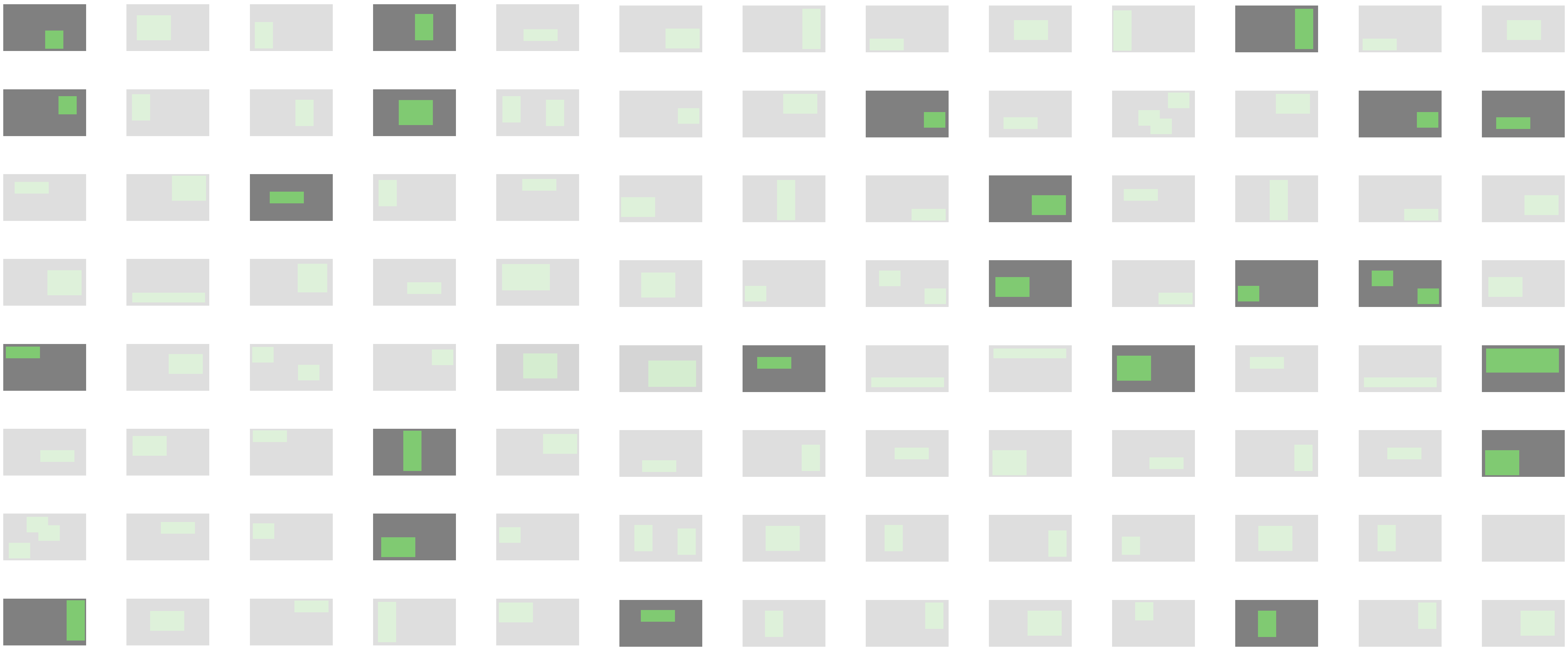




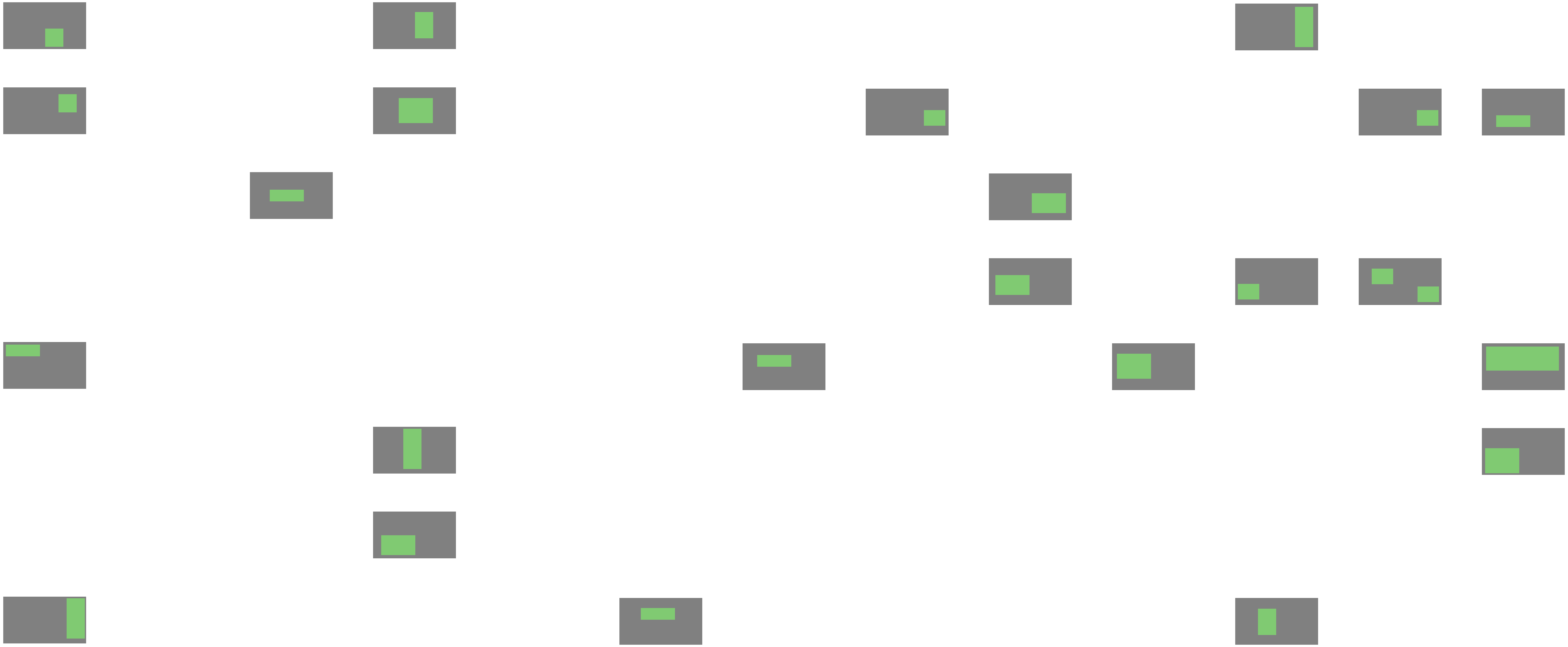
… 4000+ Test Cases without Coverage



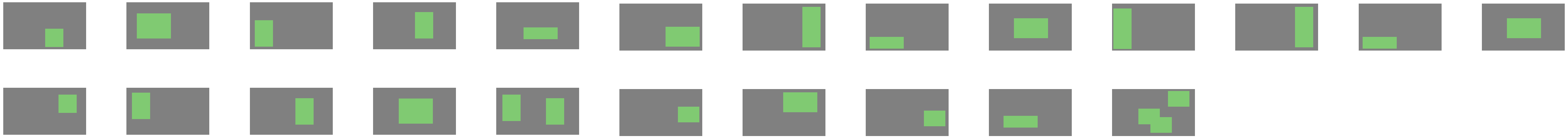
Step 1: Selection of Impacted Tests



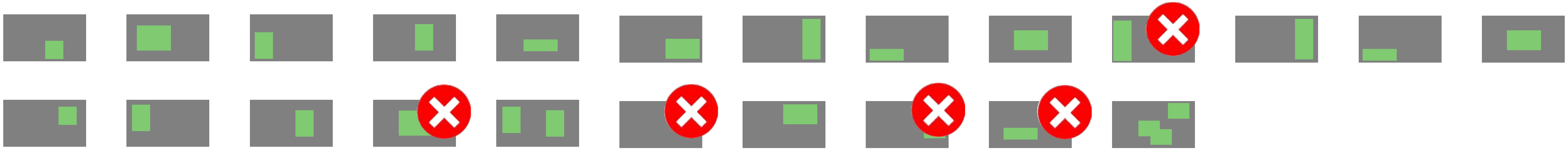
Step 1: Selection of Impacted Tests



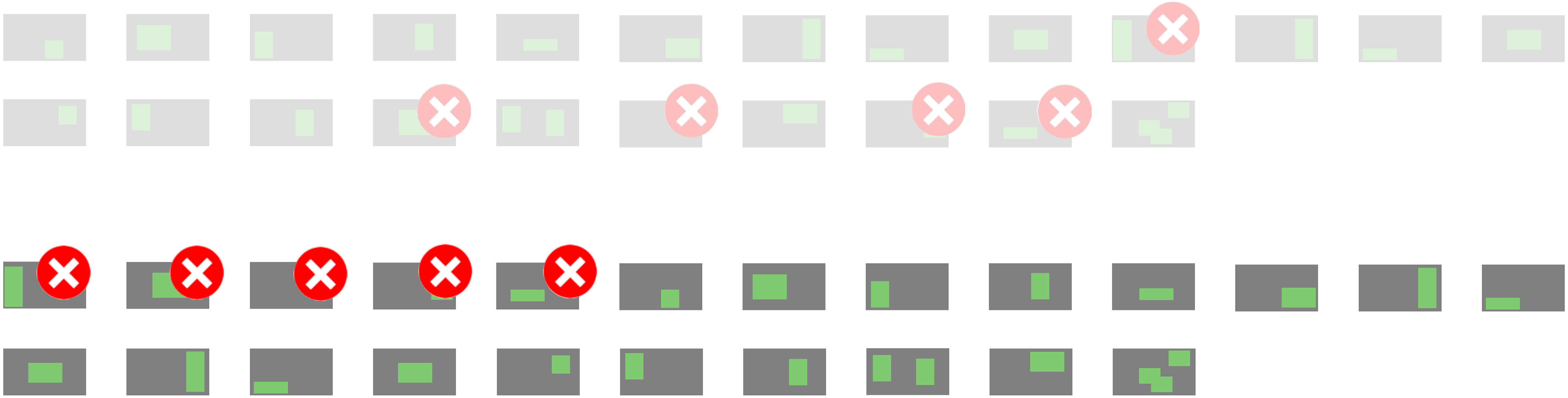
Step 1: Selection of Impacted Tests



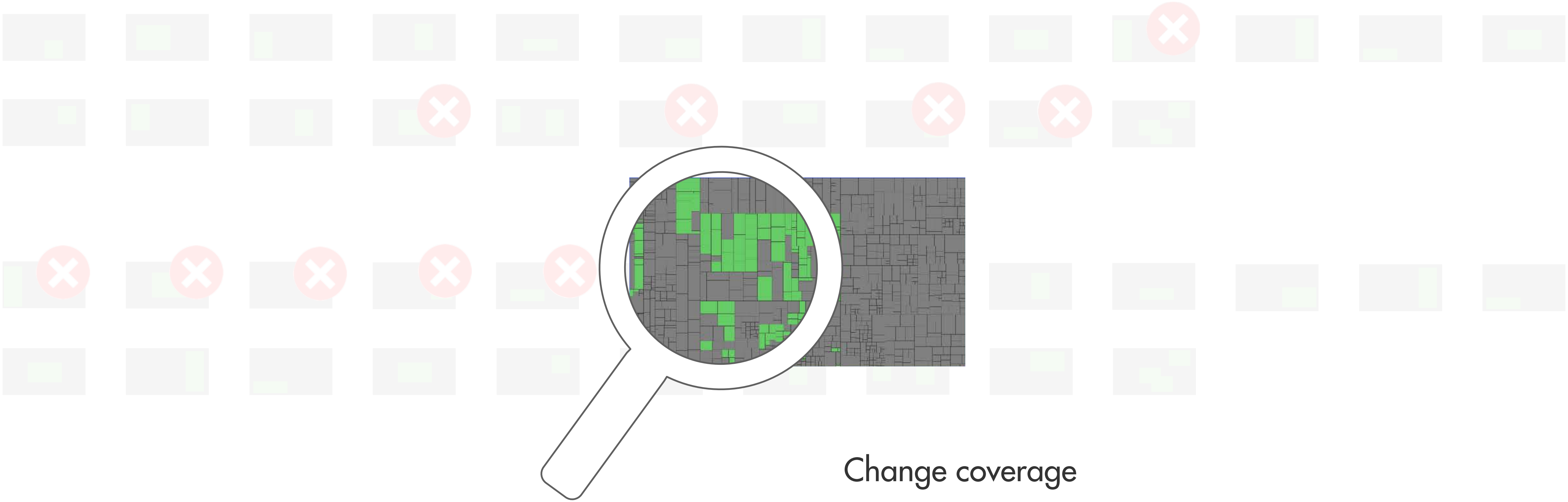
Step 2: Prioritization of Impacted Tests



Step 2: Prioritization of Impacted Tests



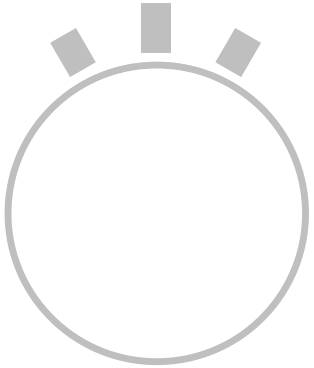
Step 2: Prioritization of Impacted Tests

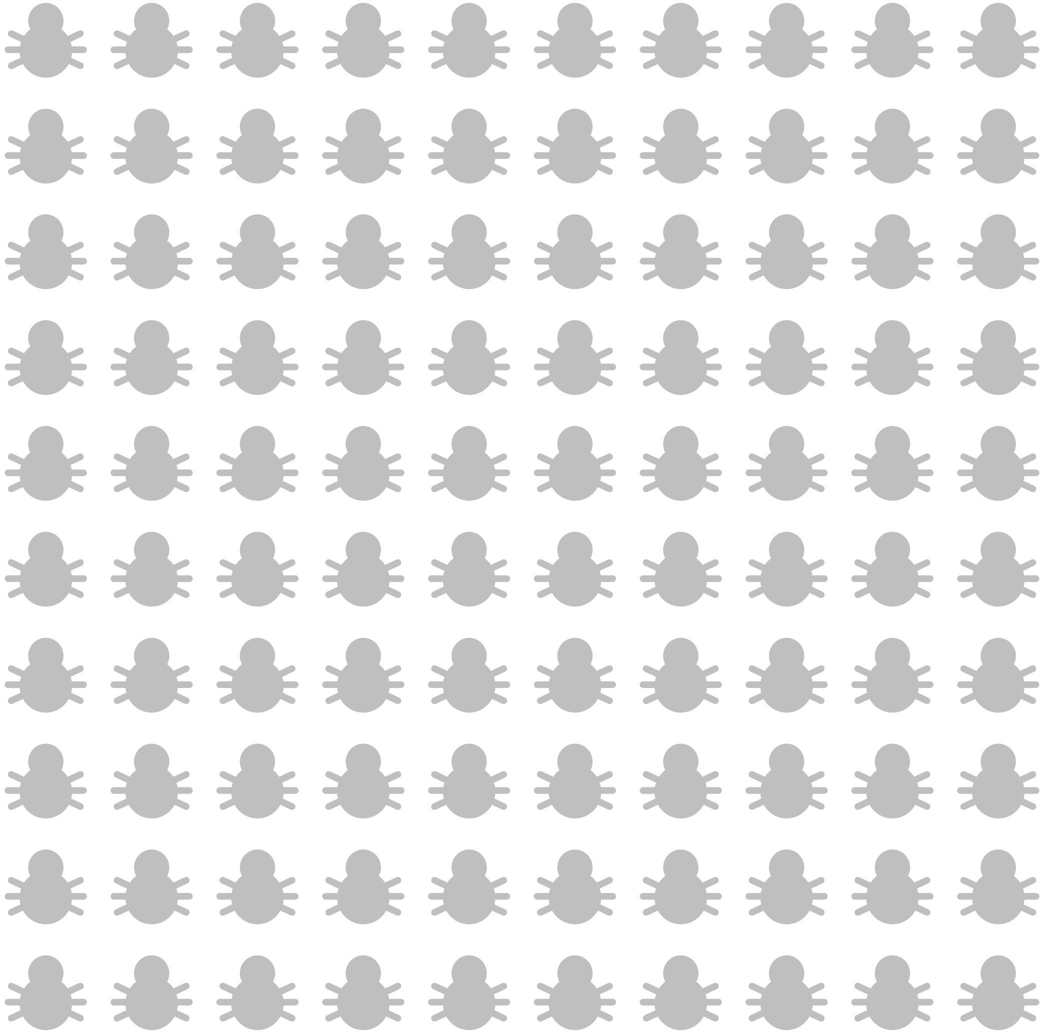


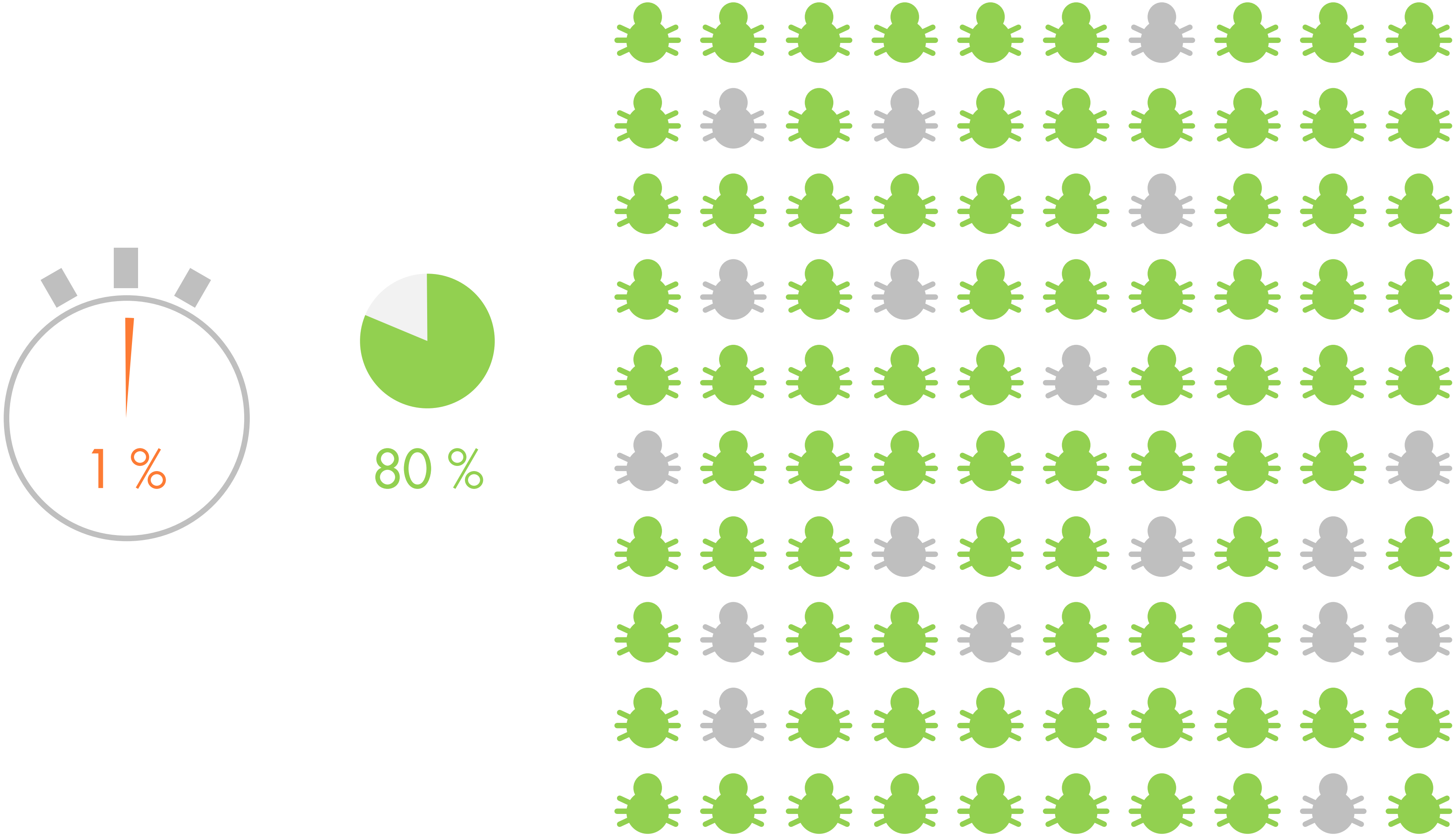
Change coverage

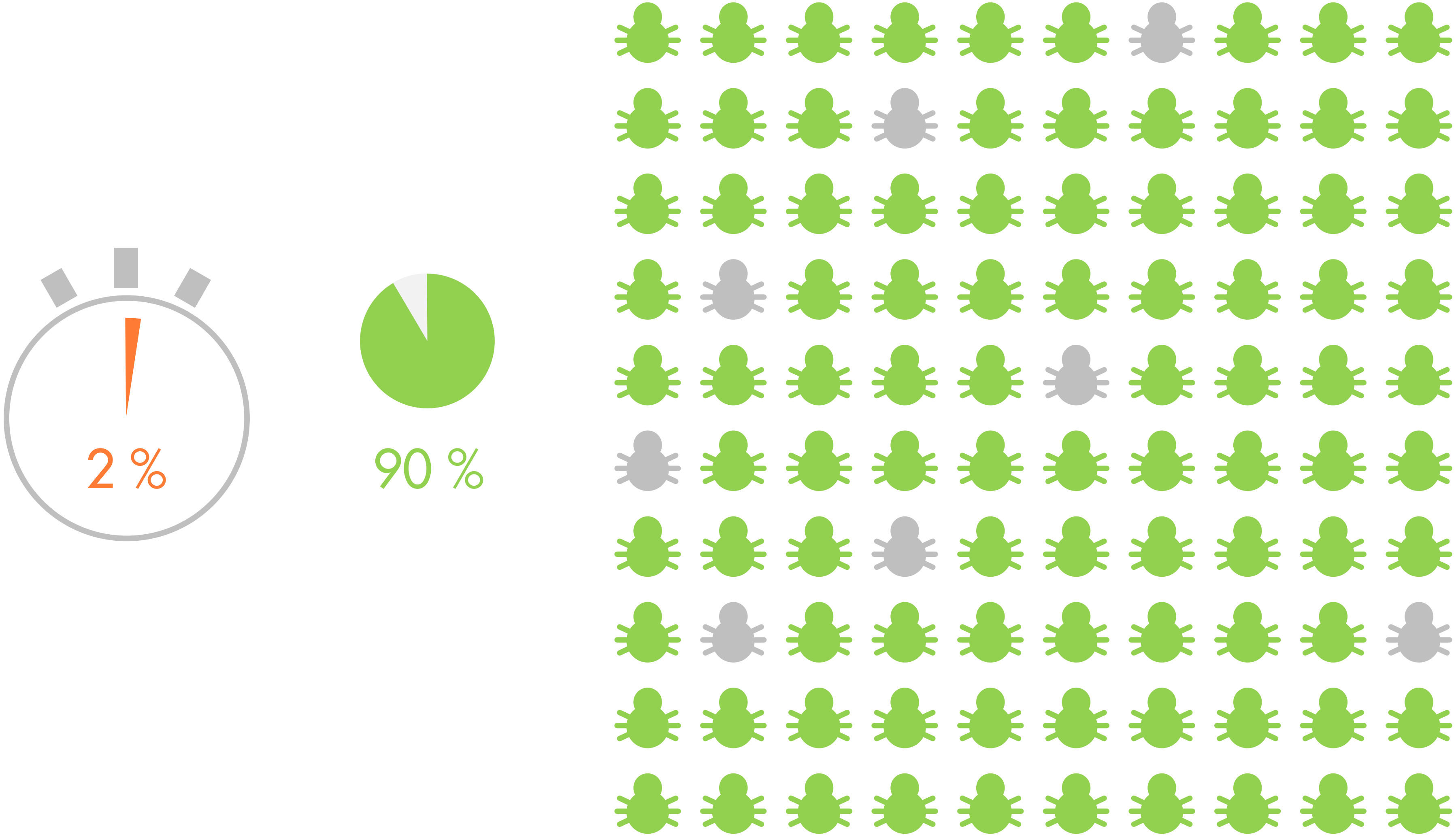
Execution time

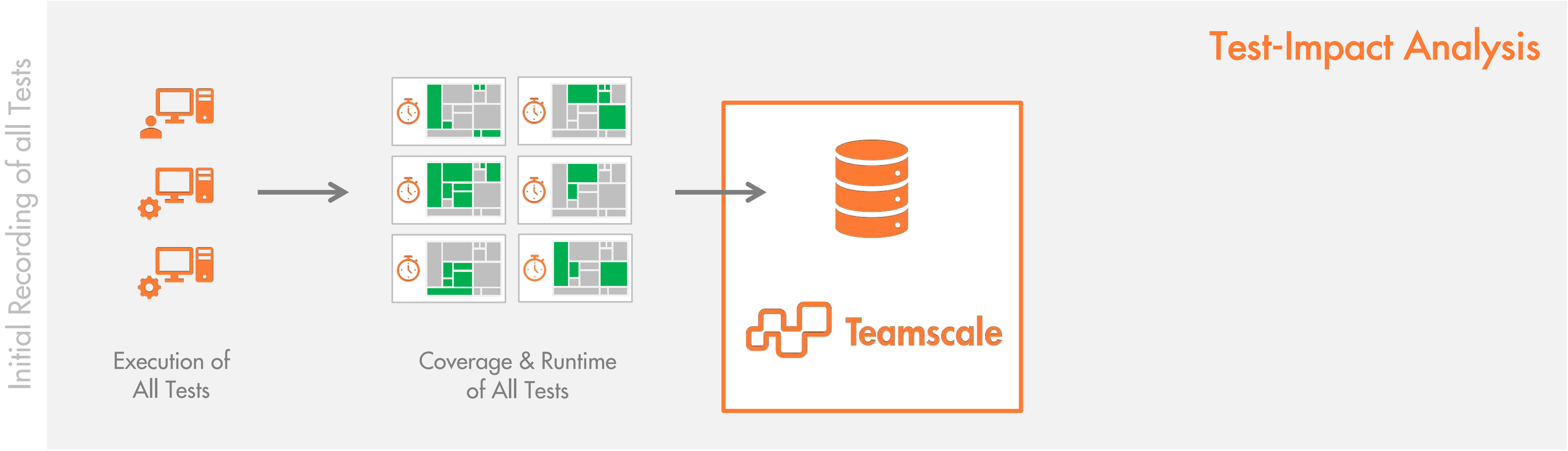
Step 2: Prioritization of Impacted Tests



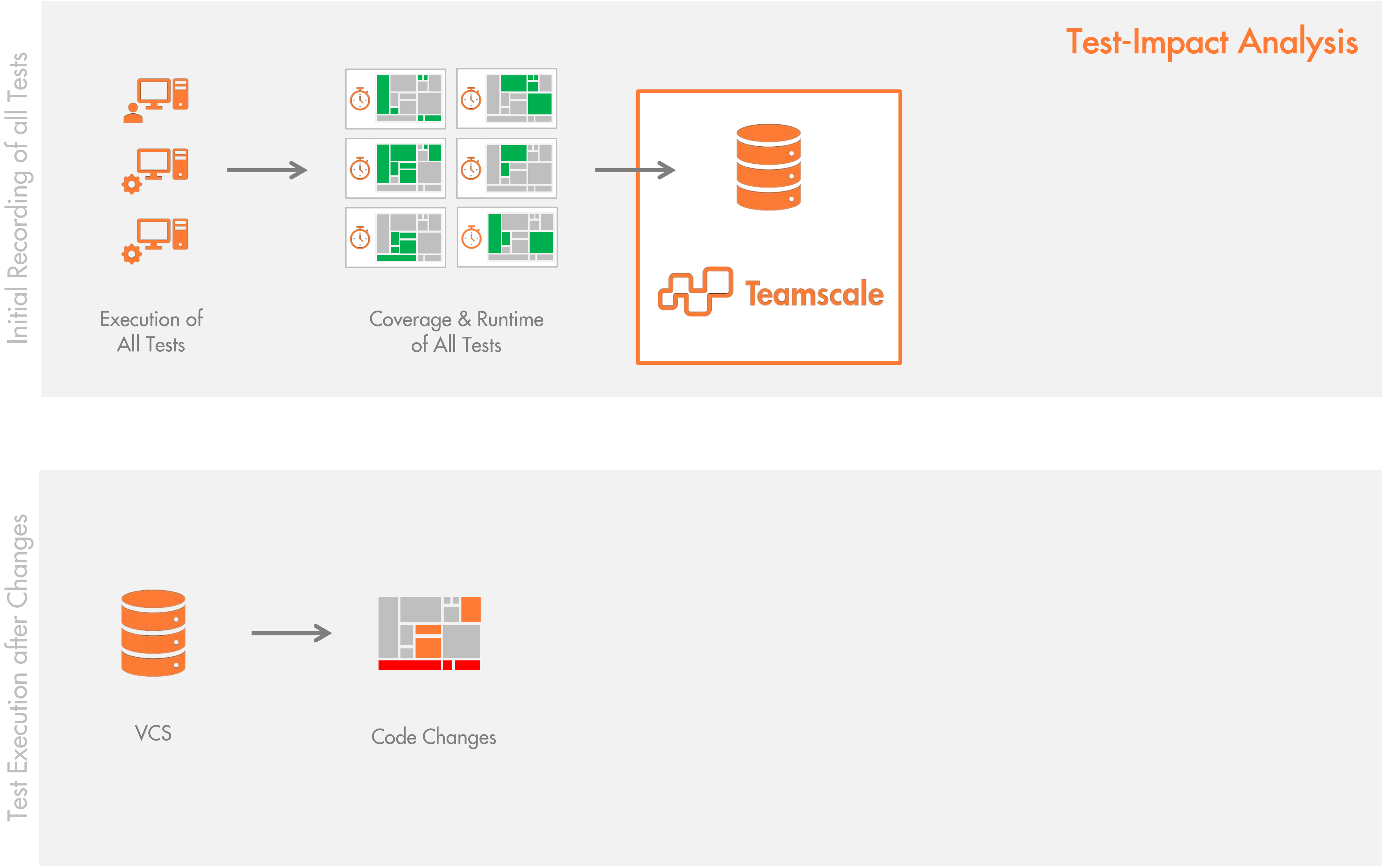


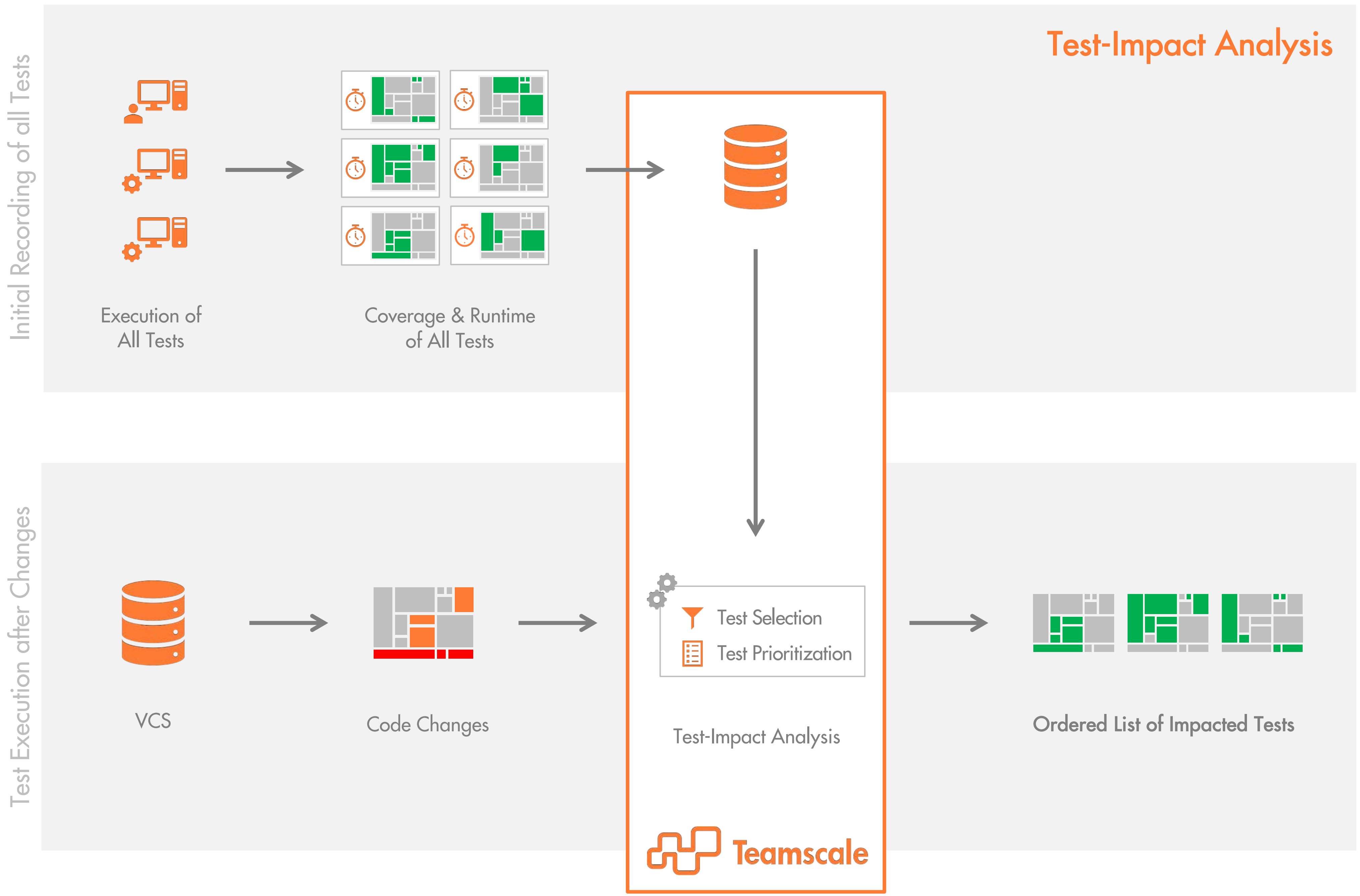

1 %
80 %


2 %
90 %

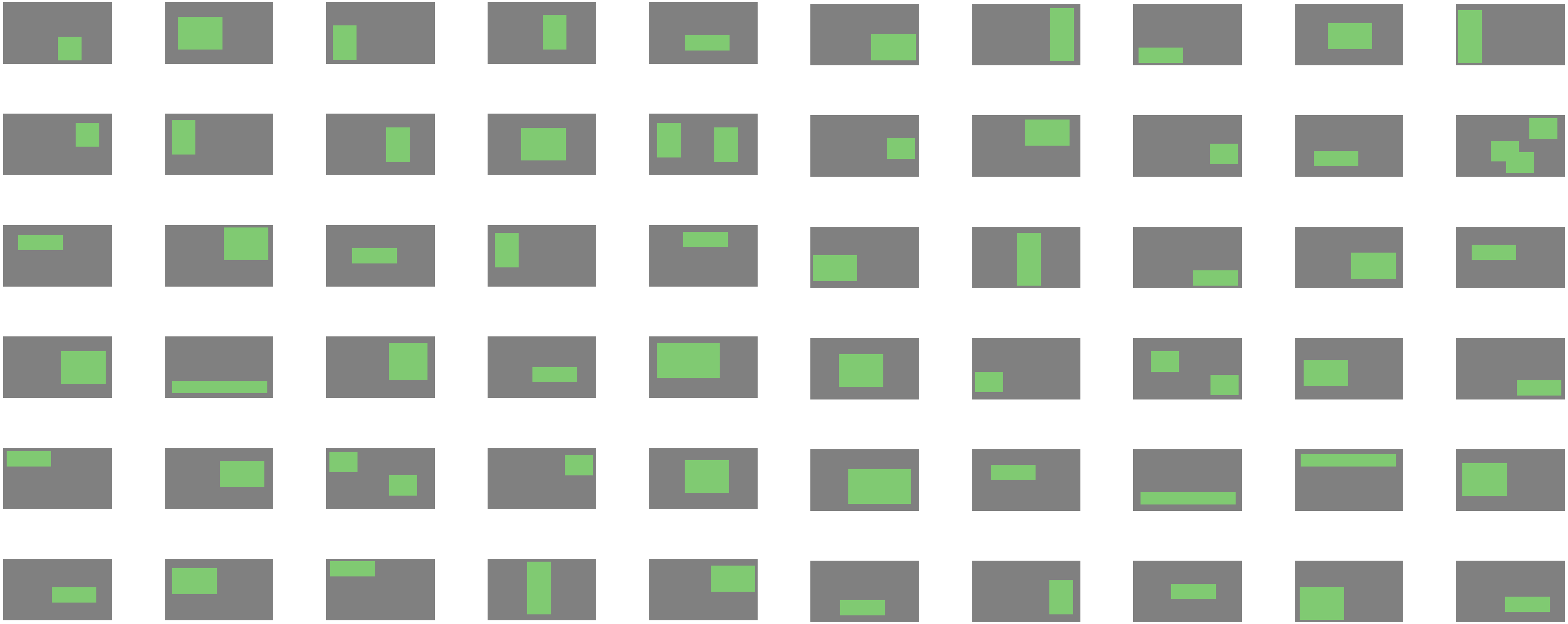


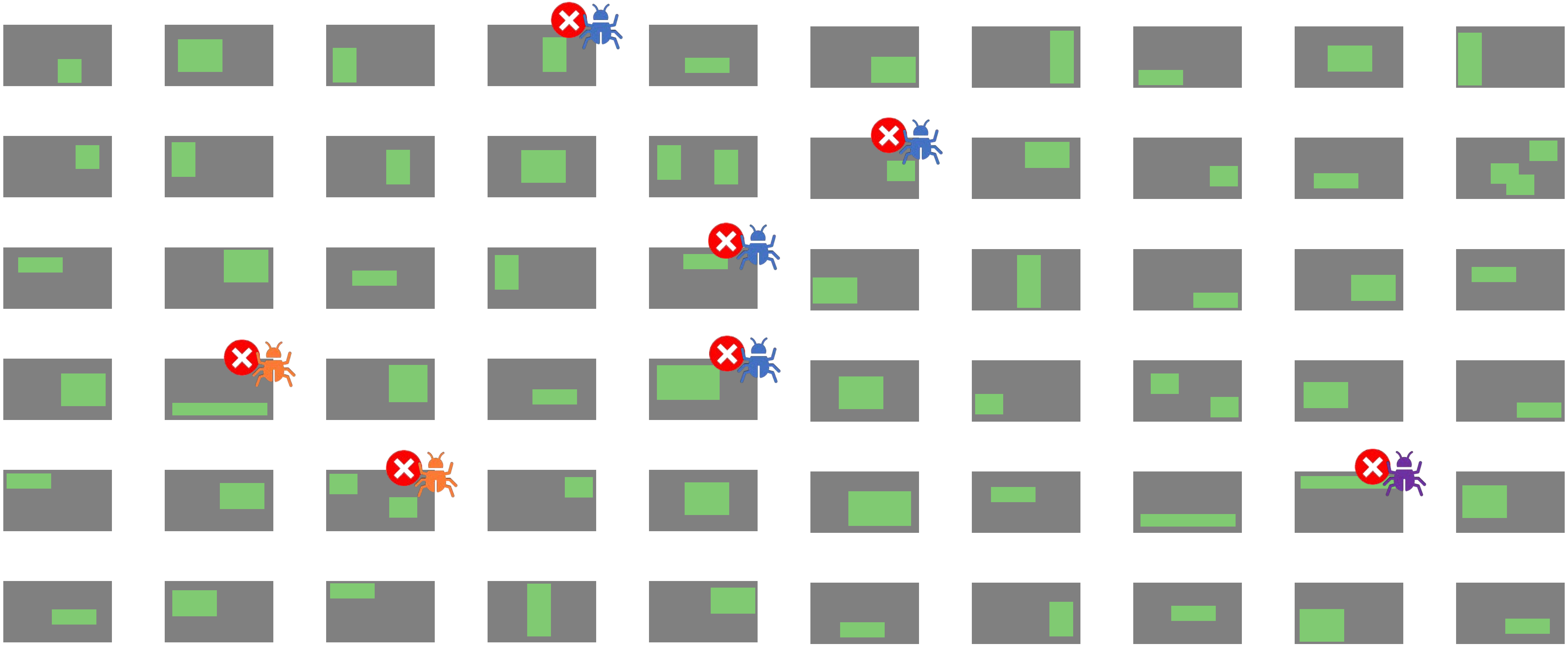
Test-Impact Analysis
Initial Recording of all Tests
Execution of All Tests
Coverage & Runtime of All Tests
Teamscale


Test-Impact Analysis
Initial Recording of all Tests
Execution of All Tests
Coverage & Runtime of All Tests
Teamscale
Test Execution after Changes
VCS
Code Changes

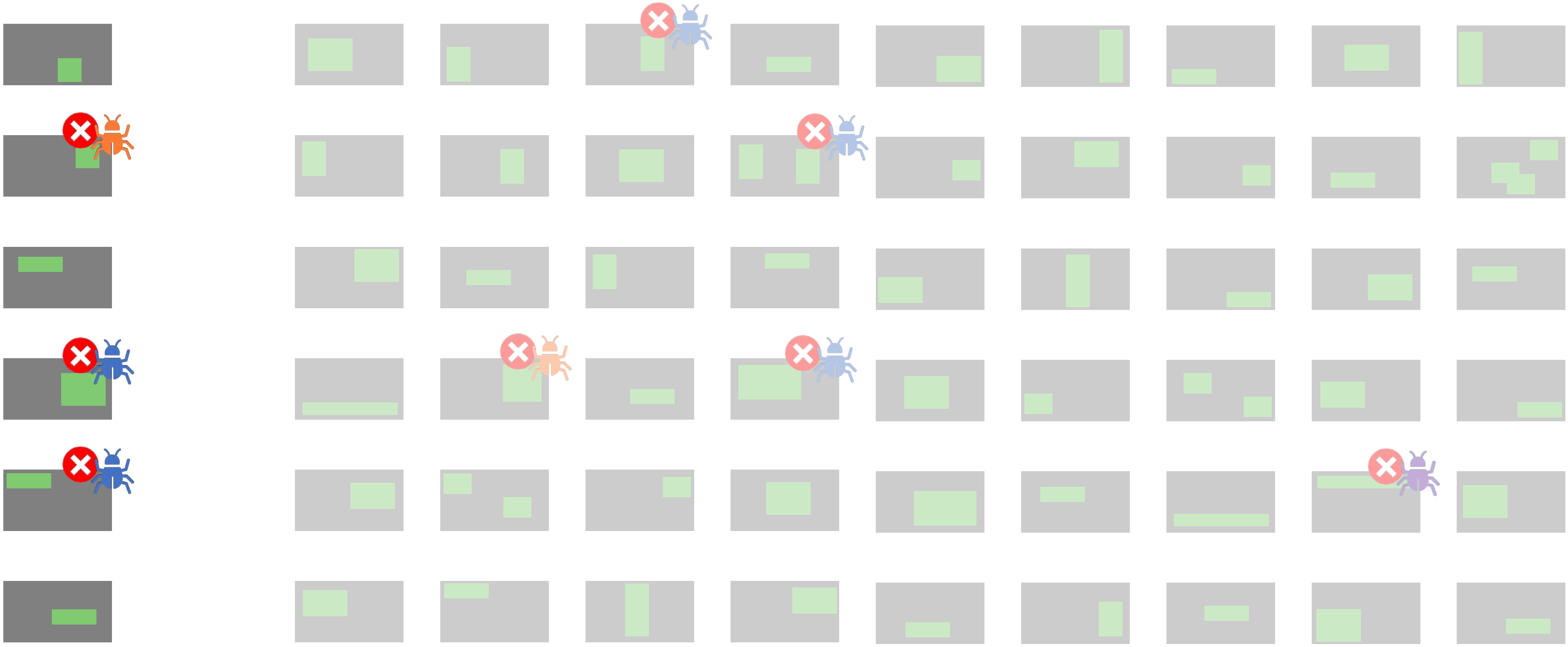

Test-Impact Analysis
Initial Recording of all Tests
Execution of All Tests
Coverage & Runtime of All Tests
Test Execution after Changes
VCS
Code Changes
Test Selection
Test Prioritization
Test-Impact Analysis
Ordered List of Impacted Tests
Teamscale

Isn‘t there a simpler option?

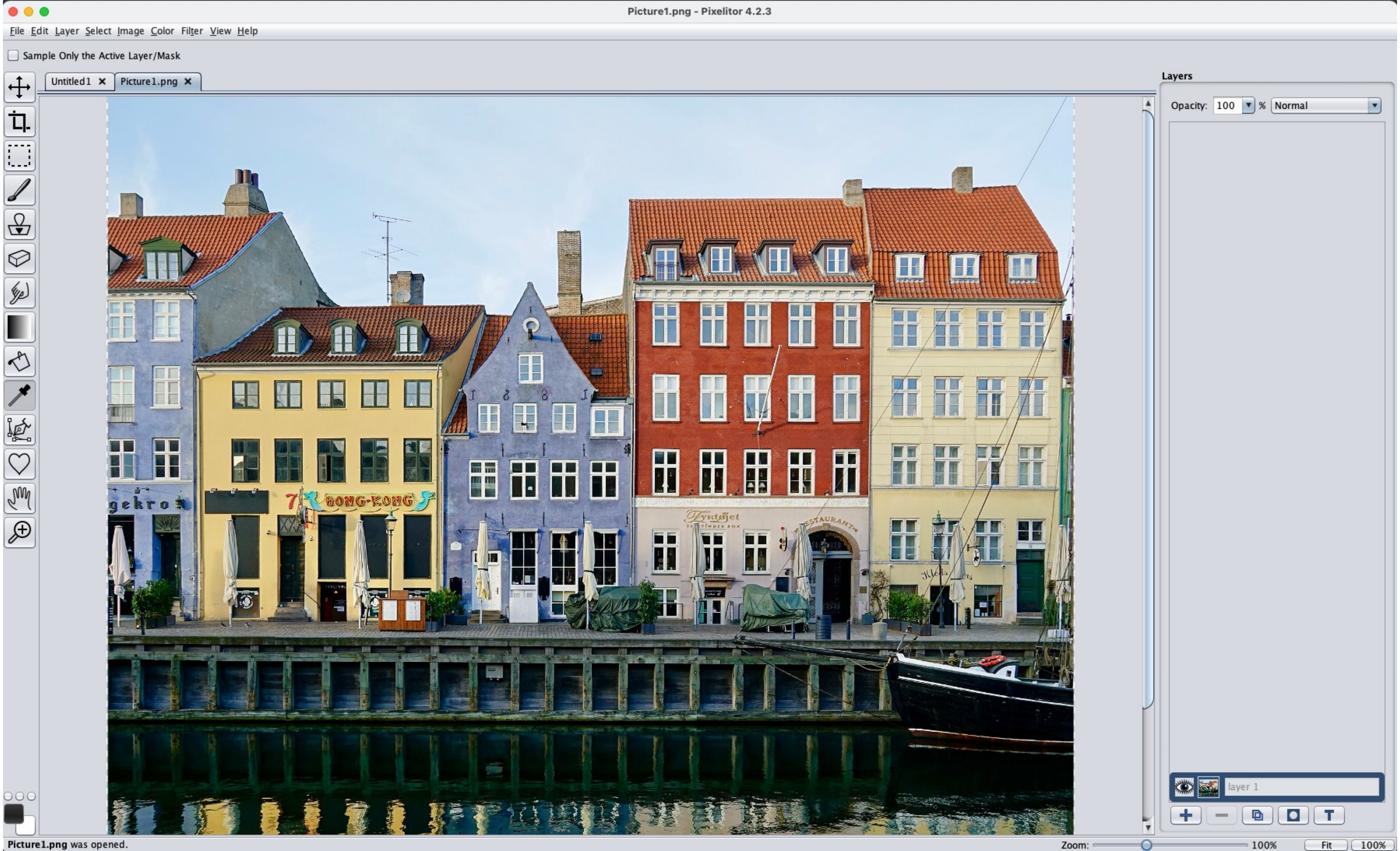


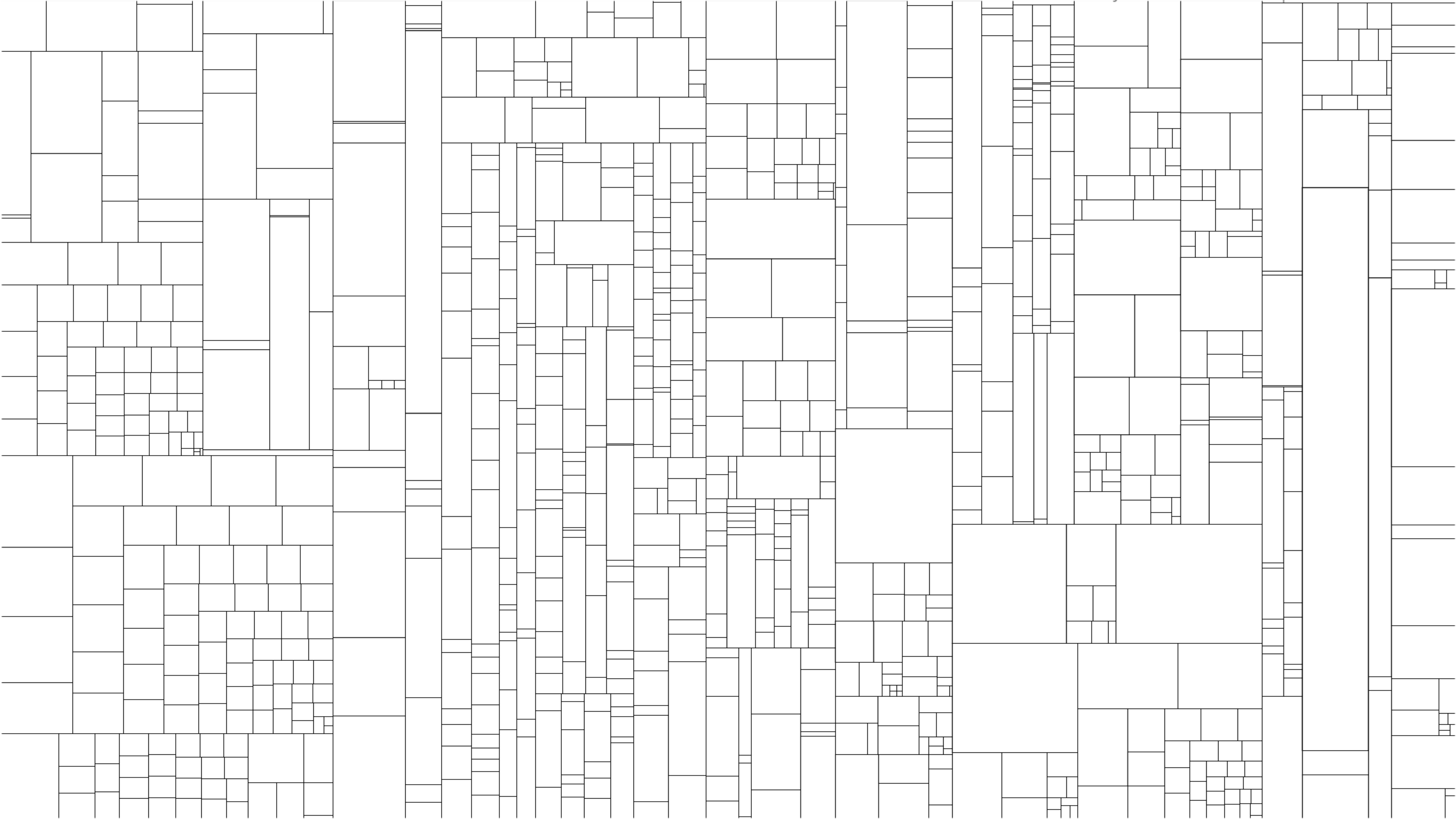


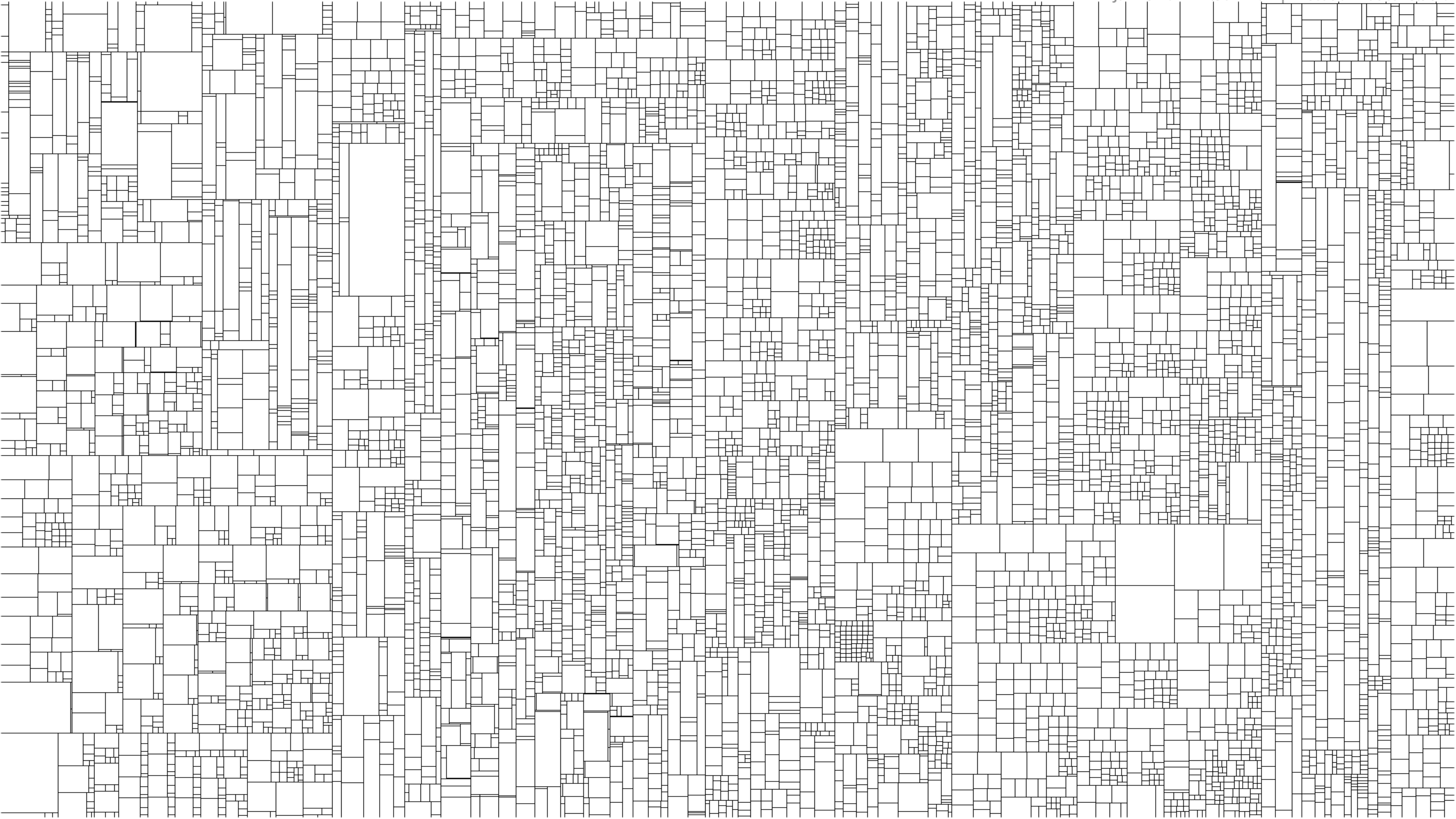
Pareto Testing: 80/20 Optimization

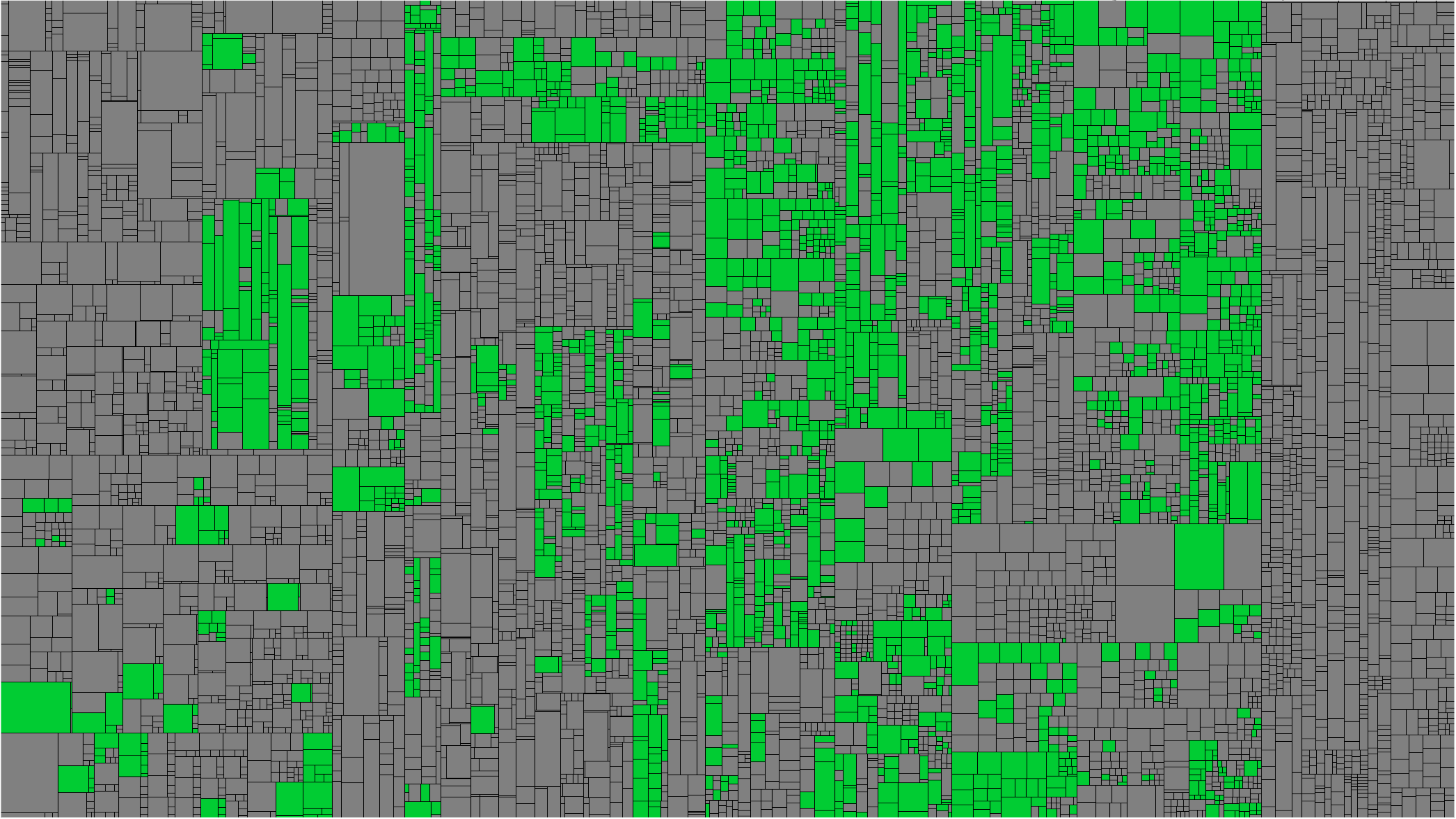


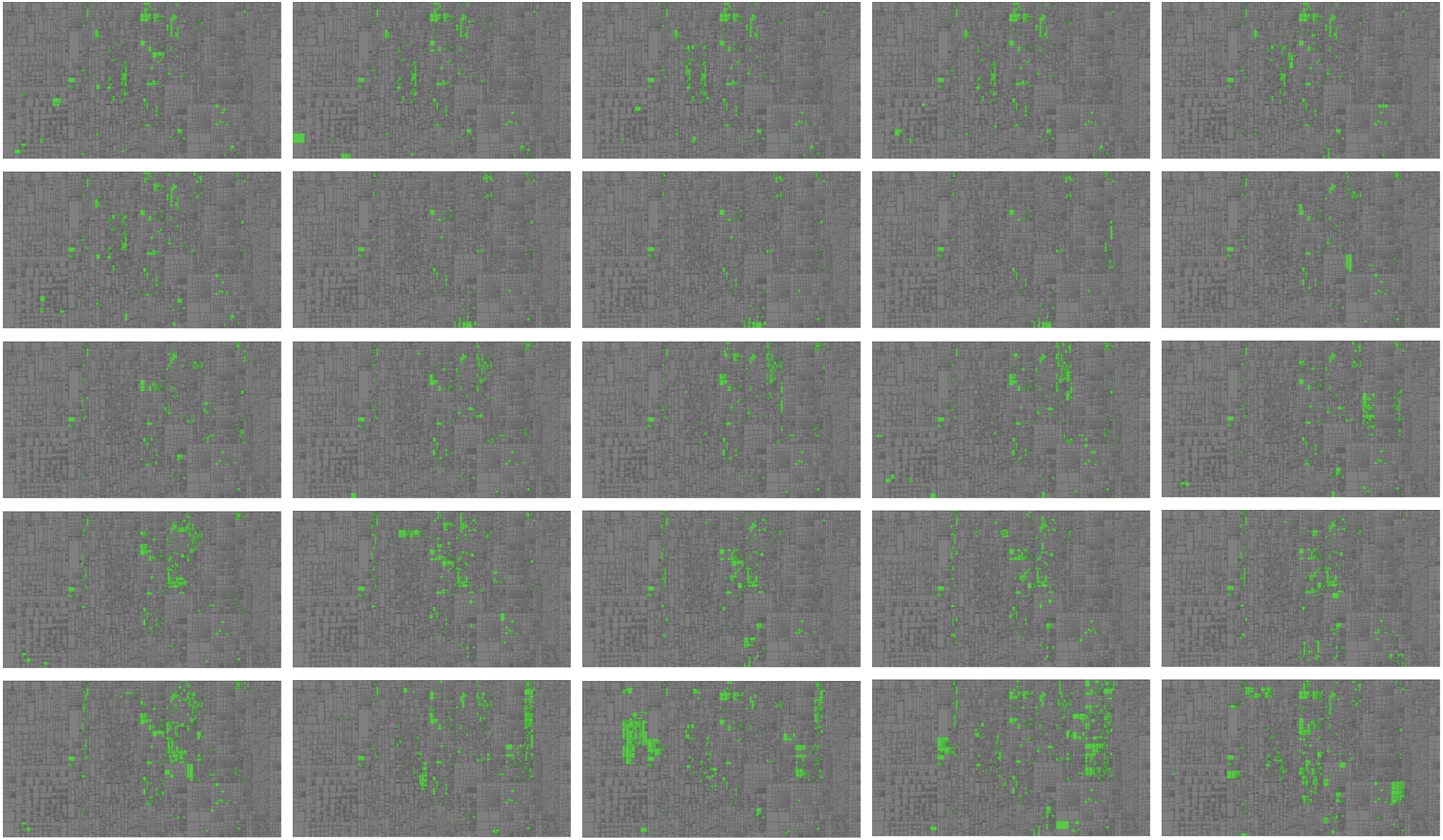
Why is there potential for optimization?

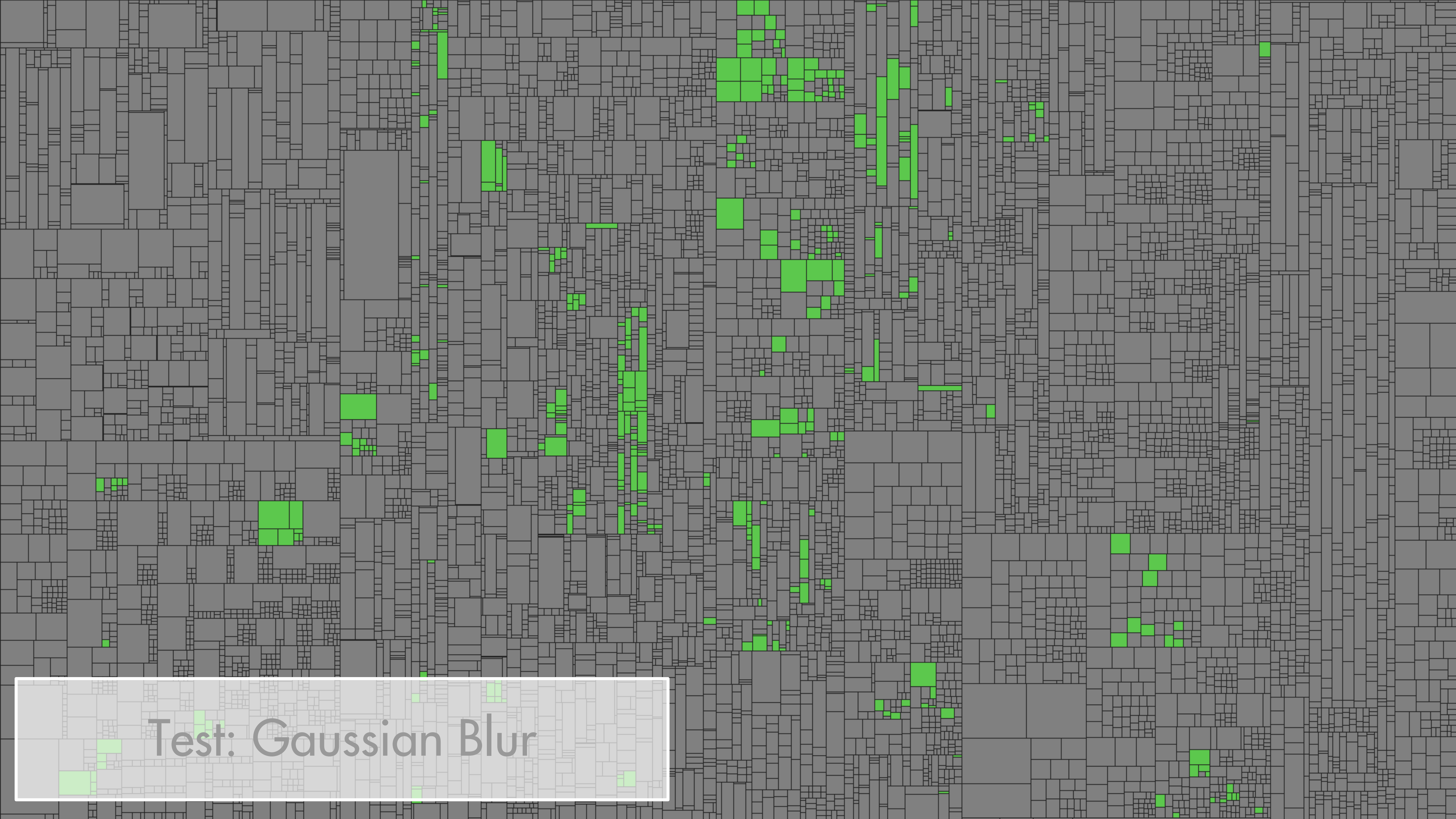

Picture1.png - Pixelitor 4.2.3
File Edit Layer Select Image Color Filter View Help
Sample Only the Active Layer/Mask
Untitled1
Picture1.png
Layers
Opacity: 100 %
Normal
layer 1
Picture1.png was opened.
Zoom:
100%
Fit
100%

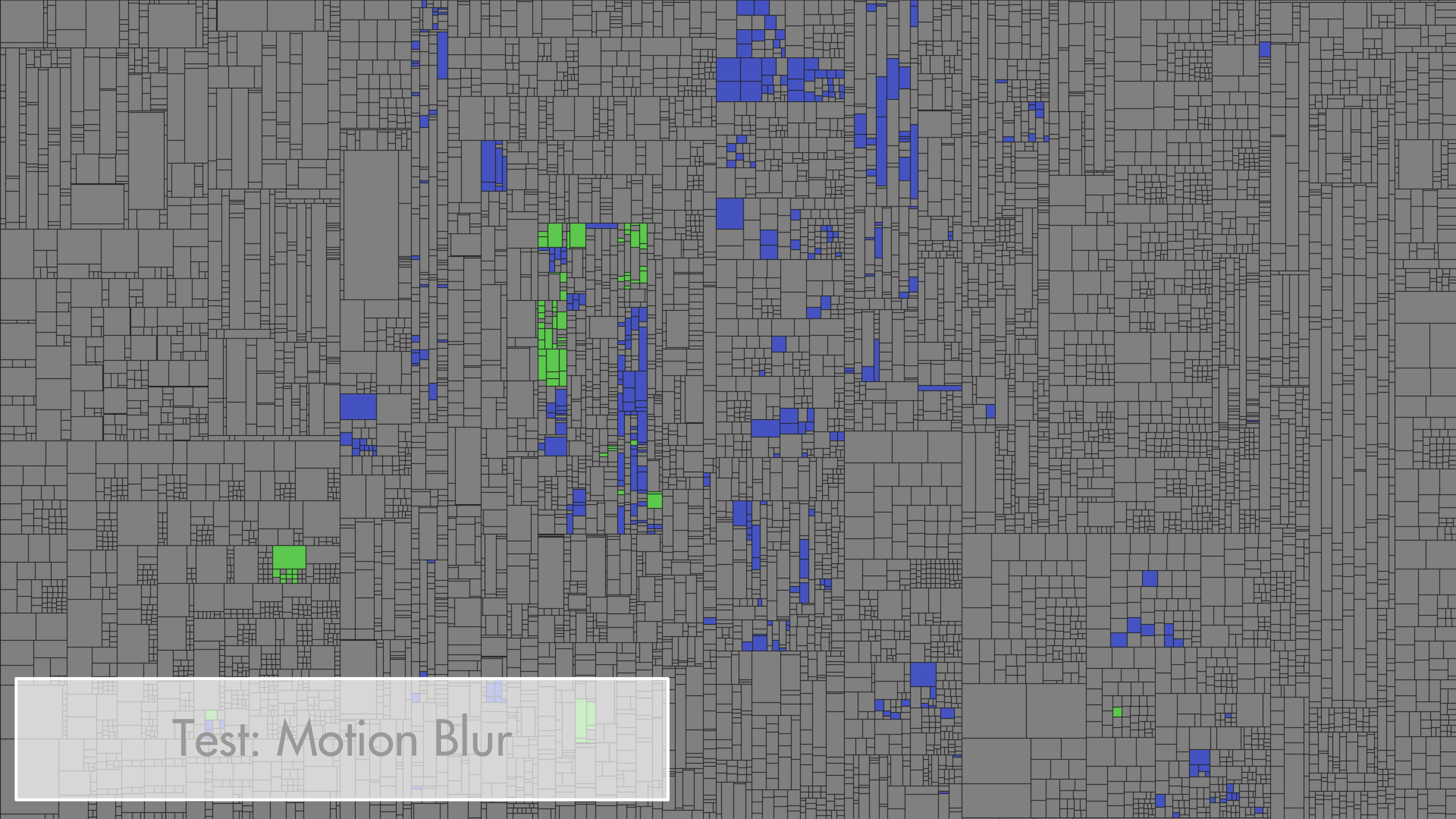


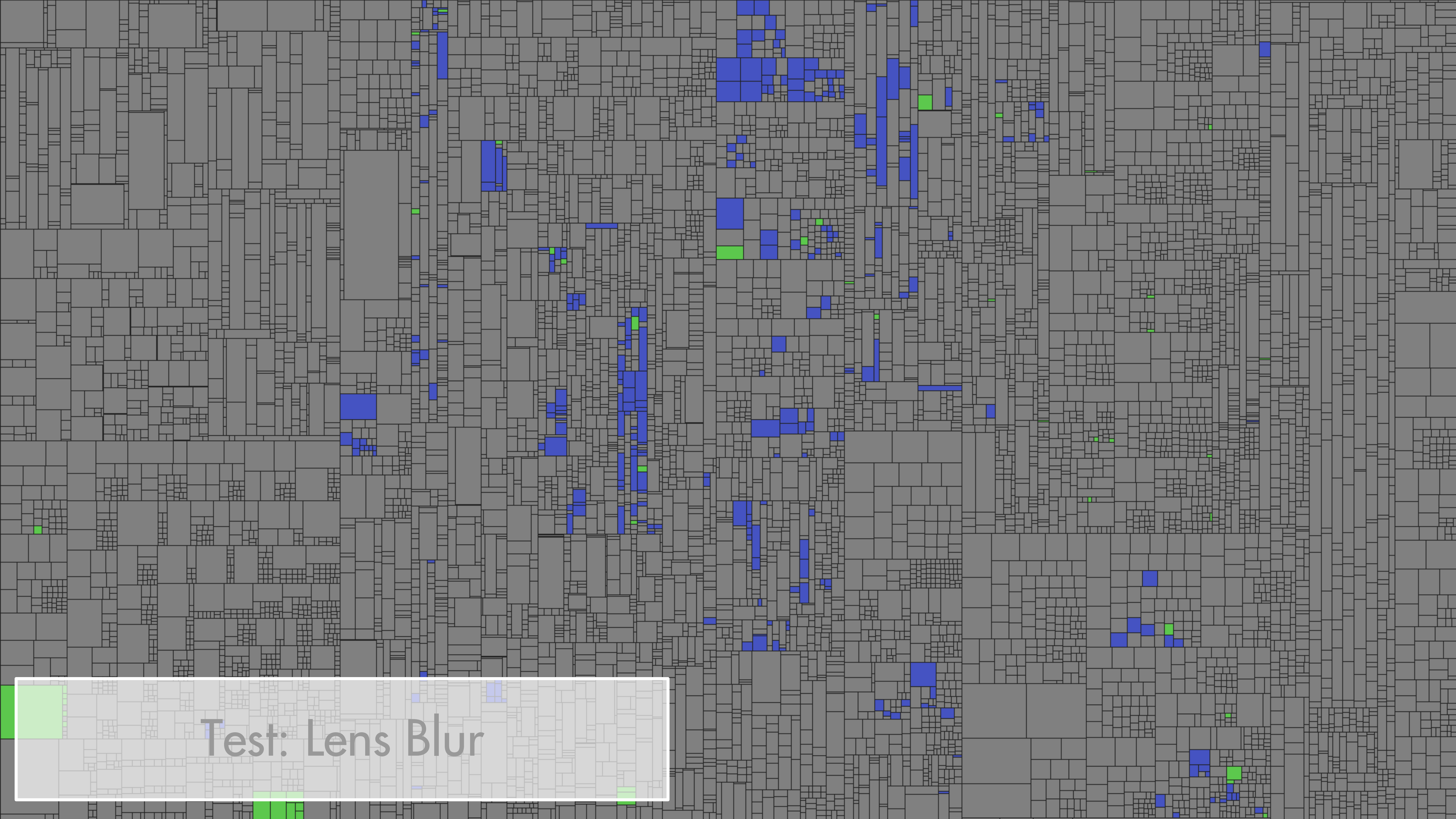




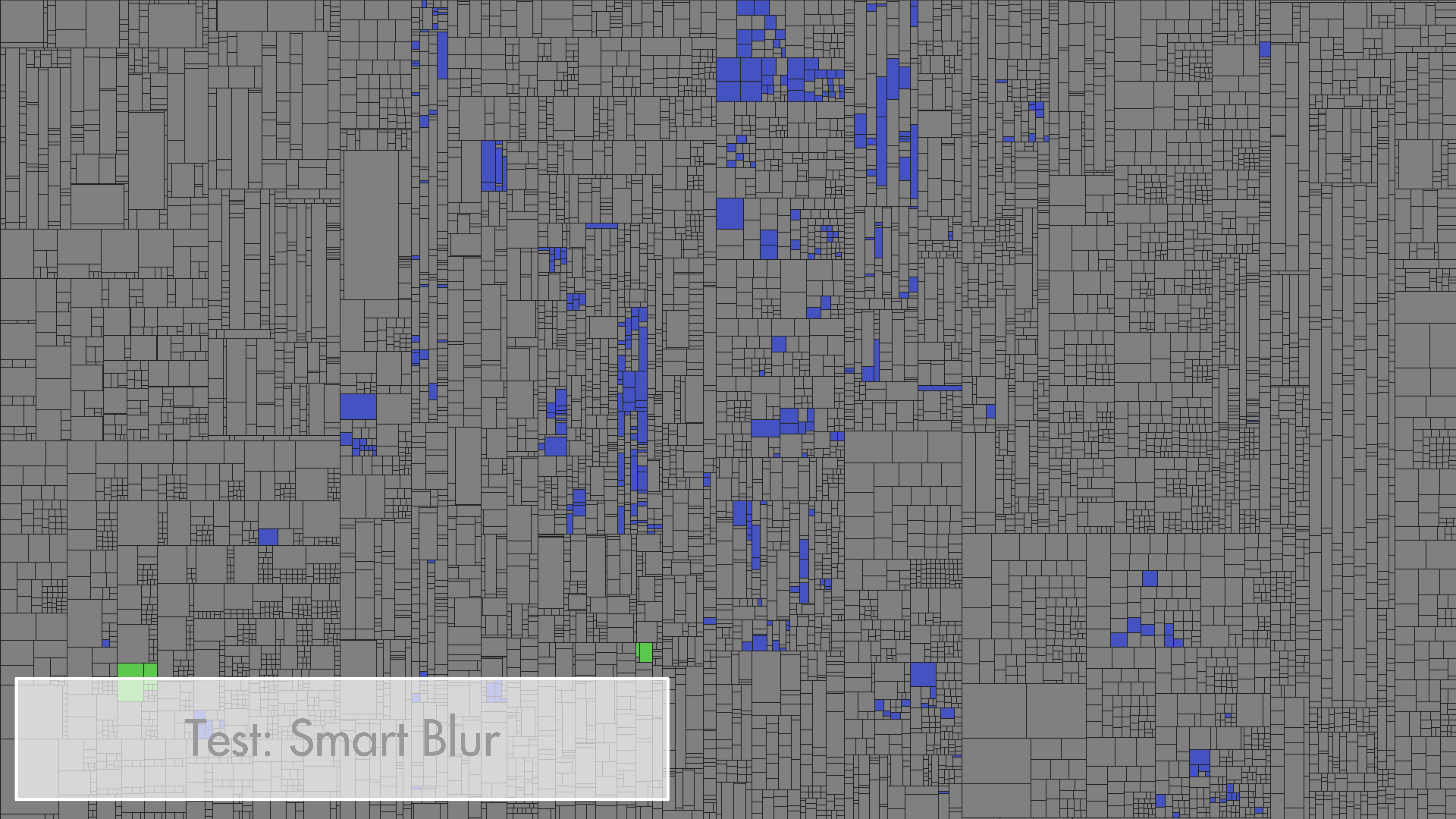



Test: Gaussian Blur

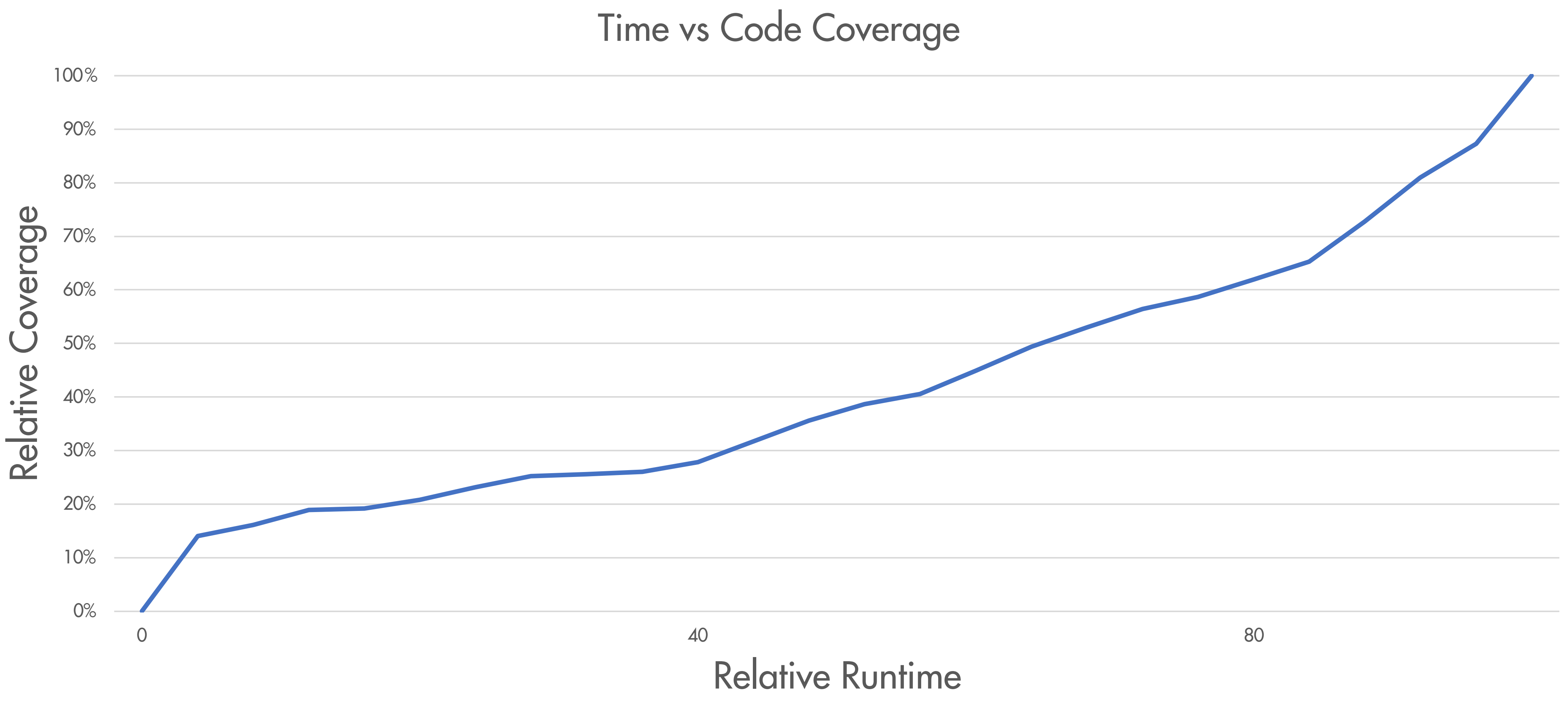

Test: Motion Blur



Test: Lens Blur

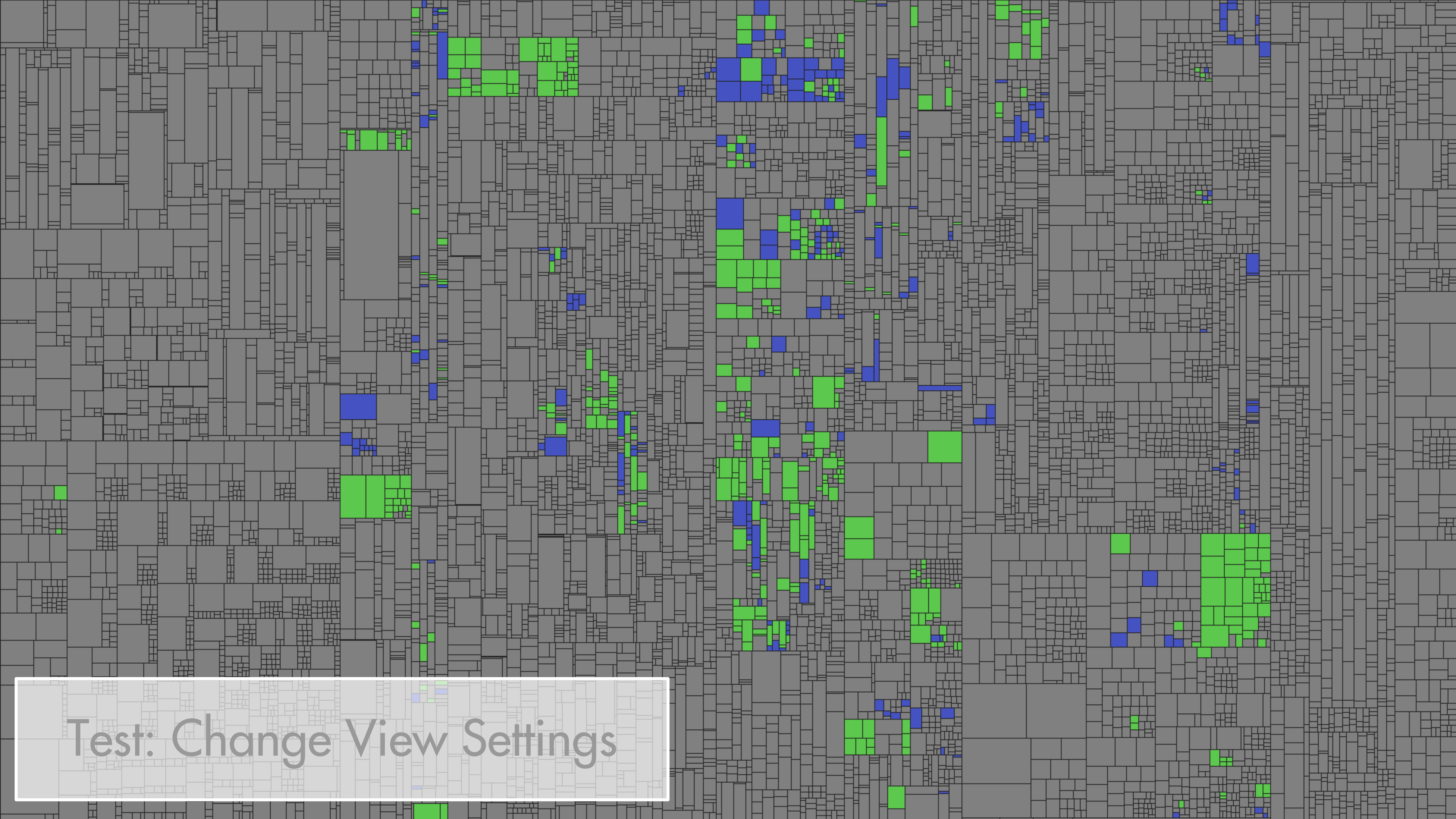


Test: Smart Blur

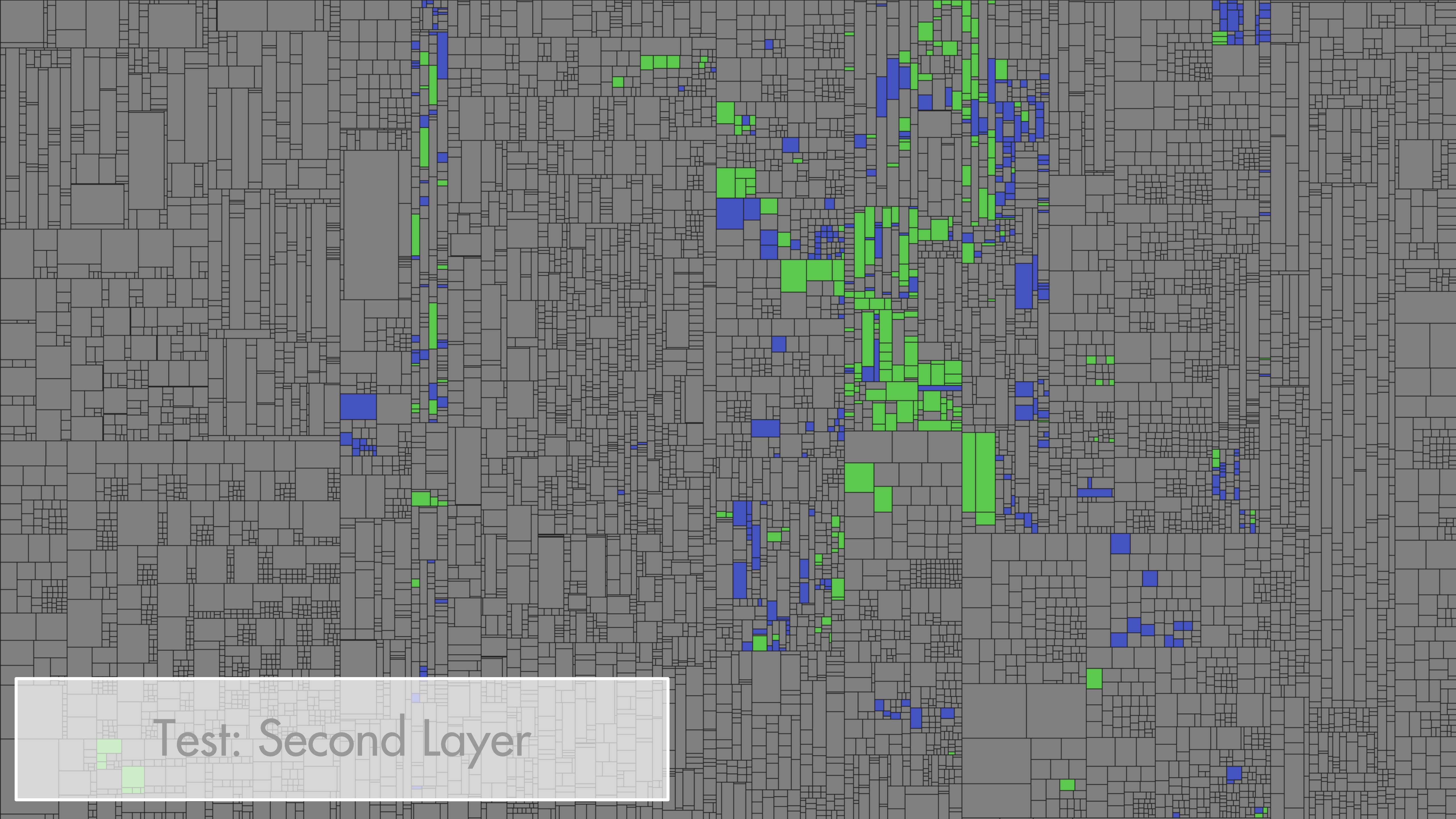


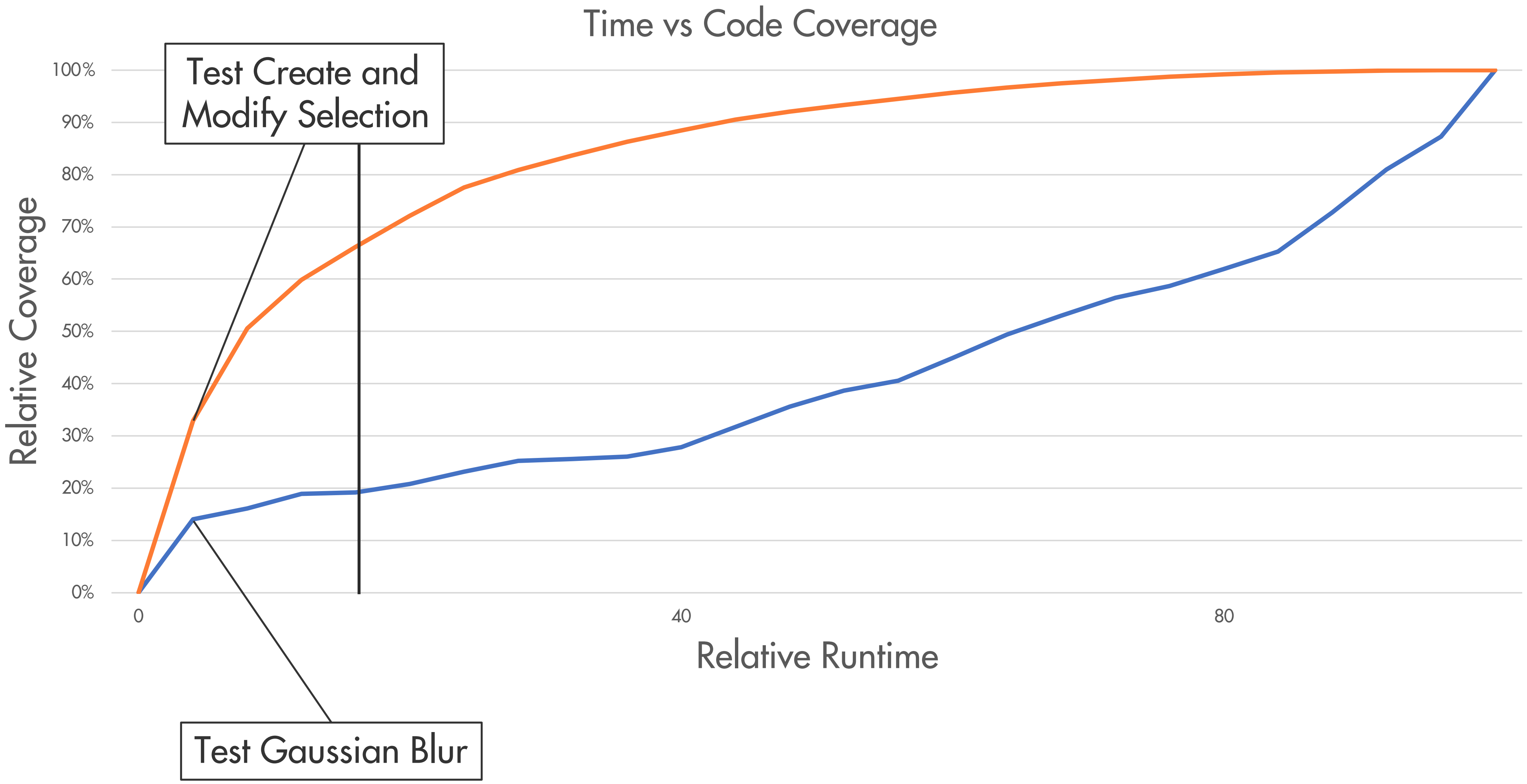
Time vs Code Coverage
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
Relative Coverage
0
40
80
Relative Runtime

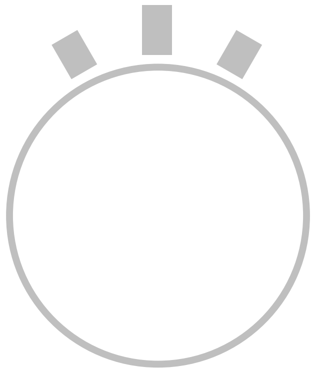
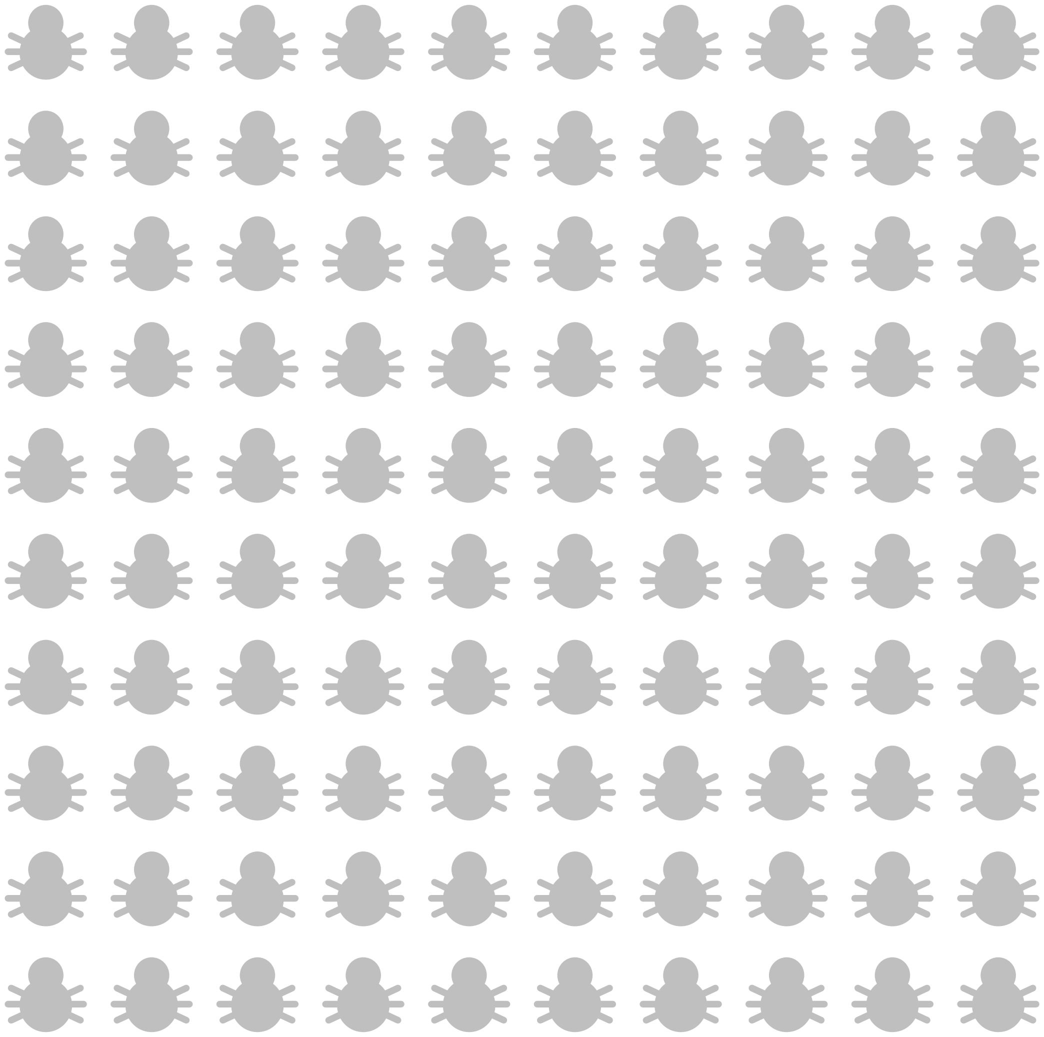
Test: Create and Modify Selection

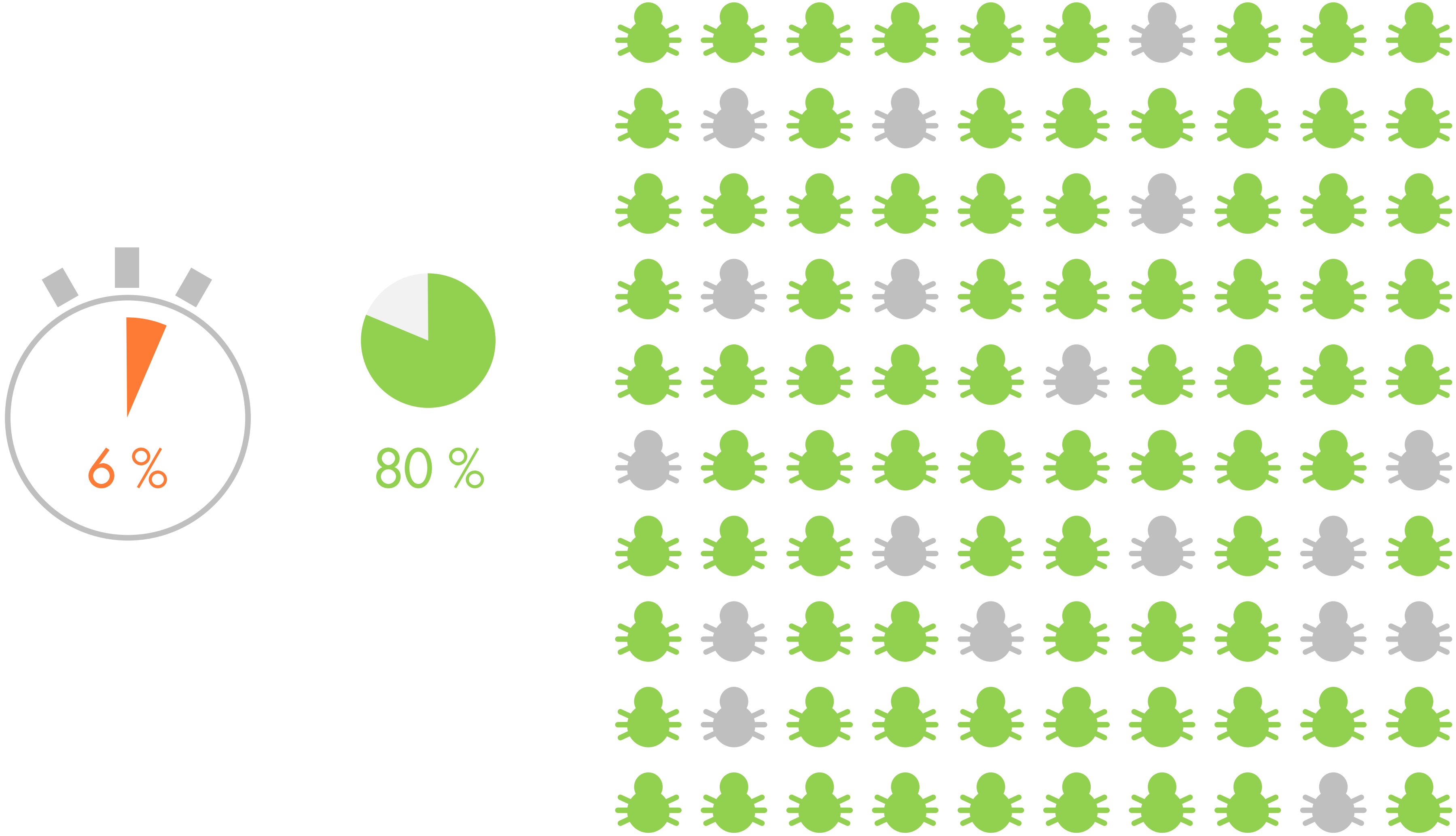

Test: Change View Settings

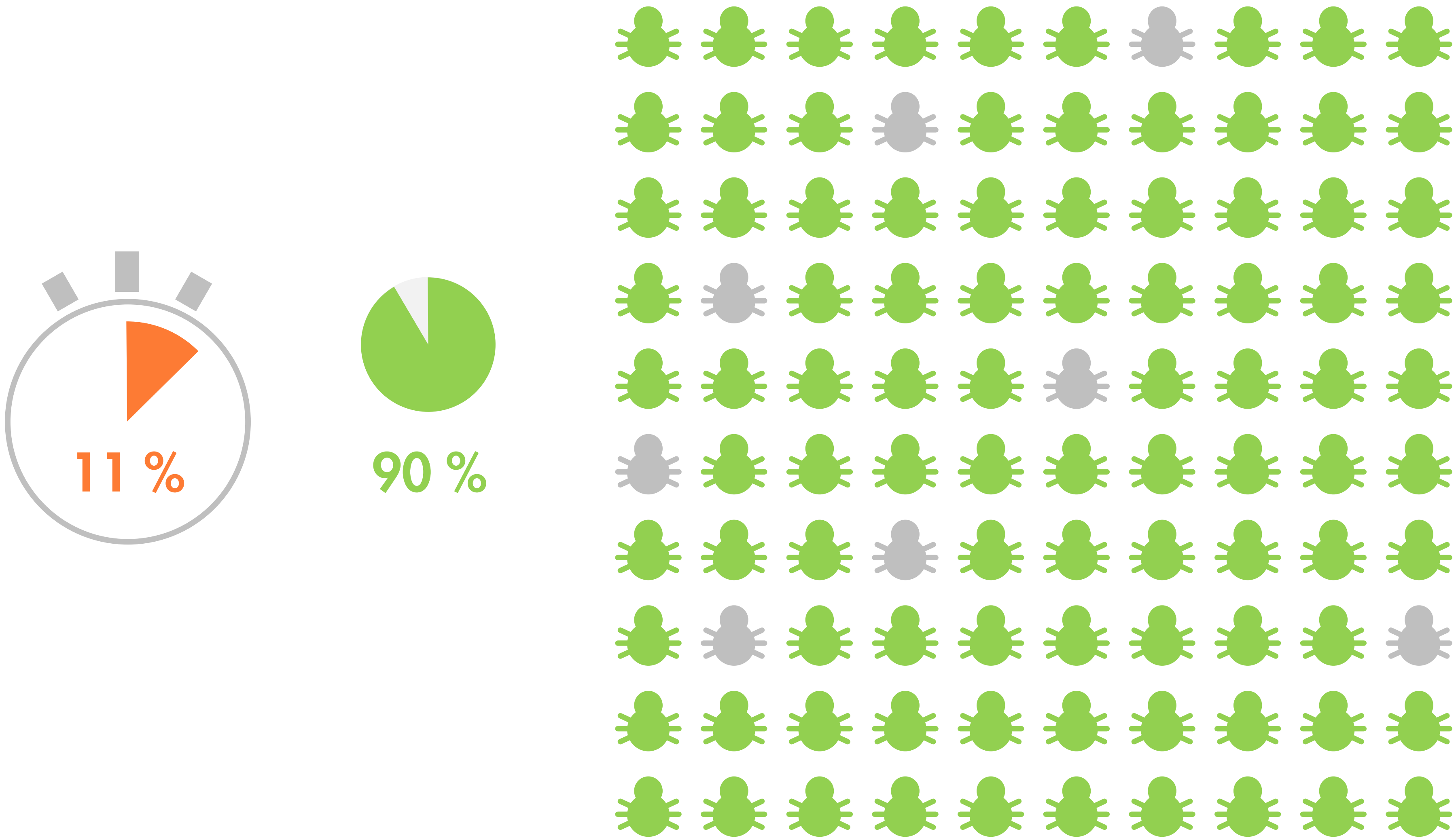
Test: Second Layer

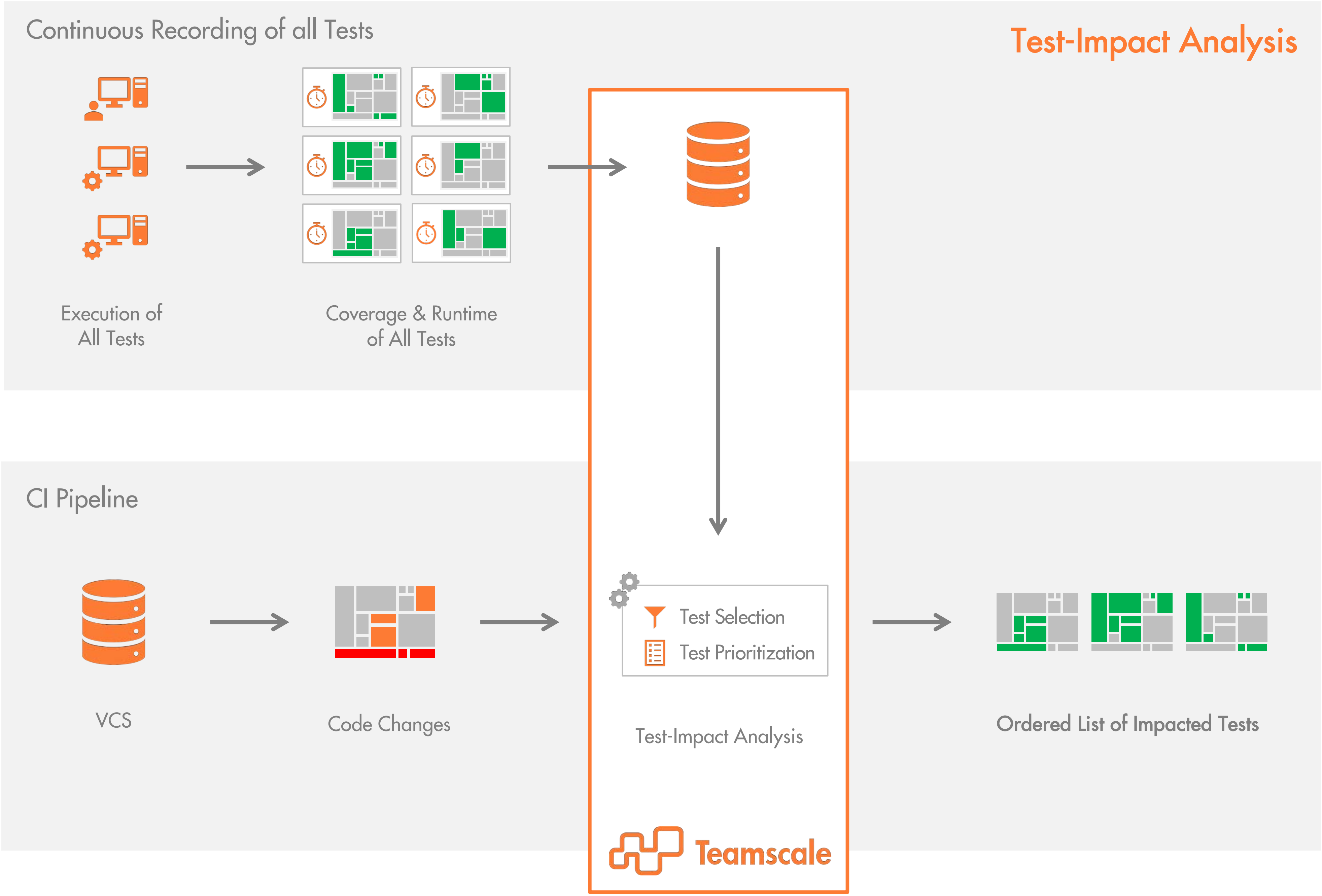



Time vs Code Coverage
Test Create and Modify Selection
Test Gaussian Blur
Relative Coverage
Relative Runtime
100%
90%
80%
70%
60%
50%
40%
30%
20%
10%
0%
0
40
80

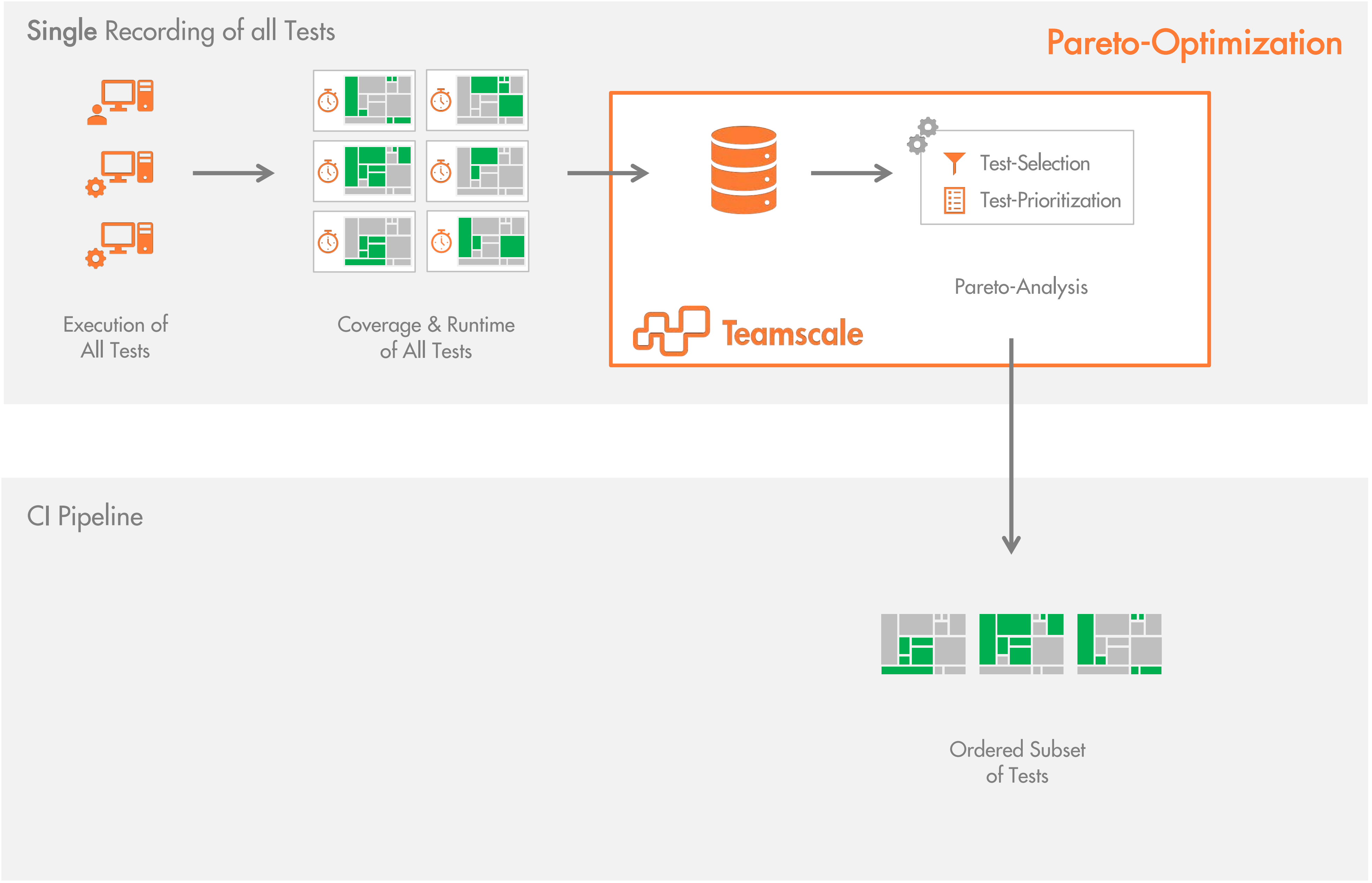




6 %
80 %


11 %
90 %



Continuous Recording of all Tests
Test-Impact Analysis
Execution of All Tests
Coverage & Runtime of All Tests
CI Pipeline
VCS
Code Changes
Test Selection
Test Prioritization
Test-Impact Analysis
Ordered List of Impacted Tests
Teamscale



Single Recording of all Tests
Pareto-Optimization
Test-Selection
Test-Prioritization
Execution of All Tests
Coverage & Runtime of All Tests
Pareto-Analysis
Teamscale
CI Pipeline
Ordered Subset of Tests

Test-Impact-Analysis

Test selection w.r.t. code changes

90% of bugs found in 2% time

Requires continuous recording of coverage and tight integration with test automation framework.

Higher speedup & higher effort

Pareto-Optimization

Tests selection independent of code changes

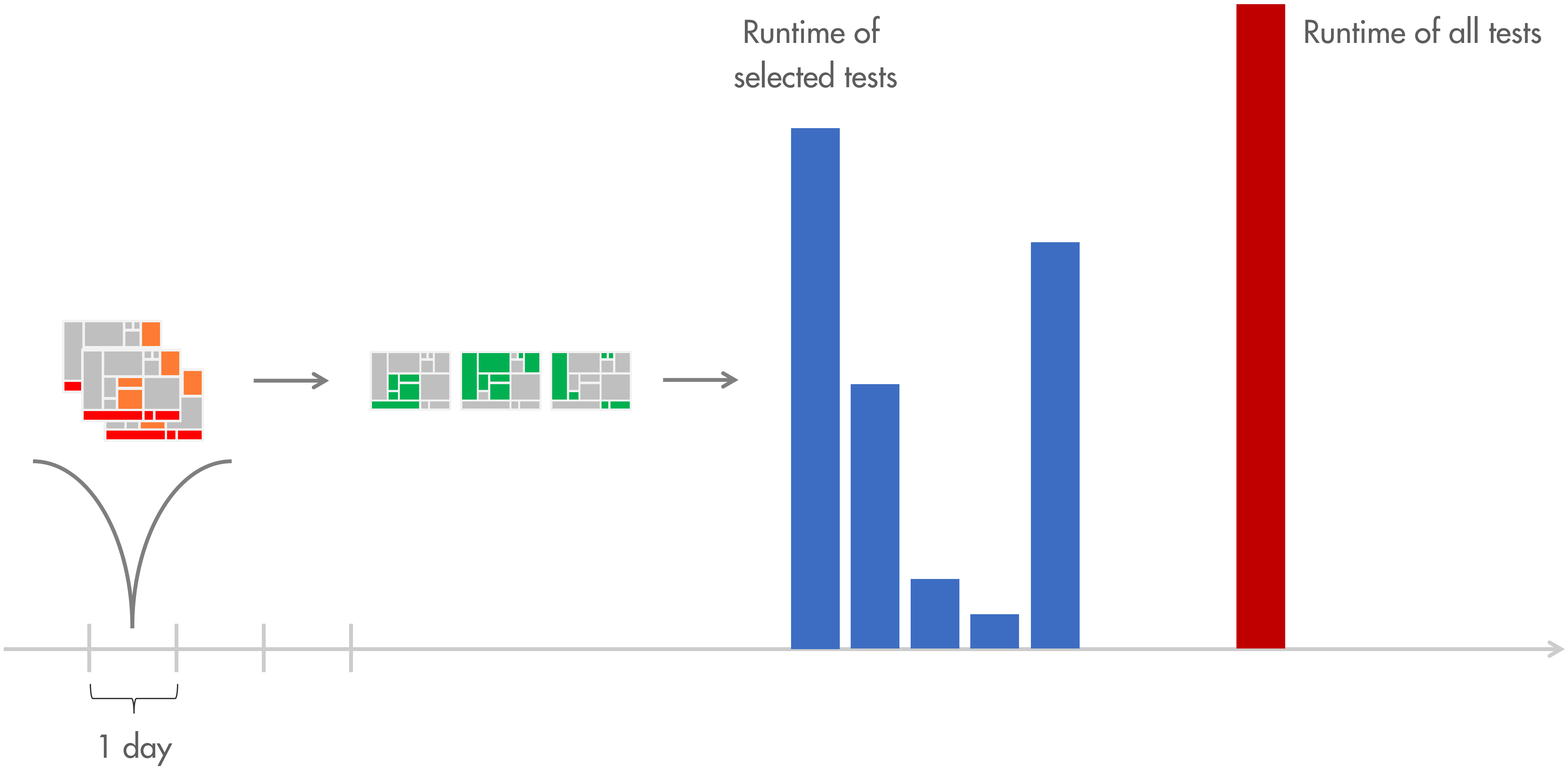
90% of bugs found in 11% time

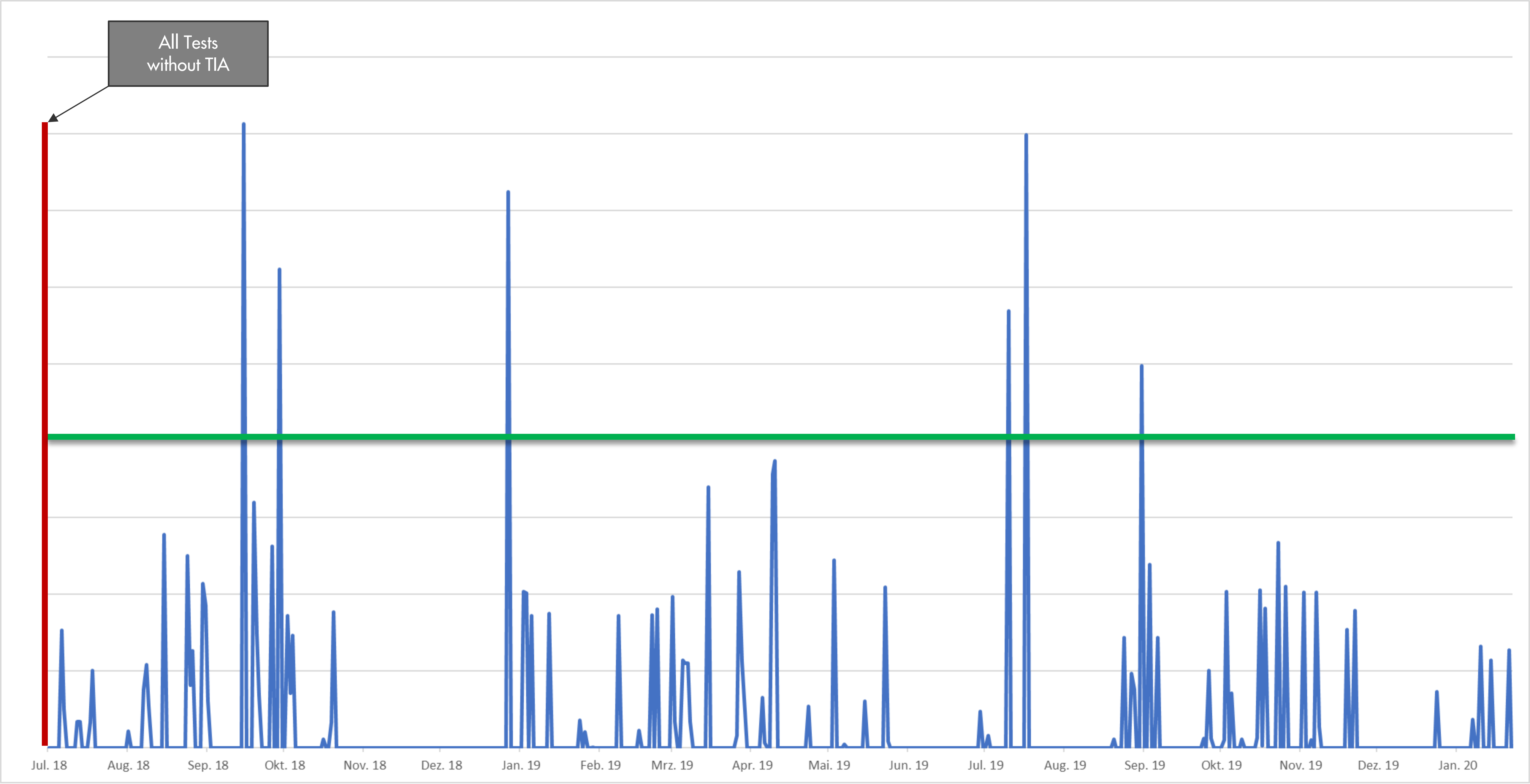
Record coverage once (per quarter)

Smaller effort & higher applicability

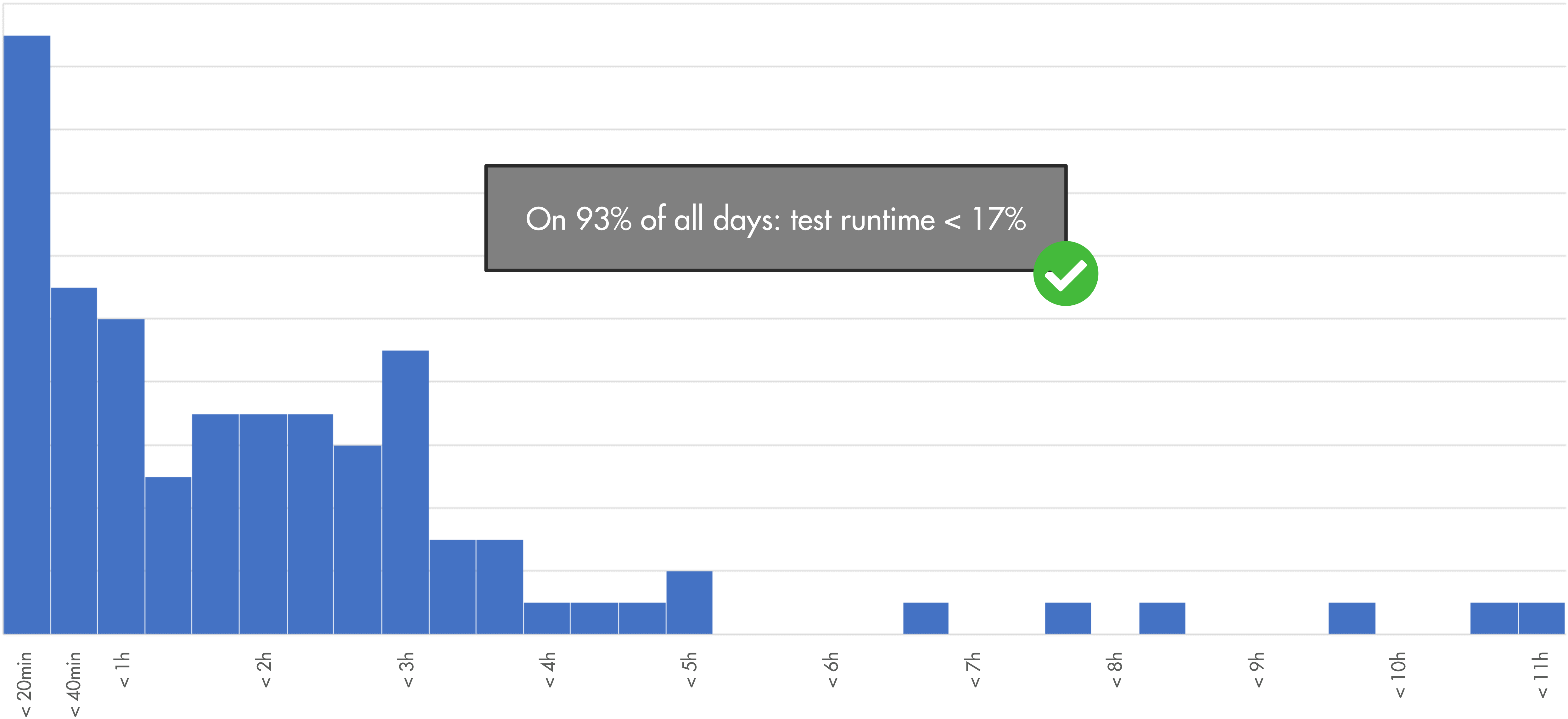
Evaluation: what's the benefit?

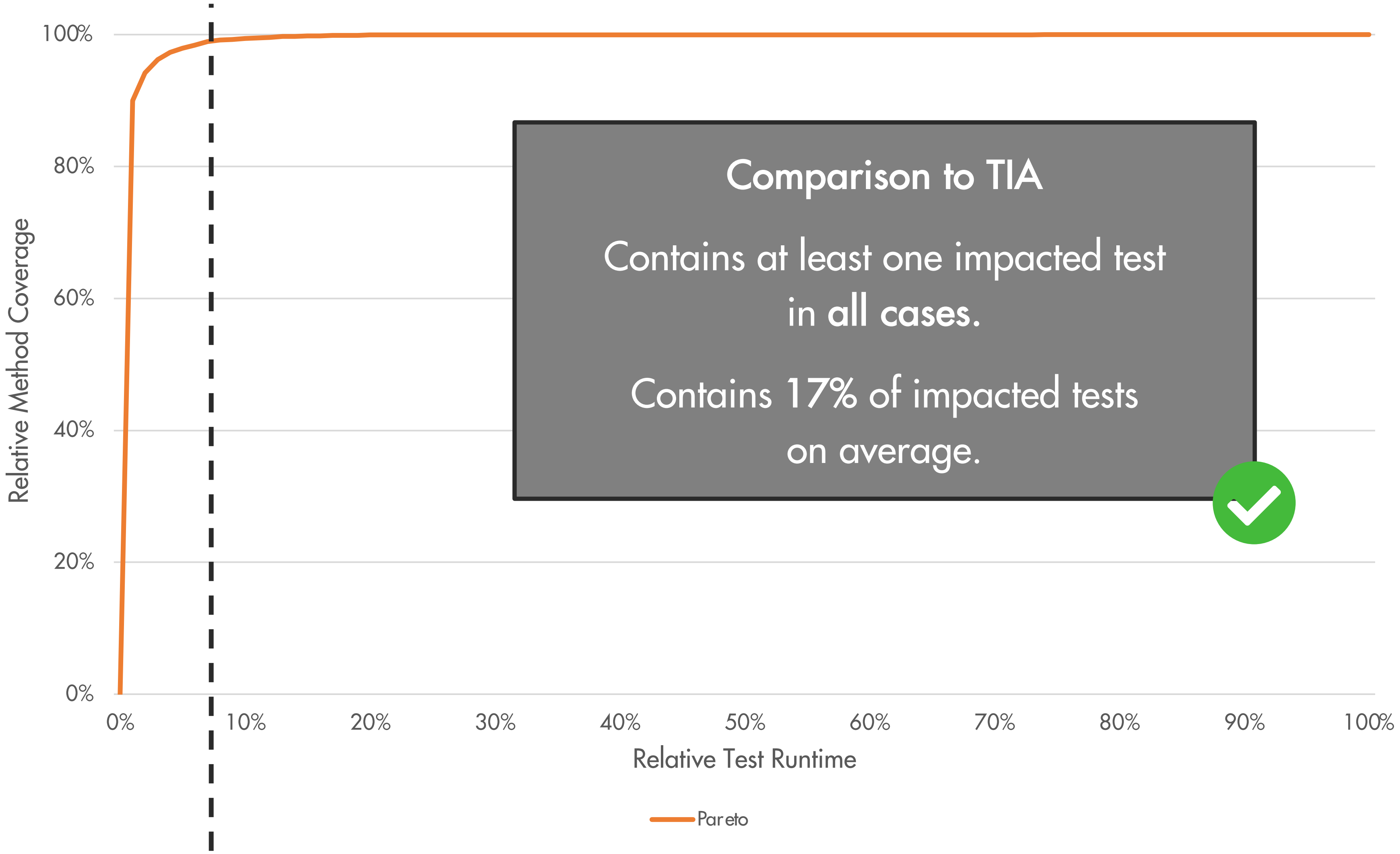
How much test time can we expect to save?


Runtime of
selected tests
Runtime of all tests
1 day



All Tests without TIA
Jul. 18
Aug. 18
Sep. 18
Okt. 18
Nov. 18
Dez. 18
Jan. 19
Feb. 19
Mrz. 19
Apr. 19
Mai. 19
Jun. 19
Jul. 19
Aug. 19
Sep. 19
Okt. 19
Nov. 19
Dez. 19
Jan. 20


On 93% of all days: test runtime < 17%
< 20min
< 40min
< 1h
< 2h
< 3h
< 4h
< 5h
< 6h
< 7h
< 8h
< 9h
< 10h
< 11h



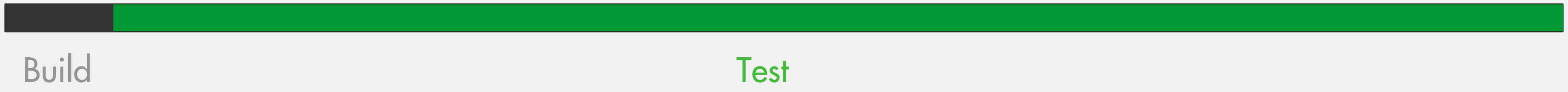
We achieve 99,2% relative method coverage with just 1h of test runtime.

CQSE

Using TIA

Before the Pilot: nightly „Re-run all“

Feedback only on the next day

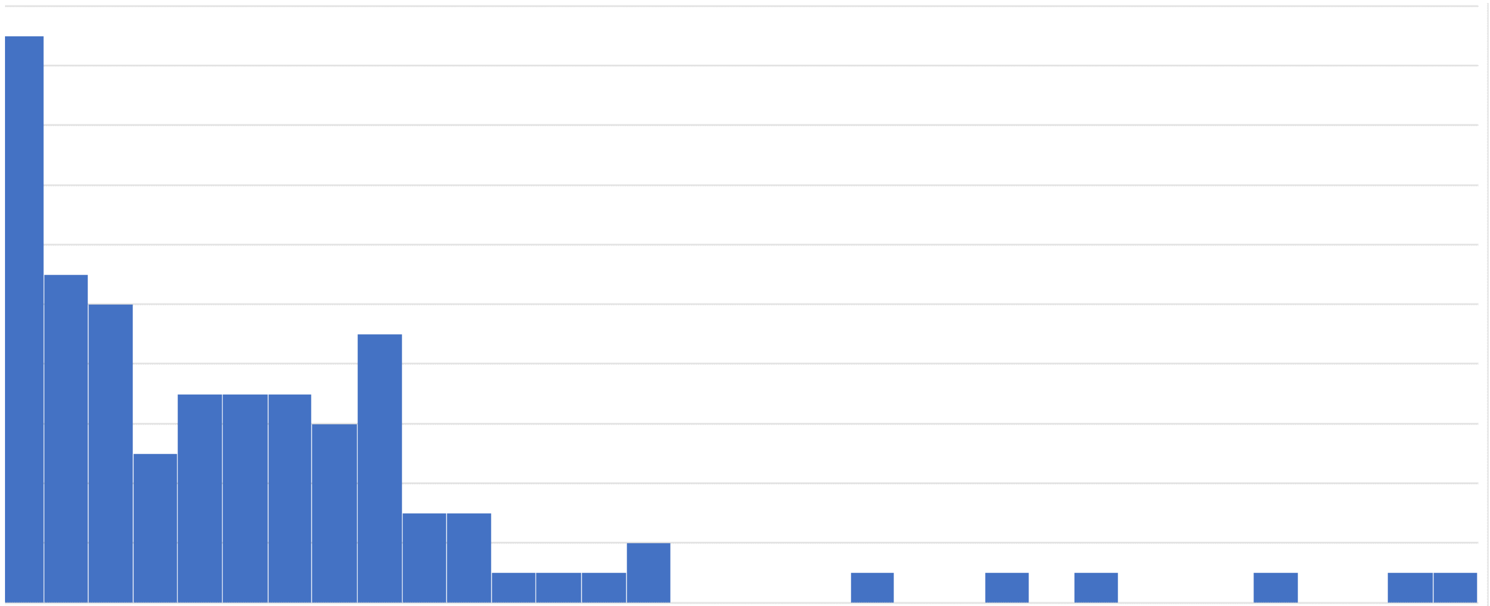


Using TIA

Fast feedback during the day: TIA Timeboxing

First feedback on checked in changes within 2h.

Build Test max. 1h



Using TIA

Fast feedback during the day: TIA Timeboxing

First feedback on checked in changes within 2h.

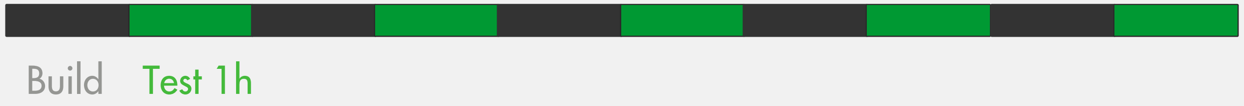
Build Test max. 1h

Still: nightly „Re-run all“, to update TIA data in Teamscale

Using Pareto Testing

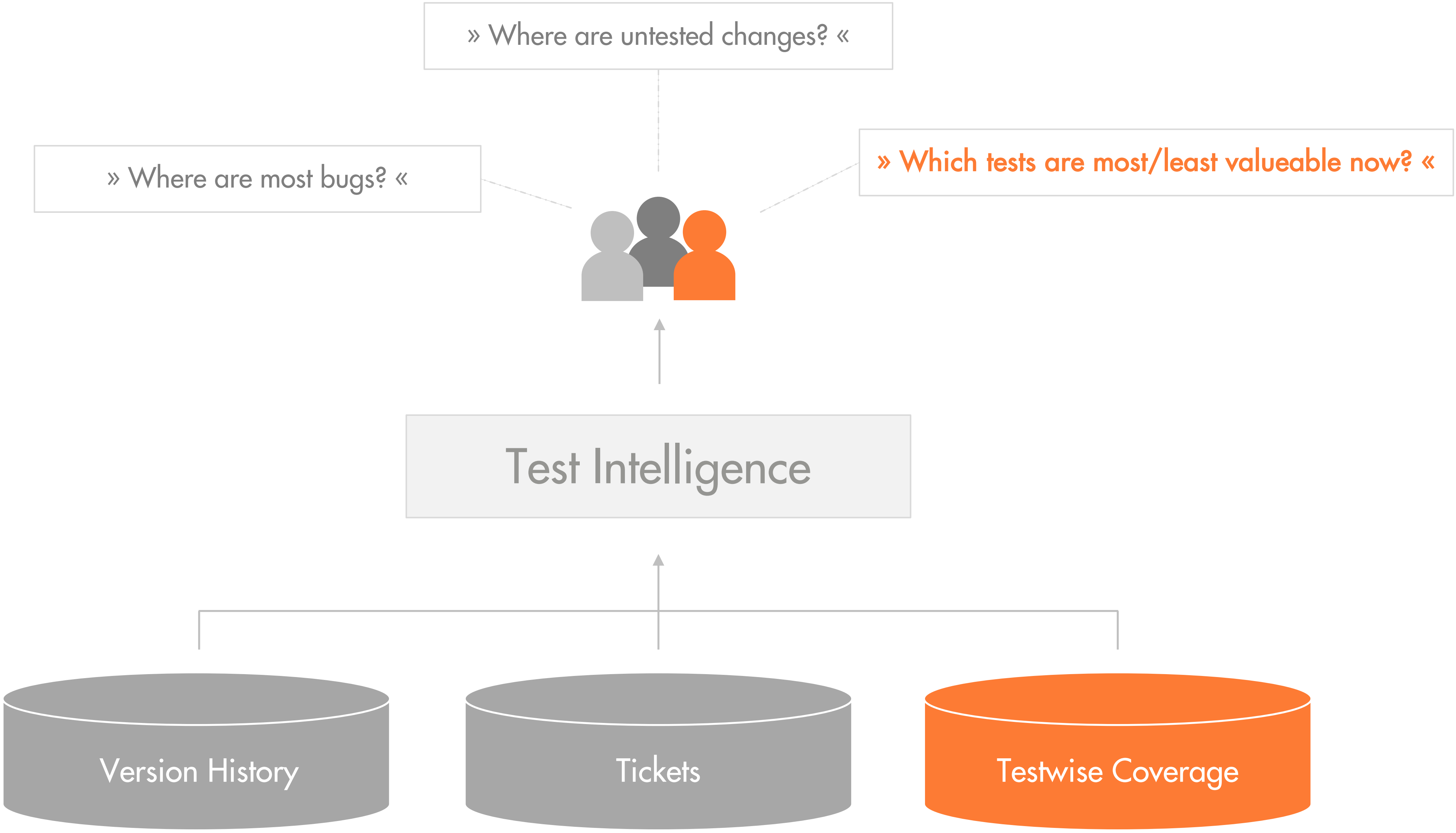
Fast feedback during the day: Pareto Timeboxing

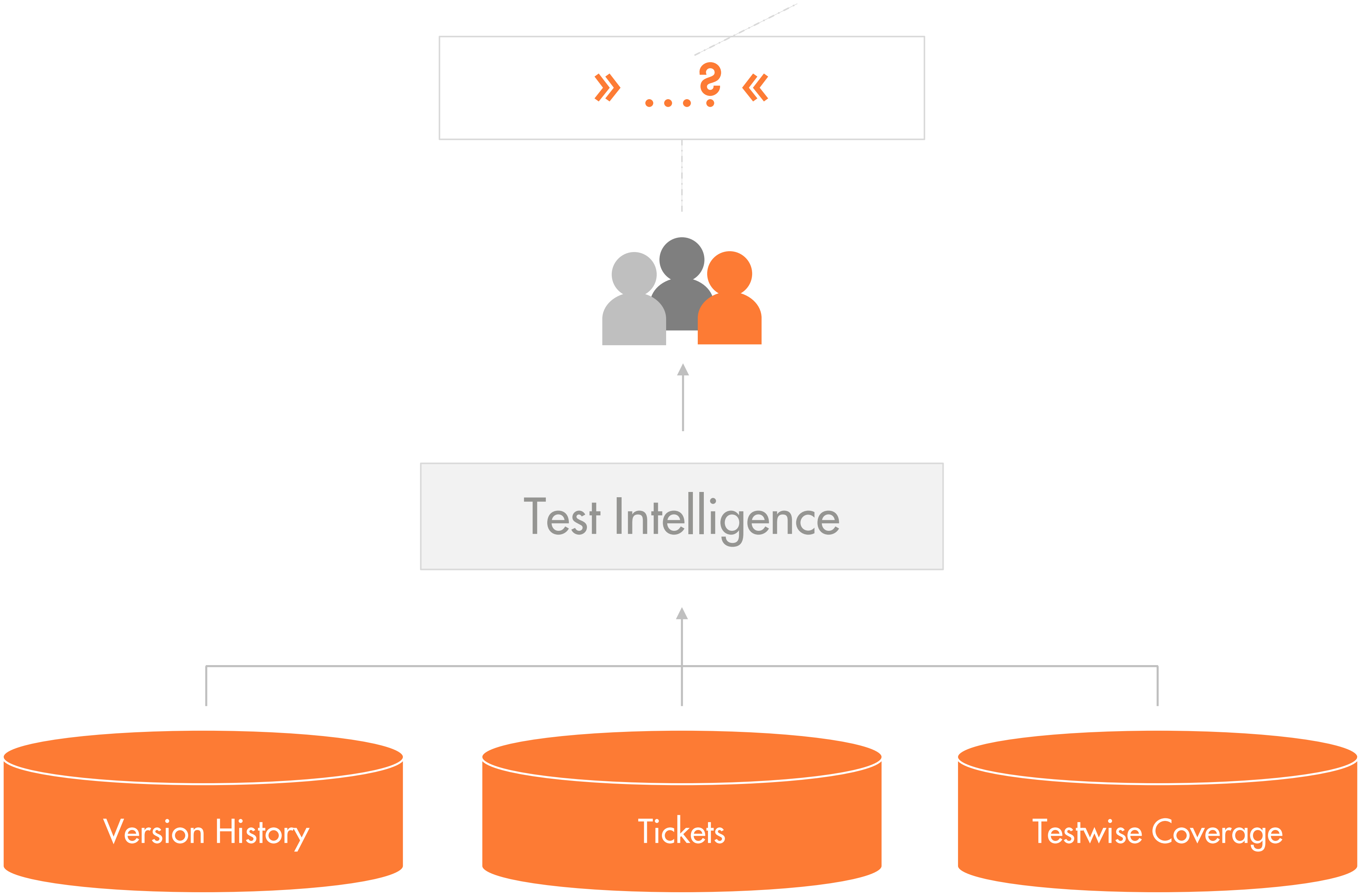
First incomplete feedback on checked in changes within 2h.

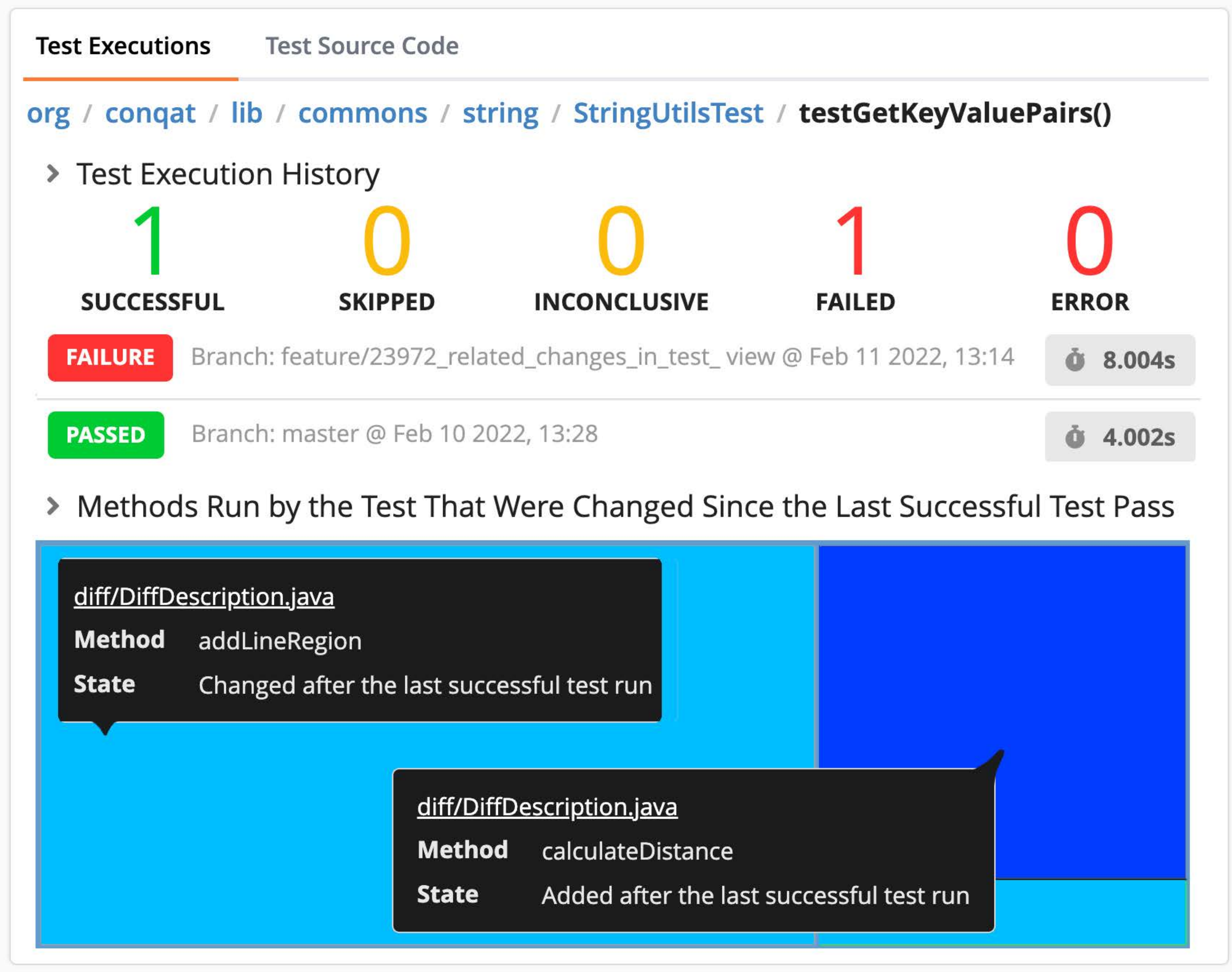


Still: nightly „Re-run all“

Per quarter year: update Pareto test list


» Where are untested changes? «
» Where are most bugs? «
» Which tests are most/least valueable now? «
Test Intelligence
Version History
Tickets
Testwise Coverage

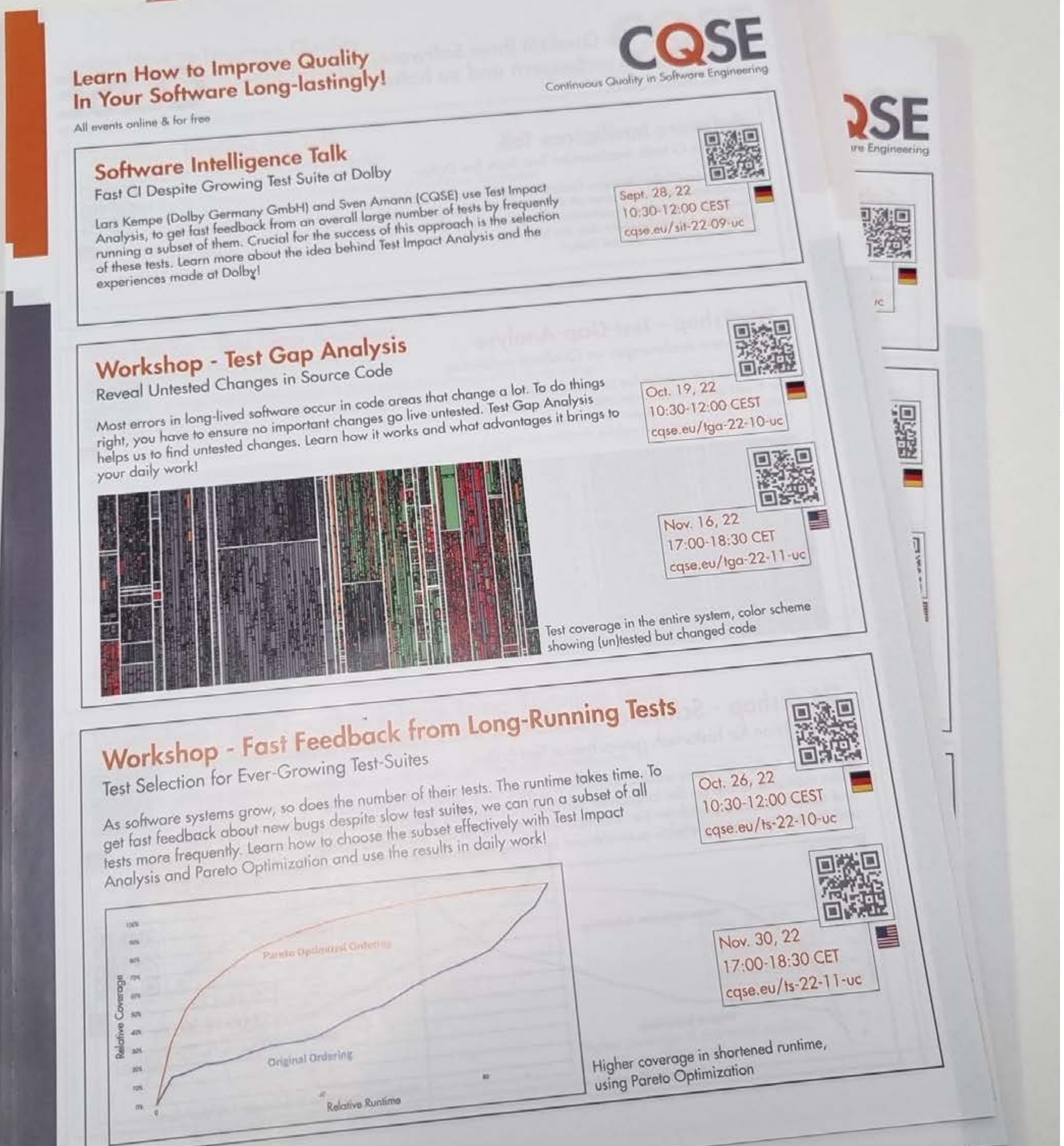

» ...? «
Test Intelligence
Version History
Tickets
Testwise Coverage


Test Executions
Test Source Code
org / conqat / lib / commons / string / StringUtilsTest / testGetKeyValuePairs()
Test Execution History
1
SUCCESSFUL
0
SKIPPED
0
INCONCLUSIVE
1
FAILED
0
ERROR
FAILURE
Branch: feature/23972_related_changes_in_test_ view @ Feb 11 2022, 13:14
8.004s
PASSED
Branch: master @ Feb 10 2022, 13:28
4.002s
Methods Run by the Test That Were Changed Since the Last Successful Test Pass
diff/DiffDescription.java
Method addLineRegion
State Changed after the last successful test run
diff/DiffDescription.java
Method calculateDistance
State Added after the last successful test run

Show test results from: **All**

Tests \ Specification Items		REQ DP-539	REQ DP-540	TST DP-547	TST DP-548
	Σ	1	0	1	1
tests/stack_tests.c/StackTests/Push	1				
tests/stack_tests.c/StackTests/Pop	1				
tests/stack_tests.c/StackTests/Peek	1				

Linux SKIPPED
Windows FAILURE



CQSE
Workshop

Test Gap Analysis

Reveal Untested Changes in Source Code

October 19
10:30-12:00 CEST
cqse.eu/tga-22-10-ut

November 16
17:00-18:30 CET
cqse.eu/tga-22-11-ut

CQSE
Workshop

Fast Feedback from Long-Running Tests

Test Selection for Ever-Growing Test Suites

October 26
10:30-12:00 CEST
cqse.eu/ts-22-10-ut

November 30
17:00-18:30 CET
cqse.eu/ts-22-11-ut

slides: cqse.eu/2022/ucaat

Happy to meet you during the breaks!

Jakob Rott · rott@cqse.eu · +49 172 186 0190

CQSE GmbH
Centa-Hafenbrädl-Straße 59
81249 Munich

CQSE