Gestern lief's doch noch

Muss ich heute wirklich schon wieder alles testen?

Forschungsergebnisse & eigene Erfahrungen mit Test-Impact-Analyse

Dr. Elmar Juergens

CQSE

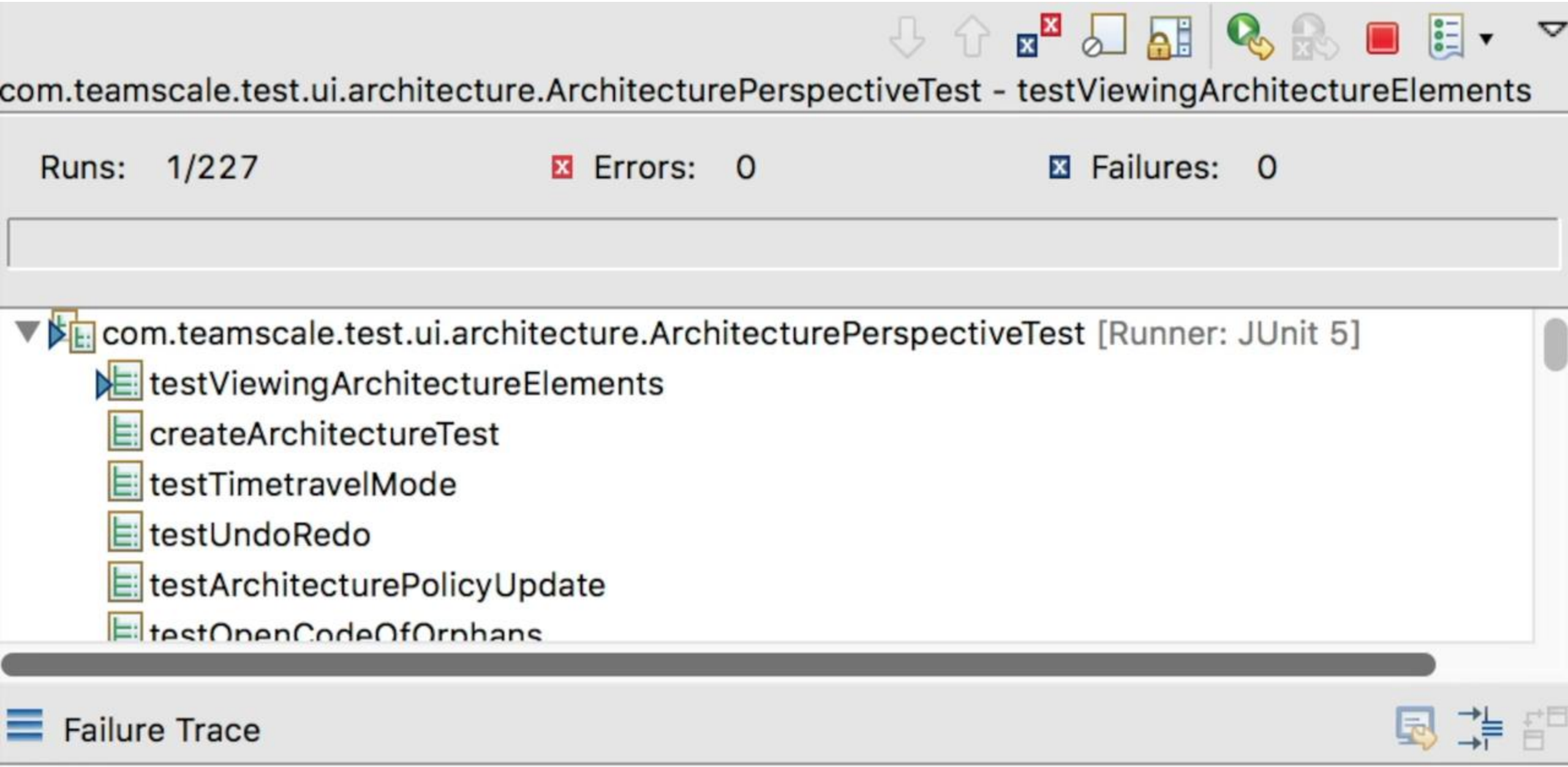
Über Mich

Forschung

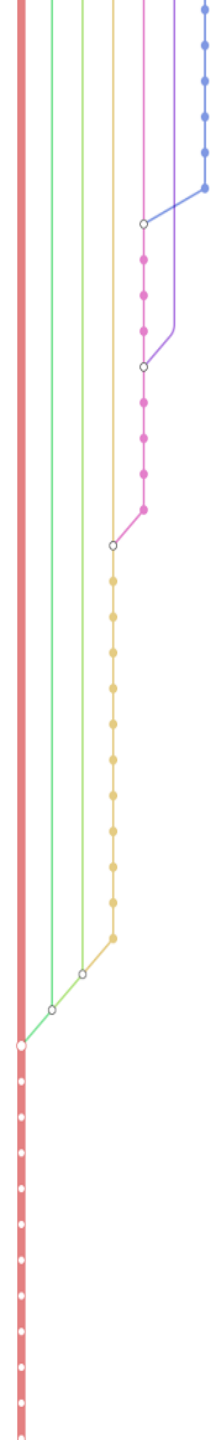
- Statische Anlyse, Test-Gap-Analyse, …
- PC Mitglied von MSR, ICPC, …

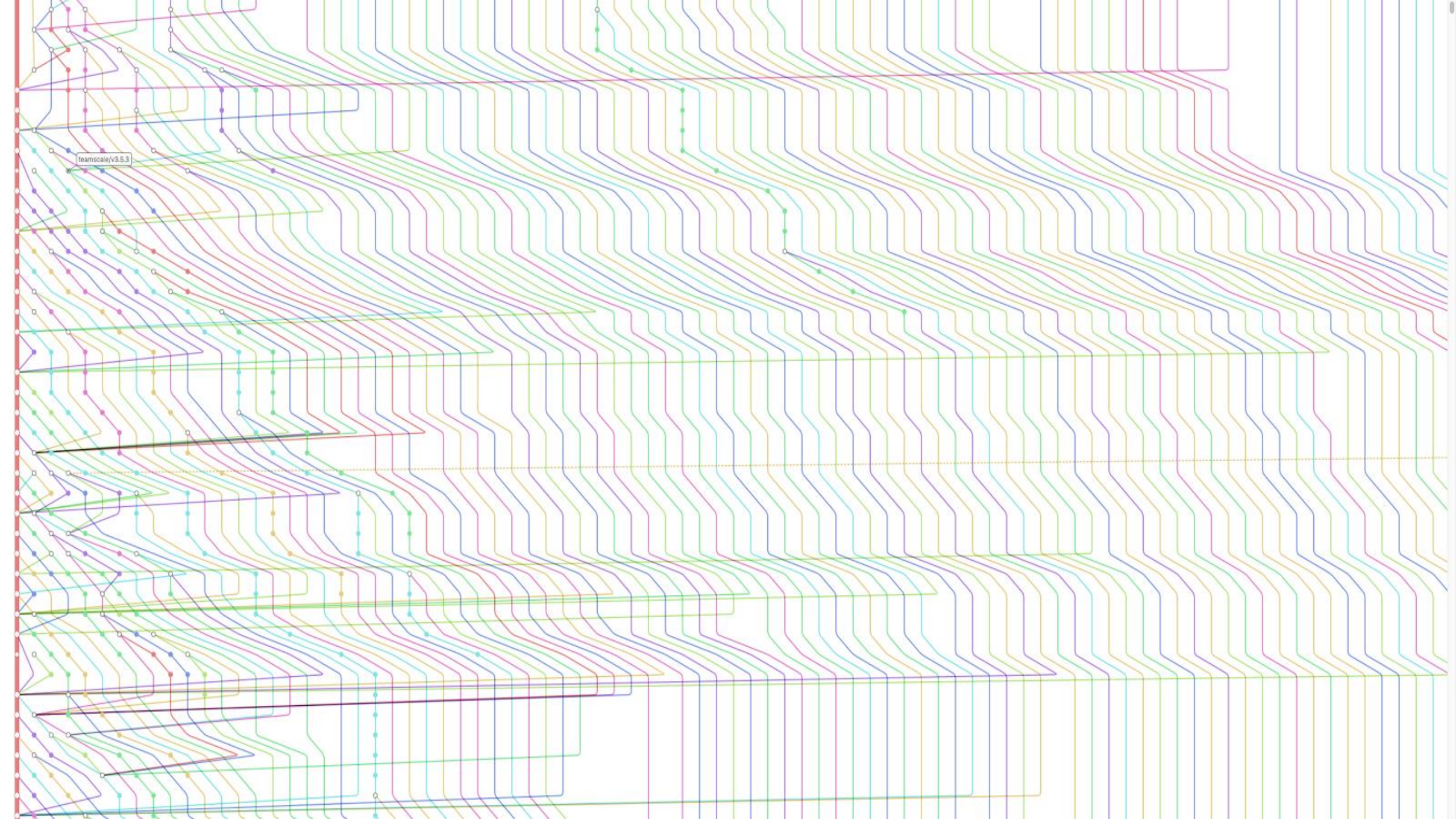
Beratung

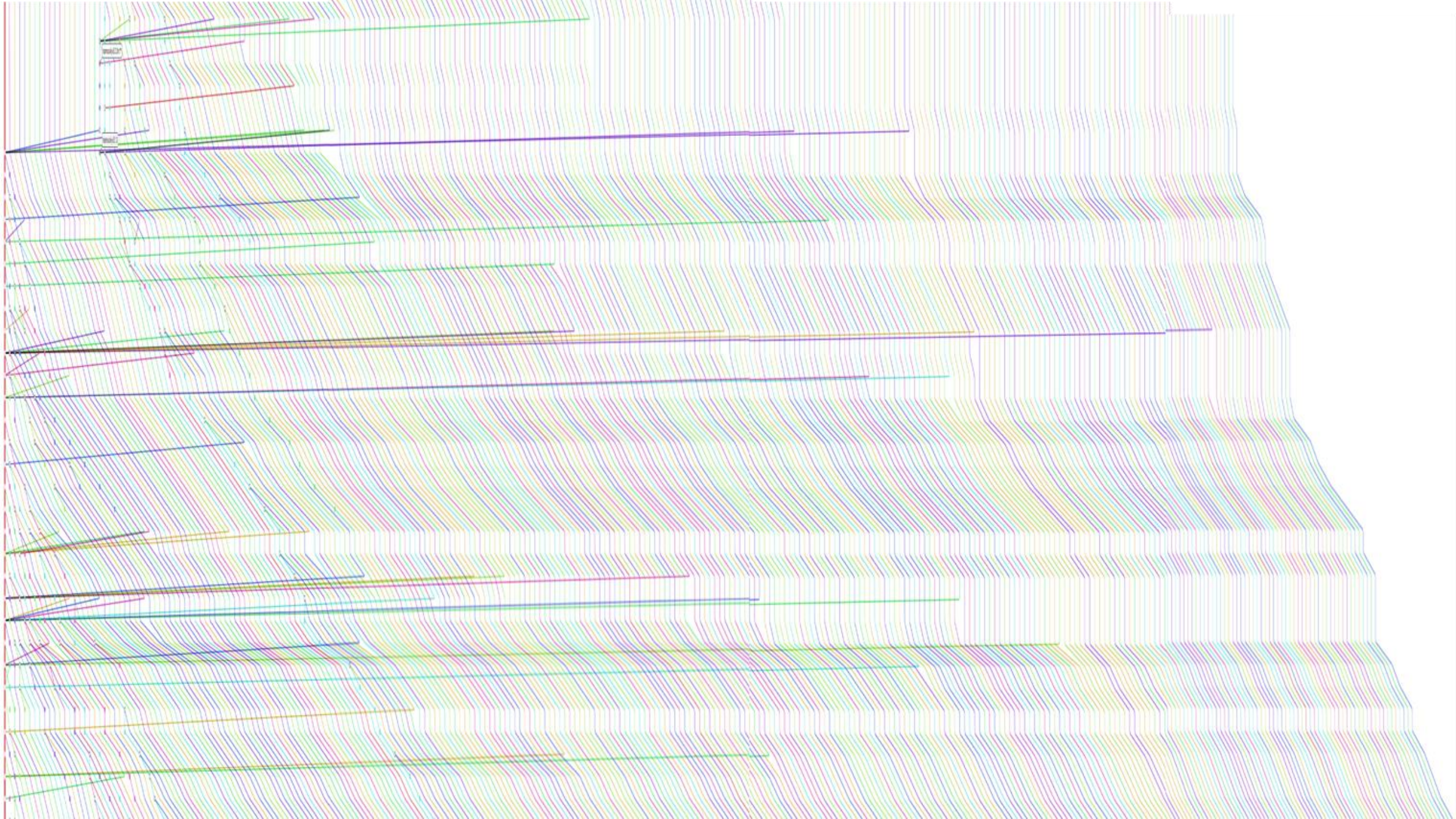
- Gründer und Mitglied der Geschäftsleitung
- Qualitäts-Bewertung & Qualitäts-Controlling

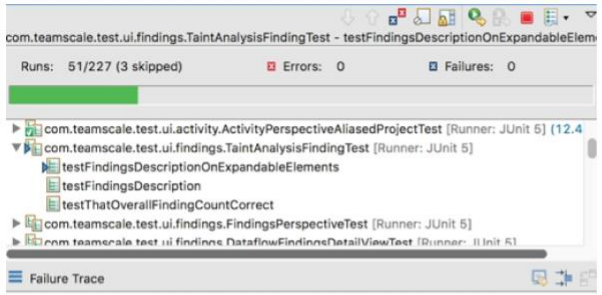

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testViewingArchitectureElements
Runs: 1/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements
createArchitectureTest
testTimetravelMode
testUndoRedo
testArchitecturePolicyUpdate
Failure Trace

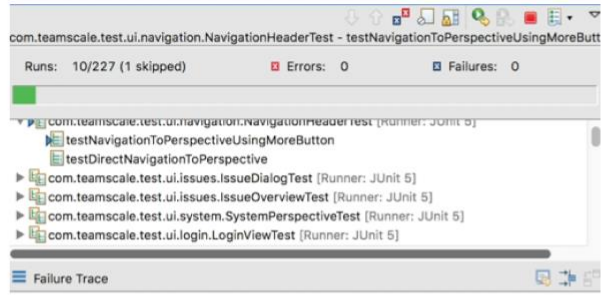
https://www.technica-engineering.de/produkte/bts-body-electronic-test-system/

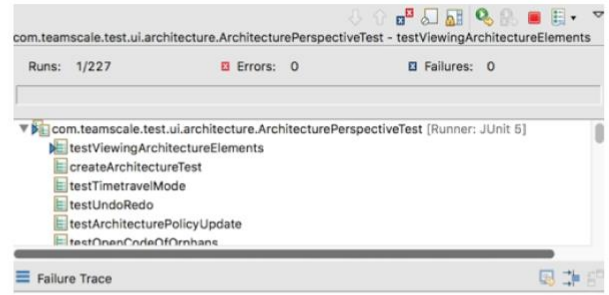


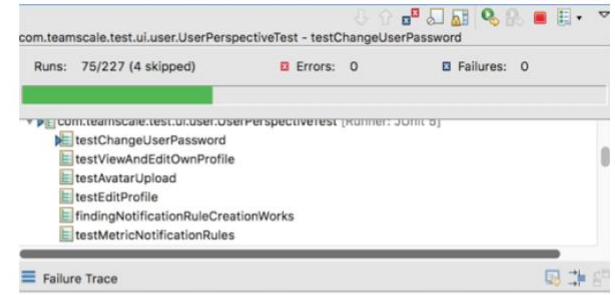

teamscale/v3.5.3

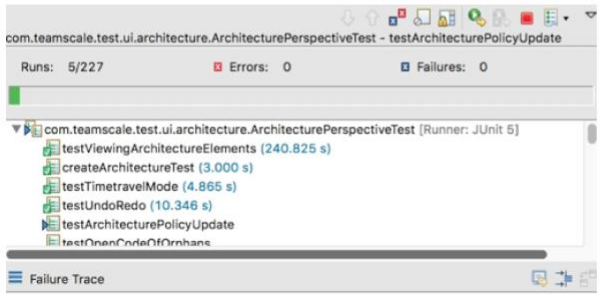


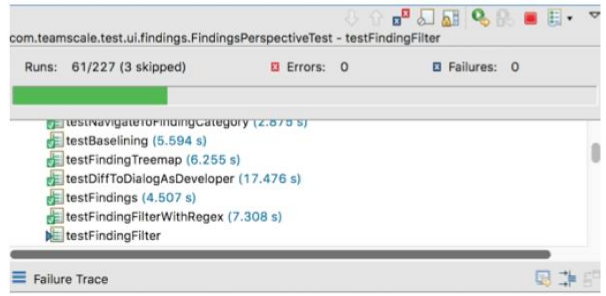

com.teamscale.test.ui.findings.TaintAnalysisFindingTest - testFindingsDescriptionOnExpandableElem
Runs: 51/227 (3 skipped)
Errors: 0
Failures: 0
com.teamscale.test.ui.activity.ActivityPerspectiveAliasedProjectTest [Runner: JUnit 5] (12.4
com.teamscale.test.ui.findings.TaintAnalysisFindingTest [Runner: JUnit 5]
testFindingsDescriptionOnExpandableElements
testFindingsDescription
testThatOverallFindingCountCorrect
com.teamscale.test.ui.findings.FindingsPerspectiveTest [Runner: JUnit 5]
Failure Trace

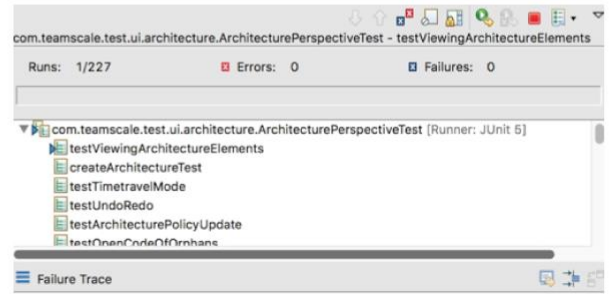

com.teamscale.test.ui.navigation.NavigationHeaderTest - testNavigationToPerspectiveUsingMoreButt
Runs: 10/227 (1 skipped)
Errors: 0
Failures: 0
testNavigationToPerspectiveUsingMoreButton
testDirectNavigationToPerspective
com.teamscale.test.ui.issues.IssueDialogTest [Runner: JUnit 5]
com.teamscale.test.ui.issues.IssueOverviewTest [Runner: JUnit 5]
com.teamscale.test.ui.system.SystemPerspectiveTest [Runner: JUnit 5]
com.teamscale.test.ui.login.LoginViewTest [Runner: JUnit 5]
Failure Trace

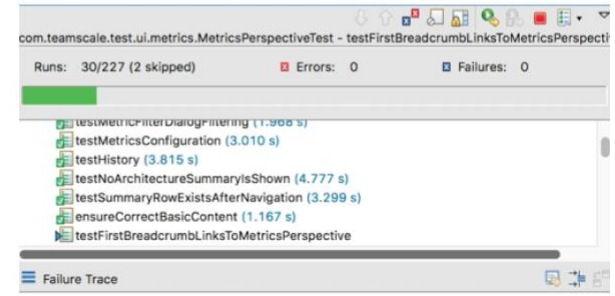

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testViewingArchitectureElements
Runs: 1/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements
createArchitectureTest
testTimetravelMode
testUndoRedo
testArchitecturePolicyUpdate
Failure Trace

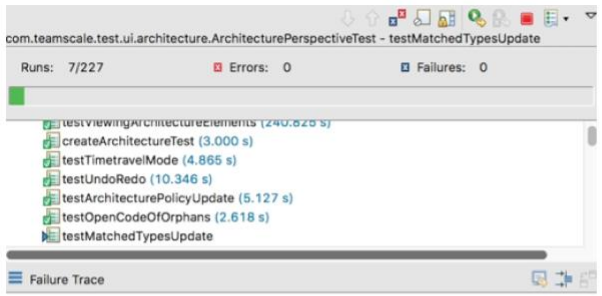

com.teamscale.test.ui.user.UserPerspectiveTest - testChangeUserPassword
Runs: 75/227 (4 skipped)
Errors: 0
Failures: 0
testChangeUserPassword
testViewAndEditOwnProfile
testAvatarUpload
testEditProfile
findingNotificationRuleCreationWorks
testMetricNotificationRules
Failure Trace

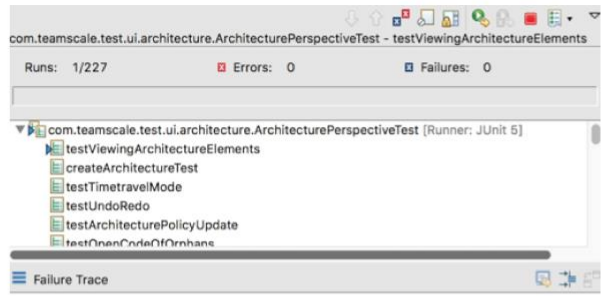

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testArchitecturePolicyUpdate
Runs: 5/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements (240.825 s)
createArchitectureTest (3.000 s)
testTimetravelMode (4.865 s)
testUndoRedo (10.346 s)
testArchitecturePolicyUpdate
Failure Trace

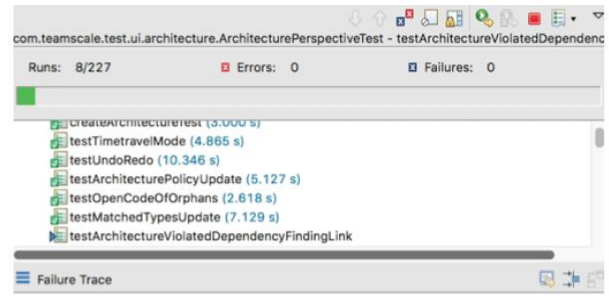

com.teamscale.test.ui.findings.FindingsPerspectiveTest - testFindingFilter
Runs: 61/227 (3 skipped)
Errors: 0
Failures: 0
testBaselining (5.594 s)
testFindingTreemap (6.255 s)
testDiffToDialogAsDeveloper (17.476 s)
testFindings (4.507 s)
testFindingFilterWithRegex (7.308 s)
testFindingFilter
Failure Trace

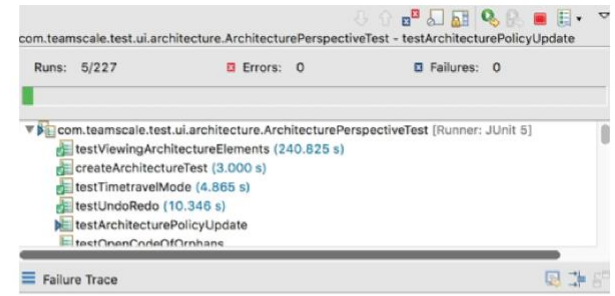

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testViewingArchitectureElements
Runs: 1/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements
createArchitectureTest
testTimetravelMode
testUndoRedo
testArchitecturePolicyUpdate
Failure Trace

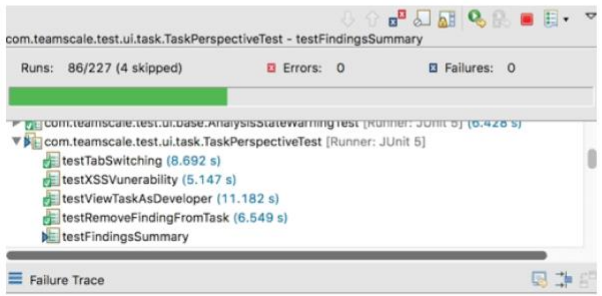

com.teamscale.test.ui.metrics.MetricsPerspectiveTest - testFirstBreadcrumbLinksToMetricsPerspecti
Runs: 30/227 (2 skipped)
Errors: 0
Failures: 0
testMetricsConfiguration (3.010 s)
testHistory (3.815 s)
testNoArchitectureSummaryIsShown (4.777 s)
testSummaryRowExistsAfterNavigation (3.299 s)
ensureCorrectBasicContent (1.167 s)
testFirstBreadcrumbLinksToMetricsPerspective
Failure Trace

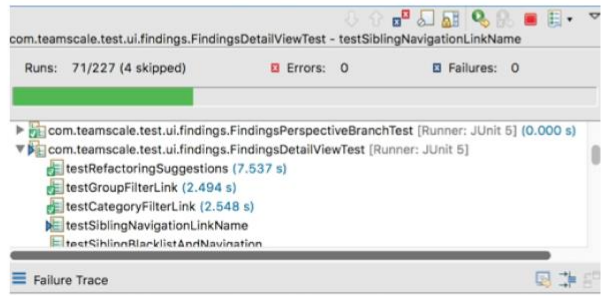

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testMatchedTypesUpdate
Runs: 7/227
Errors: 0
Failures: 0
createArchitectureTest (3.000 s)
testTimetravelMode (4.865 s)
testUndoRedo (10.346 s)
testArchitecturePolicyUpdate (5.127 s)
testOpenCodeOfOrphans (2.618 s)
testMatchedTypesUpdate
Failure Trace

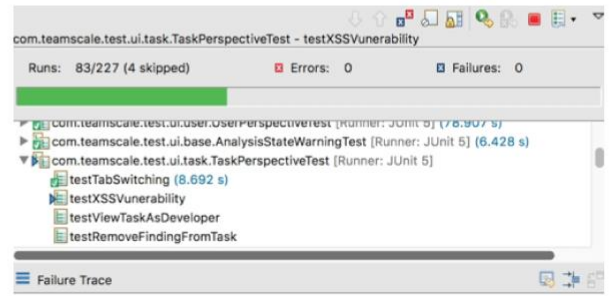

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testViewingArchitectureElements
Runs: 1/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements
createArchitectureTest
testTimetravelMode
testUndoRedo
testArchitecturePolicyUpdate
Failure Trace

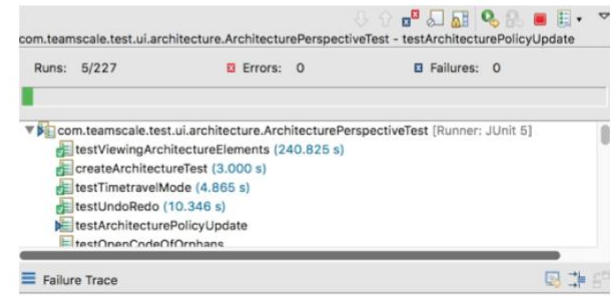

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testArchitectureViolatedDependenc
Runs: 8/227
Errors: 0
Failures: 0
testTimetravelMode (4.865 s)
testUndoRedo (10.346 s)
testArchitecturePolicyUpdate (5.127 s)
testOpenCodeOfOrphans (2.618 s)
testMatchedTypesUpdate (7.129 s)
testArchitectureViolatedDependencyFindingLink
Failure Trace

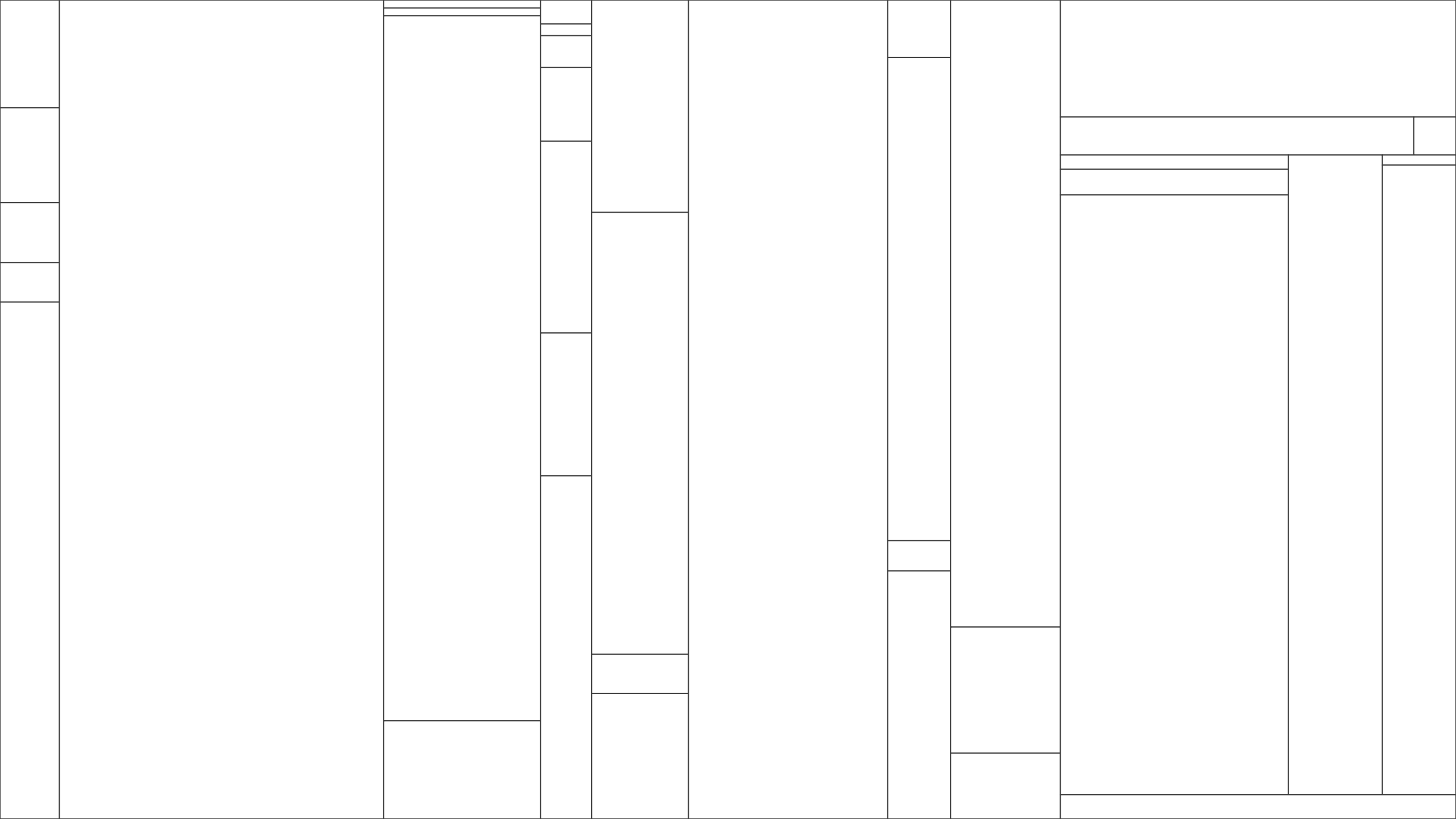

com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testArchitecturePolicyUpdate
Runs: 5/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements (240.825 s)
createArchitectureTest (3.000 s)
testTimetravelMode (4.865 s)
testUndoRedo (10.346 s)
testArchitecturePolicyUpdate
Failure Trace

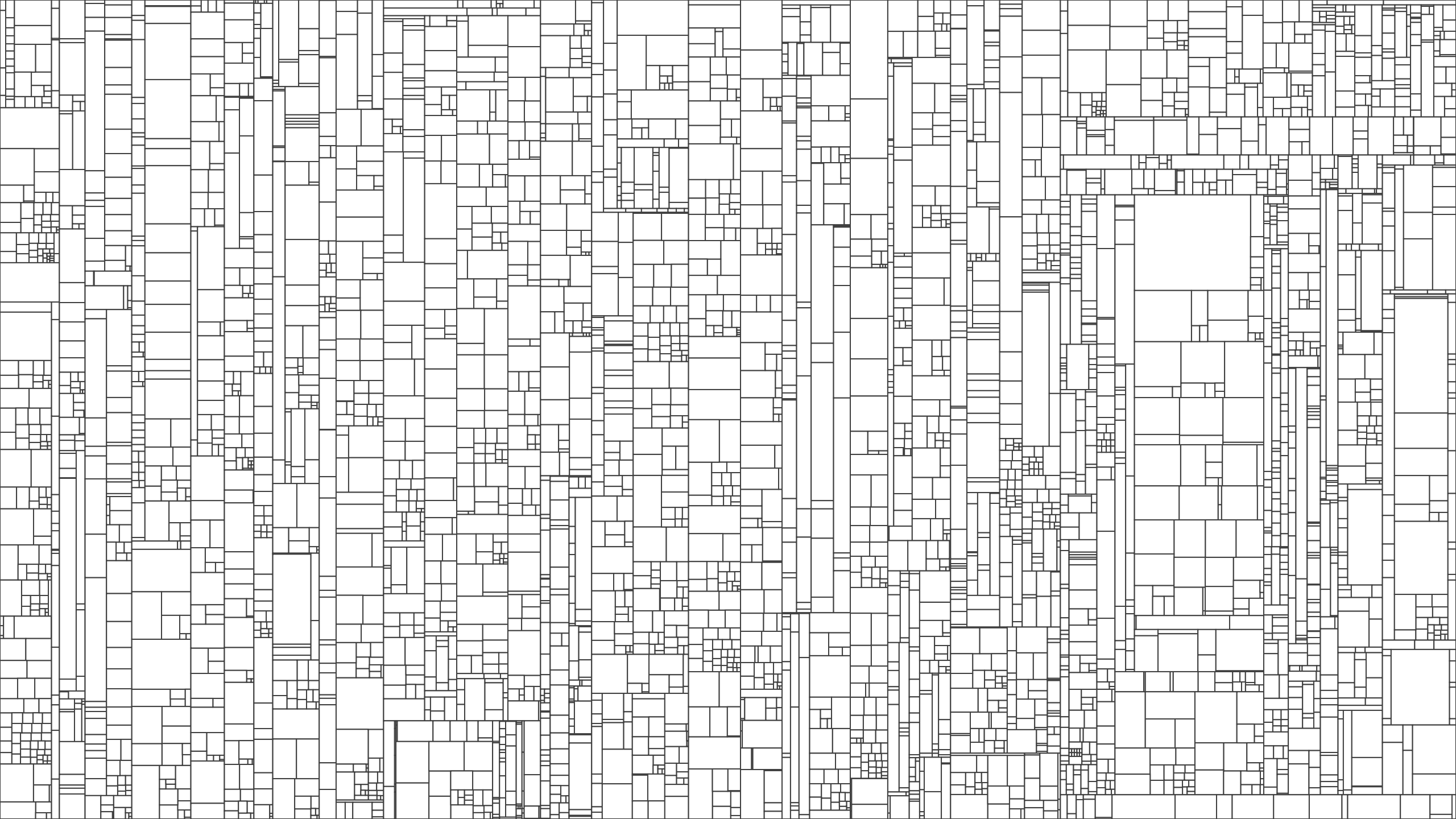

com.teamscale.test.ui.task.TaskPerspectiveTest - testFindingsSummary
Runs: 86/227 (4 skipped)
Errors: 0
Failures: 0
com.teamscale.test.ui.task.TaskPerspectiveTest [Runner: JUnit 5]
testTabSwitching (8.692 s)
testXSSVunerability (5.147 s)
testViewTaskAsDeveloper (11.182 s)
testRemoveFindingFromTask (6.549 s)
testFindingsSummary
Failure Trace

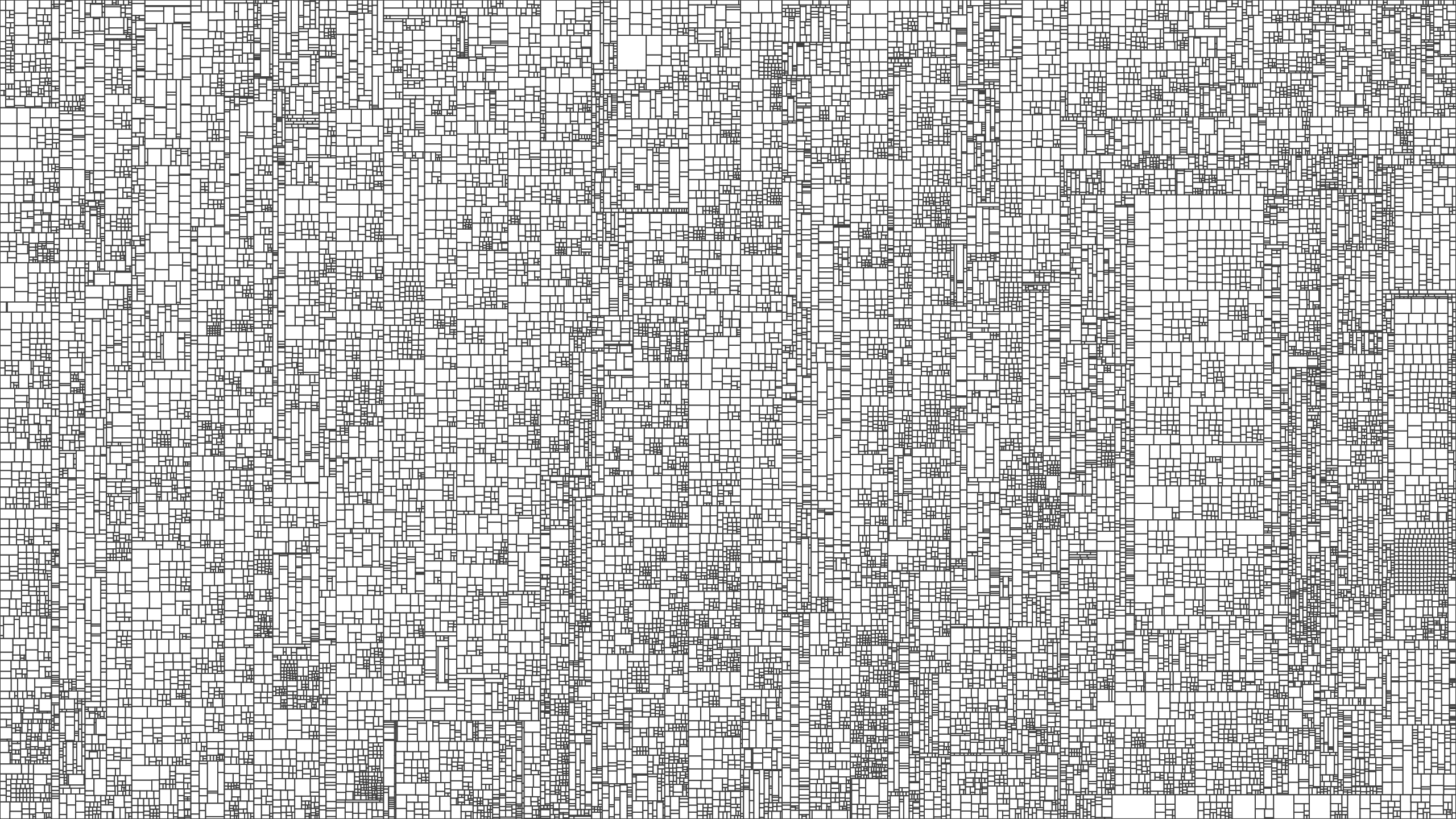

com.teamscale.test.ui.findings.FindingsDetailViewTest - testSiblingNavigationLinkName
Runs: 71/227 (4 skipped)
Errors: 0
Failures: 0
com.teamscale.test.ui.findings.FindingsPerspectiveBranchTest [Runner: JUnit 5] (0.000 s)
com.teamscale.test.ui.findings.FindingsDetailViewTest [Runner: JUnit 5]
testRefactoringSuggestions (7.537 s)
testGroupFilterLink (2.494 s)
testCategoryFilterLink (2.548 s)
testSiblingNavigationLinkName
Failure Trace


com.teamscale.test.ui.task.TaskPerspectiveTest - testXSSVunerability
Runs: 83/227 (4 skipped)
Errors: 0
Failures: 0
com.teamscale.test.ui.base.AnalysisStateWarningTest [Runner: JUnit 5] (6.428 s)
com.teamscale.test.ui.task.TaskPerspectiveTest [Runner: JUnit 5]
testTabSwitching (8.692 s)
testXSSVunerability
testViewTaskAsDeveloper
testRemoveFindingFromTask
Failure Trace


com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest - testArchitecturePolicyUpdate
Runs: 5/227
Errors: 0
Failures: 0
com.teamscale.test.ui.architecture.ArchitecturePerspectiveTest [Runner: JUnit 5]
testViewingArchitectureElements (240.825 s)
createArchitectureTest (3.000 s)
testTimetravelMode (4.865 s)
testUndoRedo (10.346 s)
testArchitecturePolicyUpdate
Failure Trace







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

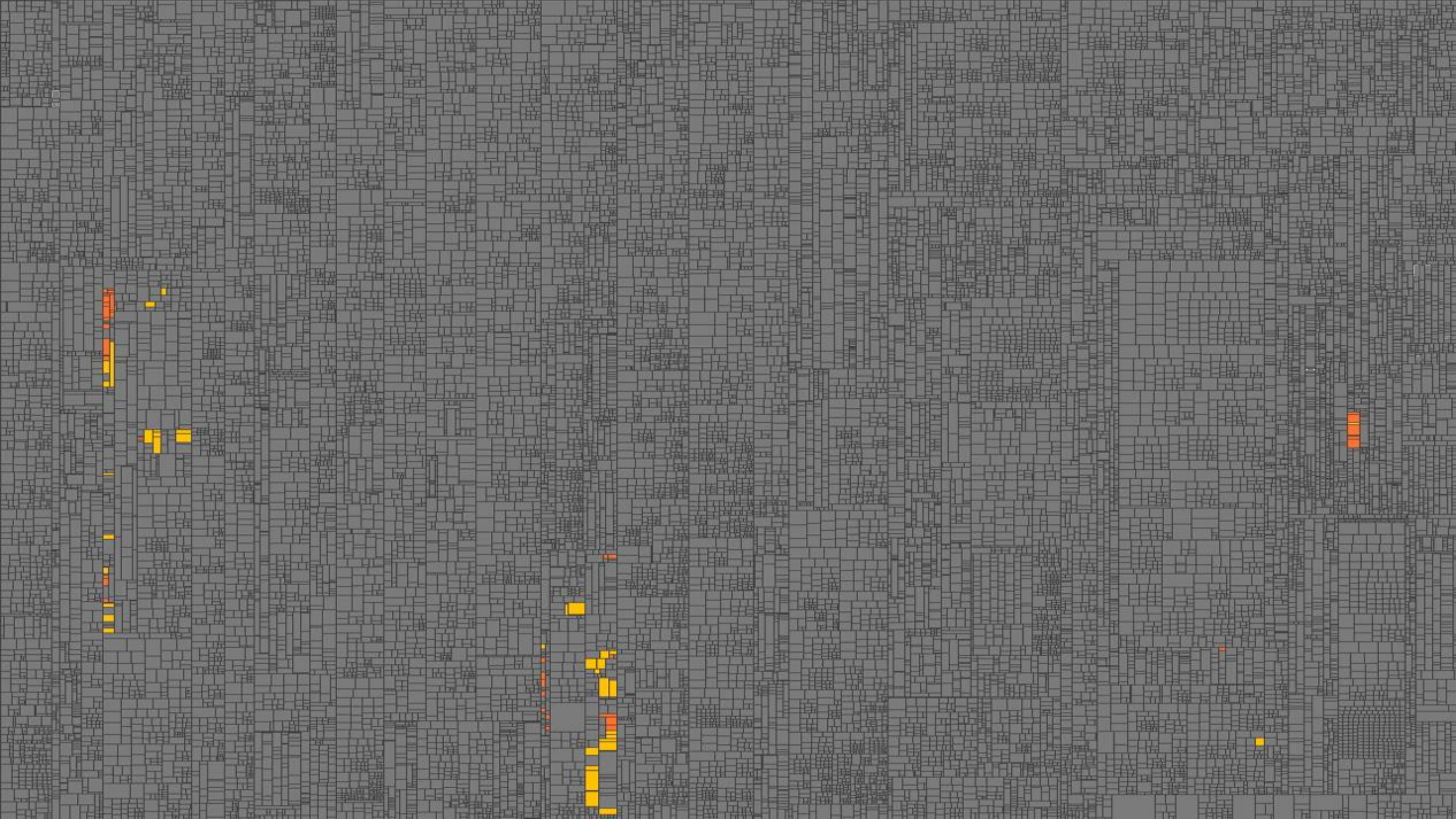
Time Tracking

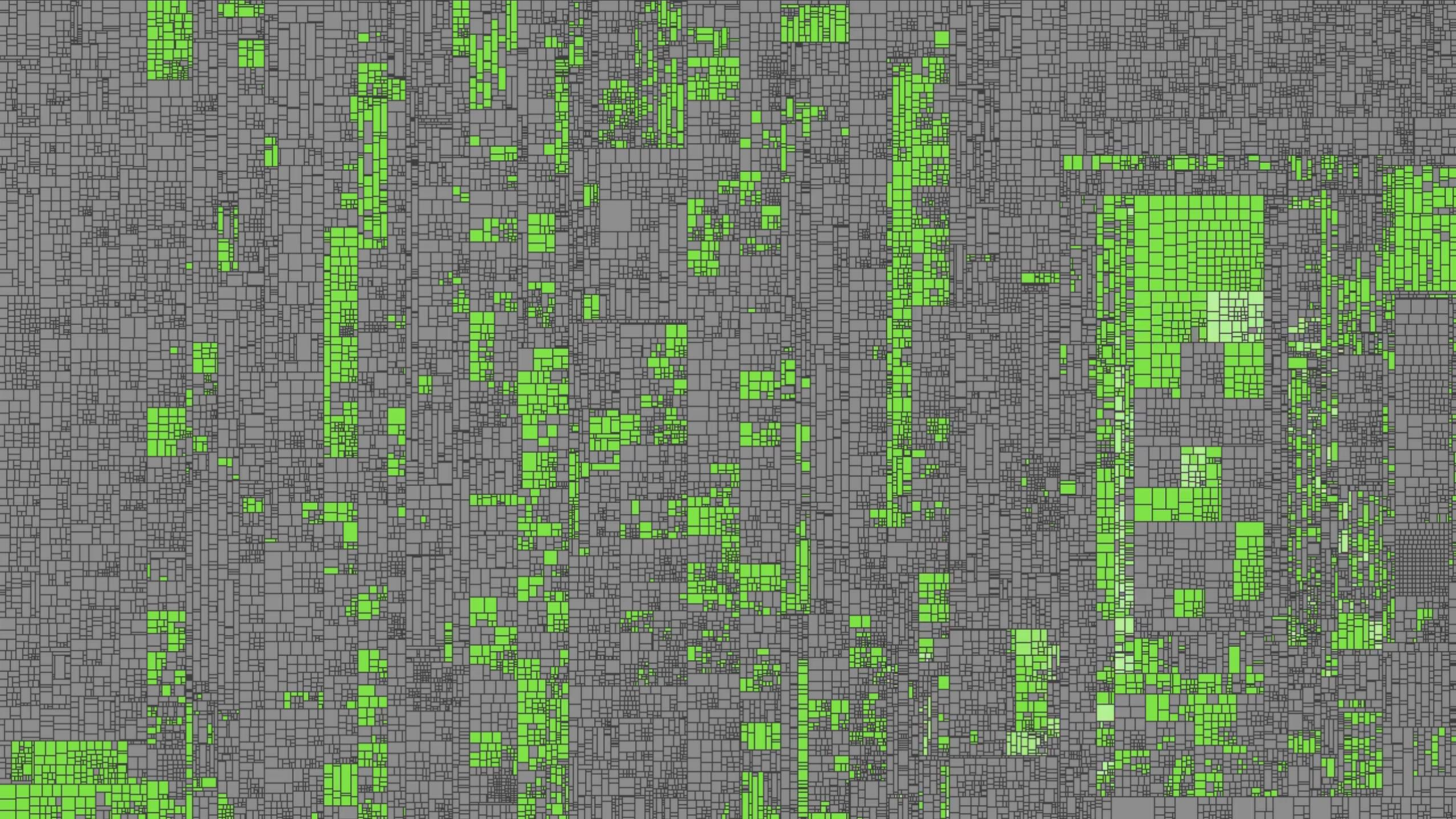
Estimated:	Not Specified
Remaining:	0m
Logged:	4d 1h 57m

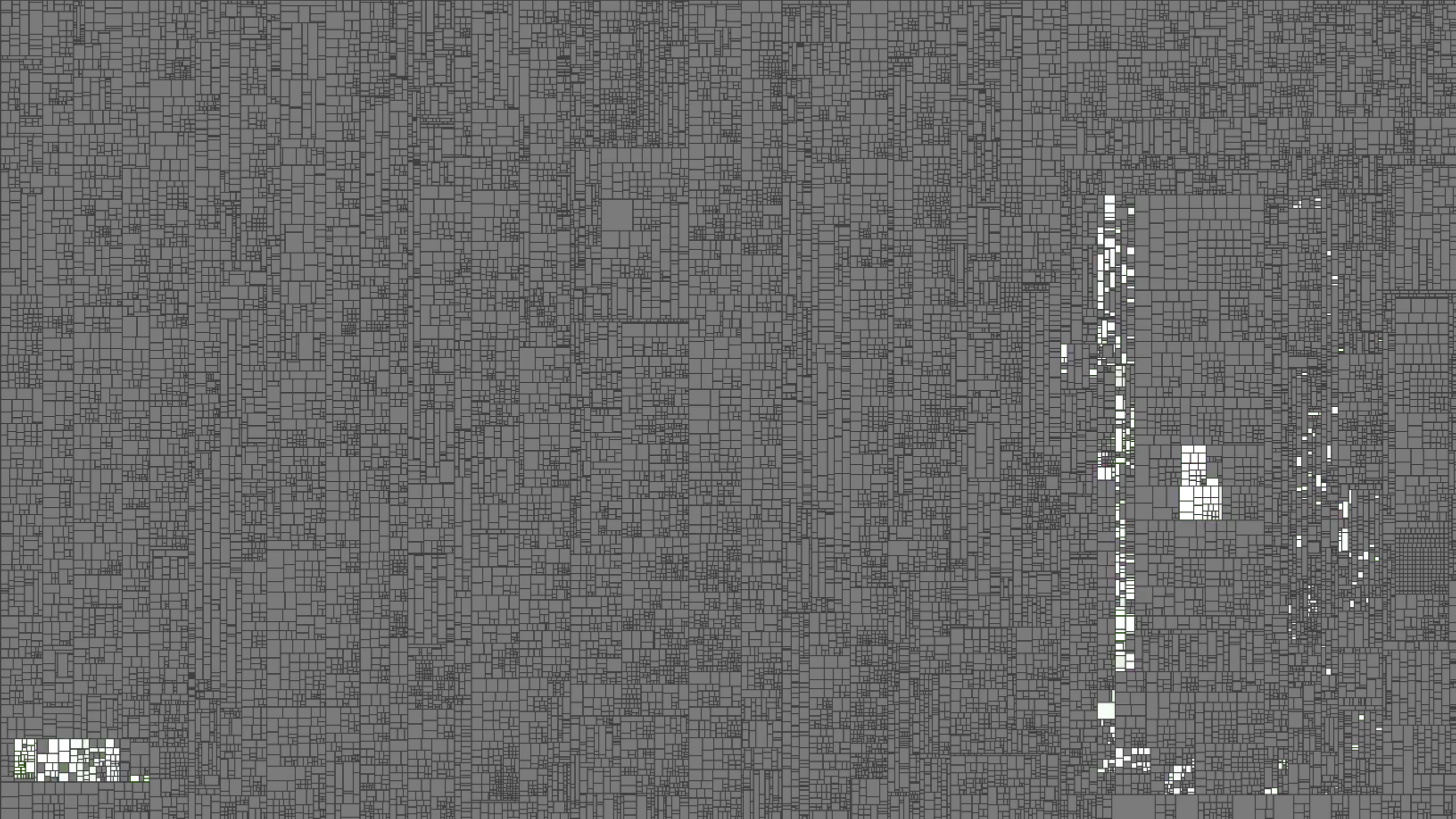
Agile

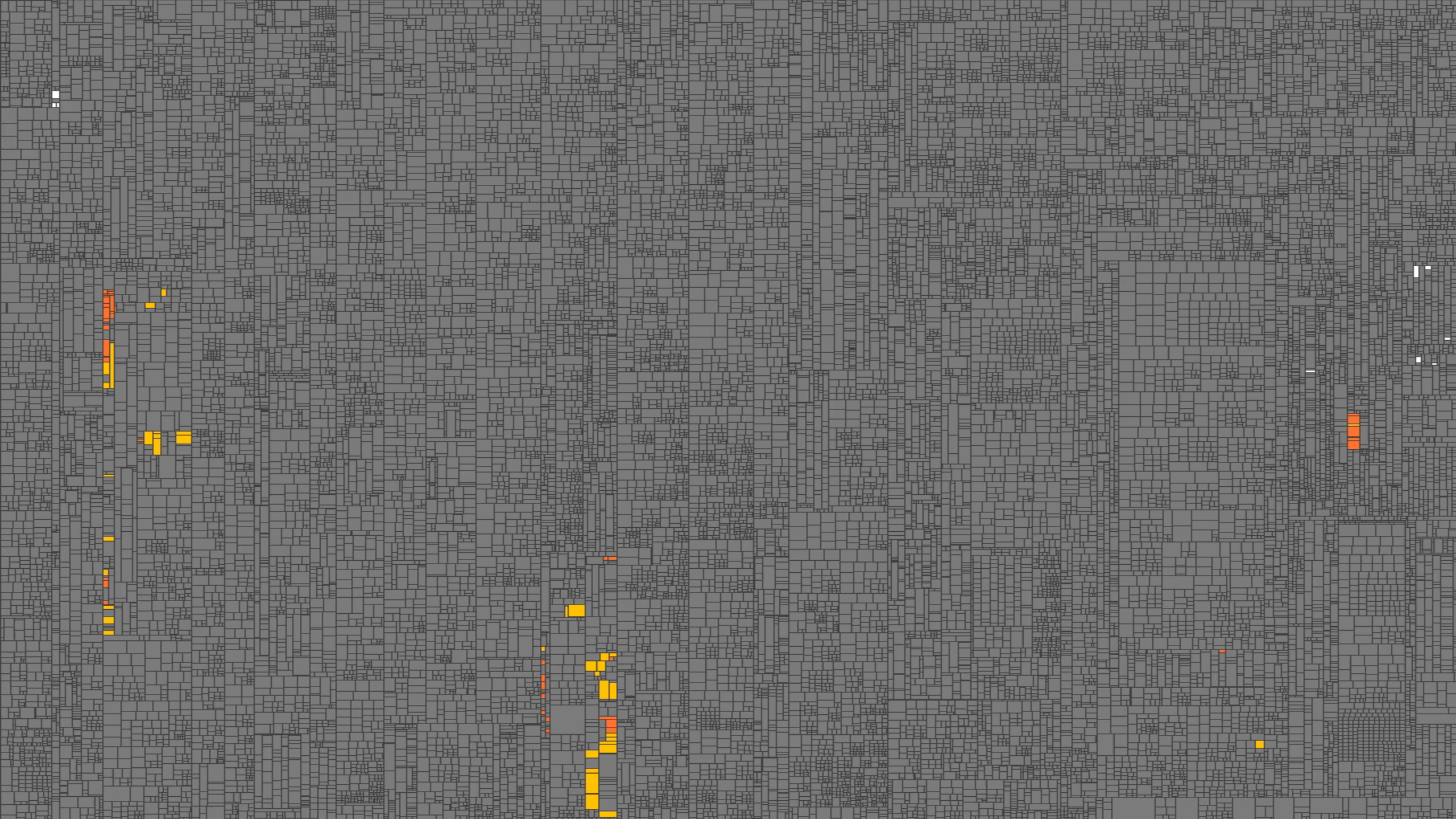
View on Board

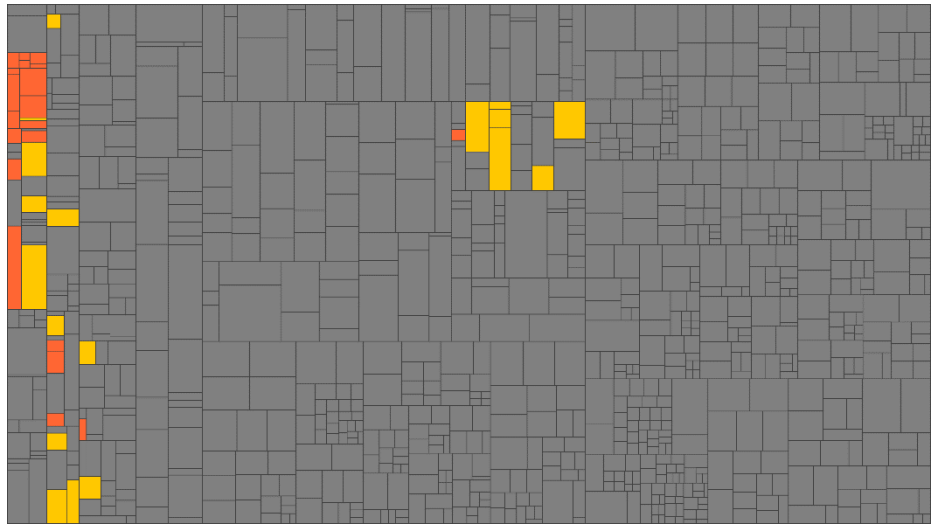
Gitlab CI

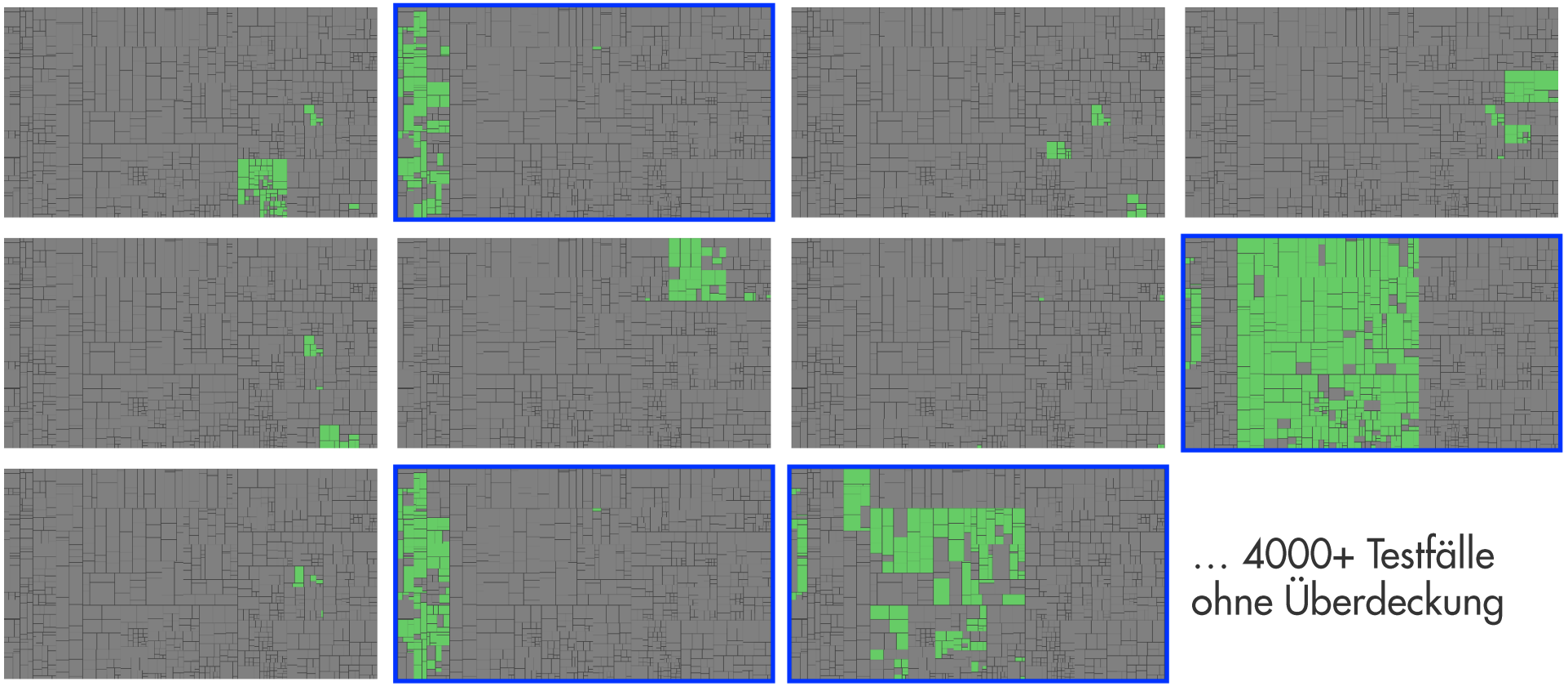




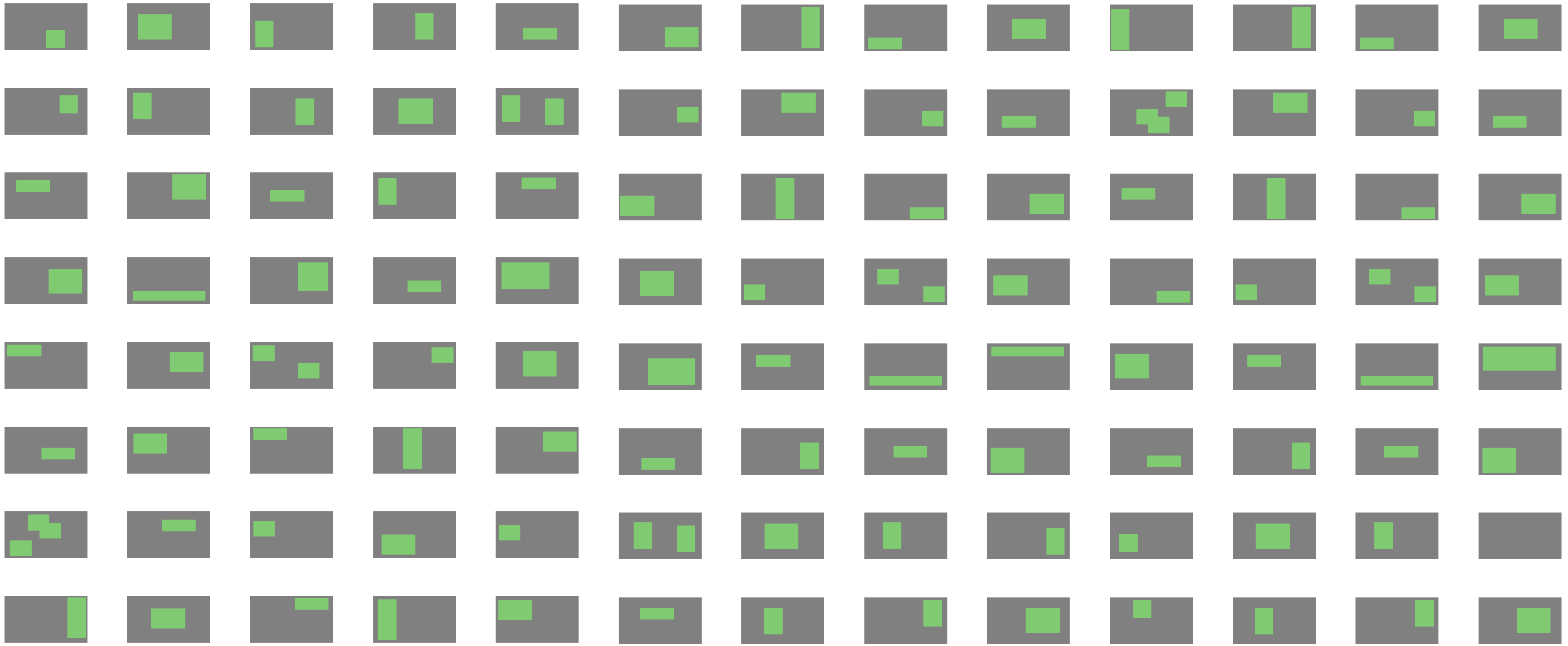




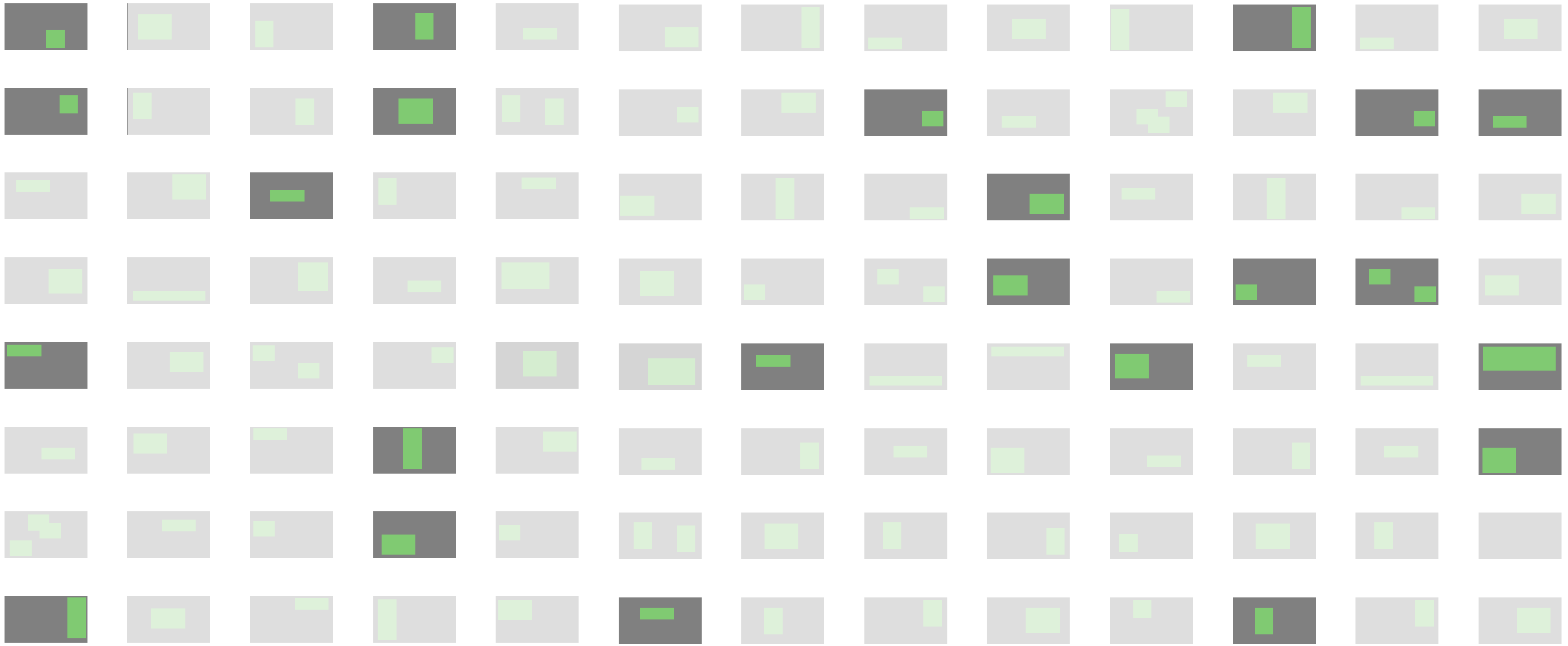




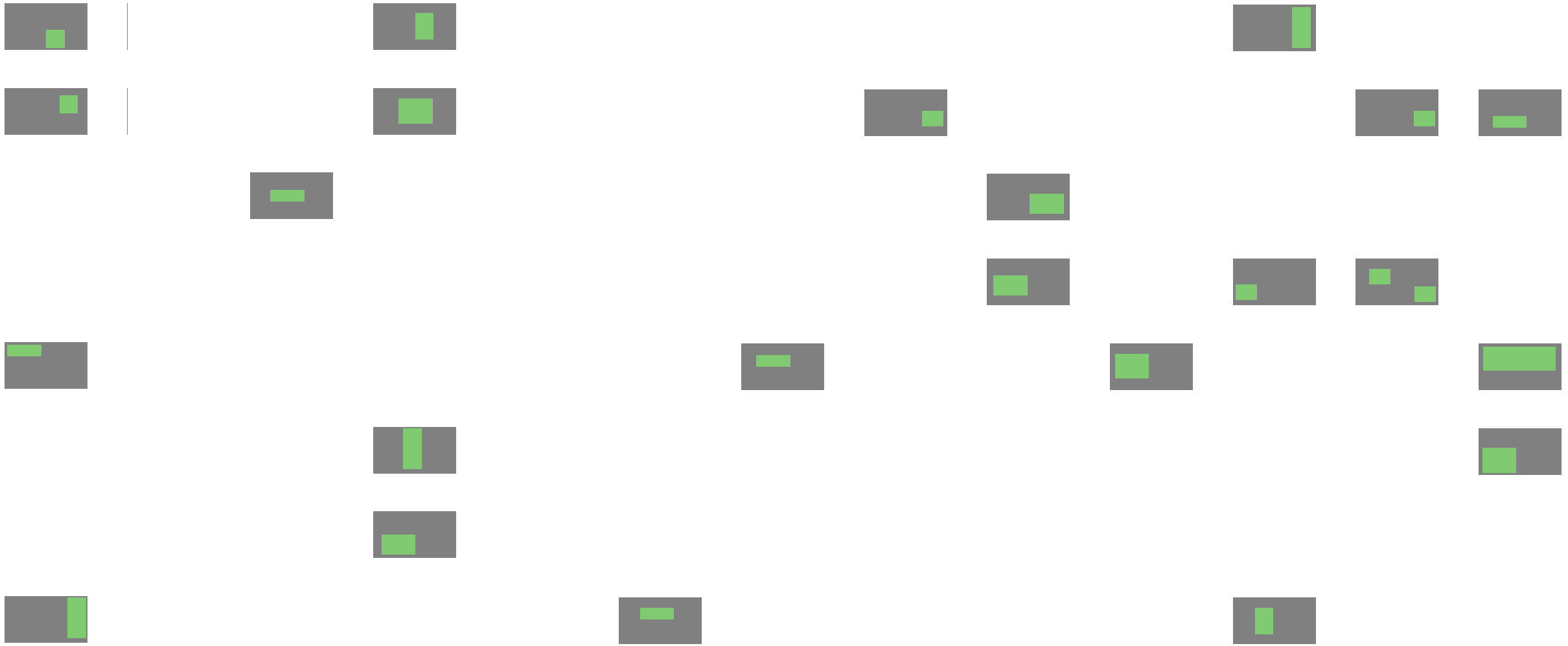
… 4000+ Testfälle
ohne Überdeckung



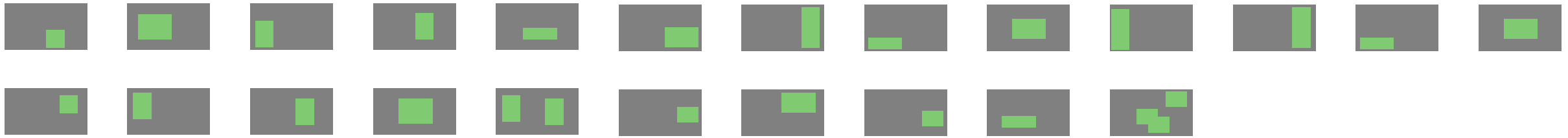
Schritt 1: Selektion betroffener Testfälle



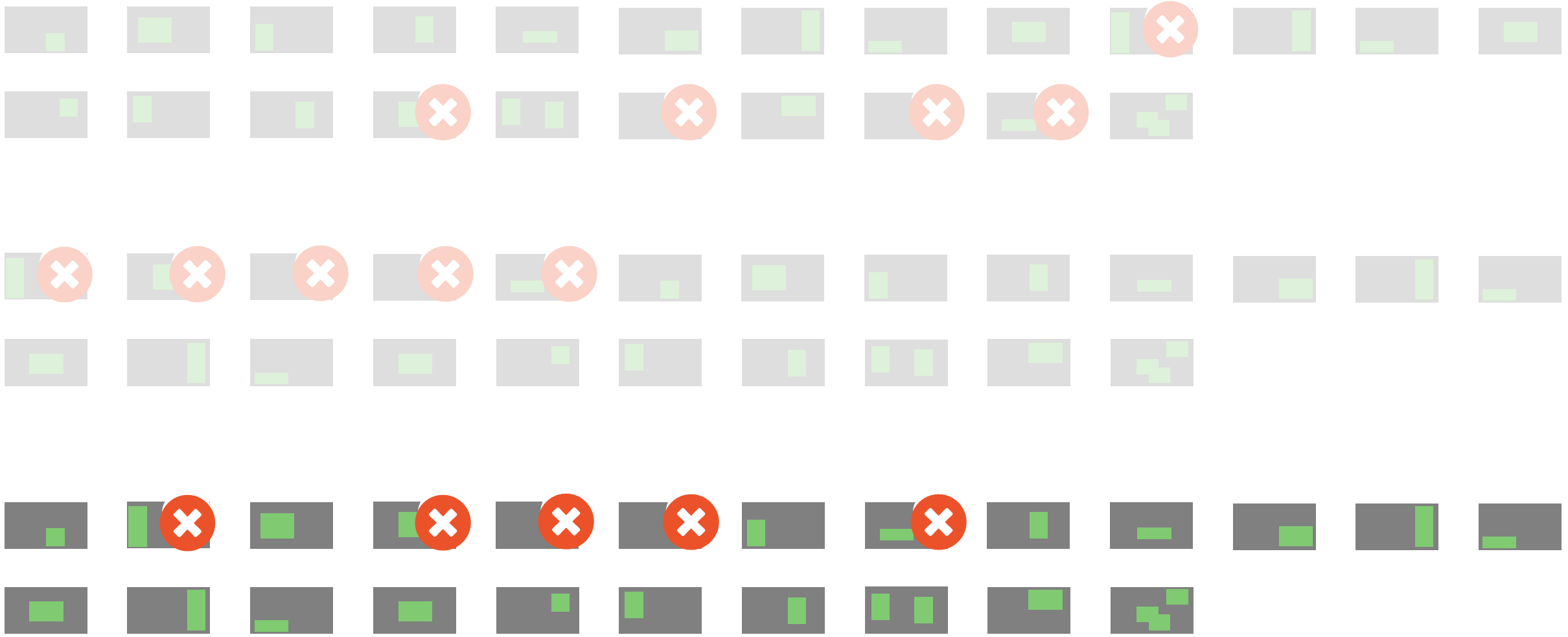
Schritt 1: Selektion betroffener Testfälle

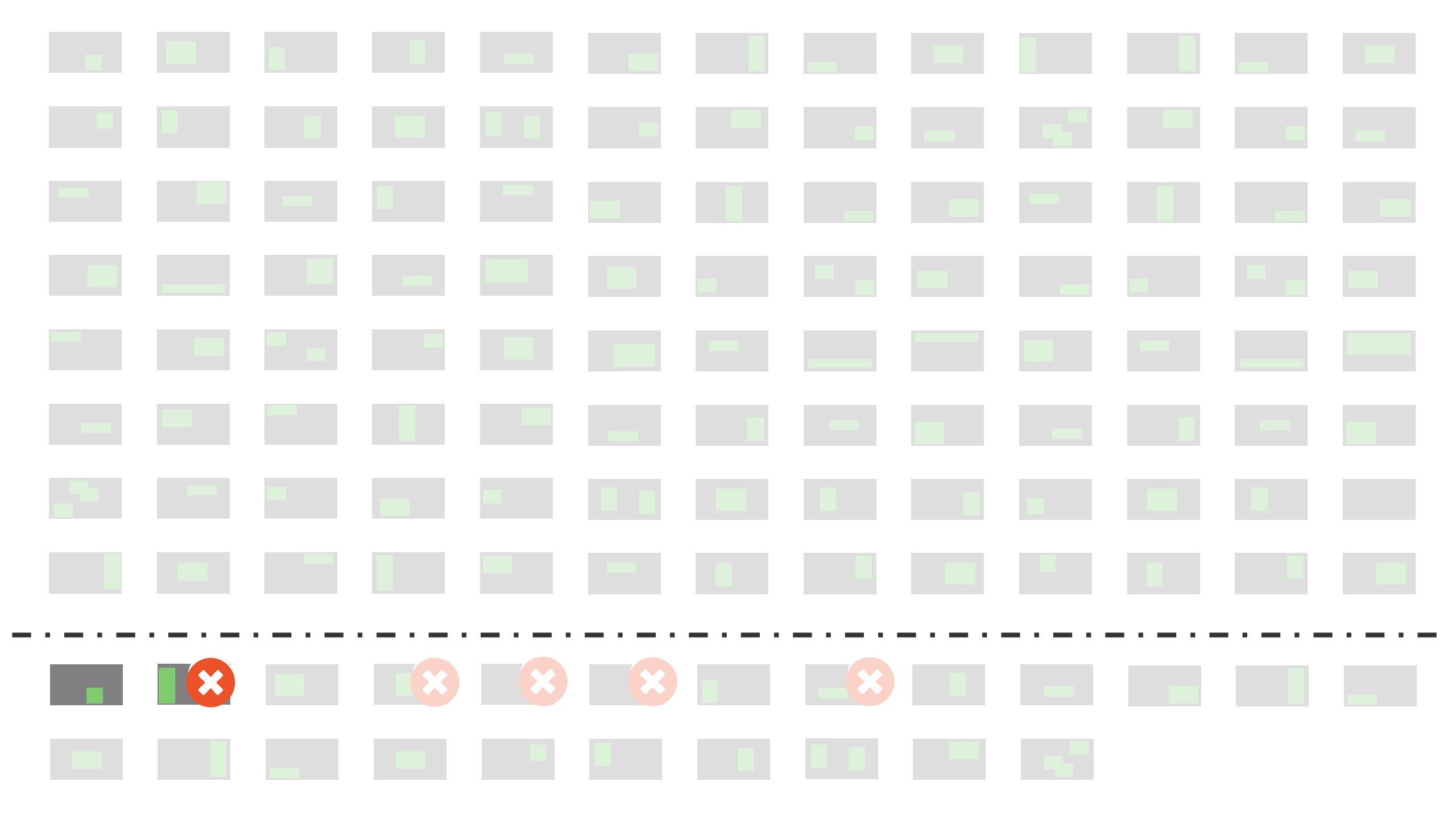


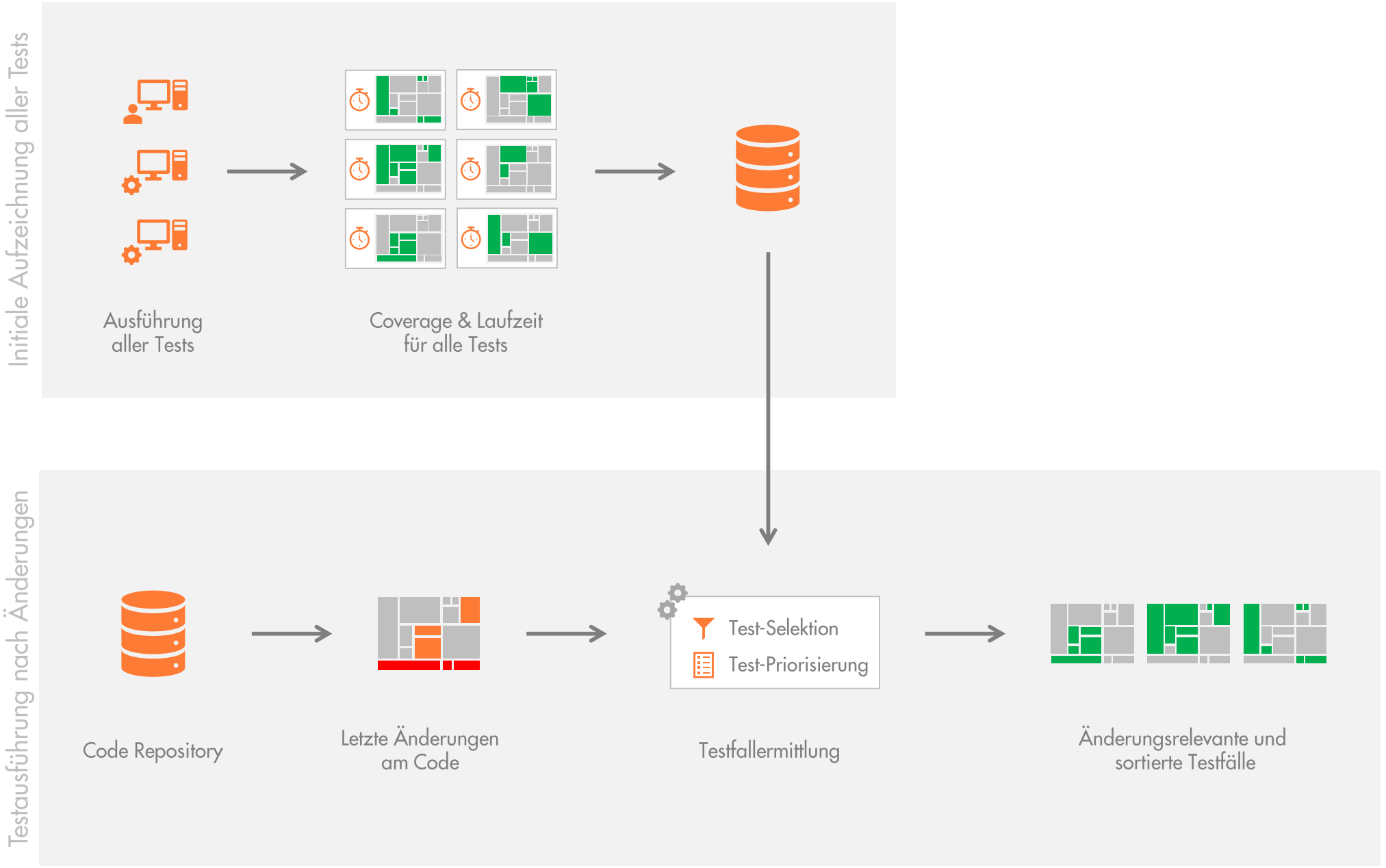
Schritt 1: Selektion betroffener Testfälle



Schritt 2: Priorisierung selektierter Testfälle






Initiale Aufzeichnung aller Tests
Ausführung aller Tests
Coverage & Laufzeit für alle Tests
Testausführung nach Änderungen
Code Repository
Letzte Änderungen am Code
Test-Selektion
Test-Priorisierung
Testfallermittlung
Änderungsrelevante und sortierte Testfälle

Grenzen von Test-Impact-Analyse

- Manche Änderungen betreffen alle Tests
  - Änderungen von Konfigurationsinformation
  - Änderungen an Testdaten

- Wenn sich die Tests selbst ändern, müssen sie erneut ausgeführt werden

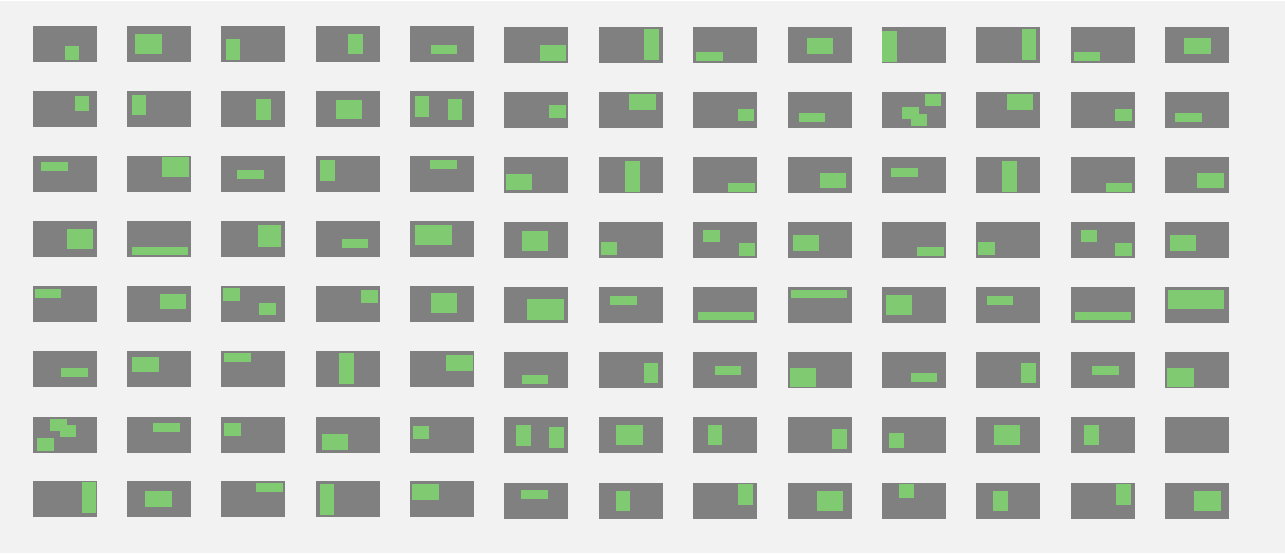
- Es gibt keine Garantie, dass die selektierten Tests alle Fehler finden

⇒ In regelmäßigen Abständen alle Tests ausführen
⇒ Hält testfallspezifische Coverage-Daten aktuell

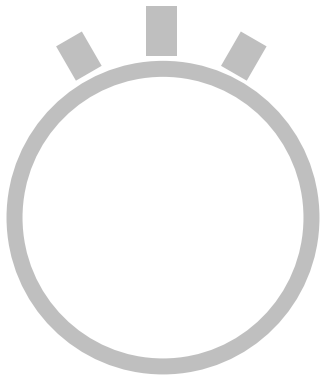
Ziel: Tests in der **Continuous Integration** wirksam machen

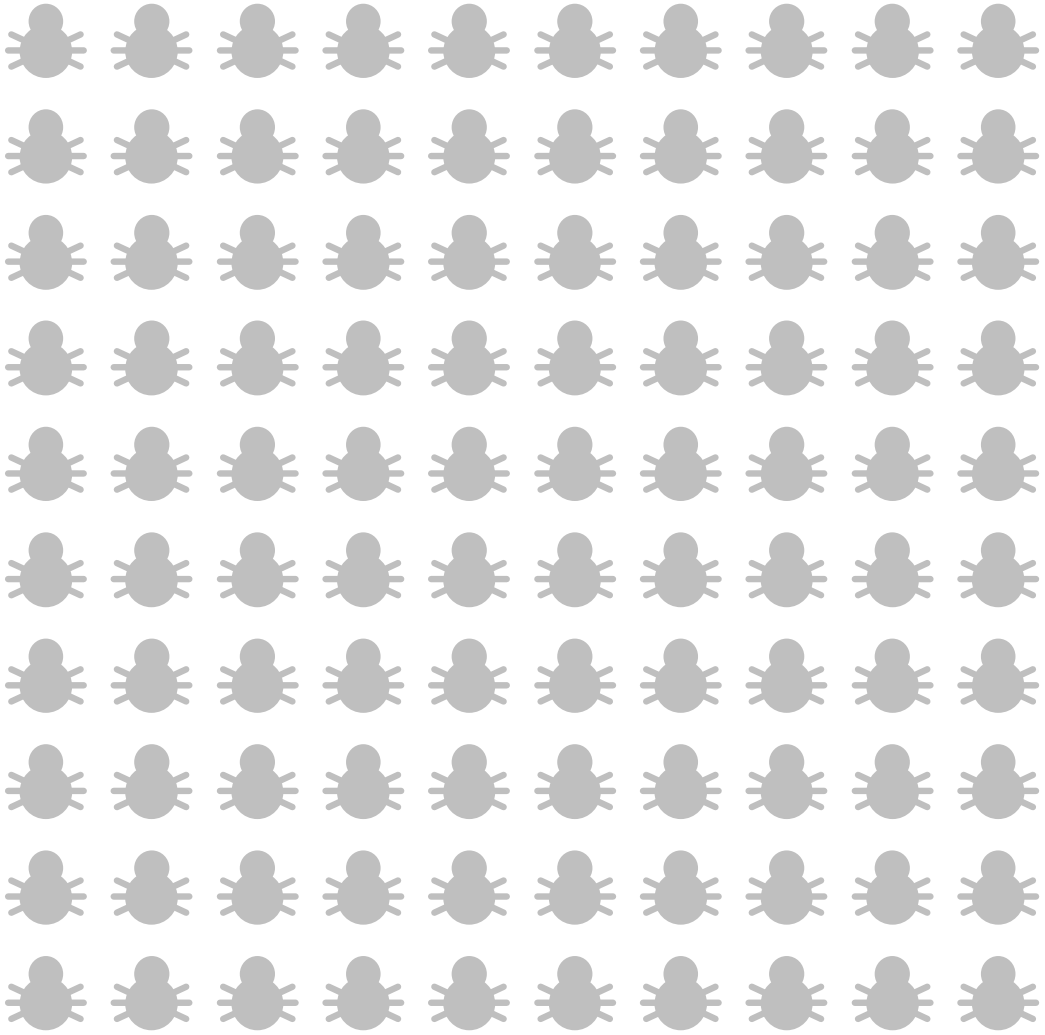
- Neue Tests immer ausführen
- Geänderte Tests immer ausführen

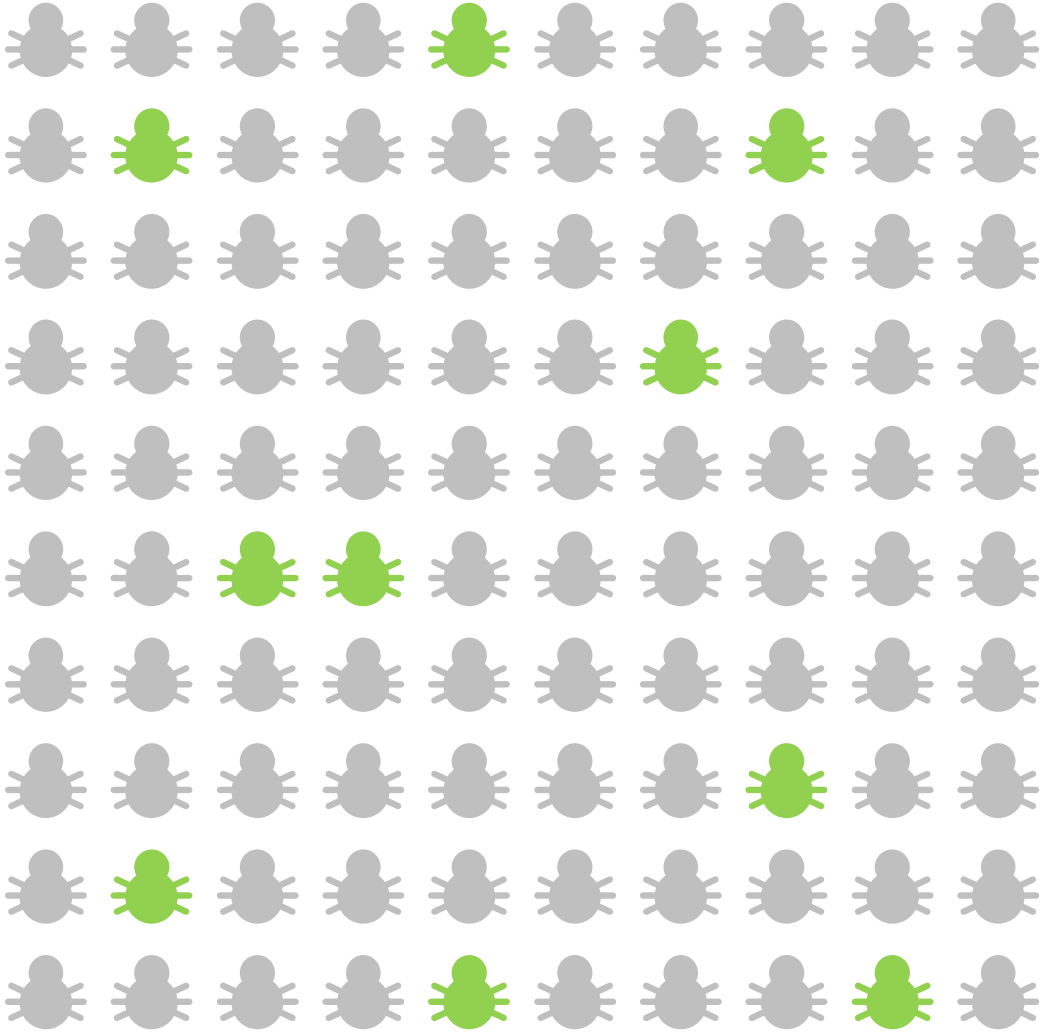
Trotzdem: Regelmäßig **alle Tests** laufen lassen

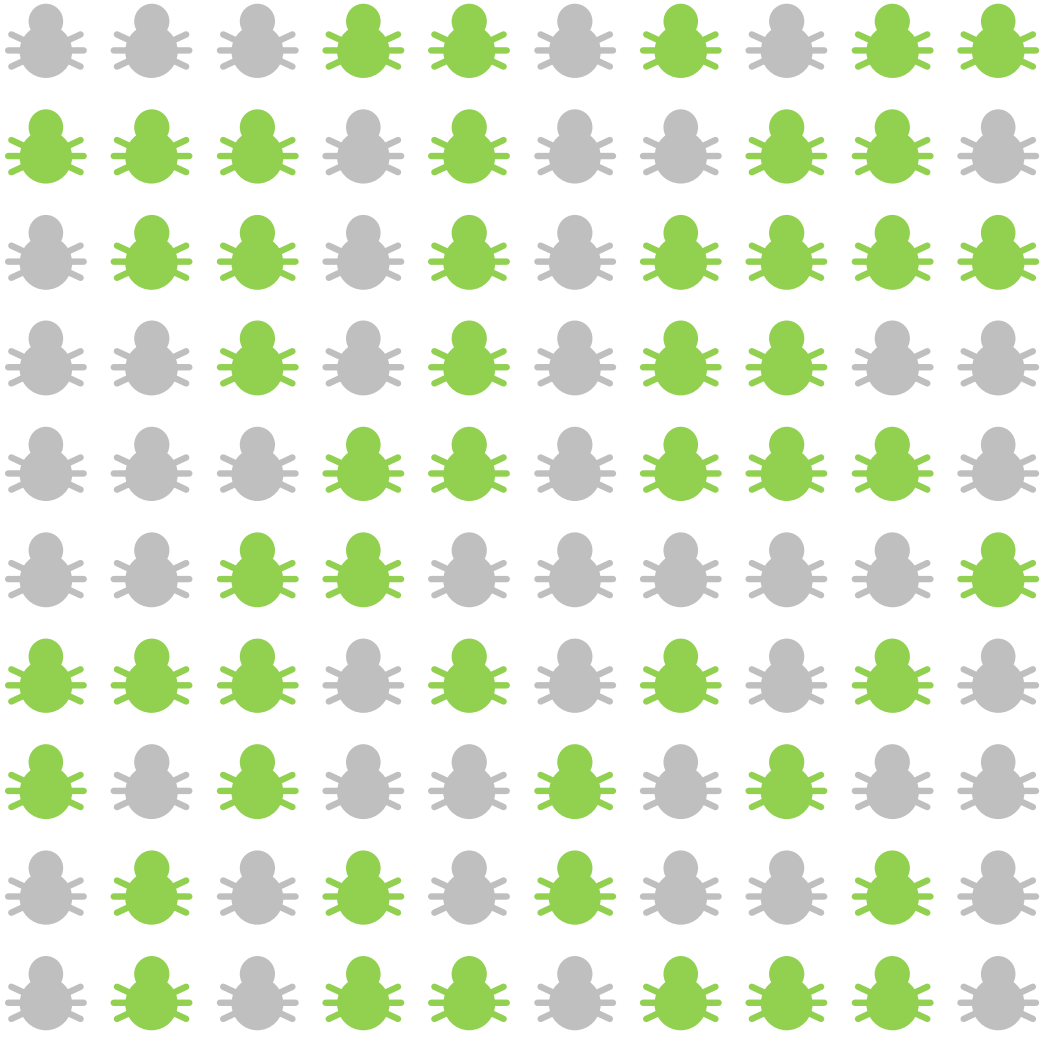


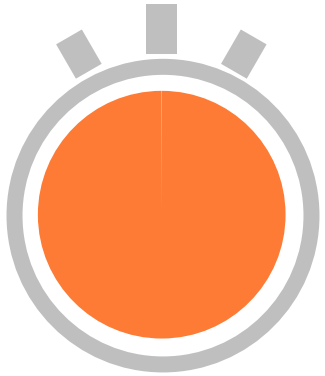
- … wenn sich Daten ändern
- … wenn sich die Konfiguration ändert

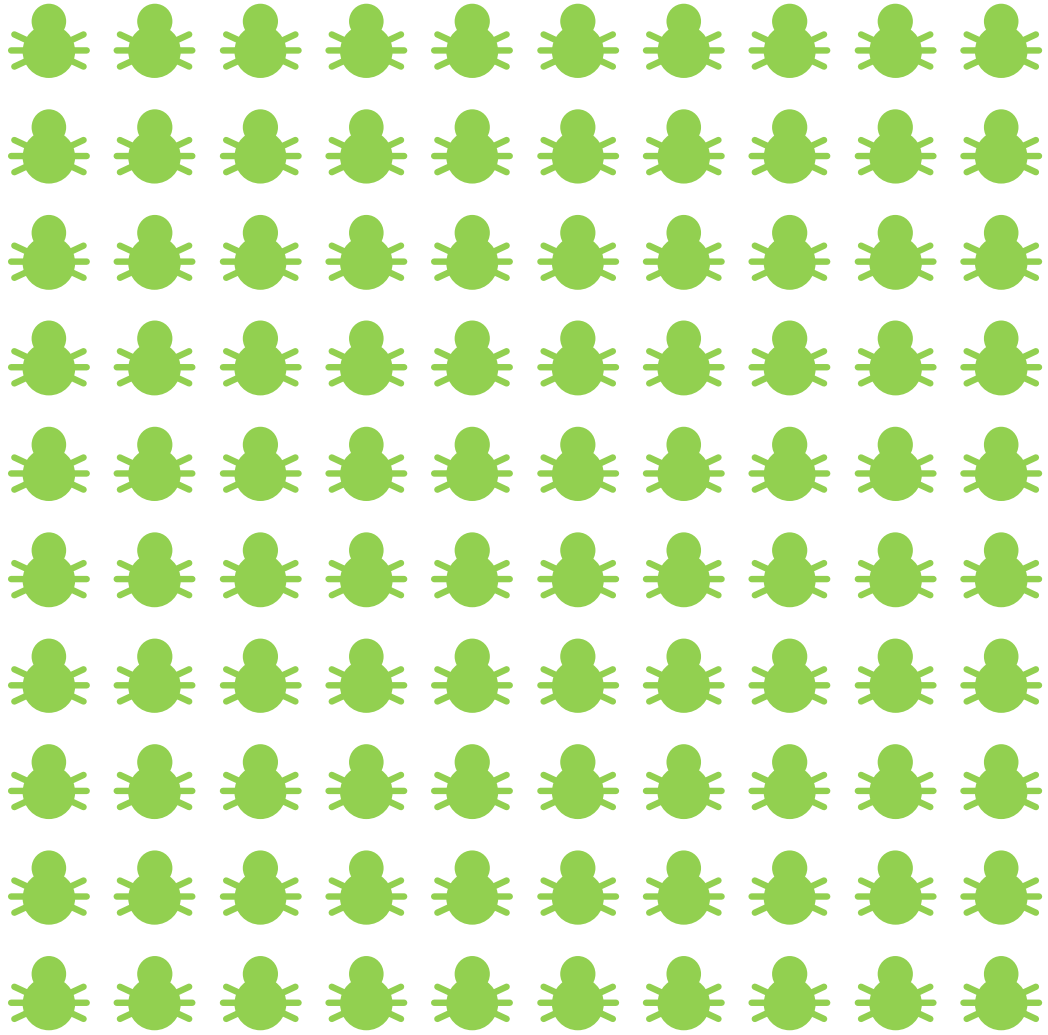






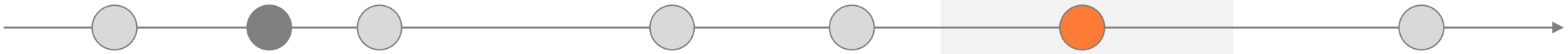


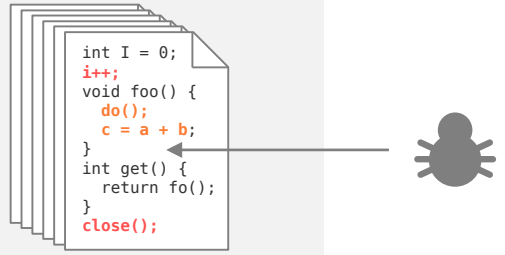




Baseline

Working Copy





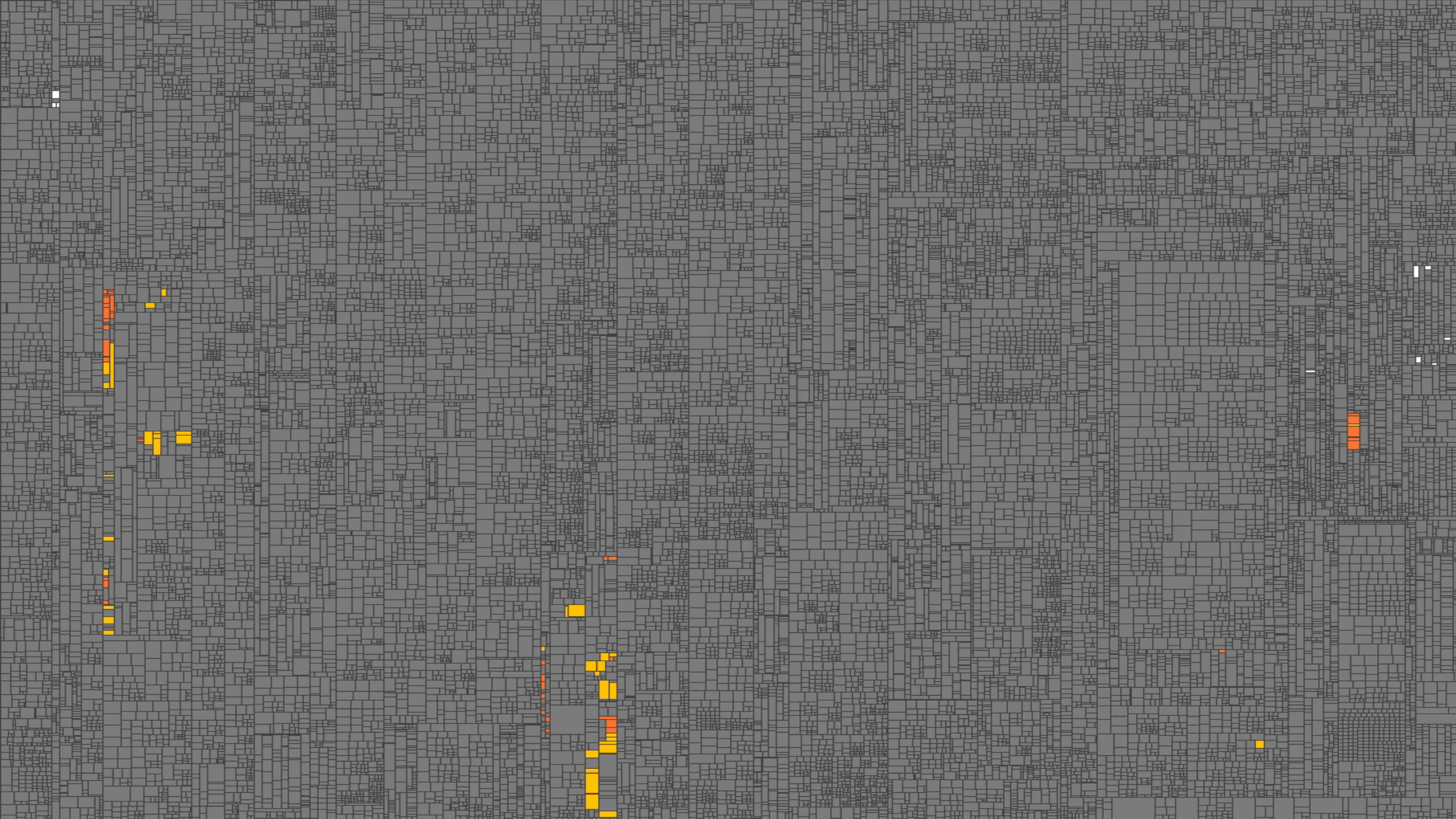
Werden mit den selektierten Tests
die gleichen Fehler gefunden,
wie mit allen Tests?

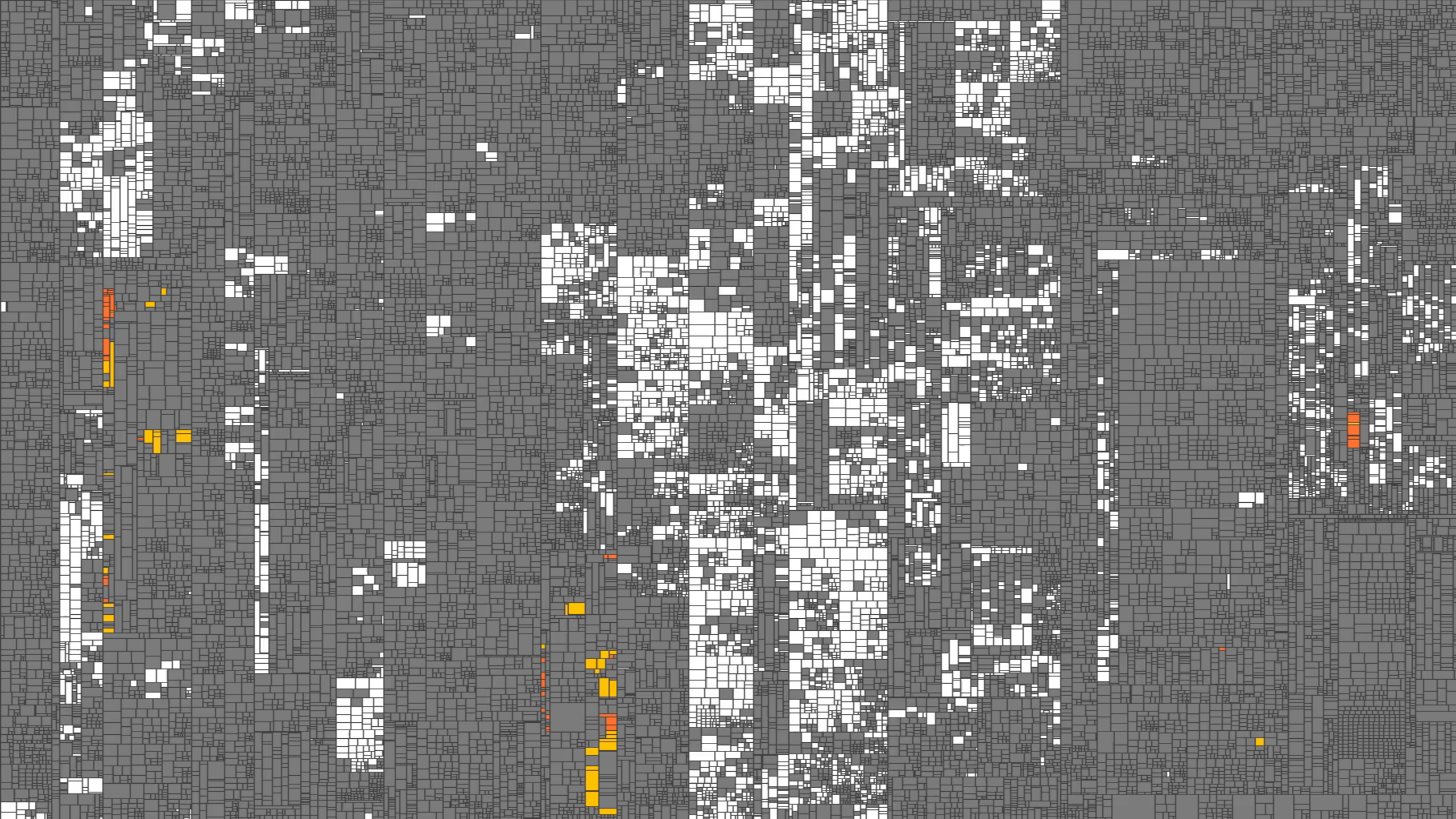
Wieviel Zeit kann mit selektierten Tests eingespart werden?

System	Alle Tests [s]	Selektierte Tests [s]	Einsparung
Apache Commons Collections	25	1	98 %
Apache Commons Lang	25	3	88 %
Apache Commons Math		114	29 %
Histone Template Engine		32	7 %
JabRef		41	66 %
Joda-Time		1	94 %
Lightweight-Stream-API		1	68 %
LittleProxy		150	3 %
OkHttp	97	76	21 %
RxJava	464	171	63 %
Symia Commons Math Parser	1	1	0 %
Teamscale	1249	197	84 %

Im Durchschnitt
64%
Einsparung

Einsparung bei Erkennung aller fehlschlagenden Tests schwankt stark

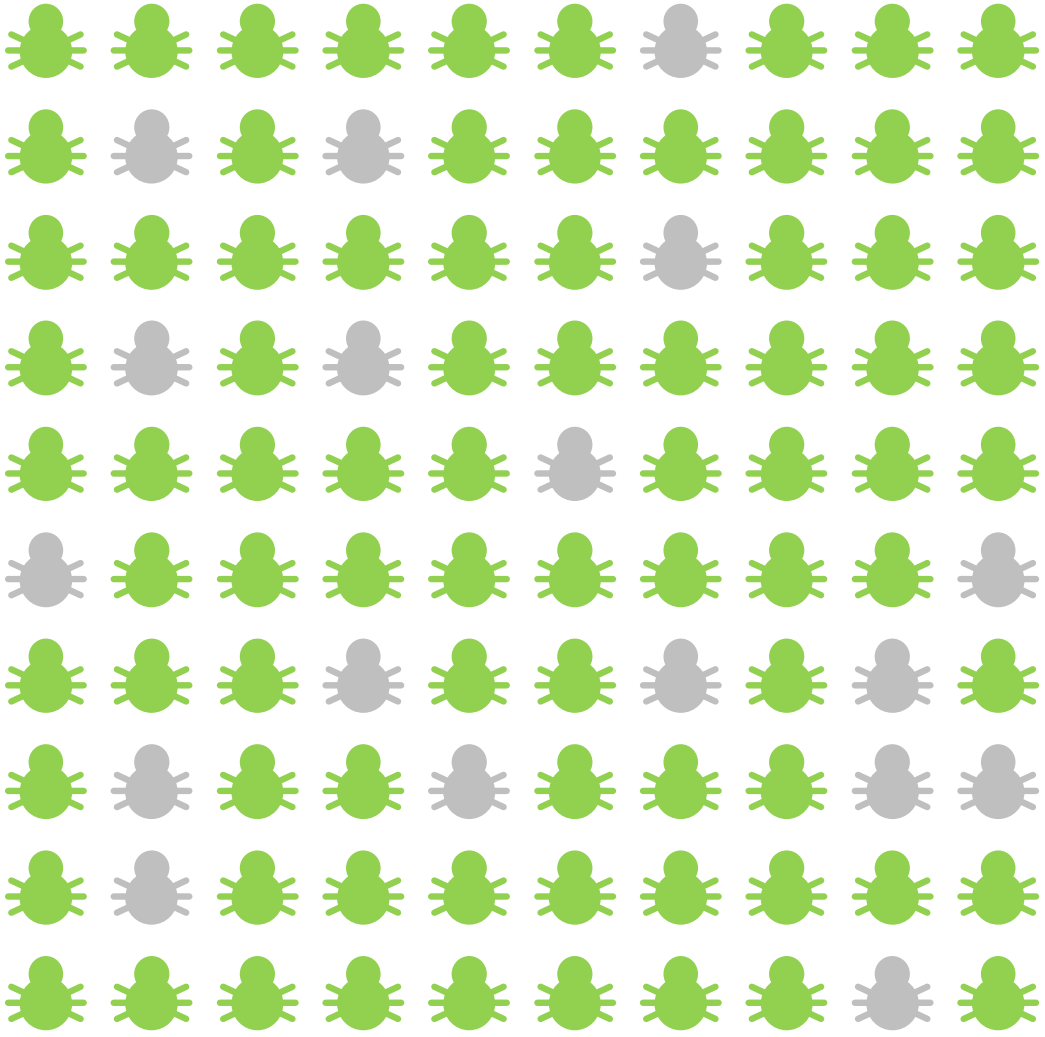


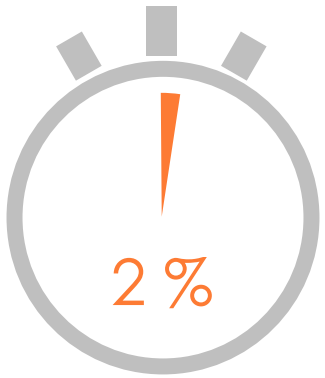


Welcher Anteil fehlerhafter Builds wird in begrenzter Zeit erkannt?

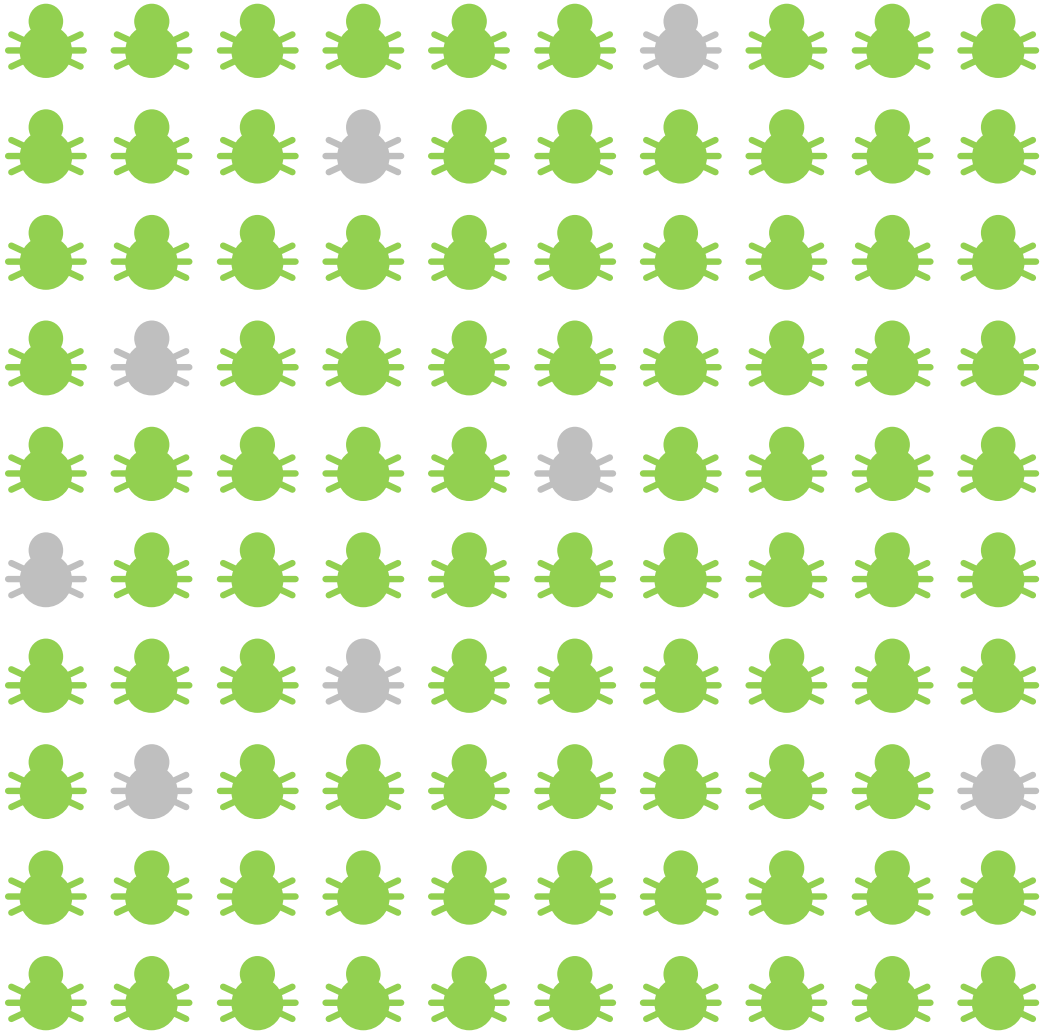
System	1 %	2 %	5 %	10 %
Apache Commons Collections	84 %	97 %	100 %	100 %
Apache Commons Lang	94 %	96 %	98 %	99 %
Apache Commons Math	80 %	86 %	92 %	93 %
Histone Template Engine	88 %	88 %	88 %	88 %
JabRef	96 %	96 %	96 %	96 %
Joda-Time	98 %	99 %	99 %	100 %
Lightweight-Stream-API	61 %	91 %	95 %	97 %
LittleProxy	74 %	77 %	80 %	80 %
OkHttp	91 %	91 %	91 %	97 %
RxJava	92 %	95 %	95 %	95 %
Symia Commons Math Parser	78 %	86 %	89 %	89 %
Teamscale	92 %	96 %	97 %	97 %

80 %

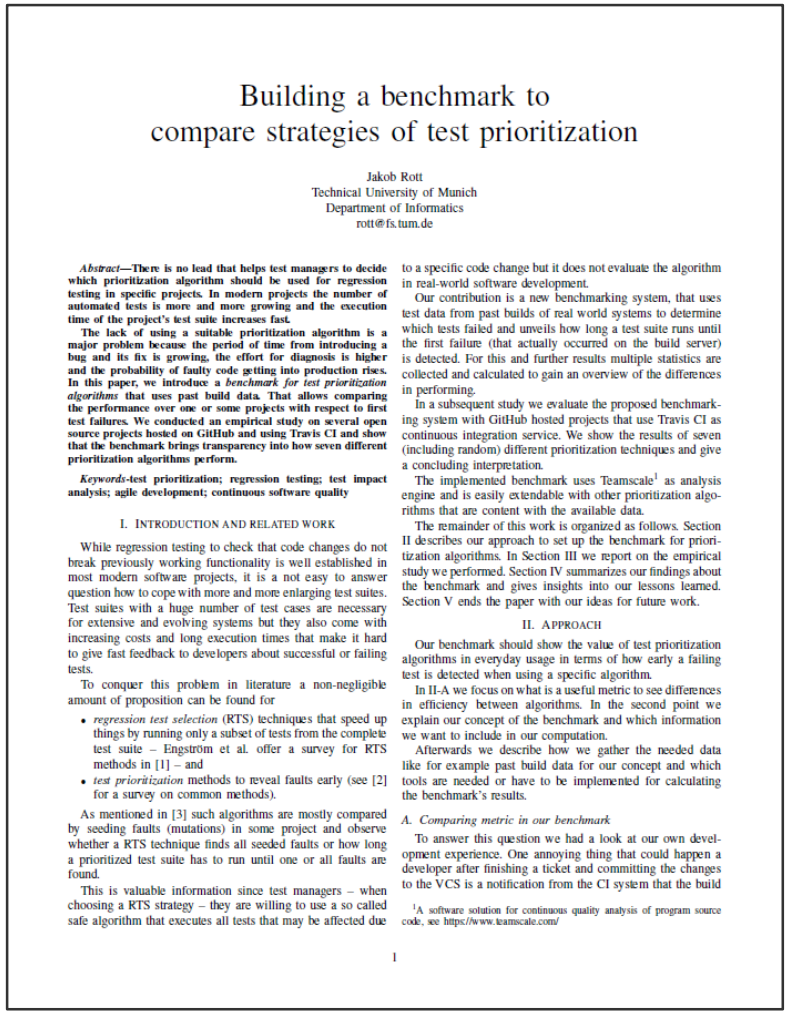




90 %



Forschungsfallstudie



Building a benchmark to
compare strategies of test prioritization

Jakob Rott
Technical University of Munich
Department of Informatics
rott@fs.tum.de

***Abstract*—There is no lead that helps test managers to decide which prioritization algorithm should be used for regression testing in specific projects. In modern projects the number of automated tests is more and more growing and the execution time of the project's test suite increases fast.**

The lack of using a suitable prioritization algorithm is a major problem because the period of time from introducing a bug and its fix is growing, the effort for diagnosis is higher and the probability of faulty code getting into production rises. In this paper, we introduce a *benchmark for test prioritization algorithms* that uses past build data. That allows comparing the performance over one or some projects with respect to first test failures. We conducted an empirical study on several open source projects hosted on GitHub and using Travis CI and show that the benchmark brings transparency into how seven different prioritization algorithms perform.

***Keywords*-test prioritization; regression testing; test impact analysis; agile development; continuous software quality**

I. Introduction and Related Work

While regression testing to check that code changes do not break previously working functionality is well established in most modern software projects, it is a not easy to answer question how to cope with more and more enlarging test suites. Test suites with a huge number of test cases are necessary for extensive and evolving systems but they also come with increasing costs and long execution times that make it hard to give fast feedback to developers about successful or failing tests.

To conquer this problem in literature a non-negligible amount of proposition can be found for

- *regression test selection* (RTS) techniques that speed up things by running only a subset of tests from the complete test suite – Engström et al. offer a survey for RTS methods in [1] – and
- *test prioritization* methods to reveal faults early (see [2] for a survey on common methods).

As mentioned in [3] such algorithms are mostly compared by seeding faults (mutations) in some project and observe whether a RTS technique finds all seeded faults or how long a prioritized test suite has to run until one or all faults are found.

This is valuable information since test managers – when choosing a RTS strategy – they are willing to use a so called safe algorithm that executes all tests that may be affected due to a specific code change but it does not evaluate the algorithm in real-world software development.

Our contribution is a new benchmarking system, that uses test data from past builds of real world systems to determine which tests failed and unveils how long a test suite runs until the first failure (that actually occurred on the build server) is detected. For this and further results multiple statistics are collected and calculated to gain an overview of the differences in performing.

In a subsequent study we evaluate the proposed benchmarking system with GitHub hosted projects that use Travis CI as continuous integration service. We show the results of seven (including random) different prioritization techniques and give a concluding interpretation.

The implemented benchmark uses Teamscale[1] as analysis engine and is easily extendable with other prioritization algorithms that are content with the available data.

The remainder of this work is organized as follows. Section II describes our approach to set up the benchmark for prioritization algorithms. In Section III we report on the empirical study we performed. Section IV summarizes our findings about the benchmark and gives insights into our lessons learned. Section V ends the paper with our ideas for future work.

II. Approach

Our benchmark should show the value of test prioritization algorithms in everyday usage in terms of how early a failing test is detected when using a specific algorithm.

In II-A we focus on what is a useful metric to see differences in efficiency between algorithms. In the second point we explain our concept of the benchmark and which information we want to include in our computation.

Afterwards we describe how we gather the needed data like for example past build data for our concept and which tools are needed or have to be implemented for calculating the benchmark's results.

A. Comparing metric in our benchmark

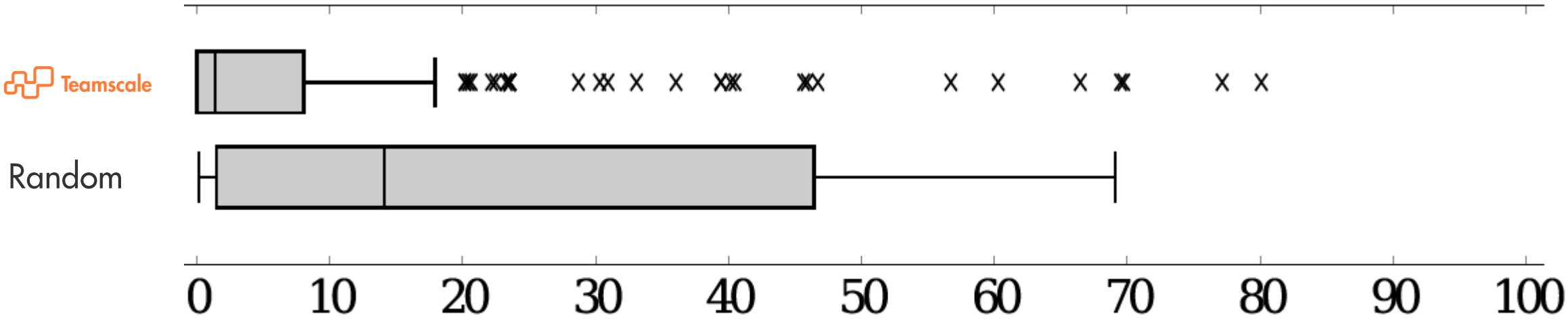
To answer this question we had a look at our own development experience. One annoying thing that could happen a developer after finishing a ticket and committing the changes to the VCS is a notification from the CI system that the build

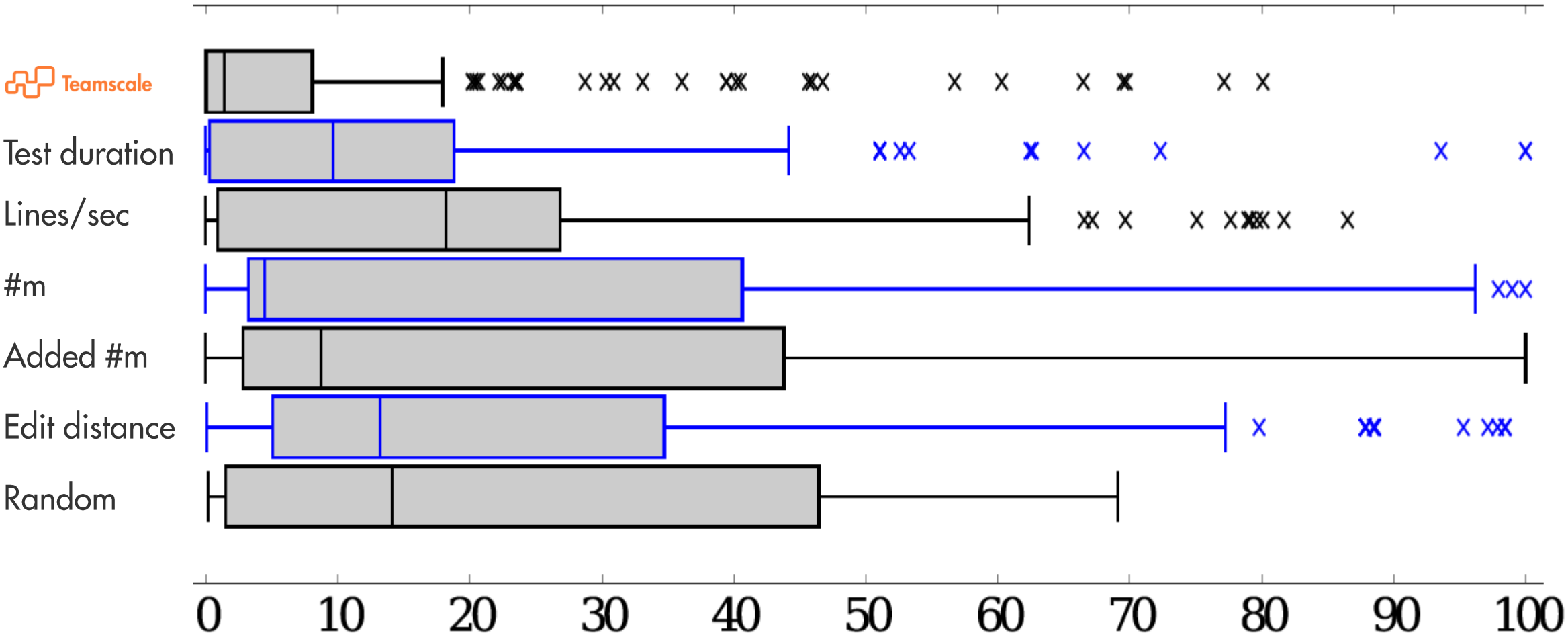
[1]A software solution for continuous quality analysis of program source code, see https://www.teamscale.com/

1

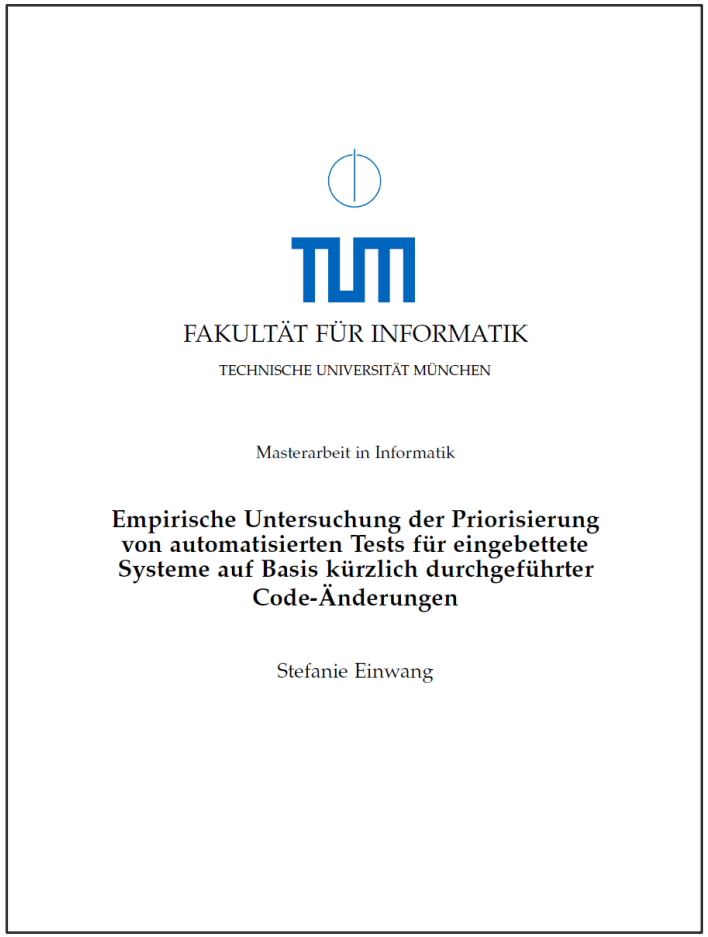
GitHub

Travis CI


Teamscale
Random
0
10
20
30
40
50
60
70
80
90
100


Teamscale
Test duration
Lines/sec
#m
Added #m
Edit distance
Random
0
10
20
30
40
50
60
70
80
90
100

Praxisfallstudie



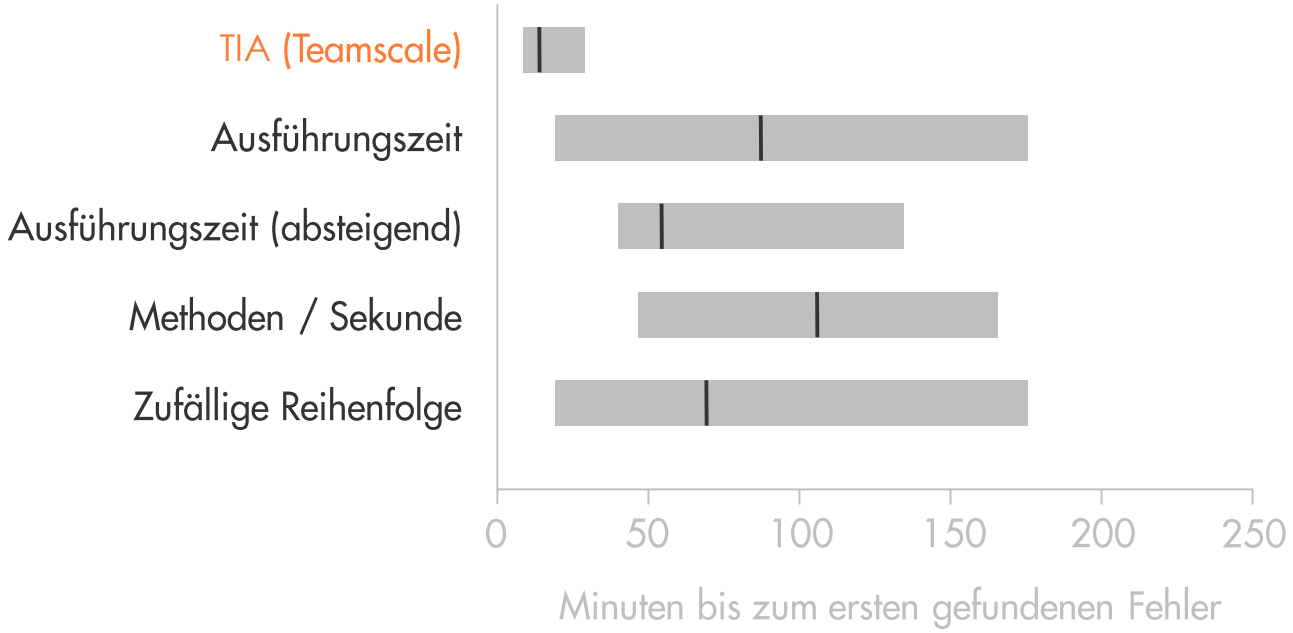
FAKULTÄT FÜR INFORMATIK

TECHNISCHE UNIVERSITÄT MÜNCHEN

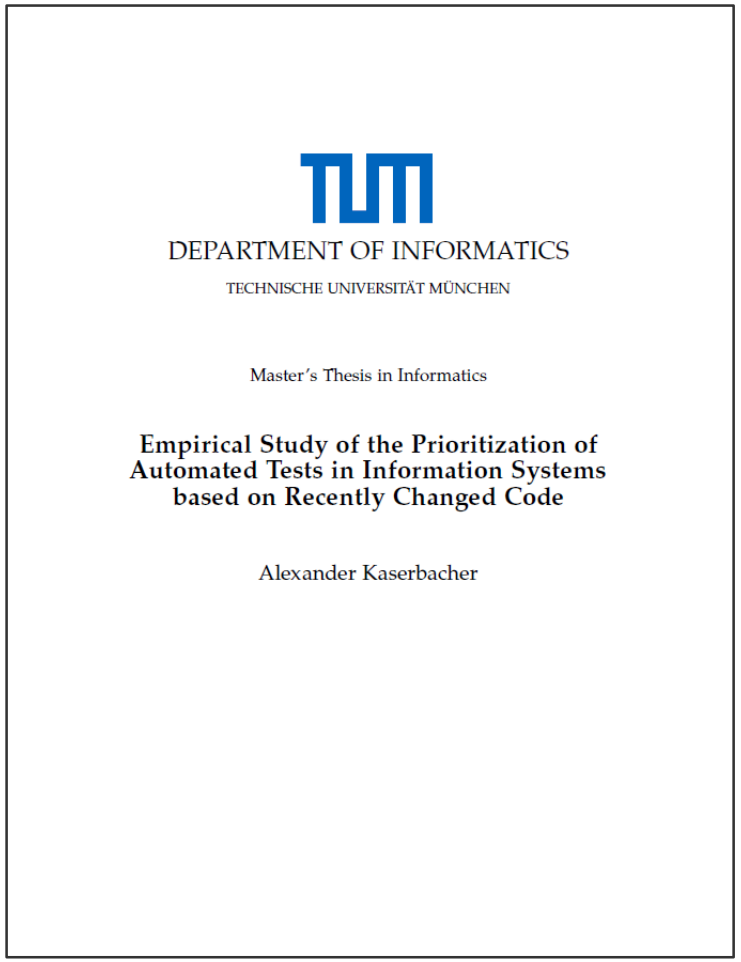
Masterarbeit in Informatik

Empirische Untersuchung der Priorisierung von automatisierten Tests für eingebettete Systeme auf Basis kürzlich durchgeführter Code-Änderungen

Stefanie Einwang


TIA (Teamscale)
Ausführungszeit
Ausführungszeit (absteigend)
Methoden / Sekunde
Zufällige Reihenfolge
0
50
100
150
200
250
Minuten bis zum ersten gefundenen Fehler

Praxisfallstudie



DEPARTMENT OF INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

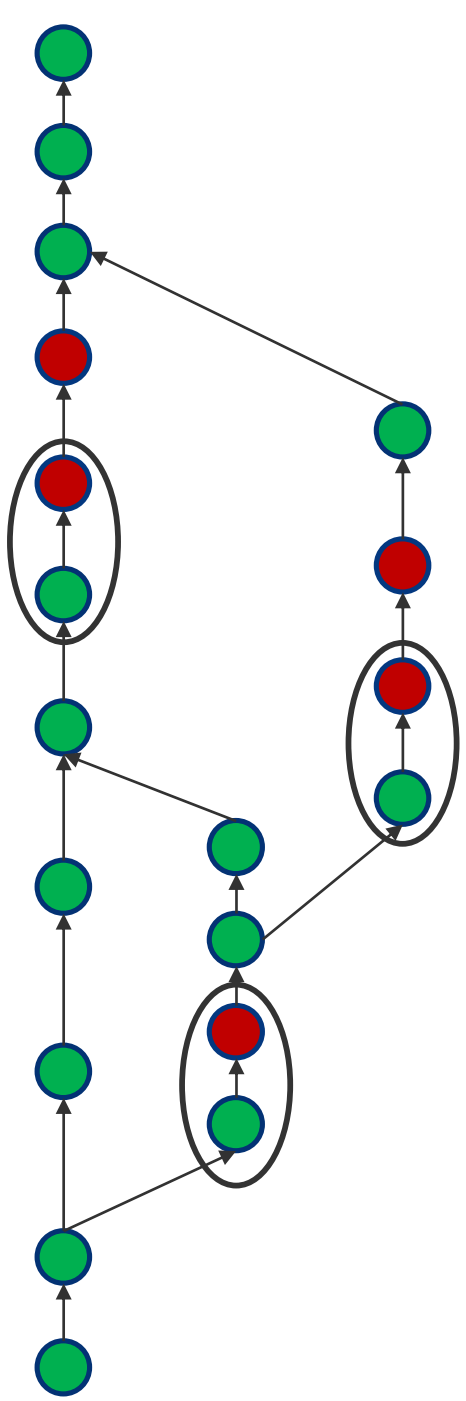
Master's Thesis in Informatics

Empirical Study of the Prioritization of Automated Tests in Information Systems based on Recently Changed Code

Alexander Kaserbacher

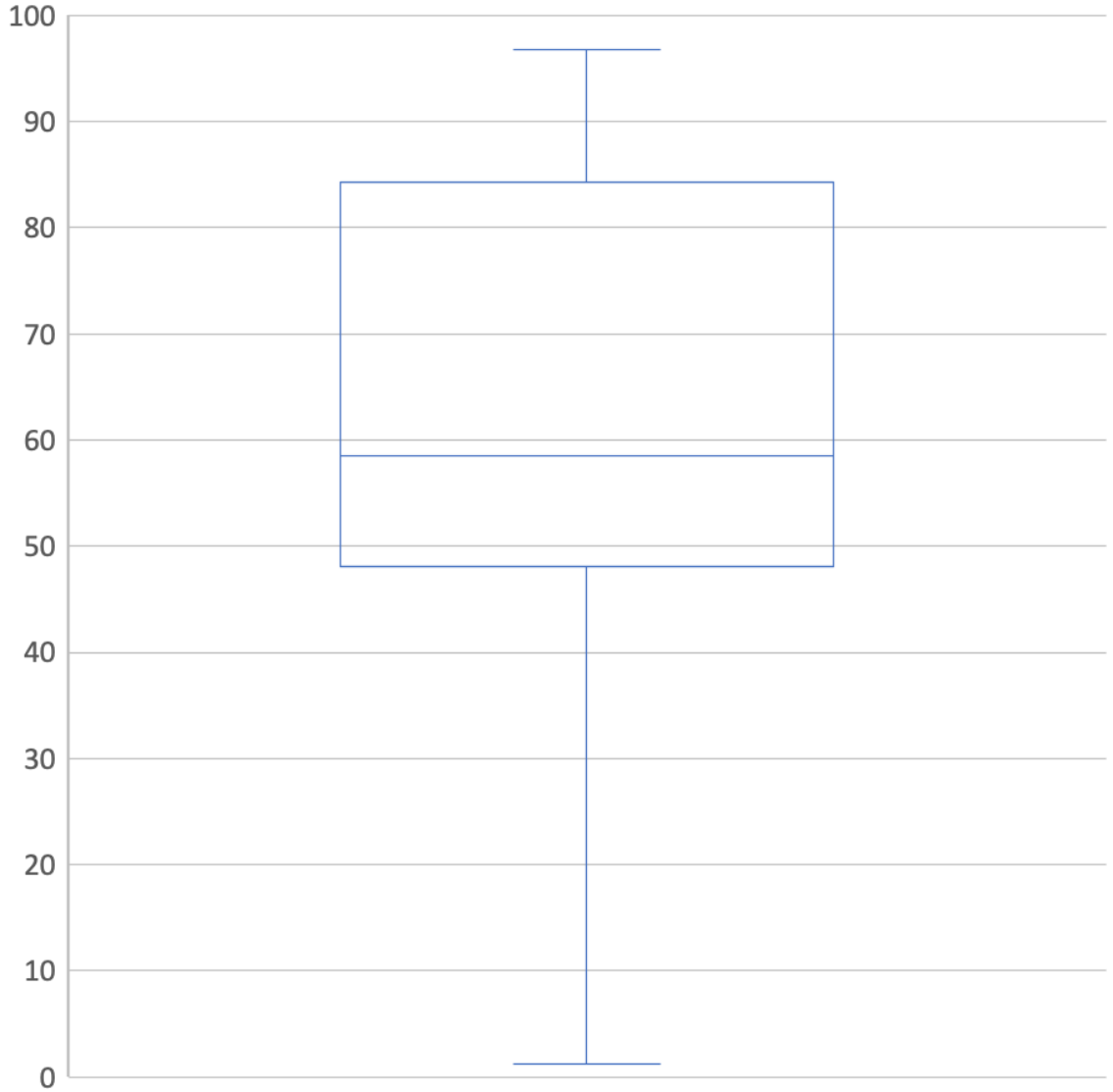
mercateo

Lines of Code	4,3 millionen
Technologien	Java, Spring, Hibernate, Maven, …
Anzahl Maven Submodule	736
Entwicklung seit	2001
Anzahl aktiver Entwickler	60
Laufzeit lokale Tests	ca. 37 Minuten
Laufzeit End-to-end Tests	ca. 7 Minuten

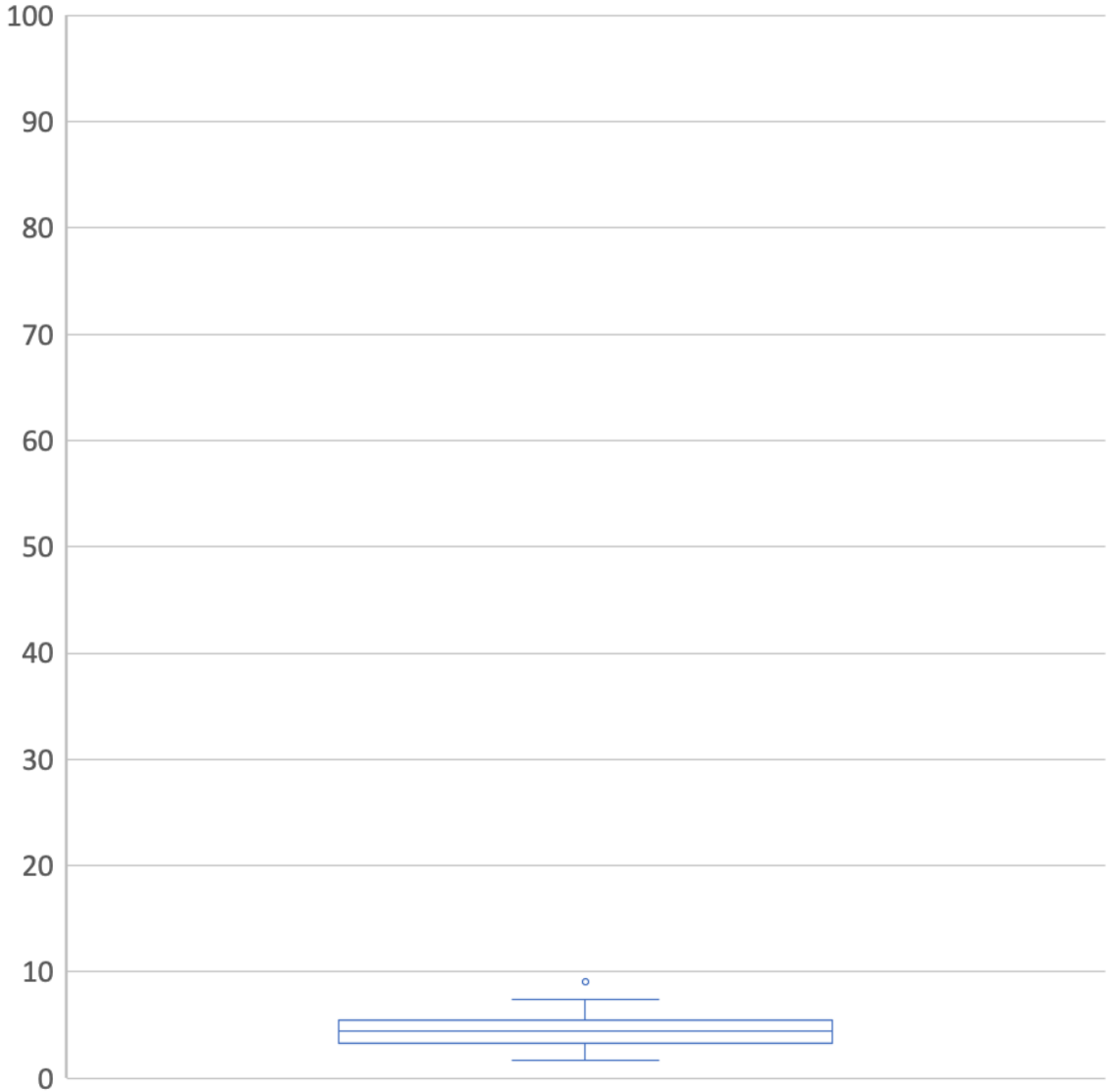


Untersuchte Commits	1605
Fehlerverursachende Änderungen	50

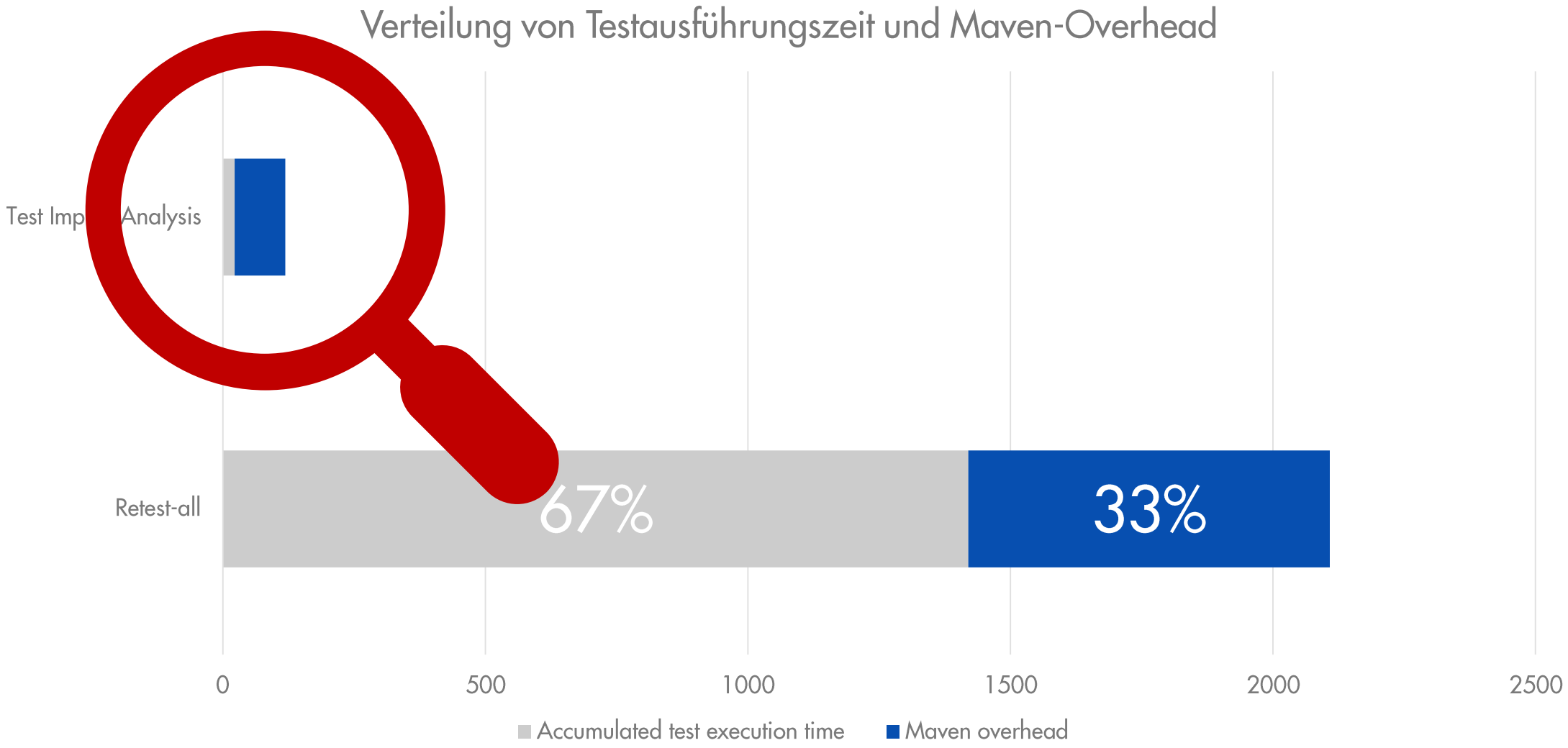
Zeitspanne bis erster Fehler gefunden wird, relativ zu gesamter Ausführungszeit

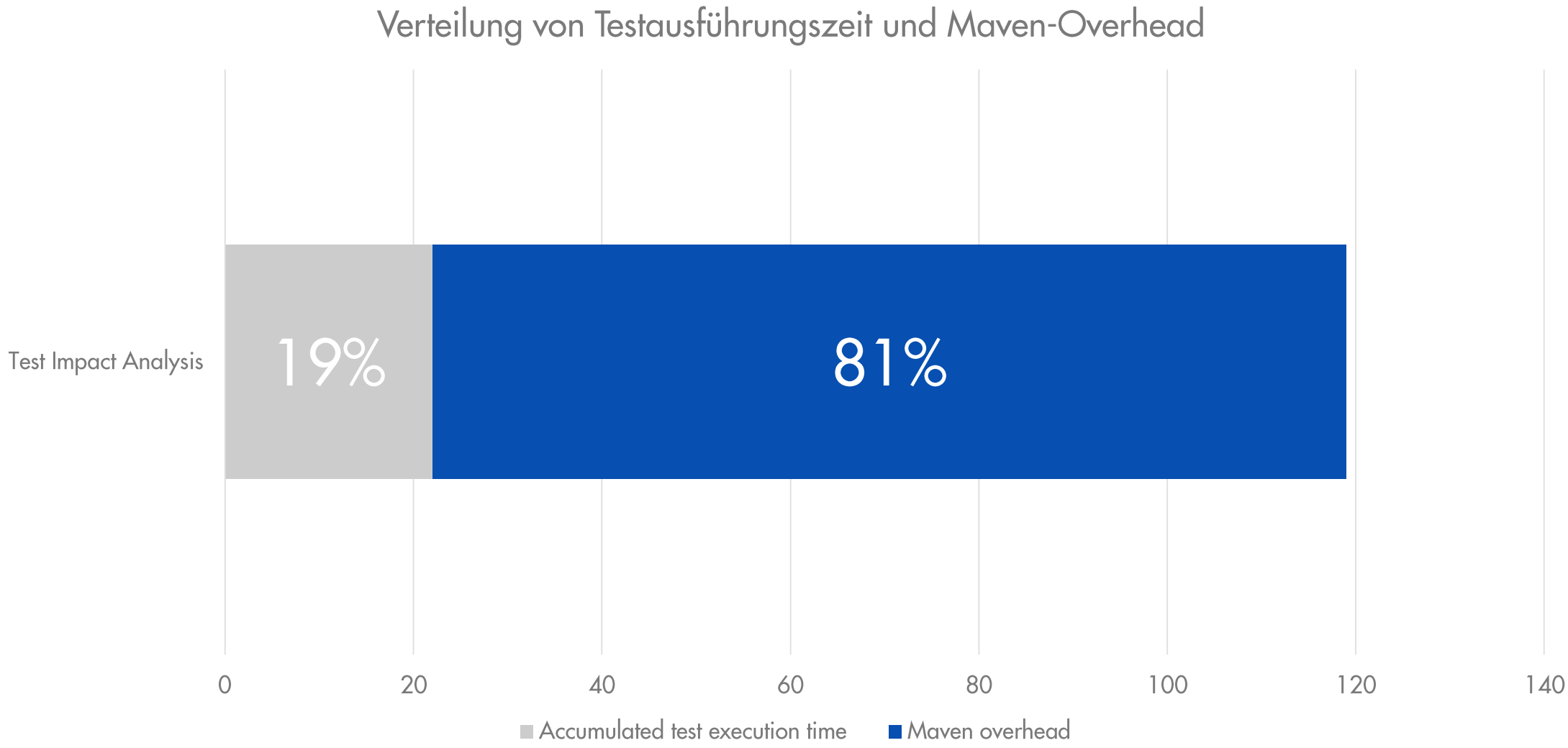


Retest-all



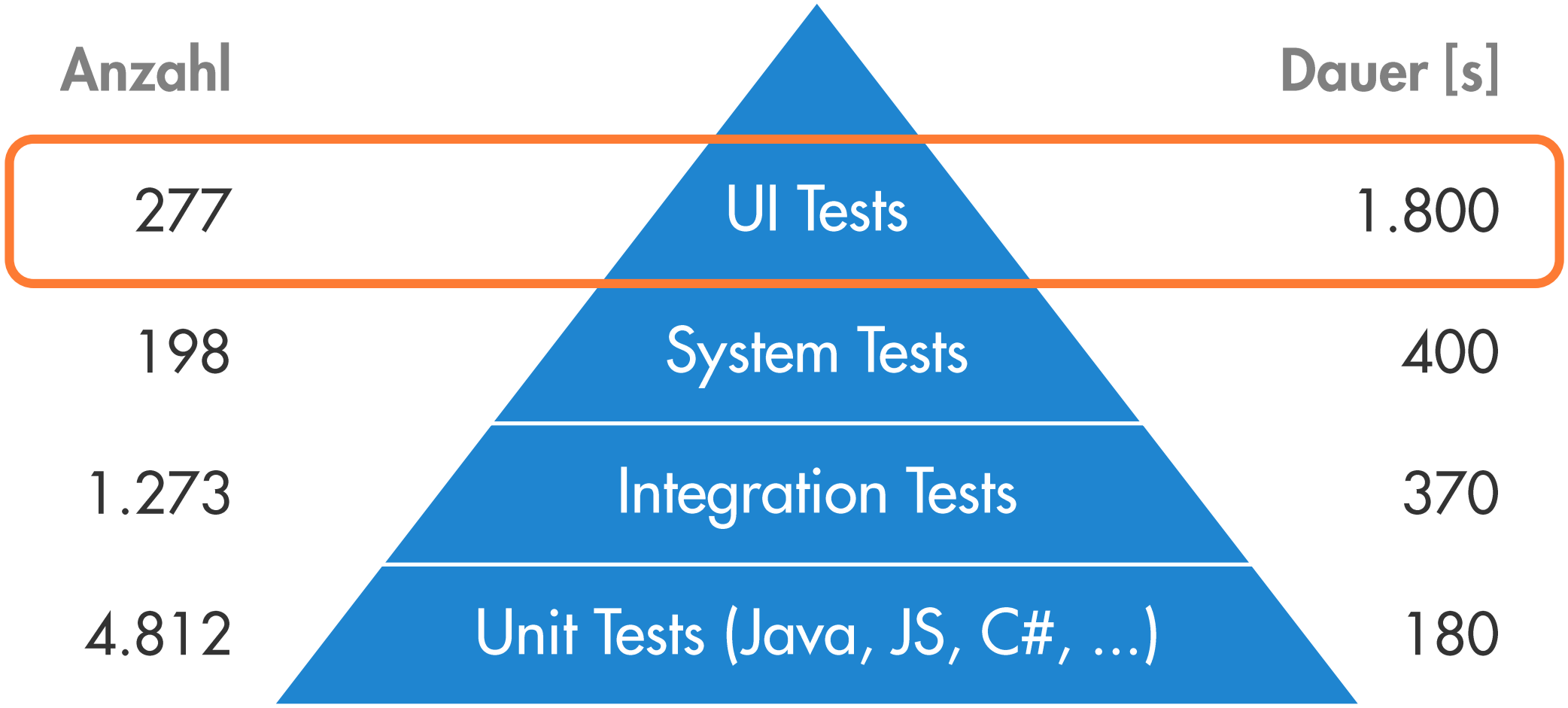
Test-Impact-Analyse


Verteilung von Testausführungszeit und Maven-Overhead
Test Imp
Analysis
Retest-all
67%
33%
0
500
1000
1500
2000
2500
Accumulated test execution time
Maven overhead


Verteilung von Testausführungszeit und Maven-Overhead
Test Impact Analysis
19%
81%
0
20
40
60
80
100
120
140
Accumulated test execution time
Maven overhead

Einsatz bei uns selbst

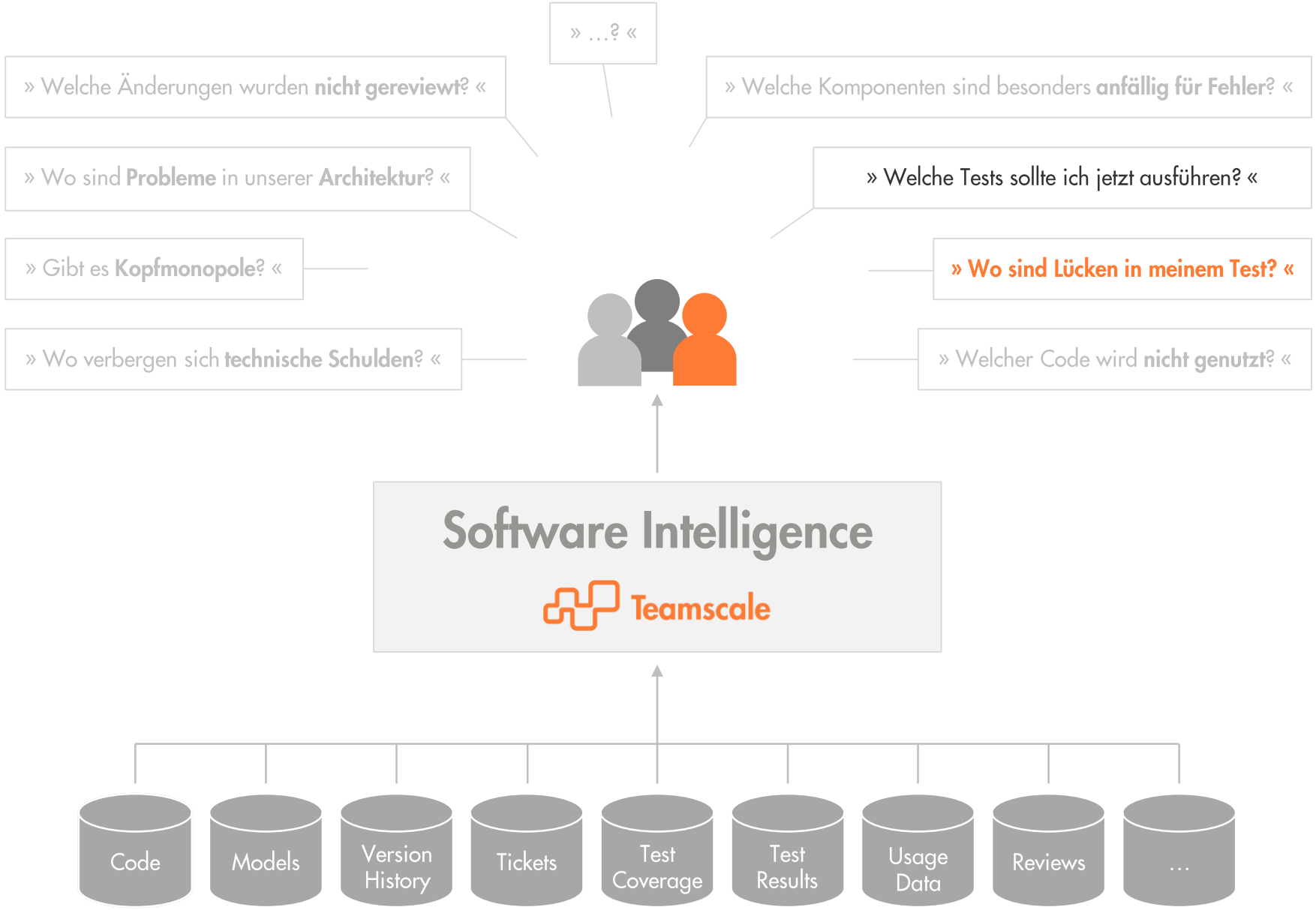
Wie wir Teamscale testen

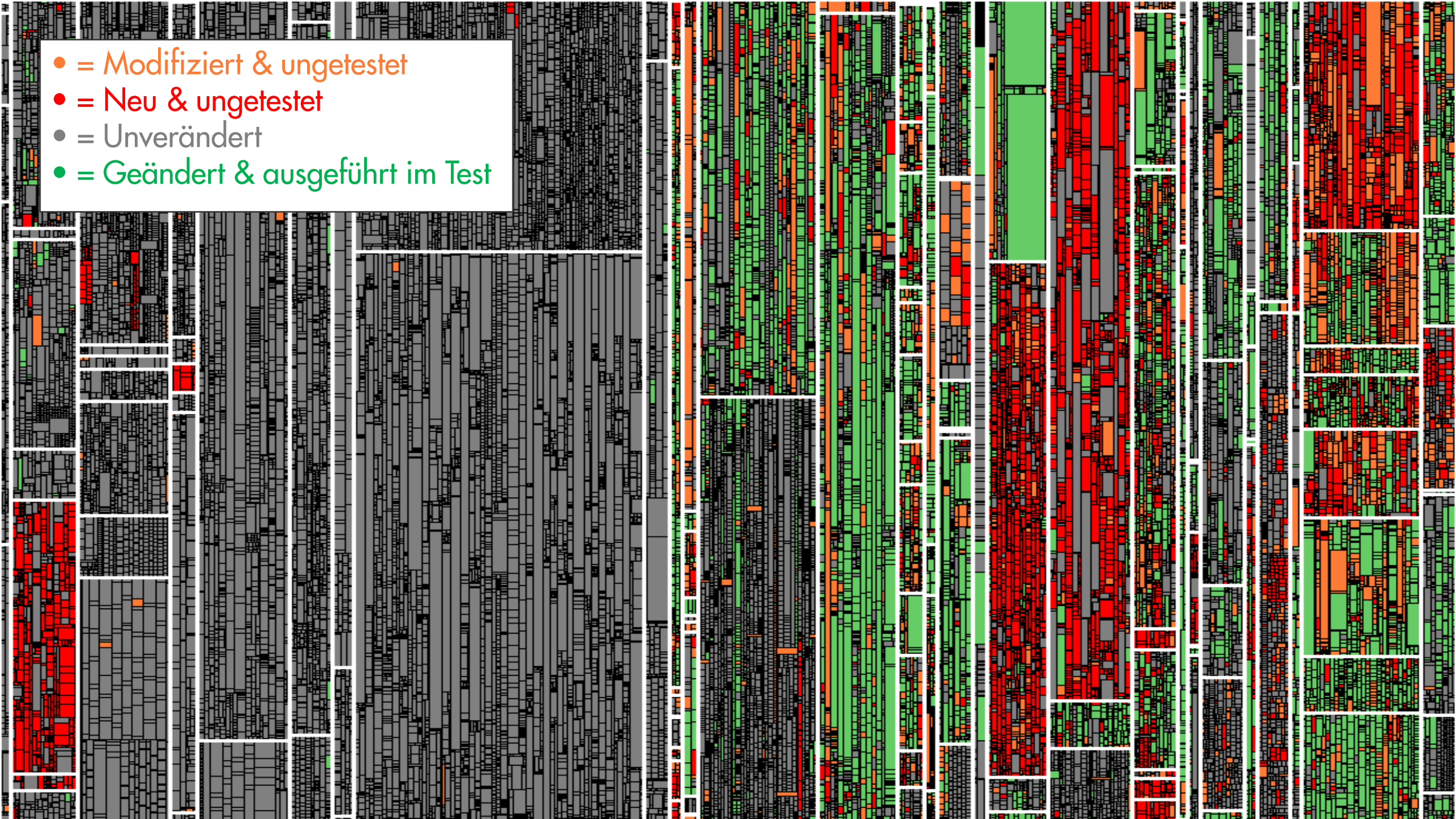


Pipeline	Id	Name	Commit	Status	Runtime	Tests executed	Time saved	TIA enabled	Runtime distribution
83265	694608	ui-tests-common-lang	master (May 8, 2019 10:54:36 AM)	SUCCESS	20m 38s	100% -> master always runs all tests	00m 30s	Yes	
83265	694607	ui-tests	master (May 8, 2019 10:54:36 AM)	SUCCESS	26m 09s	100% -> master always runs all tests	01m 58s	Yes	

Pipeline	Id	Name	Commit	Status	Runtime	Tests executed	Time saved	TIA enabled	Runtime distribution
83239	694073	ui-tests-common-lang	cr/18778_add_tasks_slide (May 7, 2019 7:03:06 PM)	SUCCESS	06m 27s	16%	14m 41s	Yes	
83239	694072	ui-tests	cr/18778_add_tasks_slide (May 7, 2019 7:03:06 PM)	SUCCESS	05m 42s	08%	22m 25s	Yes	
83237	694027	ui-tests-common-lang	cr/18985_apply_patterns_when_toggle_on (May 7, 2019 6:37:16 PM)	SUCCESS	12m 44s	47%	08m 24s	Yes	
83237	694026	ui-tests	cr/18985_apply_patterns_when_toggle_on (May 7, 2019 6:37:16 PM)	SUCCESS	16m 14s	44%	11m 53s	Yes	
83234	693958	ui-tests-common-lang	cr/18089_initial_reports_implementation (May 7, 2019 5:15:30 PM)	SUCCESS	05m 01s	01%	16m 07s	Yes	
83234	693957	ui-tests	cr/18089_initial_reports_implementation (May 7, 2019 5:15:30 PM)	SUCCESS	05m 01s	00%	23m 06s	Yes	

83059	690294	ui-tests-common-lang	cr/18089_initial_reports_implementation (May 2, 2019 10:00:07 PM)	FAILED	10m 06s	29%	11m 02s	Yes	


» …? «
» Welche Änderungen wurden **nicht gereviewt**? «
» Welche Komponenten sind besonders **anfällig für Fehler**? «
» Wo sind **Probleme** in unserer **Architektur**? «
» Welche Tests sollte ich jetzt ausführen? «
» Gibt es **Kopfmonopole**? «
» Wo sind Lücken in meinem Test? «
» Wo verbergen sich **technische Schulden**? «
» Welcher Code wird **nicht genutzt**? «
Software Intelligence
Teamscale
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…


• = Modifiziert & ungetestet
• = Neu & ungetestet
• = Unverändert
• = Geändert & ausgeführt im 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

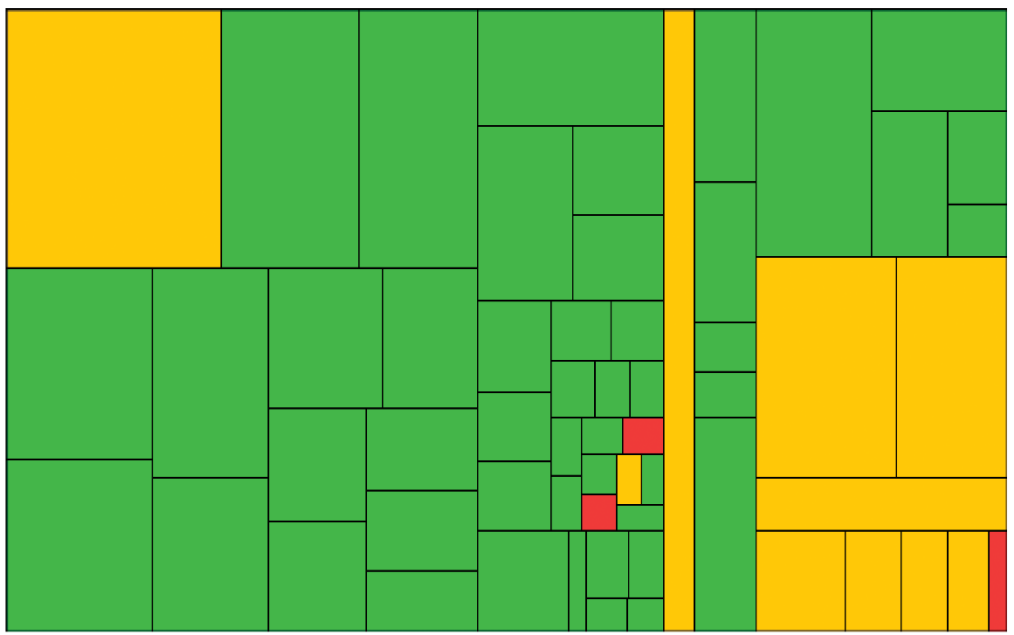
Assignee:

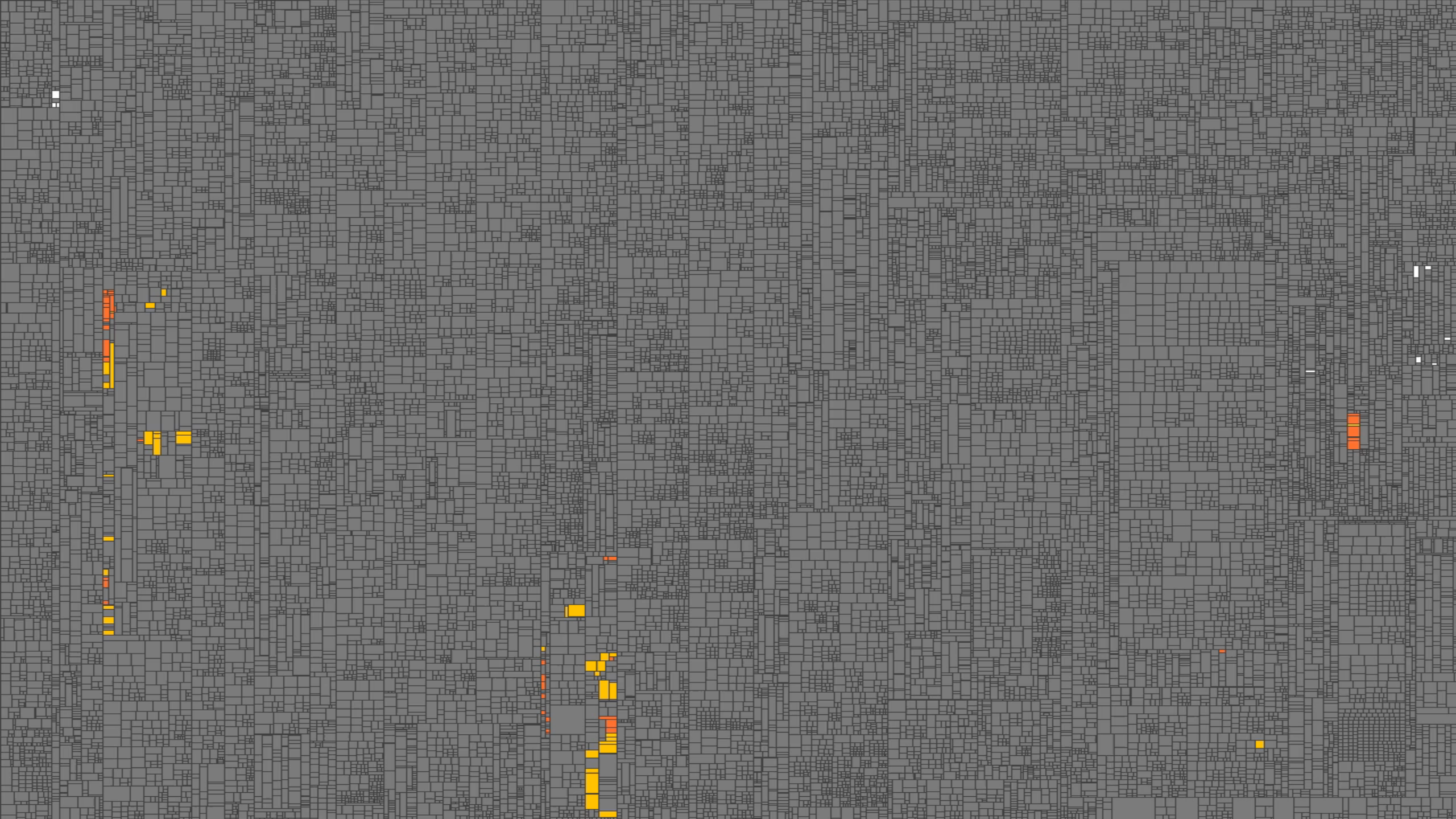
Project	Type	Priority	Resolution	Fix Version
TS	Maintenance	Normal	Green	Teamscale 4.5

Component	Labels	Affected Version	Customer	Customer Issue
Backend	Performance			

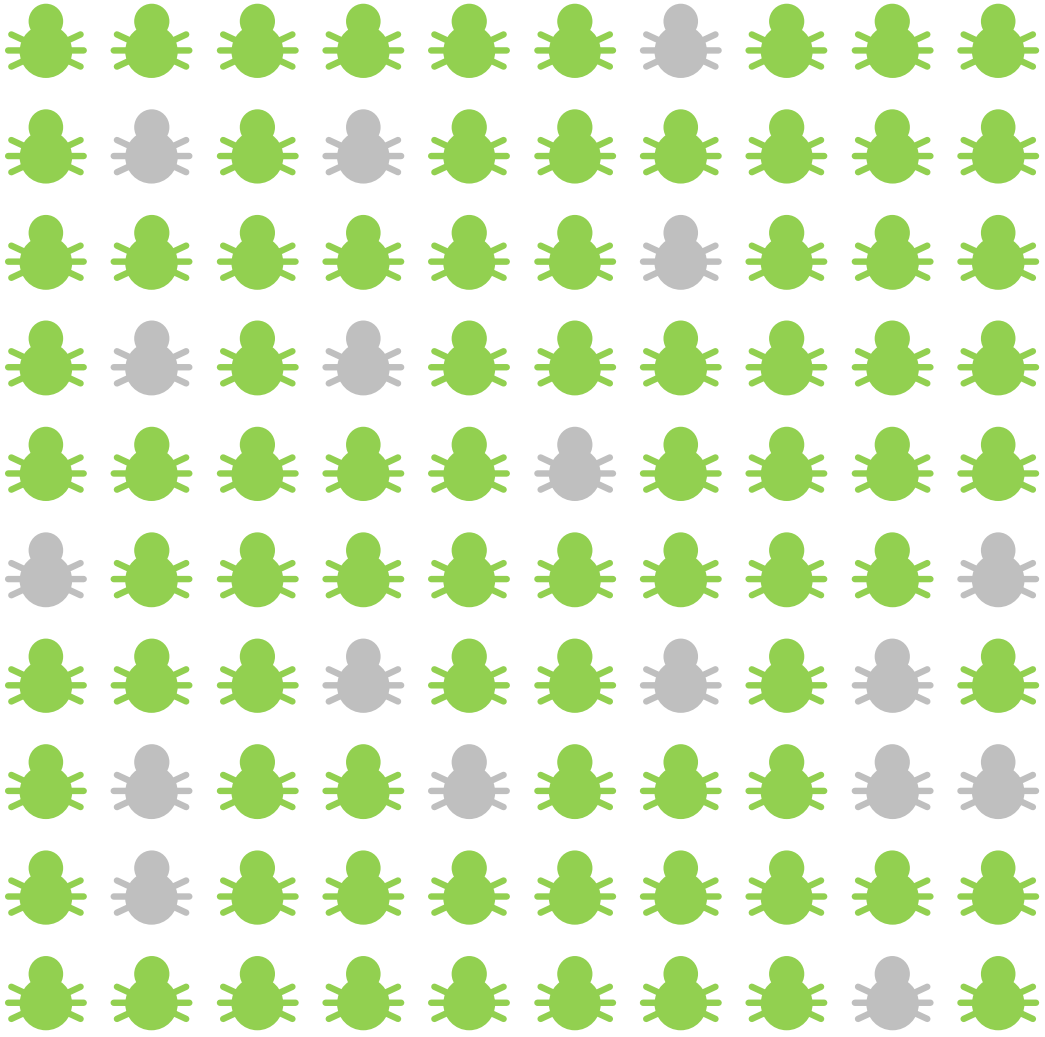
Epic Name	Freshdesk URL	Merge Request
		https://git.cqse.eu/cqse/teamscale/3621

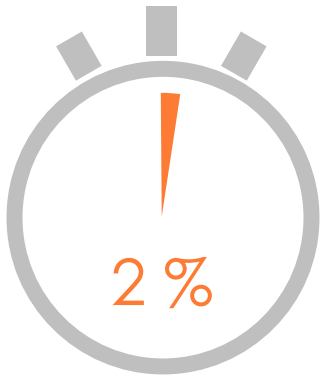
Aug 15 2018 12:37–Now | Test Gap: 22%



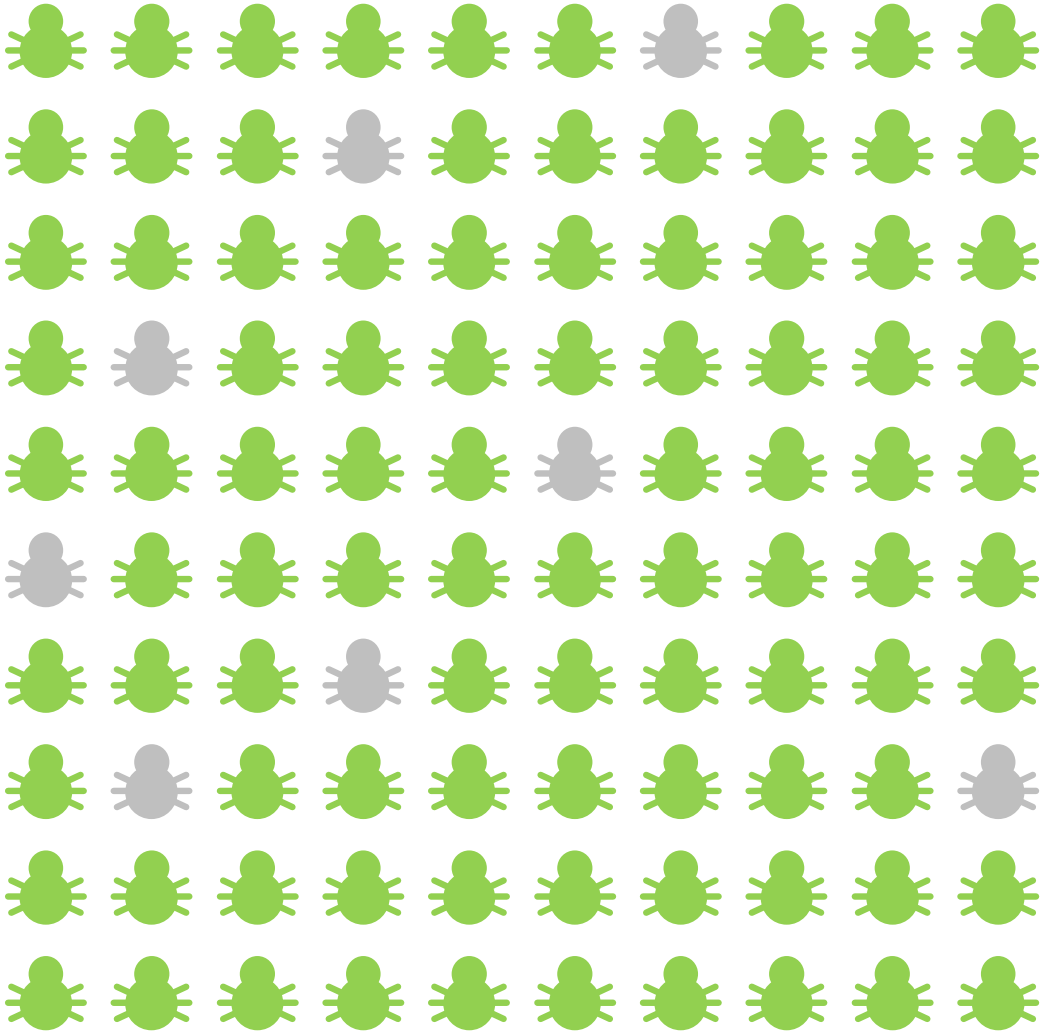


80 %





90 %



Kontakt – Ich freue mich auf Diskussionen ☺

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

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München
www.cqse.eu

CQSE