

# Test Intelligence

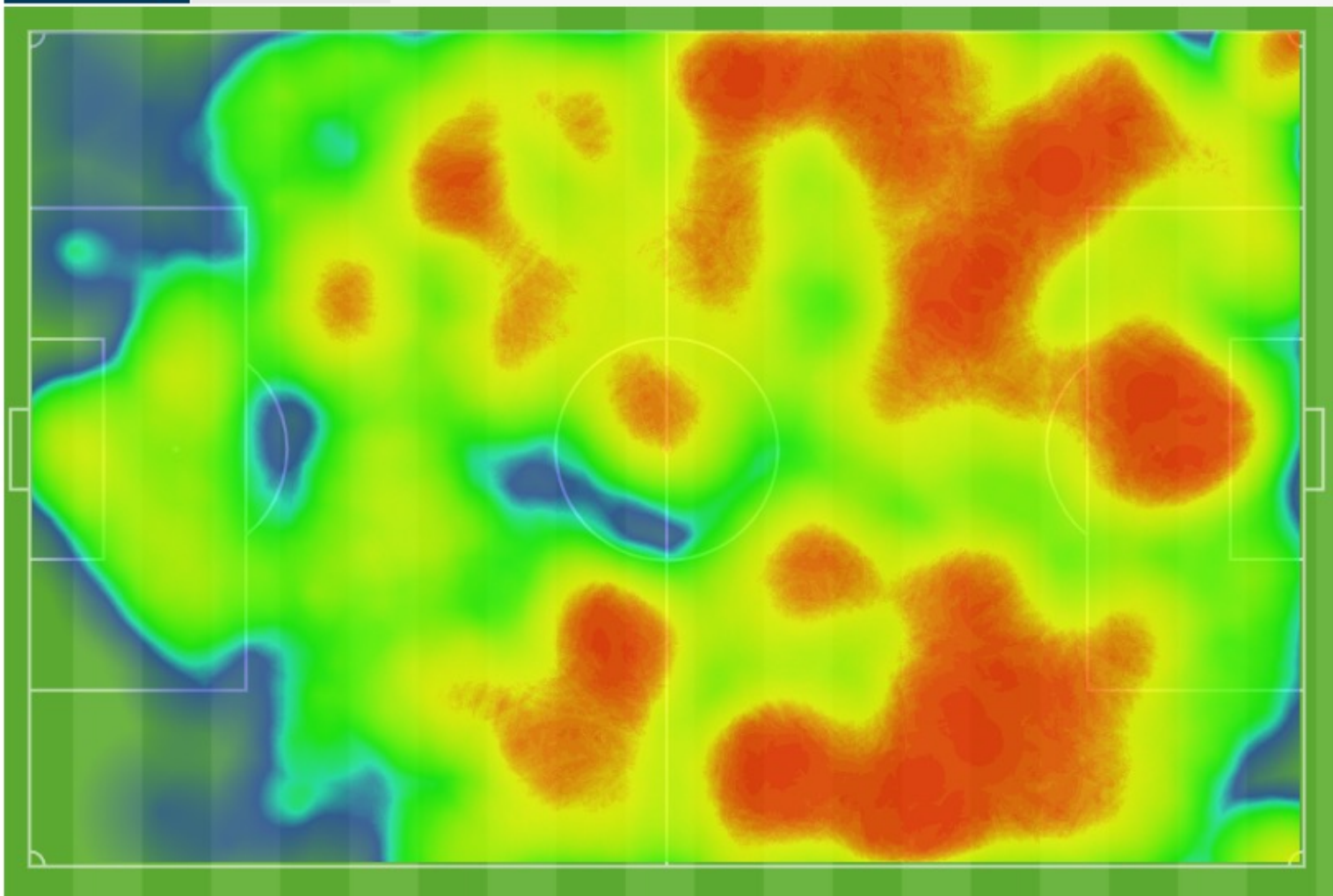
Auf Basis der eigenen Daten mehr Fehler in weniger Zeit finden



## FC Bayern München

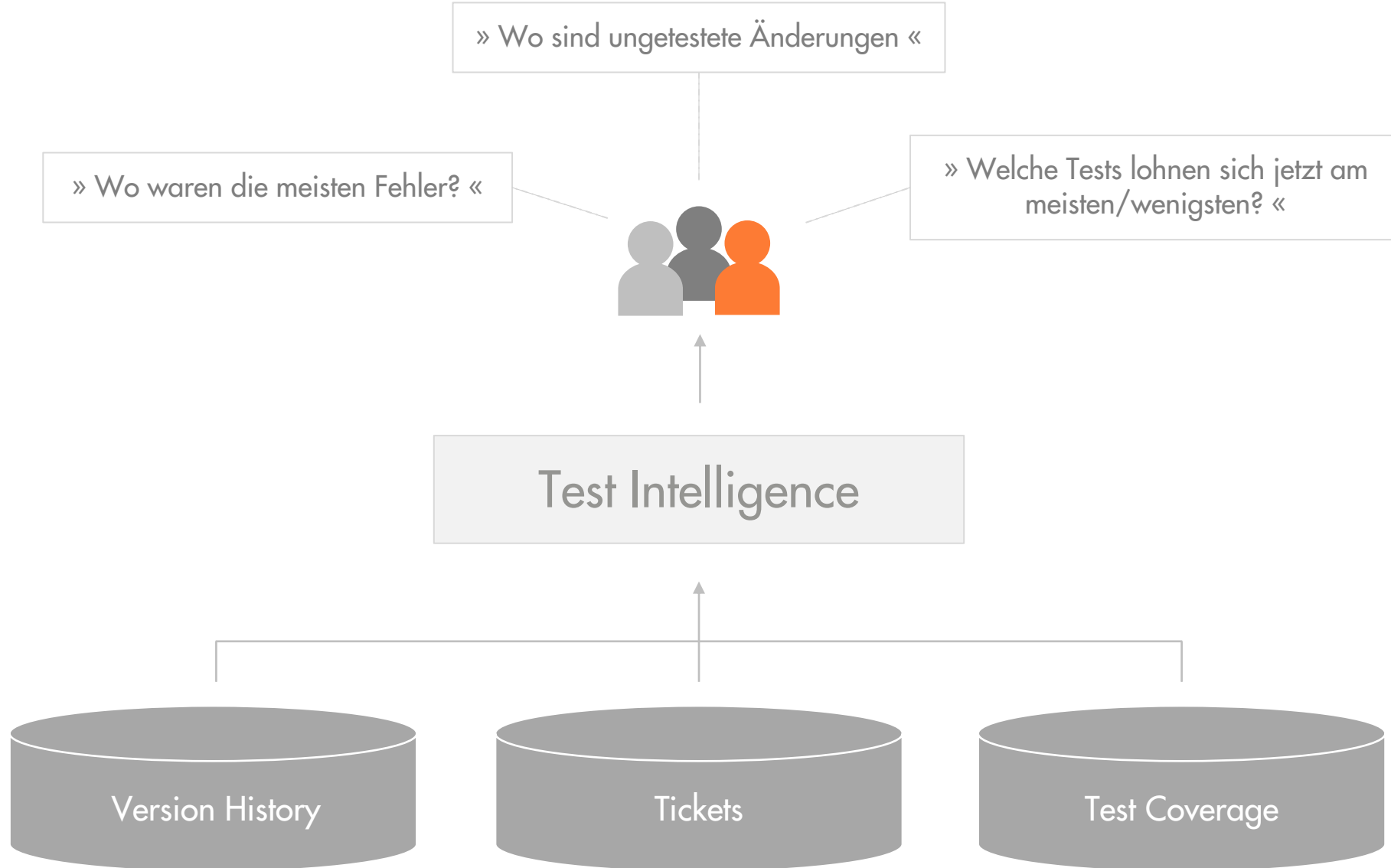
- FC Bayern München
- TW 1 Manuel Neuer
- AW 21 Lucas Hernández
- AW 5 Benjamin Pavard
- AW 4 Niklas Süle
- AW 27 David Alaba
- MF 29 Kingsley Coman
- MF 10 Leroy Sané
- MF 25 Thomas Müller
- MF 6 Joshua Kimmich
- MF 18 Leon Goretzka
- ST 13 Eric Maxim Choupo-Moting
- Einwechselfspieler
- 17 Jérôme Boateng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

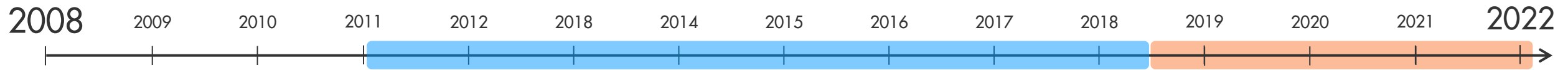
## Heatmap Touchmap

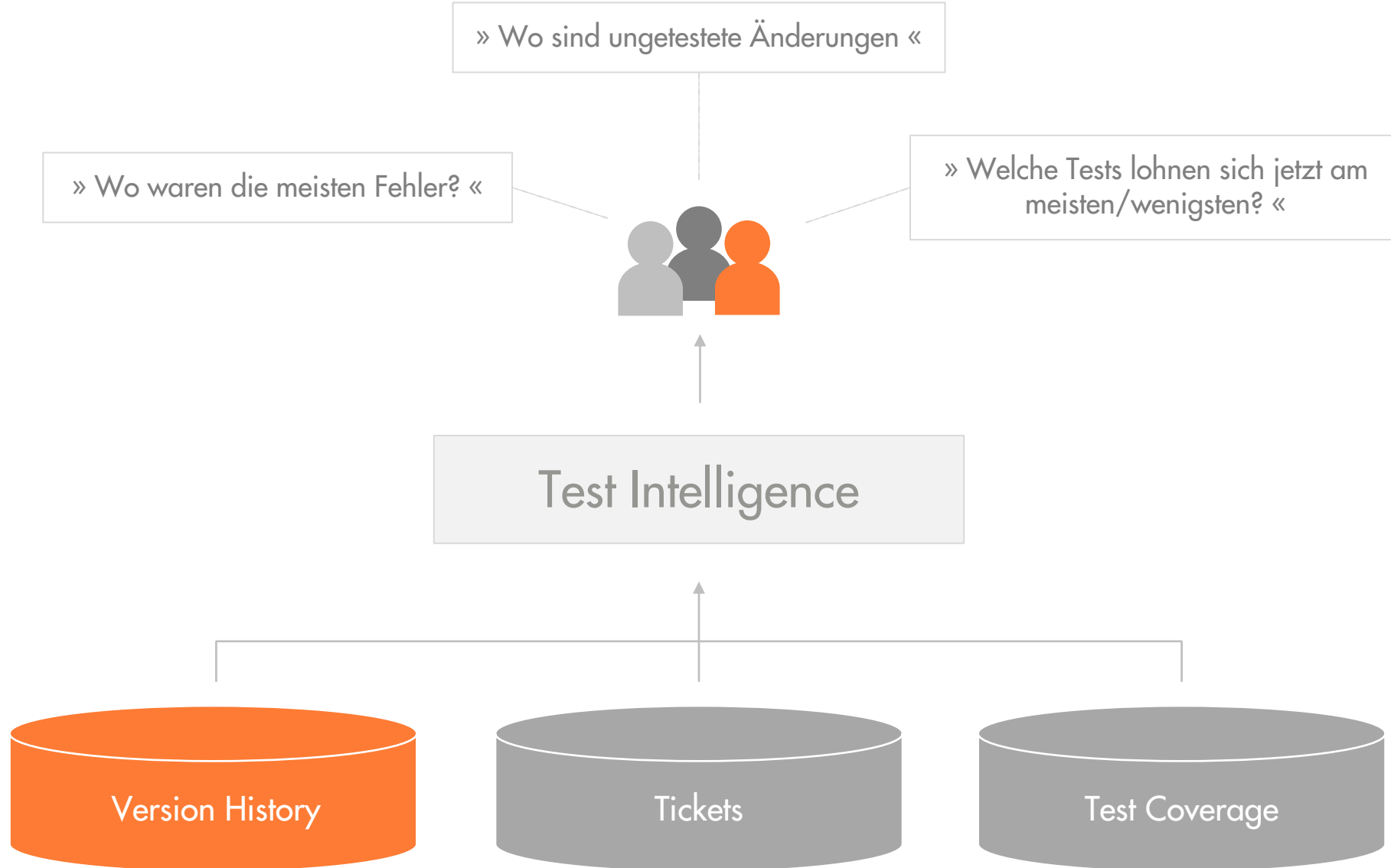


## Paris Saint-Germain

- 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
- Einwechselfspieler
- 25 Mitchel Bakker
- 18 Moise Kean
- 21 Ander Herrera
- 12 Rafinha







```

23
24
25
26 /* package */class NamespaceRenames {
27
28     /** Maps from old name fragment to new name fragment */
29     public Set<ImmutablePair<String, String>> namespaceRenames = new HashSet<
30
31     /**
32      * Computes a rename rule from an old and a new namespace name based on a
33      * type name correspondence. <code>
34      * For name1 = a.oldnamespace.b.c.D and name2 = a.newnamespace.b.c.D, res
35      * </code>
36      * */
37     public static ImmutablePair<String, String> computeRenameRule(
38         String fqTypeName1, String fqTypeName2, String separator) {
39
40         if (fqTypeName1.equals(fqTypeName2)) {
41             return null;
42         }
43
44         String commonSuffix = StringUtils.longestCommonSuffix(fqTypeName1,
45             fqTypeName2);
46         if (StringUtils.isEmpty(commonSuffix)) {
47             return null;
48         }
49
50         int separatorPosition = commonSuffix.indexOf(separator);
51         if (separatorPosition != -1) {
52             commonSuffix = commonSuffix.substring(separatorPosition);
53         }
54
55         String from = StringUtils.stripSuffix(commonSuffix, fqTypeName1);
56         String to = StringUtils.stripSuffix(commonSuffix, fqTypeName2);
57
58         return new ImmutablePair<String, String>(from, to);
59     }
60
61
62
63
64

```

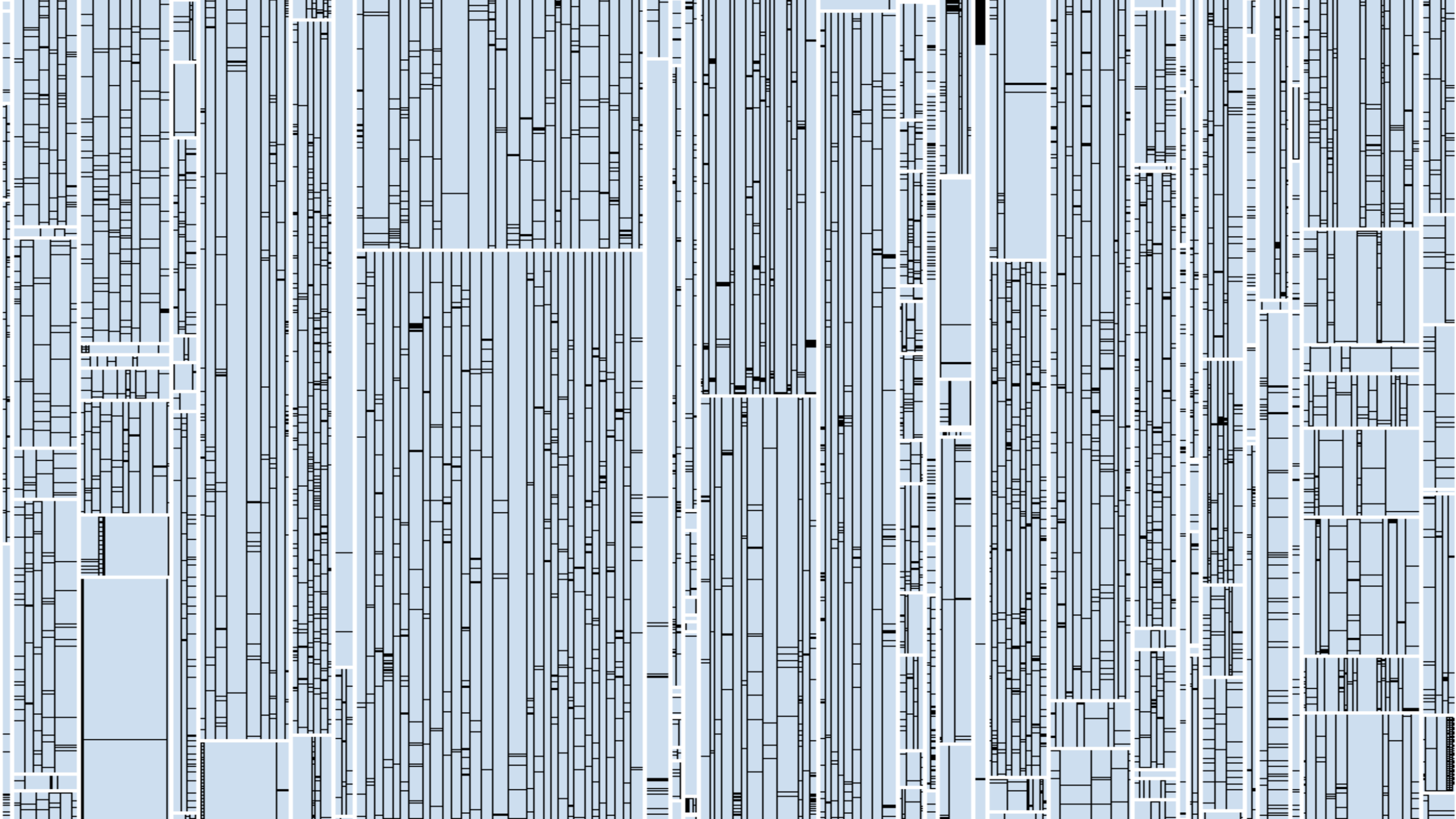
```

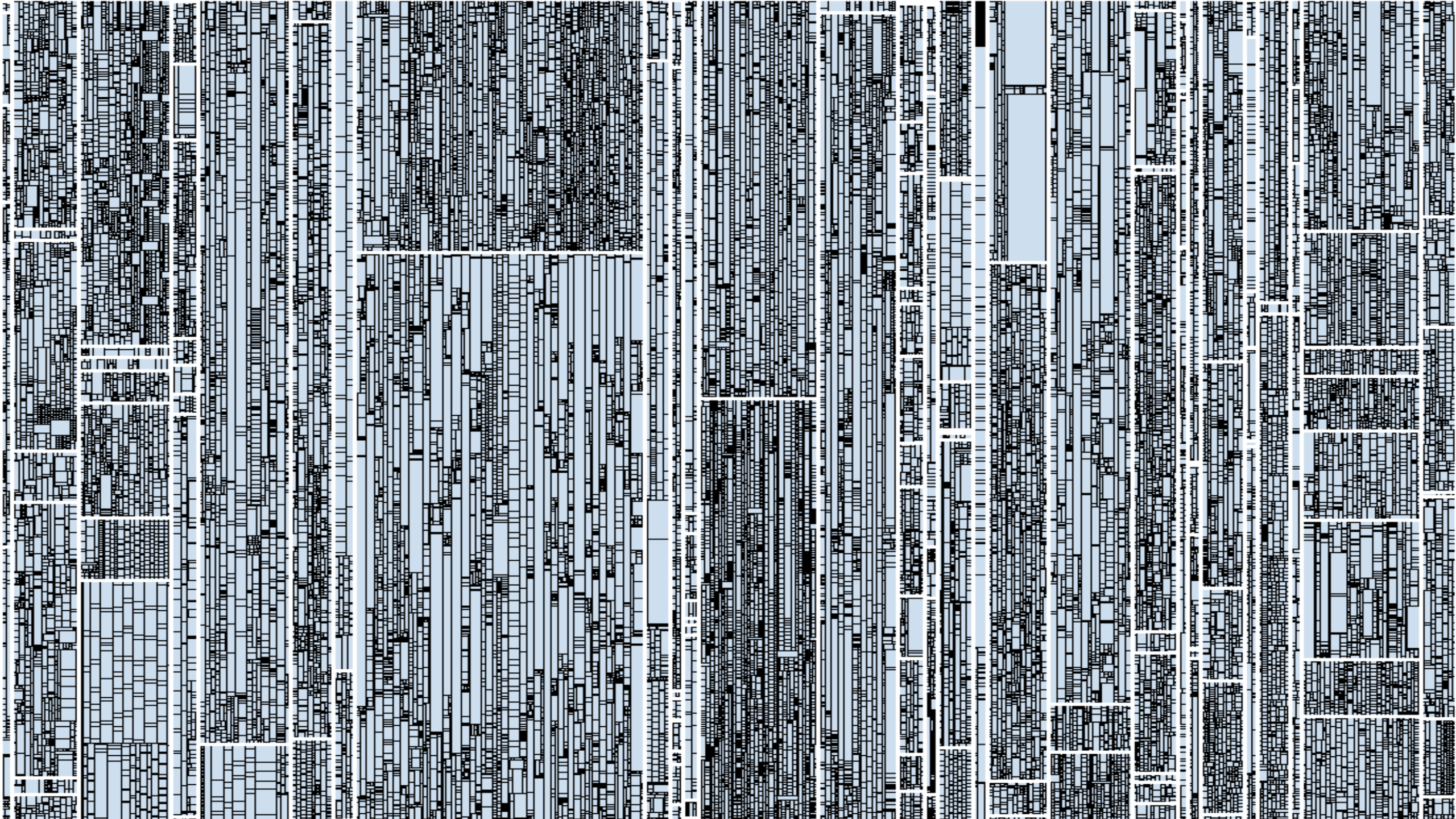
24
25
26 /* package */class NamespaceRenames {
27
28     /** Maps from old name fragment to new name fragment */
29     public Set<ImmutablePair<String, String>> namespaceRenames = new Has
30
31     // TODO (LH) Doc vs. method name : 'Compute' vs. 'find'
32     /**
33      * Computes a rename rule from an old and a new namespace name based
34      * type name correspondence. <code>
35      * For name1 = a.oldnamespace.b.c.D and name2 = a.newnamespace.b.c.D
36      * </code>
37      * */
38     // TODO (LH) Please reflect in identifier names that they refer to
39     // namespaces
40     public static ImmutablePair<String, String> findRenameRule(String na
41         String name2, String separator) {
42
43         if (name1.equals(name2)) {
44             return null;
45         }
46
47         // TODO (LH) Looks like StringUtils is missing a 'longestCommon
48         String commonSuffix = reverse(StringUtils.longestCommonPrefix(
49             reverse(name1), reverse(name2)));
50         if (StringUtils.isEmpty(commonSuffix)) {
51             return null;
52         }
53
54         int separatorPosition = commonSuffix.indexOf(separator);
55         if (separatorPosition != -1) {
56             commonSuffix = commonSuffix.substring(separatorPosition);
57         }
58
59         // TODO (LH) Please reflect in identifier names that these are
60         // namespaces
61         String from = StringUtils.stripSuffix(commonSuffix, name1);
62         String to = StringUtils.stripSuffix(commonSuffix, name2);
63
64         return new ImmutablePair<String, String>(from, to);
65

```

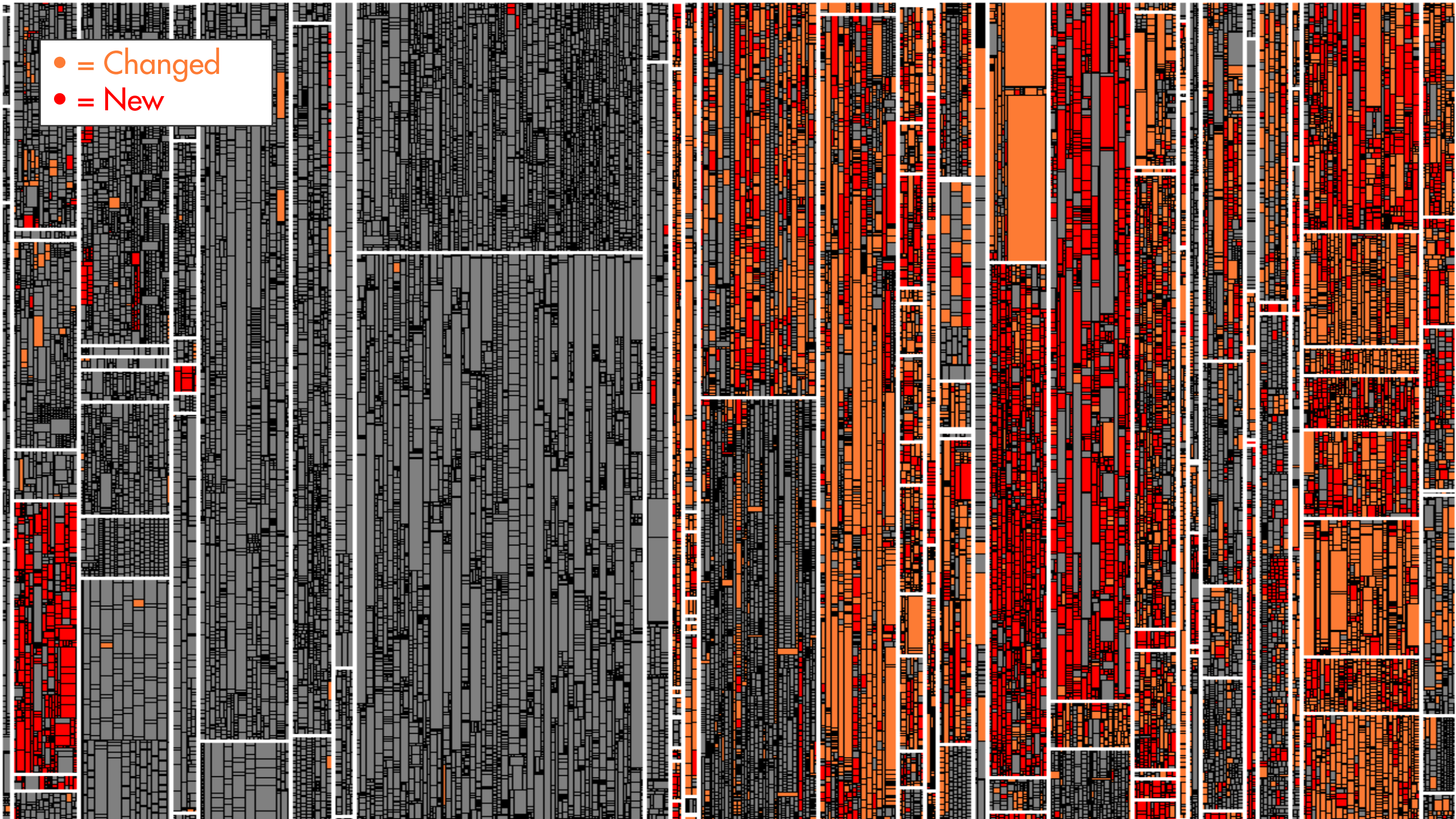


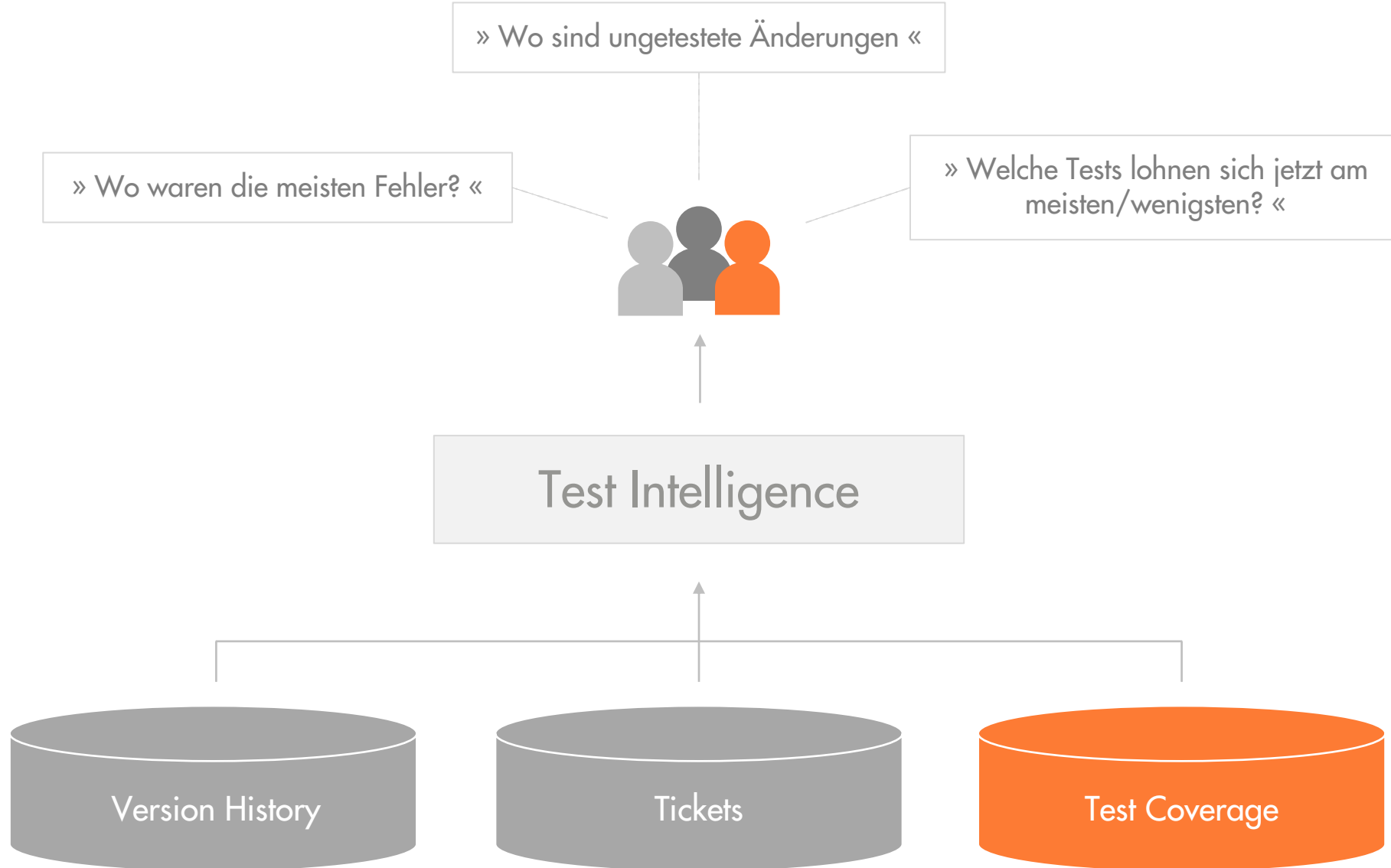






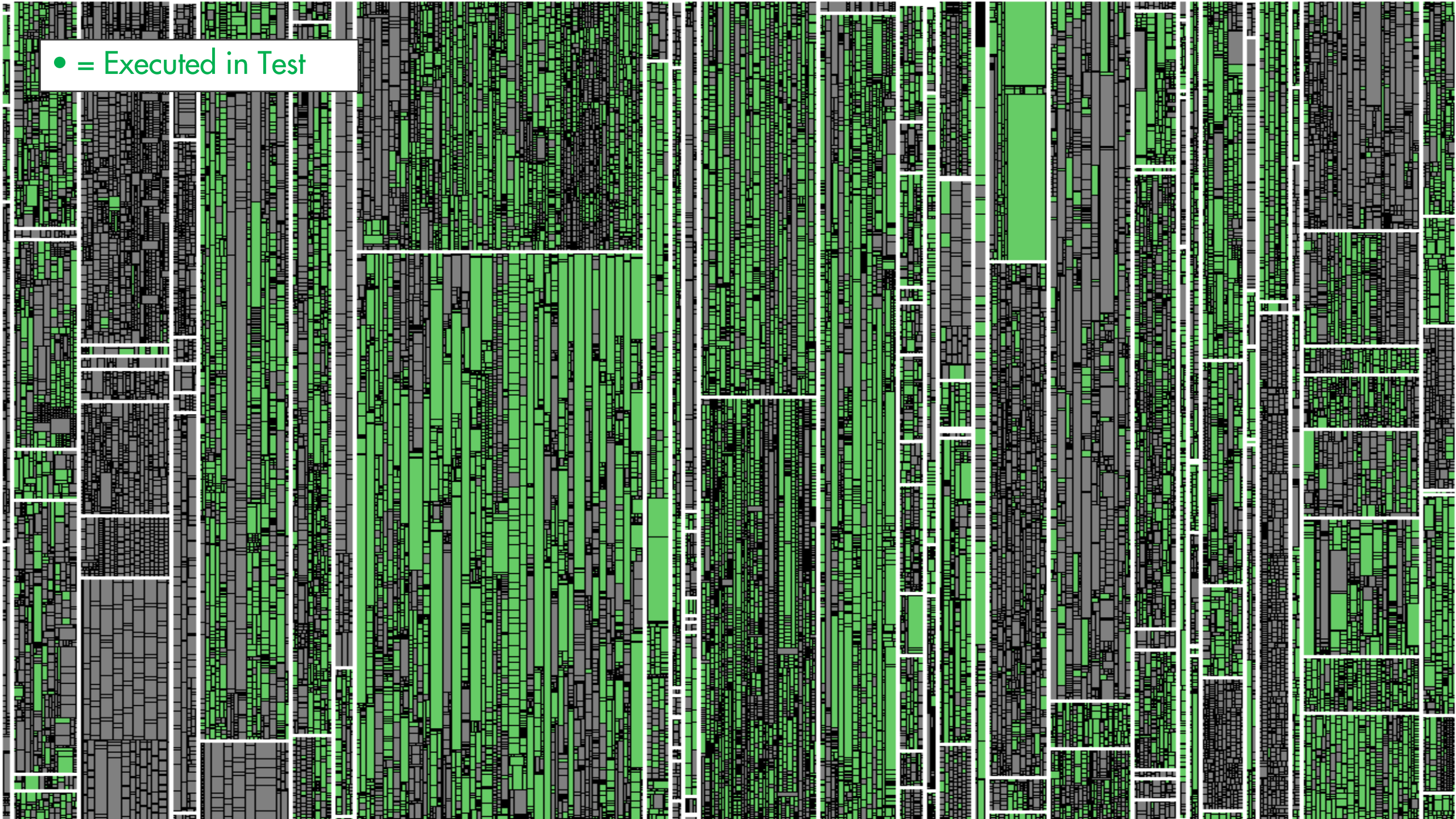
- = Changed
- = New

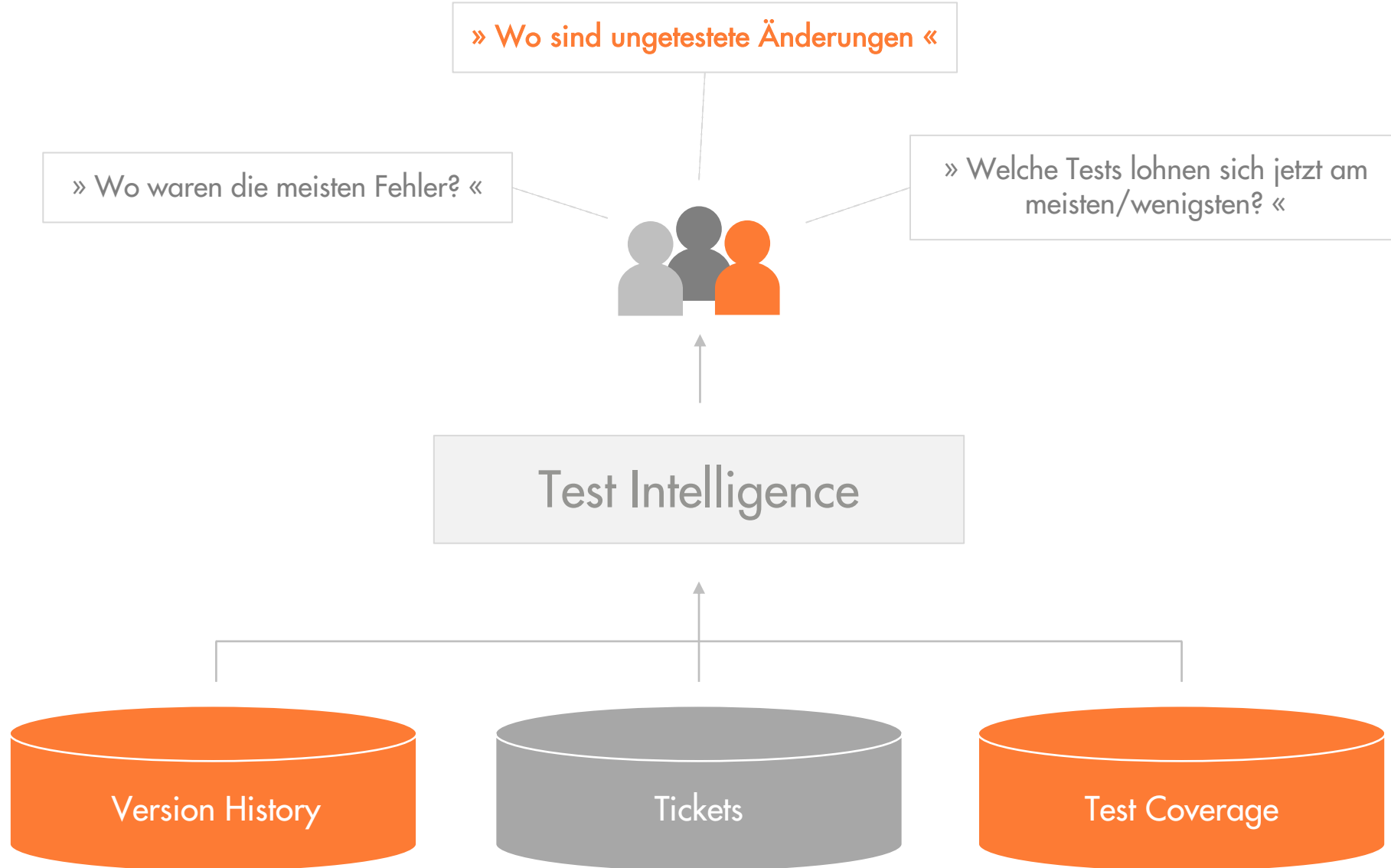




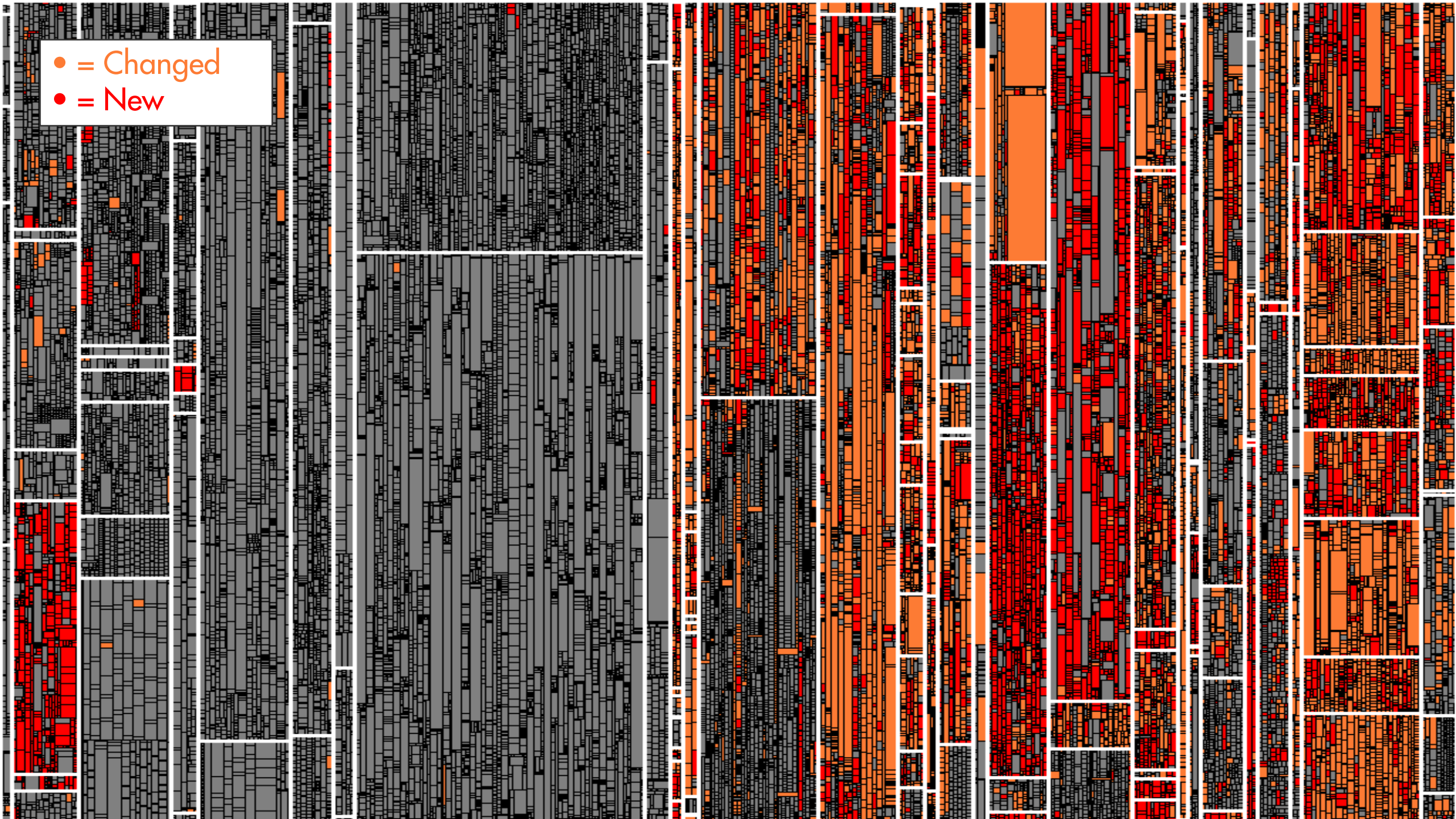
```
1  protected void calculateIndirectAmbiguities() {
2      Map<NucleotideCompound, List<NucleotideCompound>> equivalentMap = new HashMap<>>();
3
4      List<NucleotideCompound> ambiguousCompounds = new ArrayList<NucleotideCompound>();
5      for (NucleotideCompound compound : getAllCompounds()) {
6          if (!compound.isAmbiguous()) {
7              continue;
8          }
9          ambiguousCompounds.add(compound);
10     }
11
12     for (NucleotideCompound sourceCompound : ambiguousCompounds) {
13         Set<NucleotideCompound> compoundConstituents = sourceCompound.getConstituents();
14         for (NucleotideCompound targetCompound : ambiguousCompounds) {
15             Set<NucleotideCompound> targetConstituents = targetCompound.getConstituents();
16             if (targetConstituents.containsAll(compoundConstituents)) {
17                 NucleotideCompound lcSourceCompound = toLowerCase(sourceCompound);
18                 NucleotideCompound lcTargetCompound = toLowerCase(targetCompound);
19                 checkAdd(equivalentMap, sourceCompound, targetCompound);
20                 checkAdd(equivalentMap, sourceCompound, lcTargetCompound);
21                 checkAdd(equivalentMap, targetCompound, sourceCompound);
22                 checkAdd(equivalentMap, lcTargetCompound, sourceCompound);
23                 checkAdd(equivalentMap, lcSourceCompound, targetCompound);
24                 checkAdd(equivalentMap, lcSourceCompound, lcTargetCompound);
25             }
26         }
27     }
28
29     for (NucleotideCompound key : equivalentMap.keySet()) {
30         List<NucleotideCompound> vals = equivalentMap.get(key);
31         for (NucleotideCompound value : vals) {
32             addEquivalent((C) key, (C) value);
33             addEquivalent((C) value, (C) key);
34         }
35     }
36 }
```

● = Executed in Test





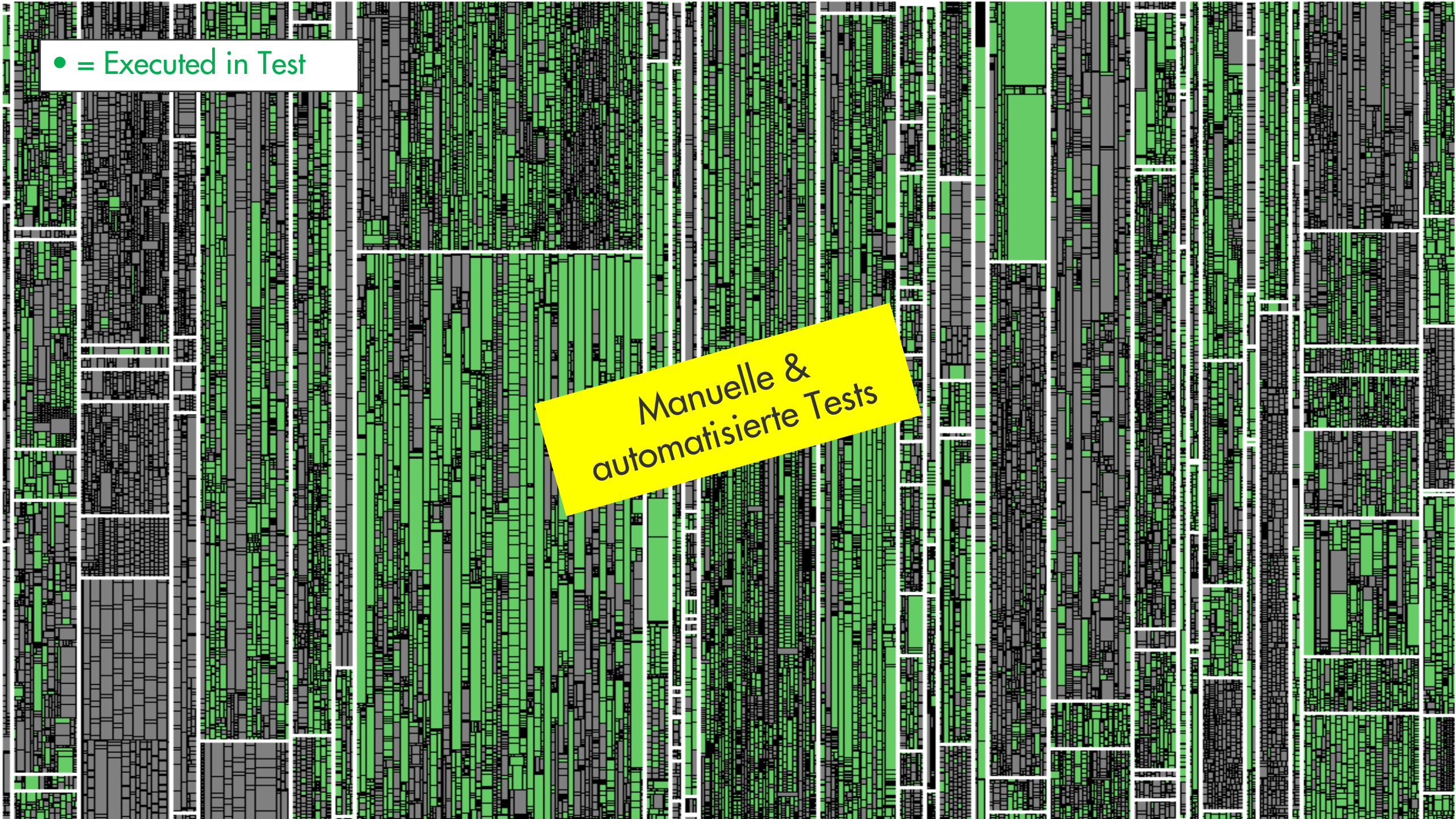
- = Changed
- = New



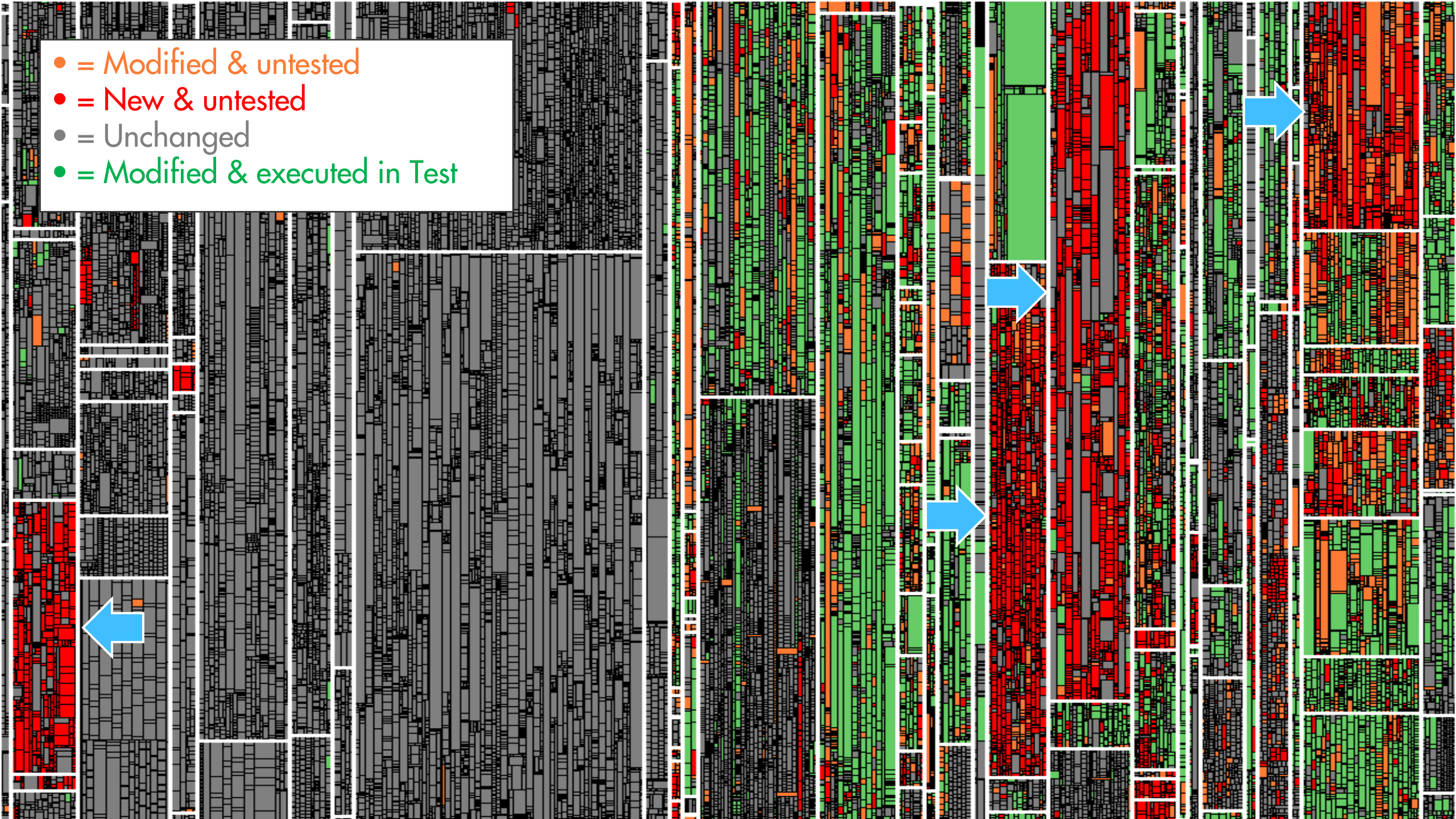



















● = Executed in Test





Manuelle &  
automatisierte Tests




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




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

Issue # ▾	Subject		Test Gap
<a href="#">TS-14421</a>	Get rid of TestGapSynchronizer block	Done 	0% 
<a href="#">TS-14733</a>	Remove Dataflow blocks	Done 	22% 

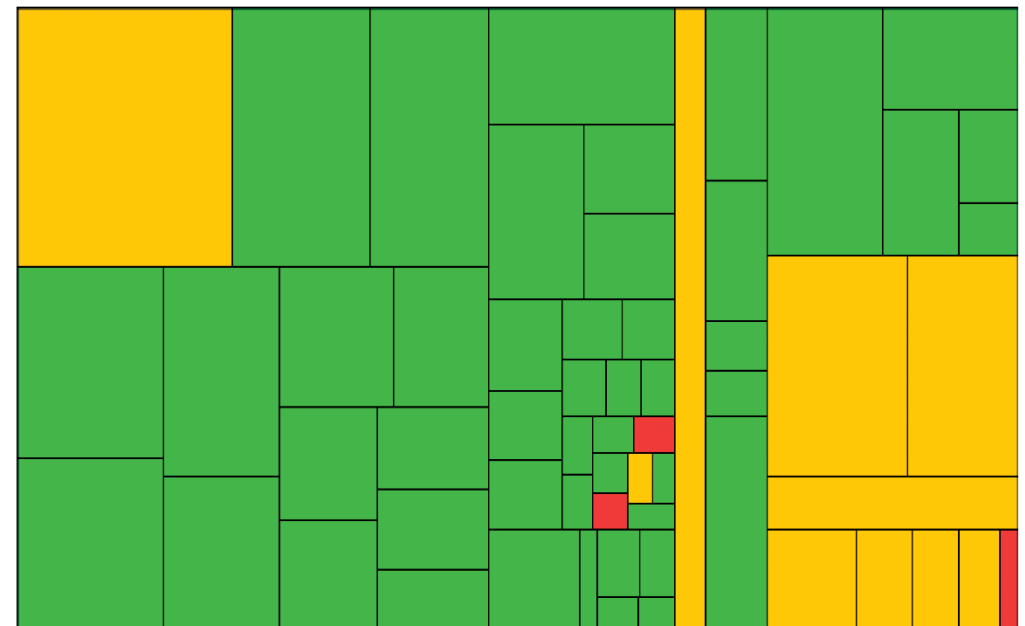
**Done Issue TS-14733 - Remove Dataflow blocks**

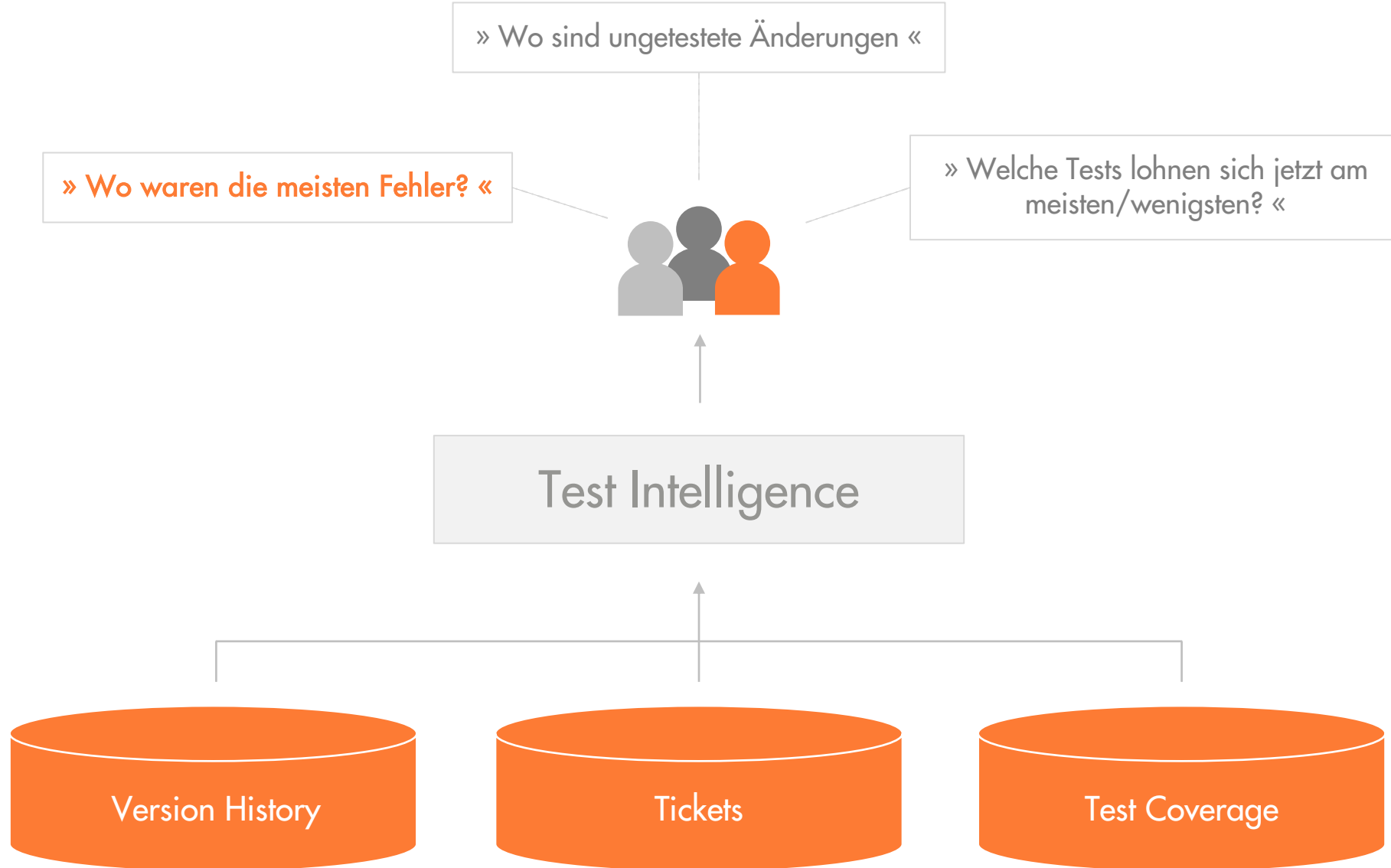
Creator:  (on Apr 06 2018 19:44) Last update: Aug 24 2018 09:32

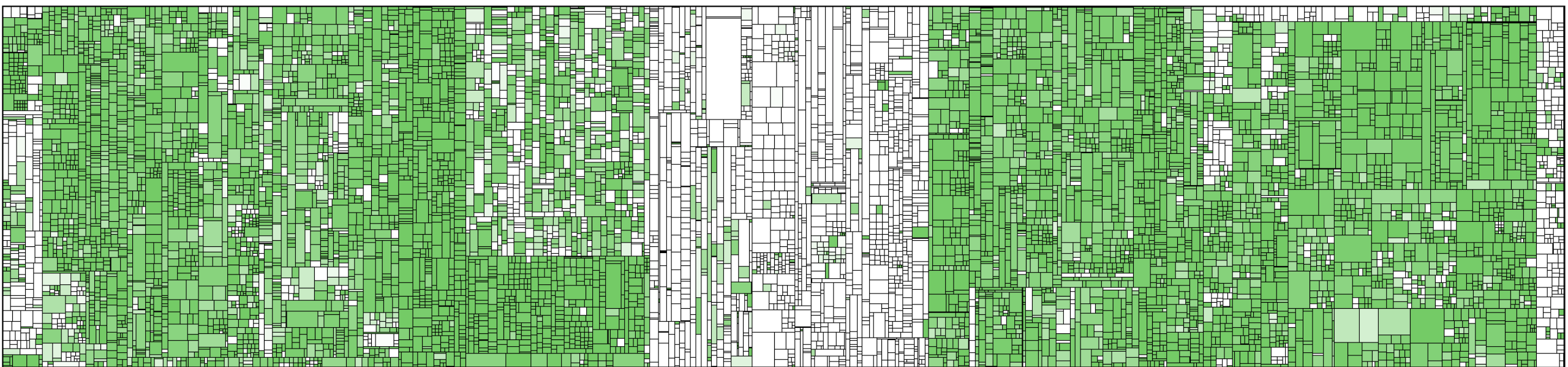
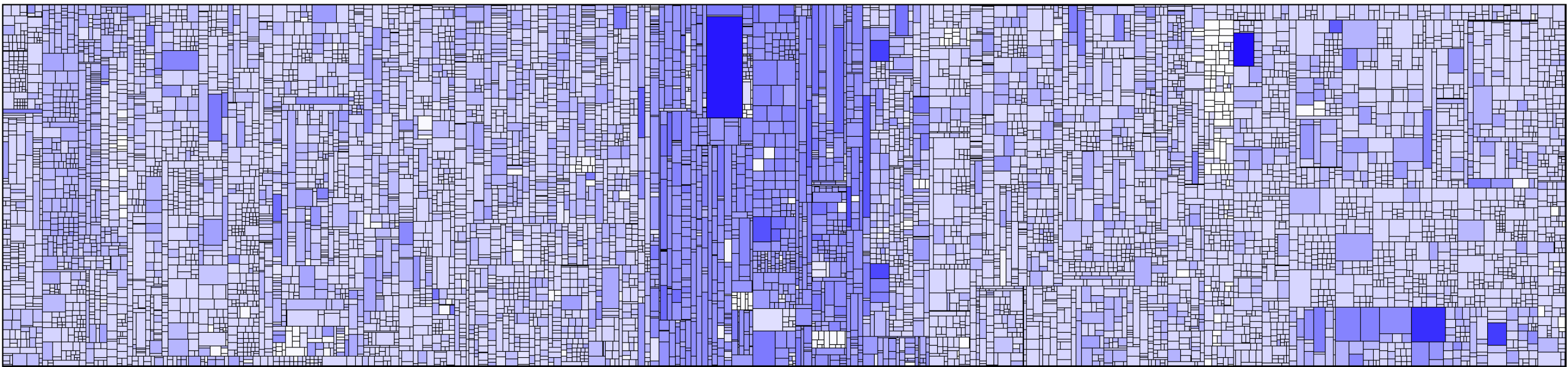
Assignee: 

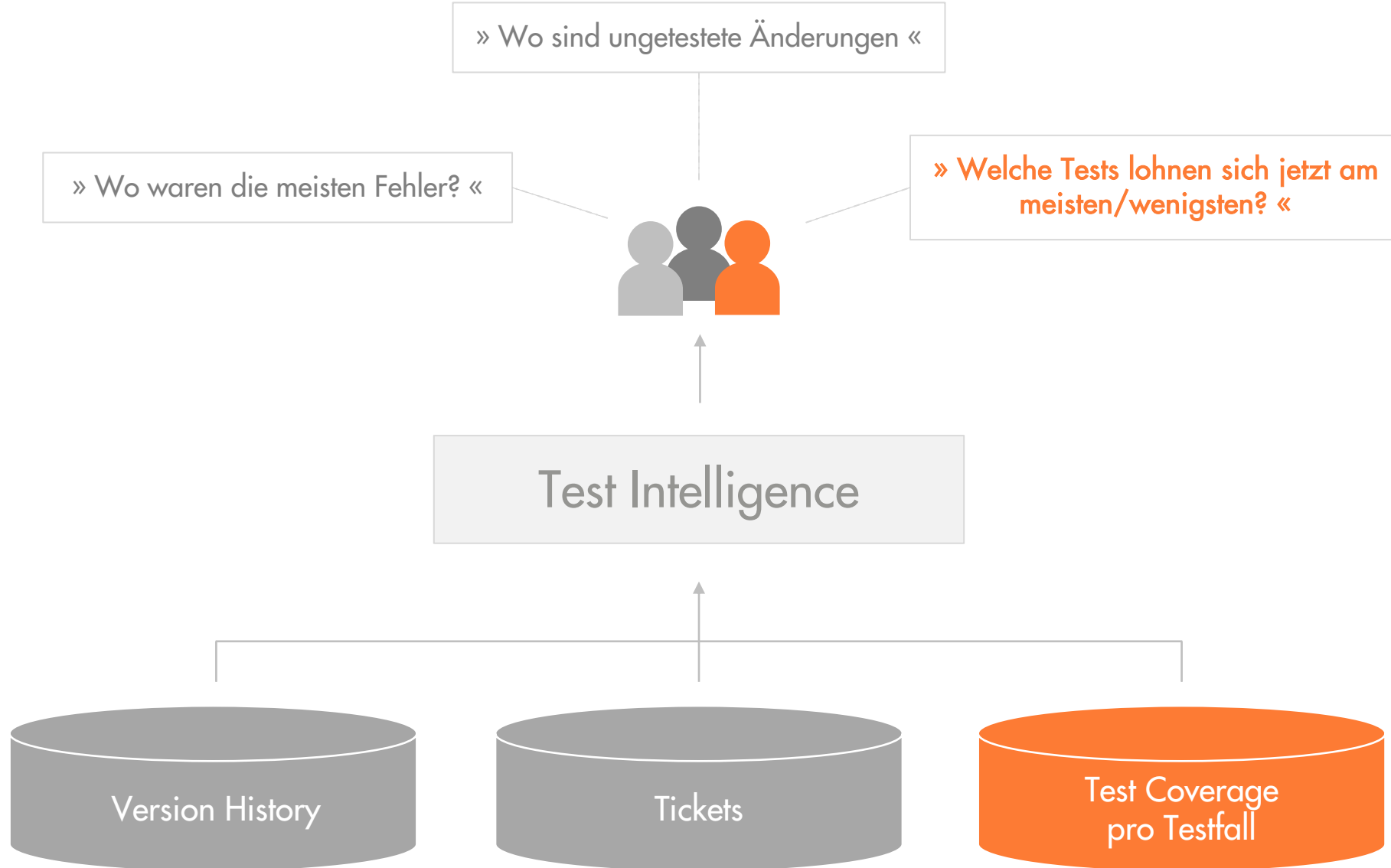
Project	Type	Priority	Resolution	Fix Version
TS	Maintenance	Normal	Green	Teamscale 4.5
Component	Labels	Affected Version	Customer	Customer Issue
Backend	Performance			
Epic Name	Freshdesk URL	Merge Request		
		<a href="https://git.cqse.eu/cqse/teamscale/3621">https://git.cqse.eu/cqse/teamscale/3621</a>		

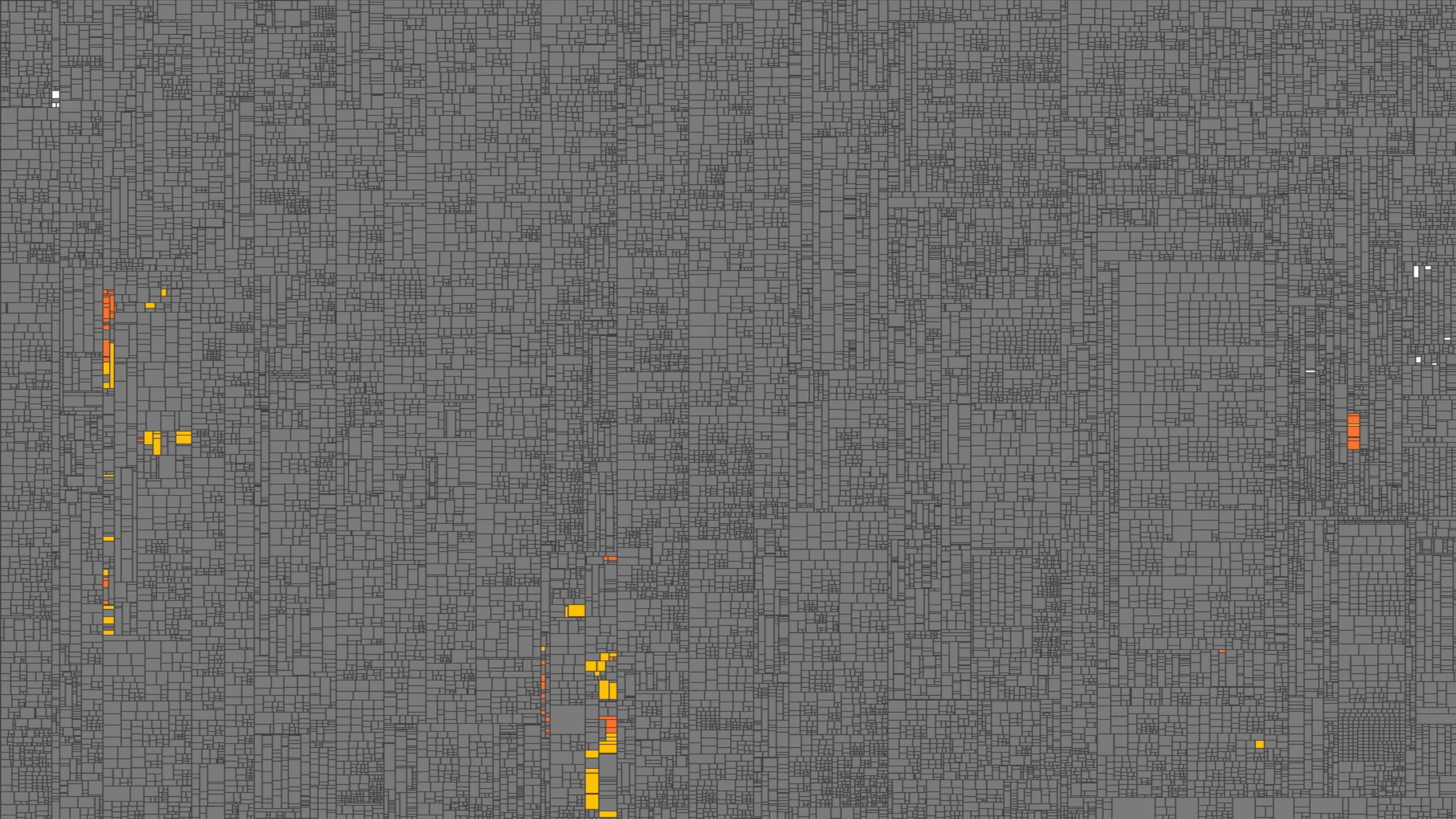
Aug 15 2018 12:37–Now | Test Gap: 22%



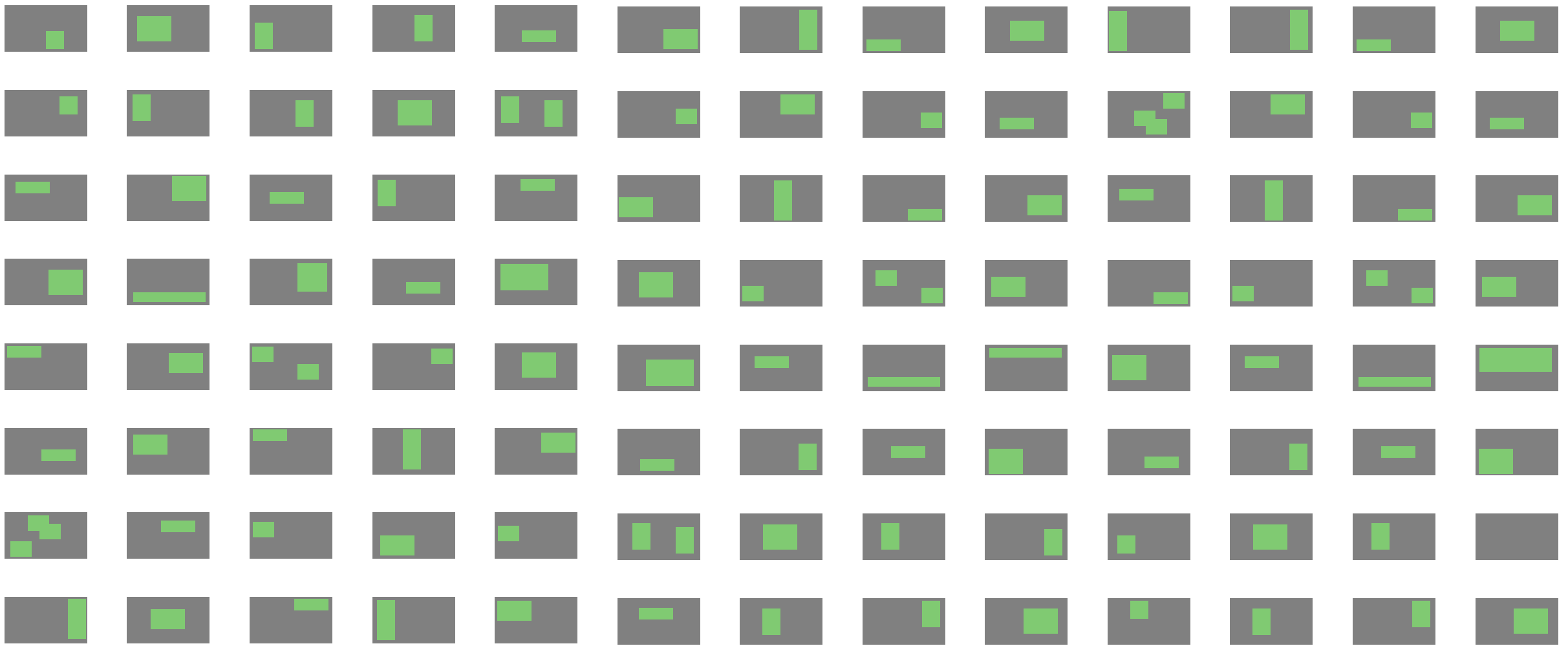








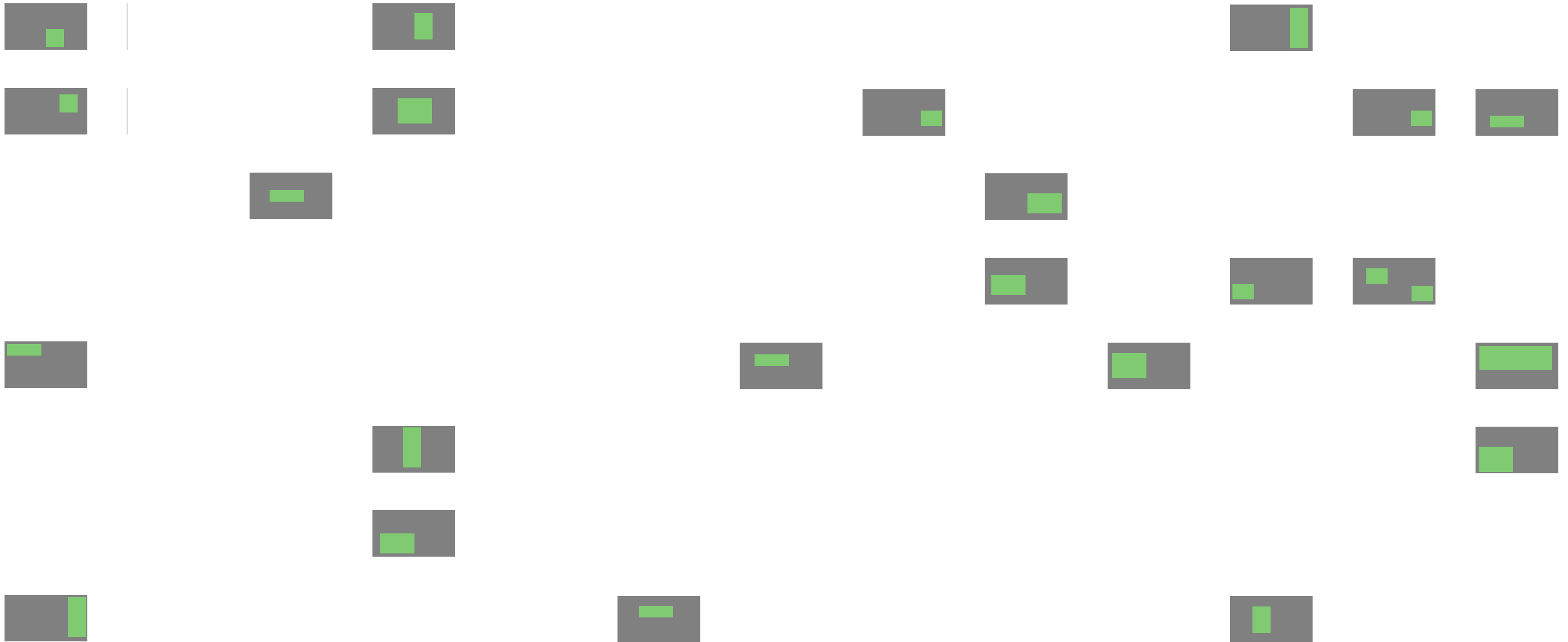




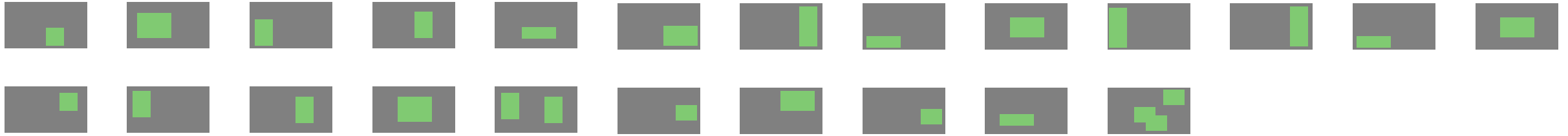
# Schritt 1: Selektion betroffener Testfälle



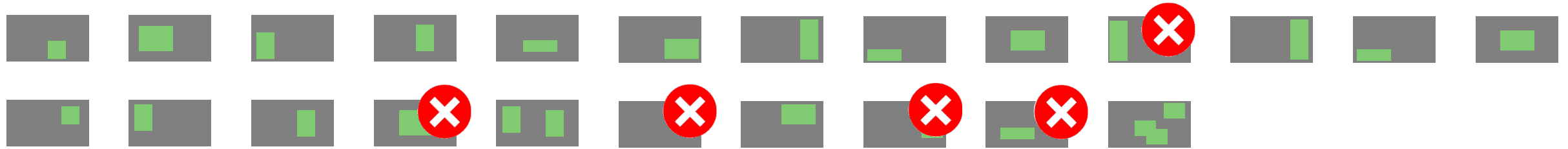
# Schritt 1: Selektion betroffener Testfälle



# Schritt 1: Selektion betroffener Testfälle

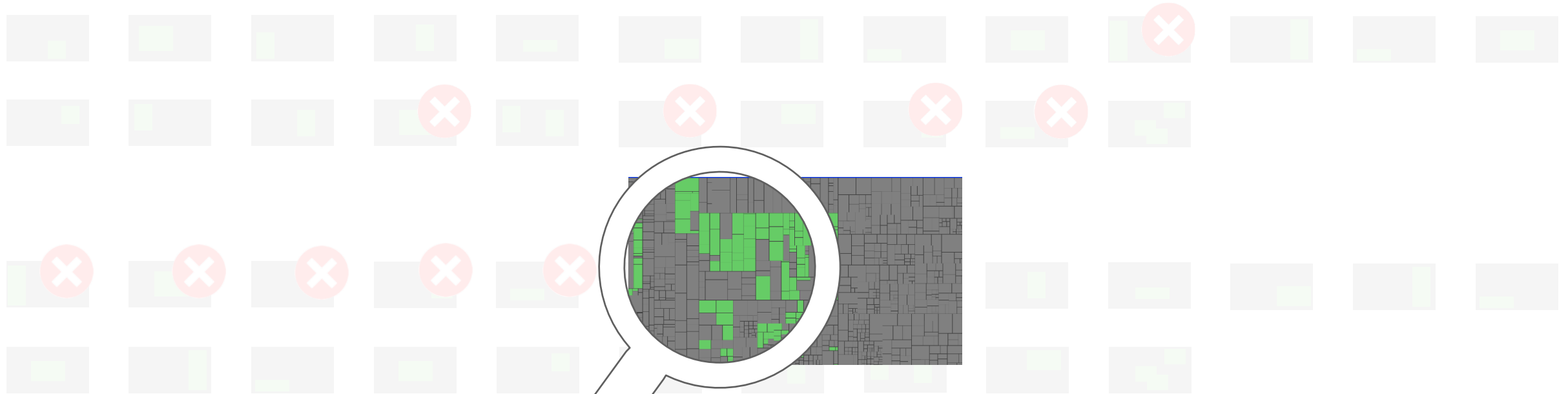


## Schritt 2: Priorisierung selektierter Testfälle





## Schritt 2: Priorisierung selektierter Testfälle



Change coverage

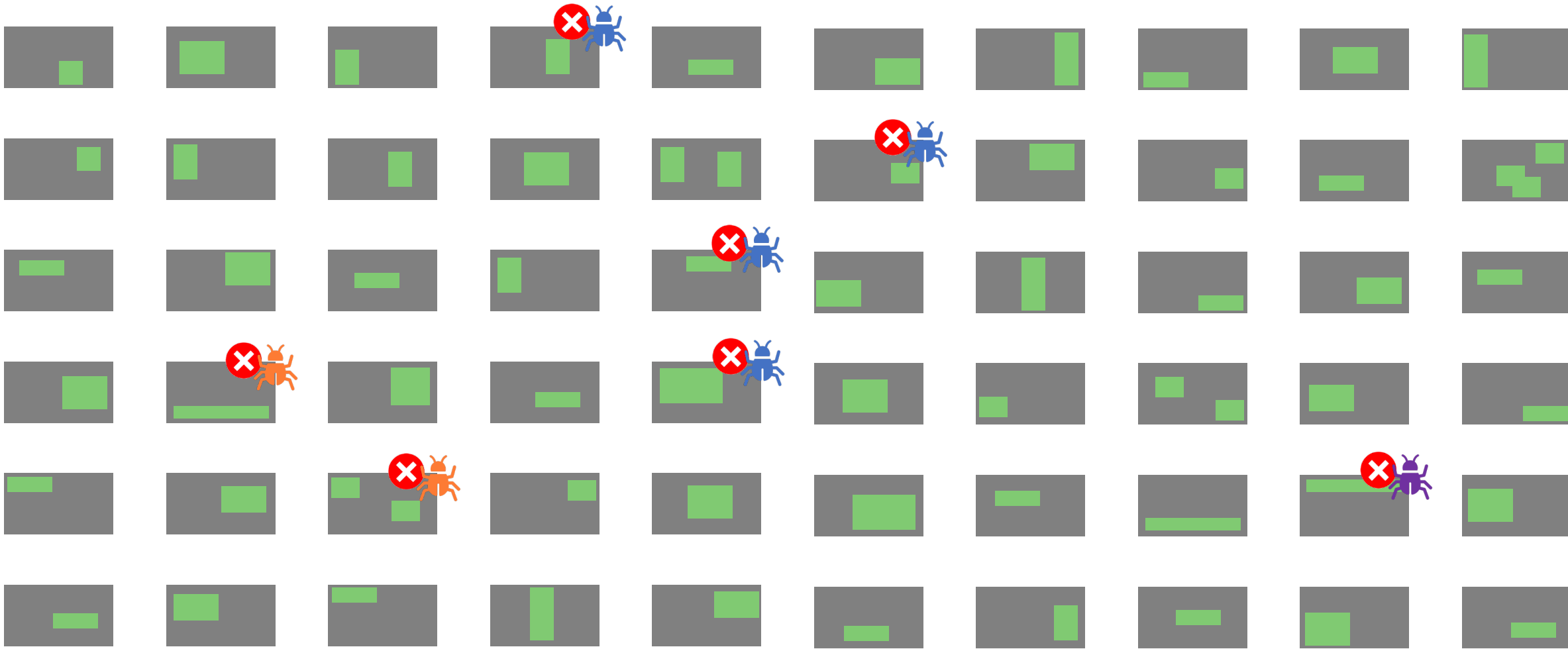
Execution time

## Schritt 2: Priorisierung selektierter Testfälle

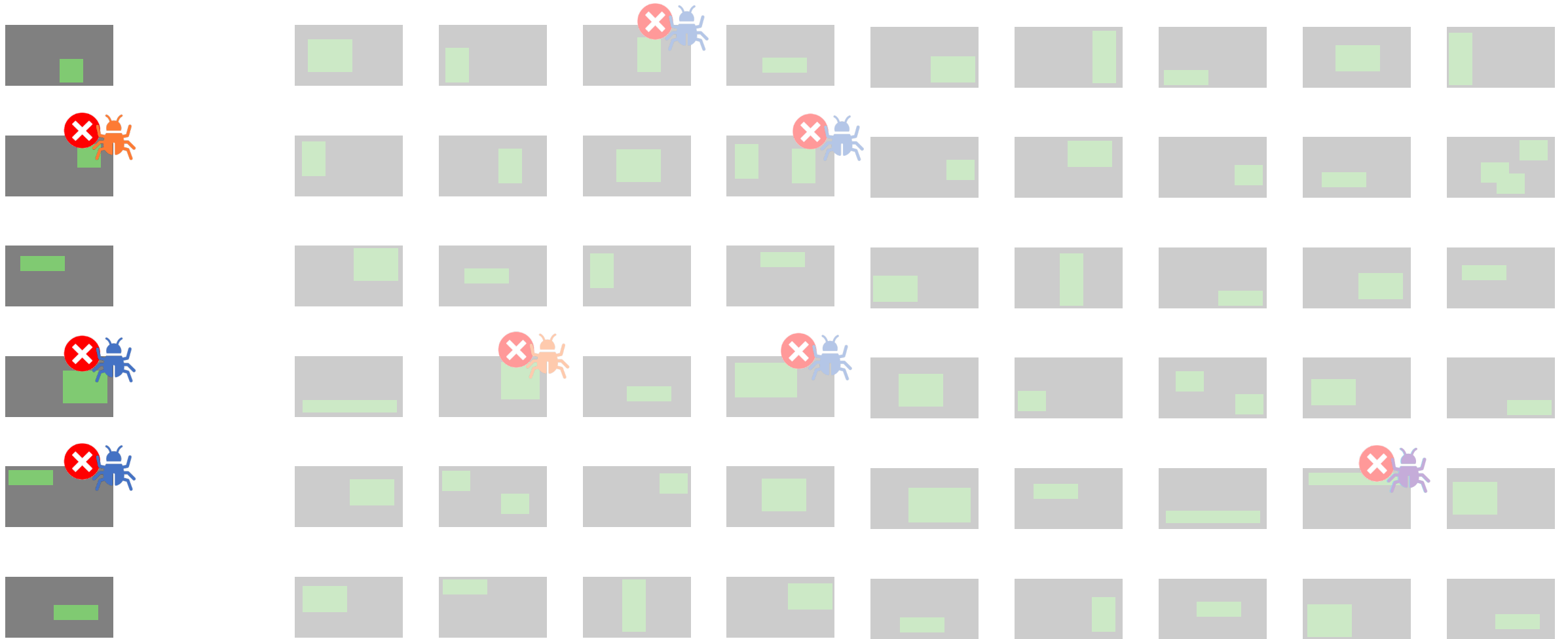


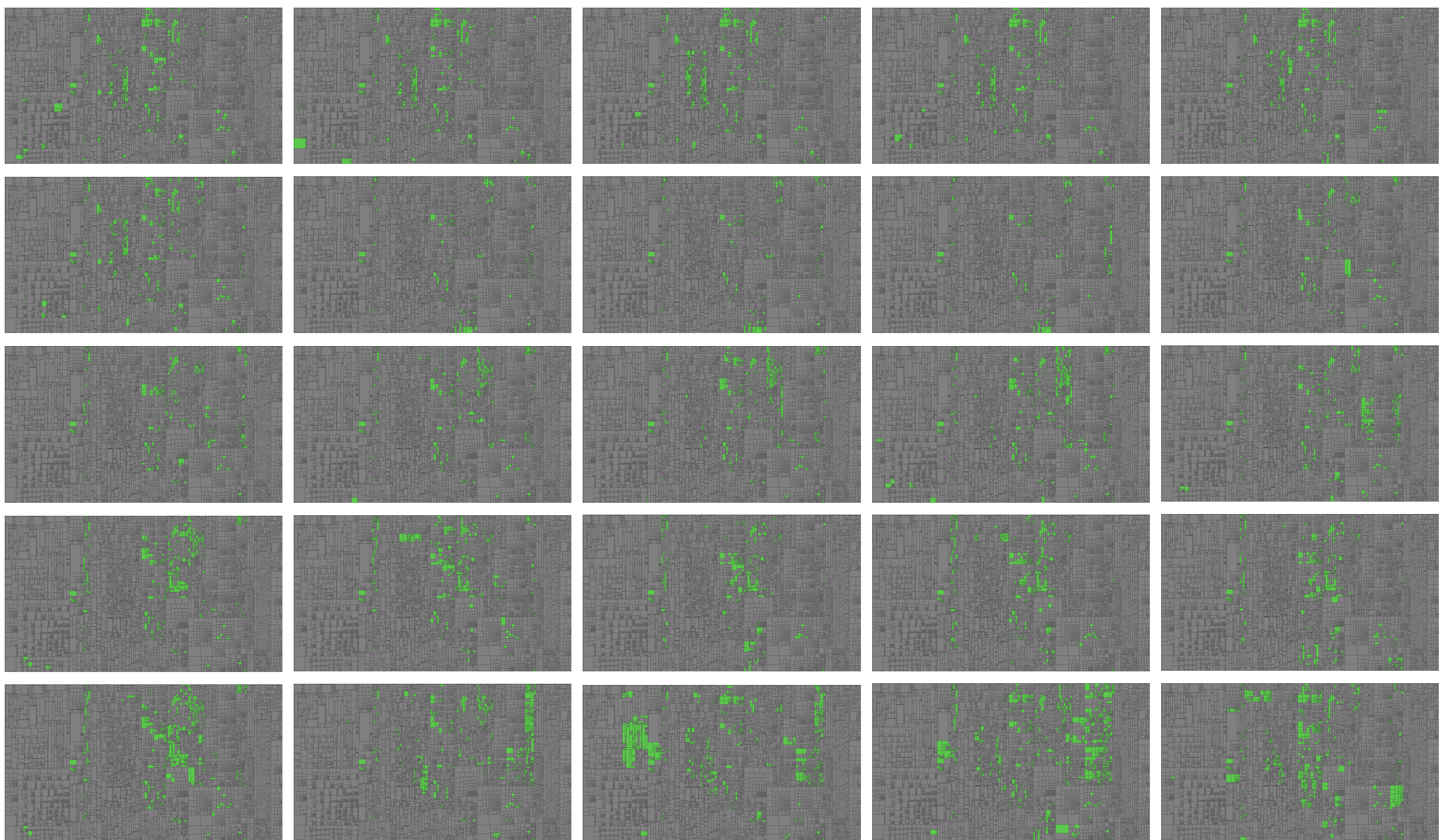


Geht das auch einfacher?



# Pareto-Testing: 80/20 Optimierung





Dashboard

Activity

Findings

Metrics

Test Gaps

Files

Components

Issues

Unlinked Changes

Pareto Ranking

Quality Control

Architecture

Delta

Project Configuration

# Pareto Ranking

Switch to query language

Show tests from:

Batch TestsNightly

GeVo TestsNightly

KettentestsNightly

Name

Duration

> 0s

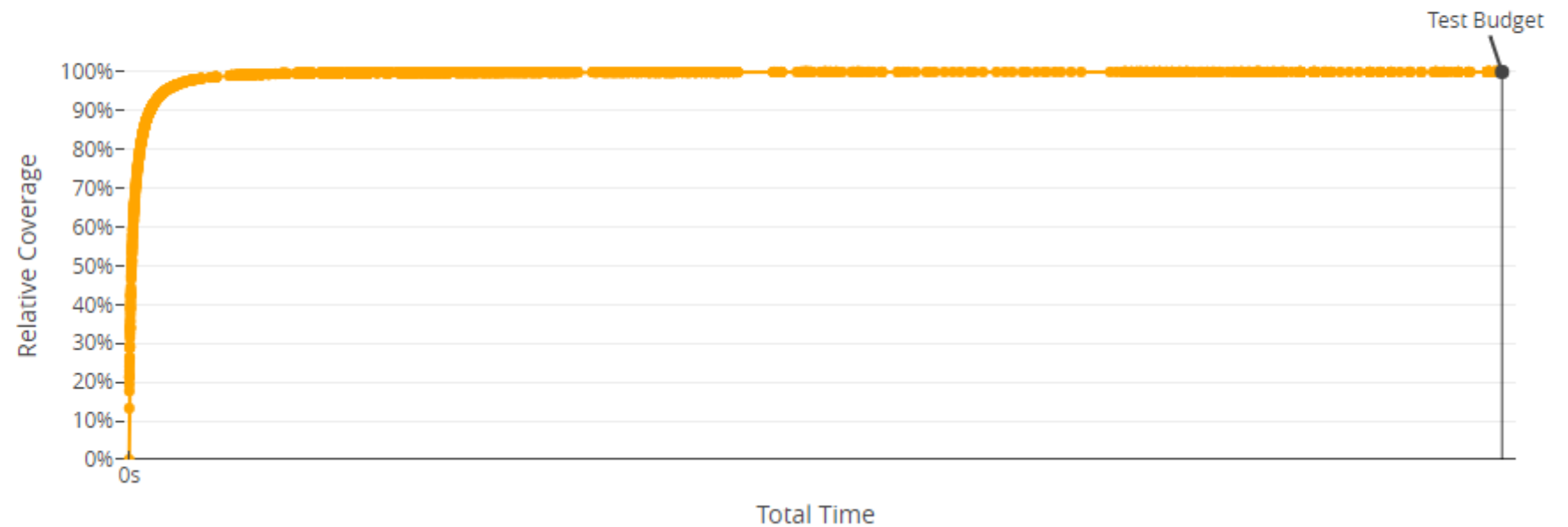
Result

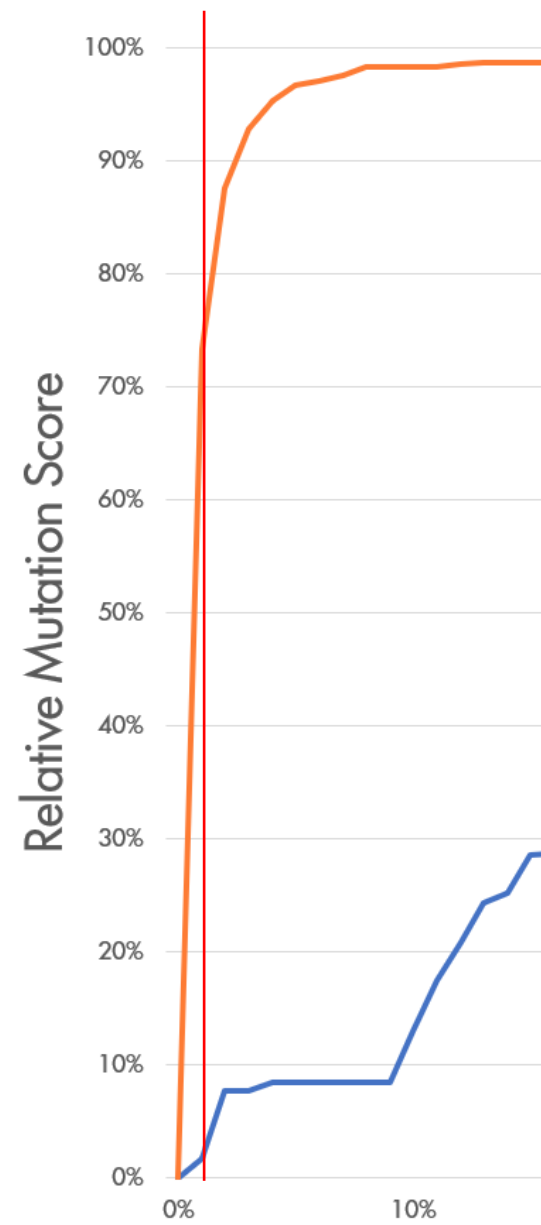
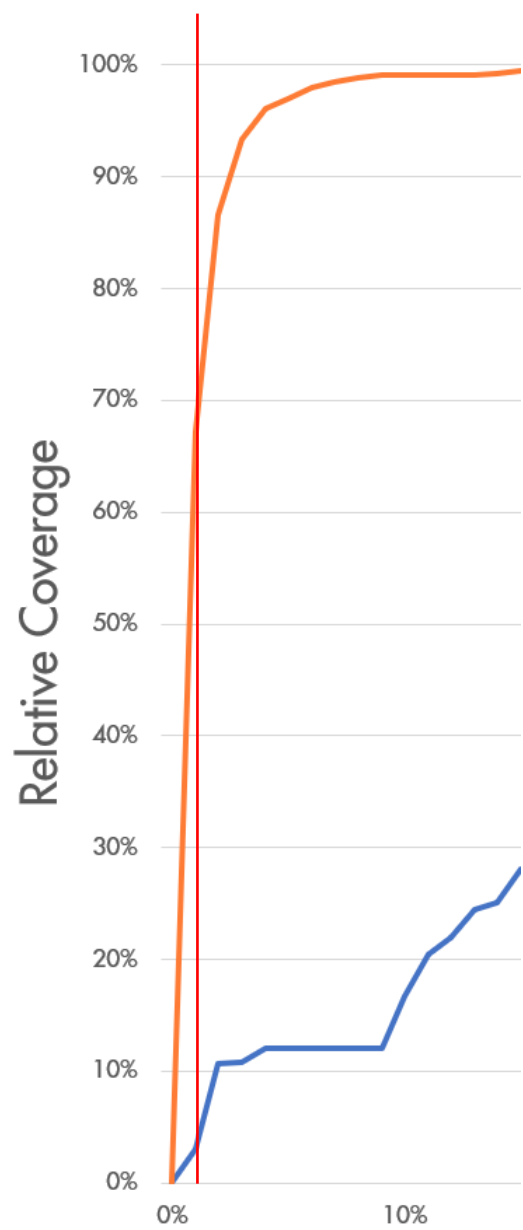
All results

+ Save query

Compute Ranking

Pareto Ranking computes a subset of the selected tests that **maximises test coverage** for the given time budget.





# Test-Impact-Analyse

Tests werden für jeden Lauf **passend zu Änderungen** ausgewählt

90% der Mutanten in 2% der Zeit

Erfordert kontinuierliche Messung der Coverage und Integration der Test-Auswahl in die CI / Testautomatisierung.

**Stärkere Beschleunigung des Feedbacks (bei höherem Aufwand)**

# Pareto-Optimierung

Tests werden **unabhängig von Änderungen** ausgewählt

90% der Mutanten in 11% der Zeit

Einmalige Messung der Coverage reicht aus.

**Viel geringerer Aufwand**

# Fast Feedback From Long-Running Tests

Test-Impact Analysis and Pareto Optimization  
for Ever-Growing Test Suites

Wednesday, July 20, 2022

17:00 bis 18:30 CEST

Registration:

[cqse.eu/workshop-8020-en-2206-saec](https://cqse.eu/workshop-8020-en-2206-saec)

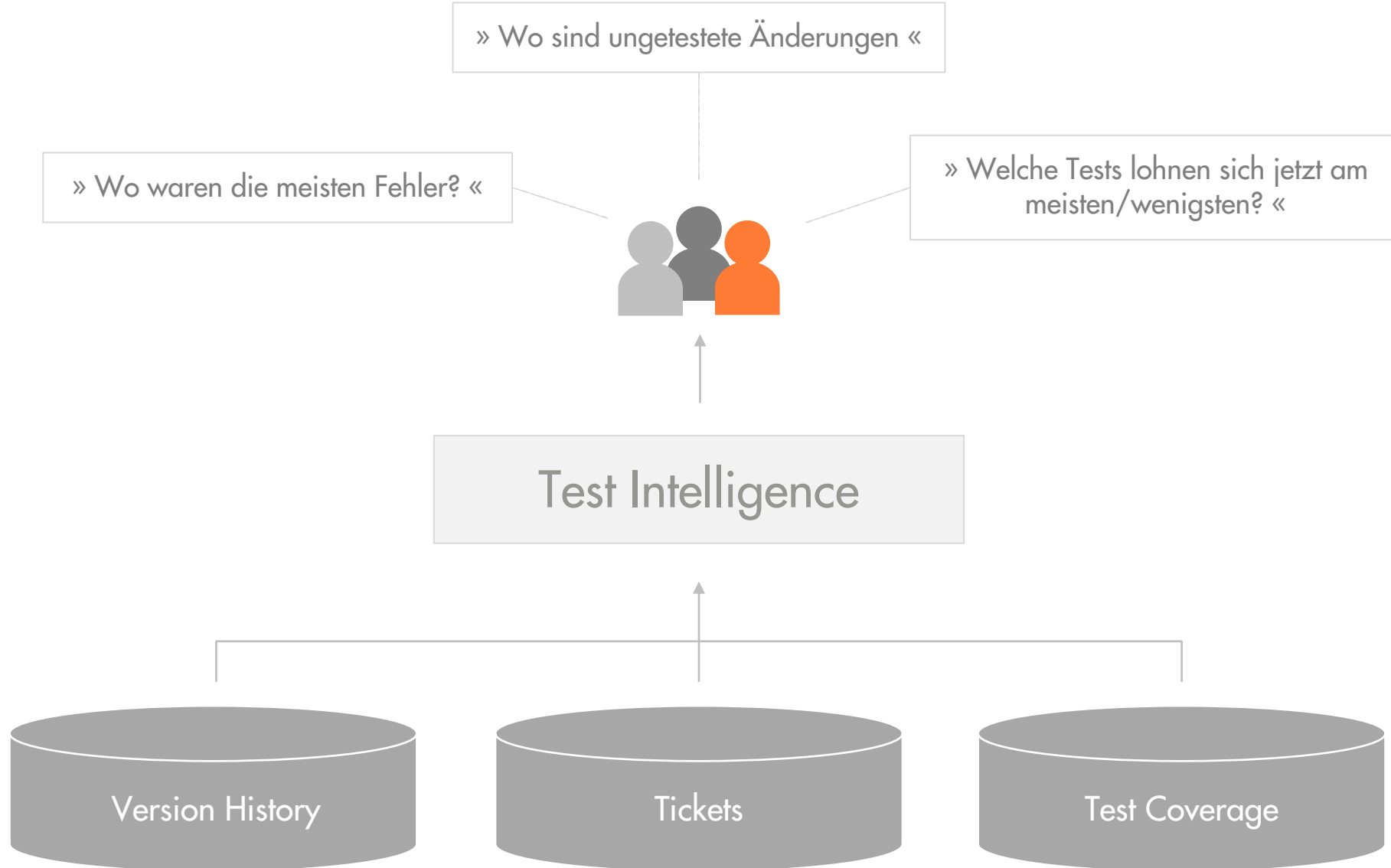


Dr. Elmar Jürgens



Fabian Streitl





# Kontakt – Ich freue mich auf Fragen 😊



Dr. Sven Amann

amann@cqse.eu +49 172 1860063

CQSE GmbH  
Centa-Hafenbrädl-Str 59  
81249 München  
[www.cqse.eu](http://www.cqse.eu)

