Immer kürzere Testphasen?

Mit Ticket Coverage verhindern, dass wichtige Änderungen ungetestet bleiben

Dr. Elmar Juergens

CQSE

Über Mich

Forschung

- Clone Detection, Test-Gap-Analyse, …
- PC Mitglied von MSR, ICPC, ICSE, …

Beratung

- Gründer
- Qualitäts-Bewertung & Qualitäts-Controlling

Gesellschaft für Informatik

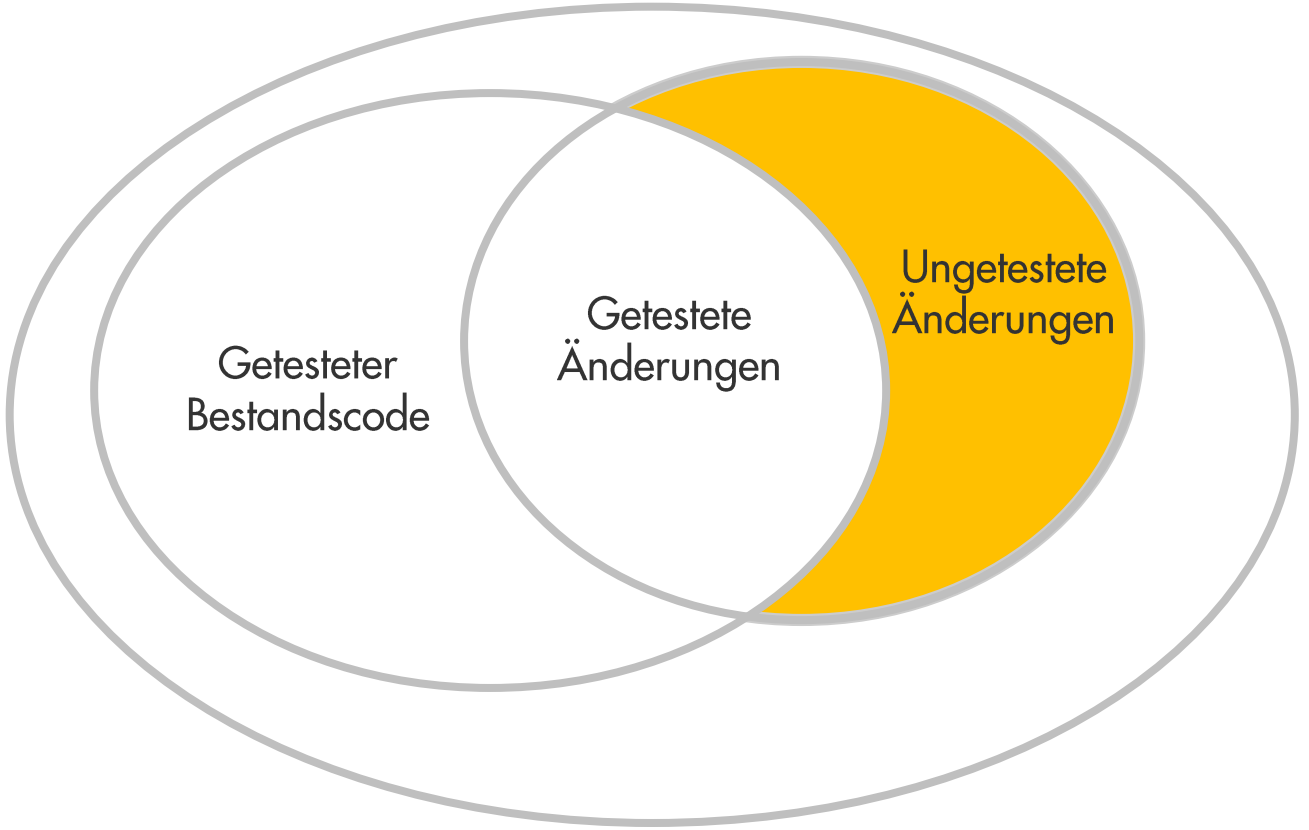
- Zum Junior-Fellow ernannt
- Erfahrungsaustausch Forschung <-> Praxis

Wieviele Änderungen sind ungetestet?

Studie: C# System @ Munich Re

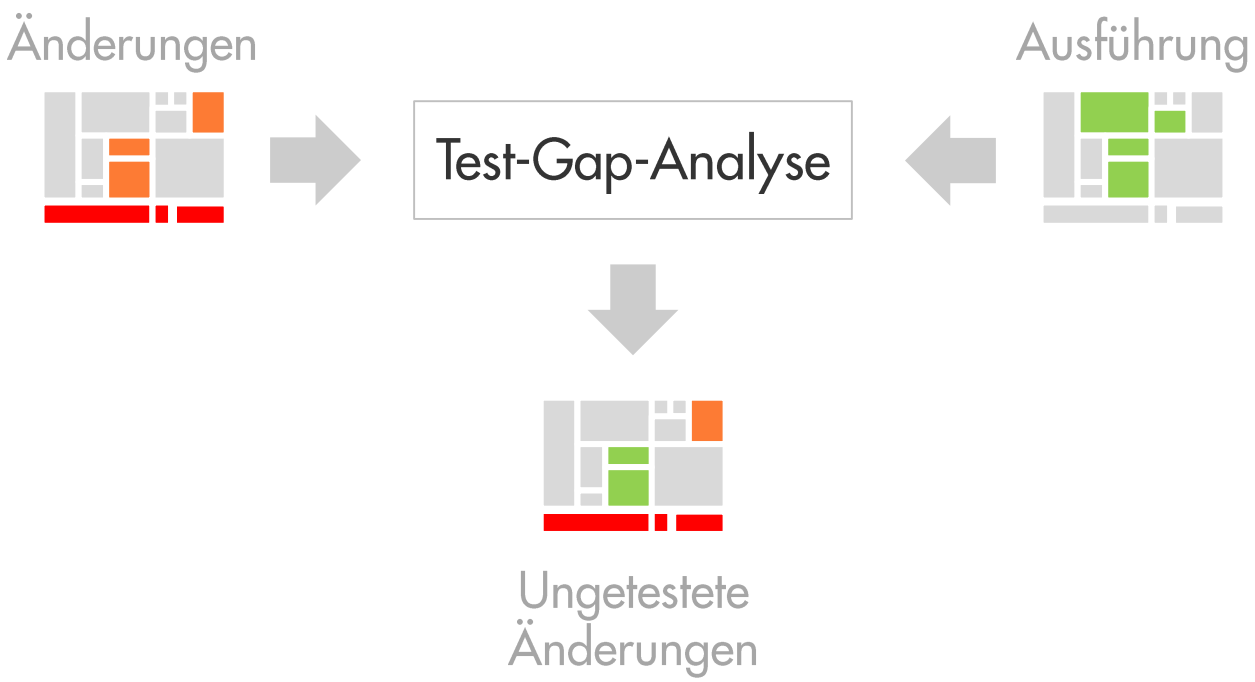
Release A:
15% Code neu/geändert,
>50% ungetestet

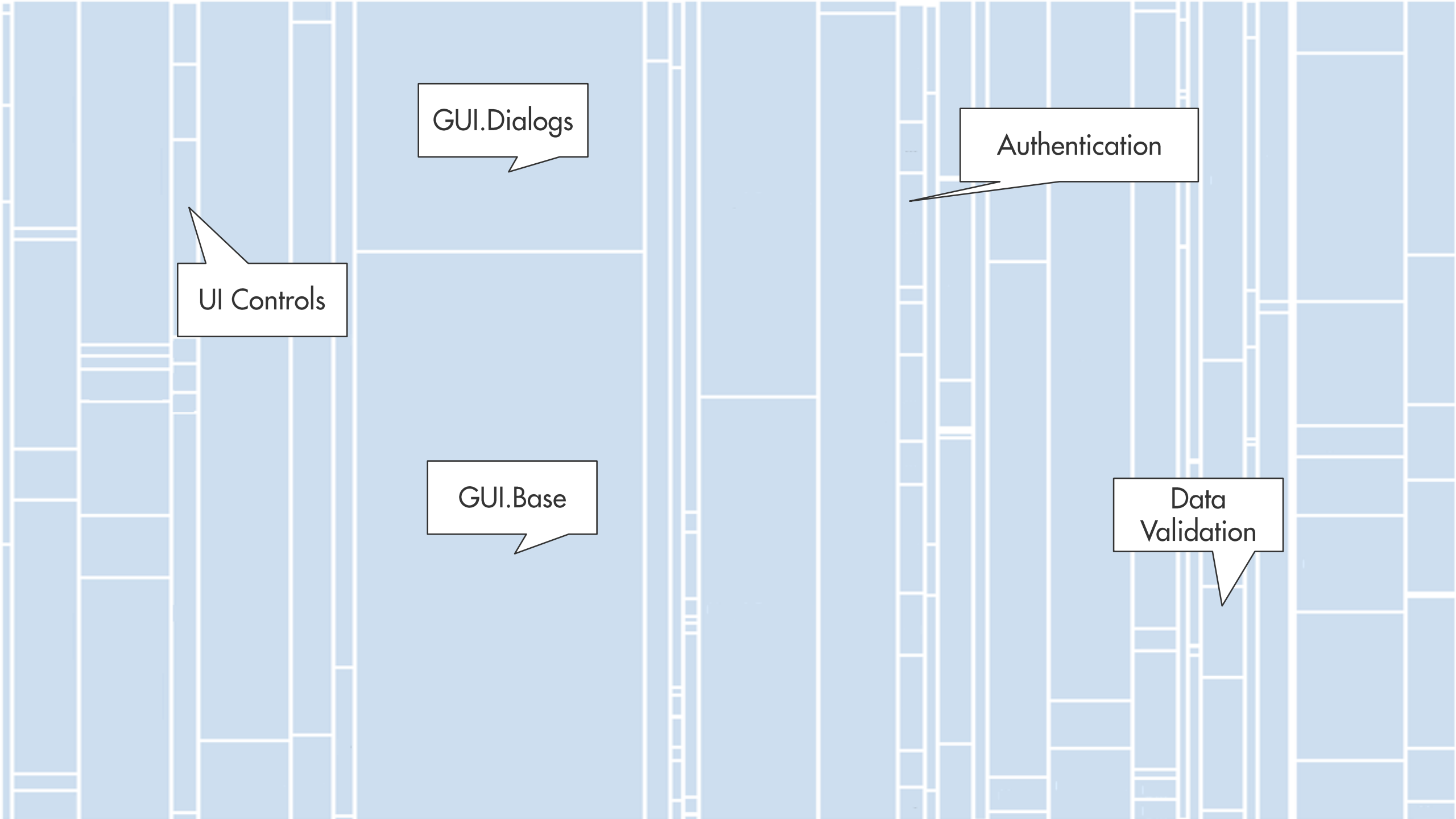
Release B:
15% Code neu/geändert,
>60% ungetestet

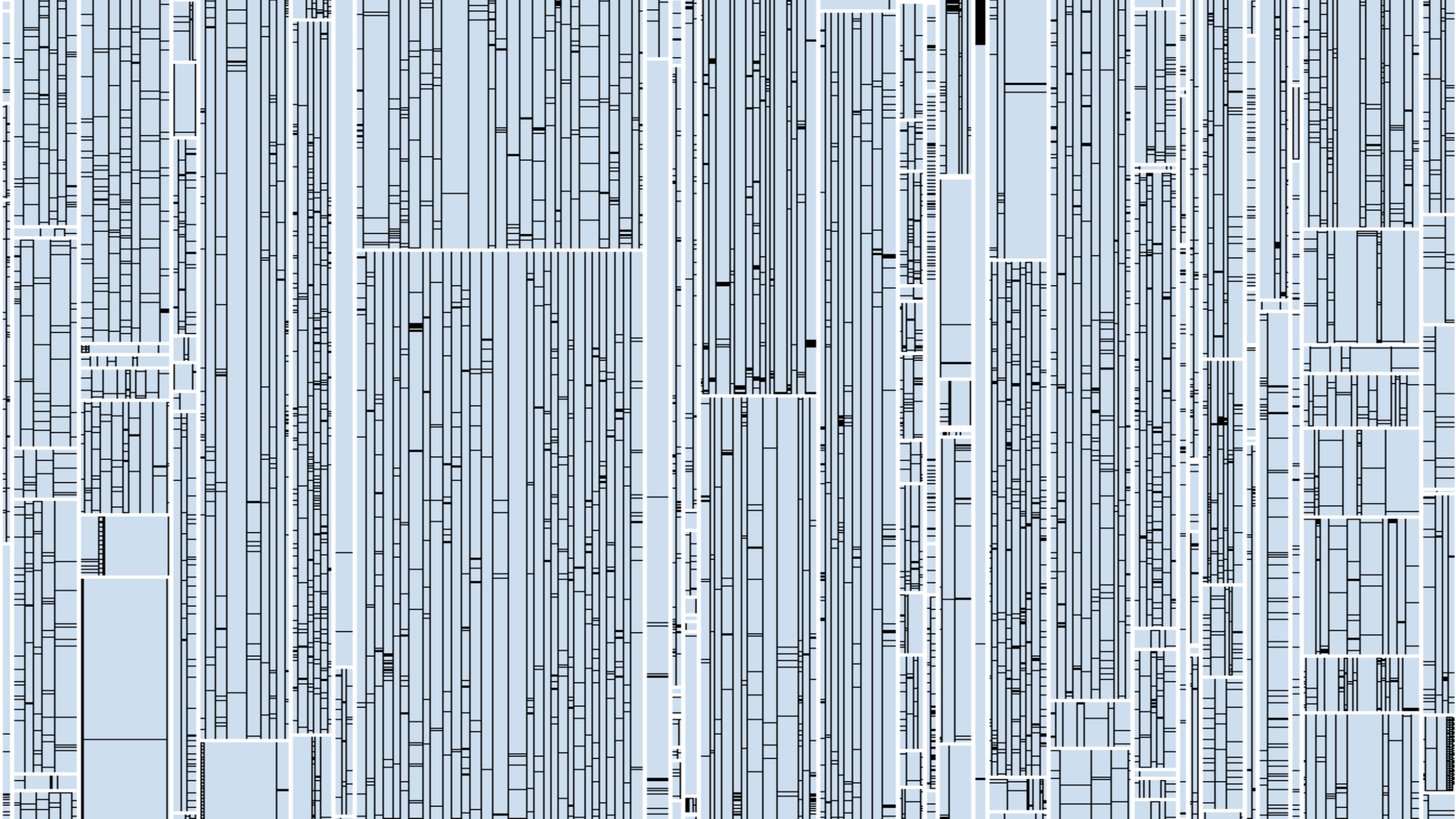


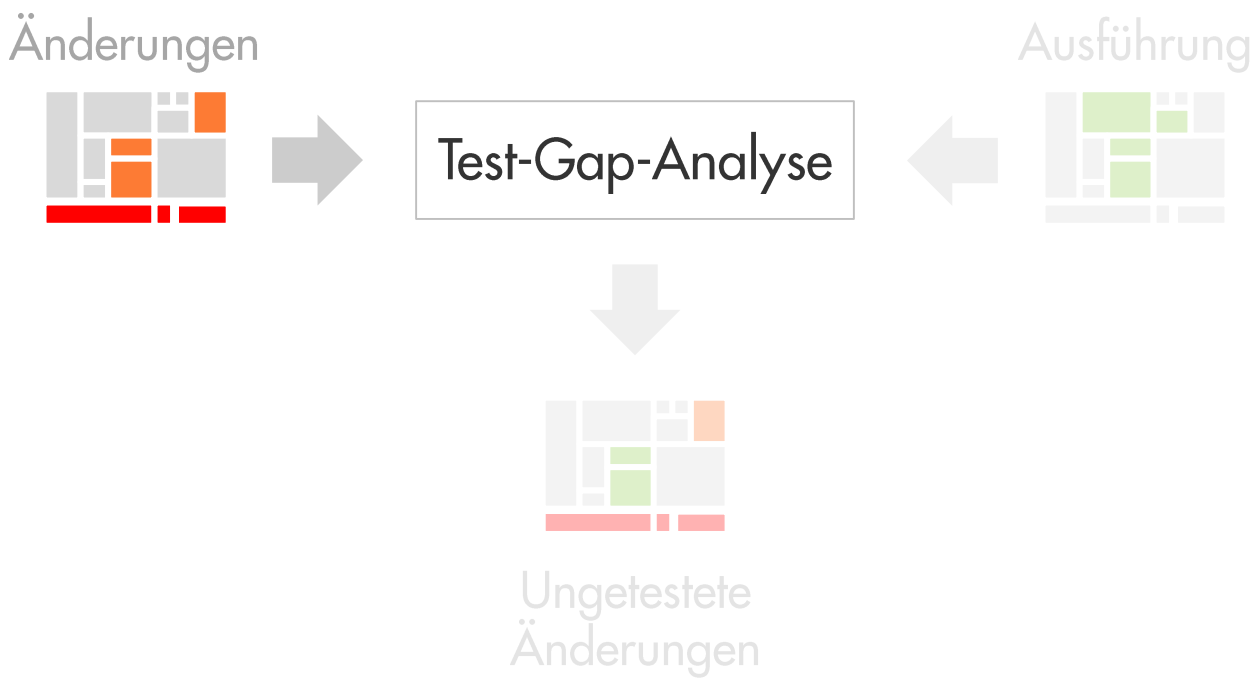
Feldfehlerwahrscheinlichkeit 5x höher für ungetestete Änderungen!

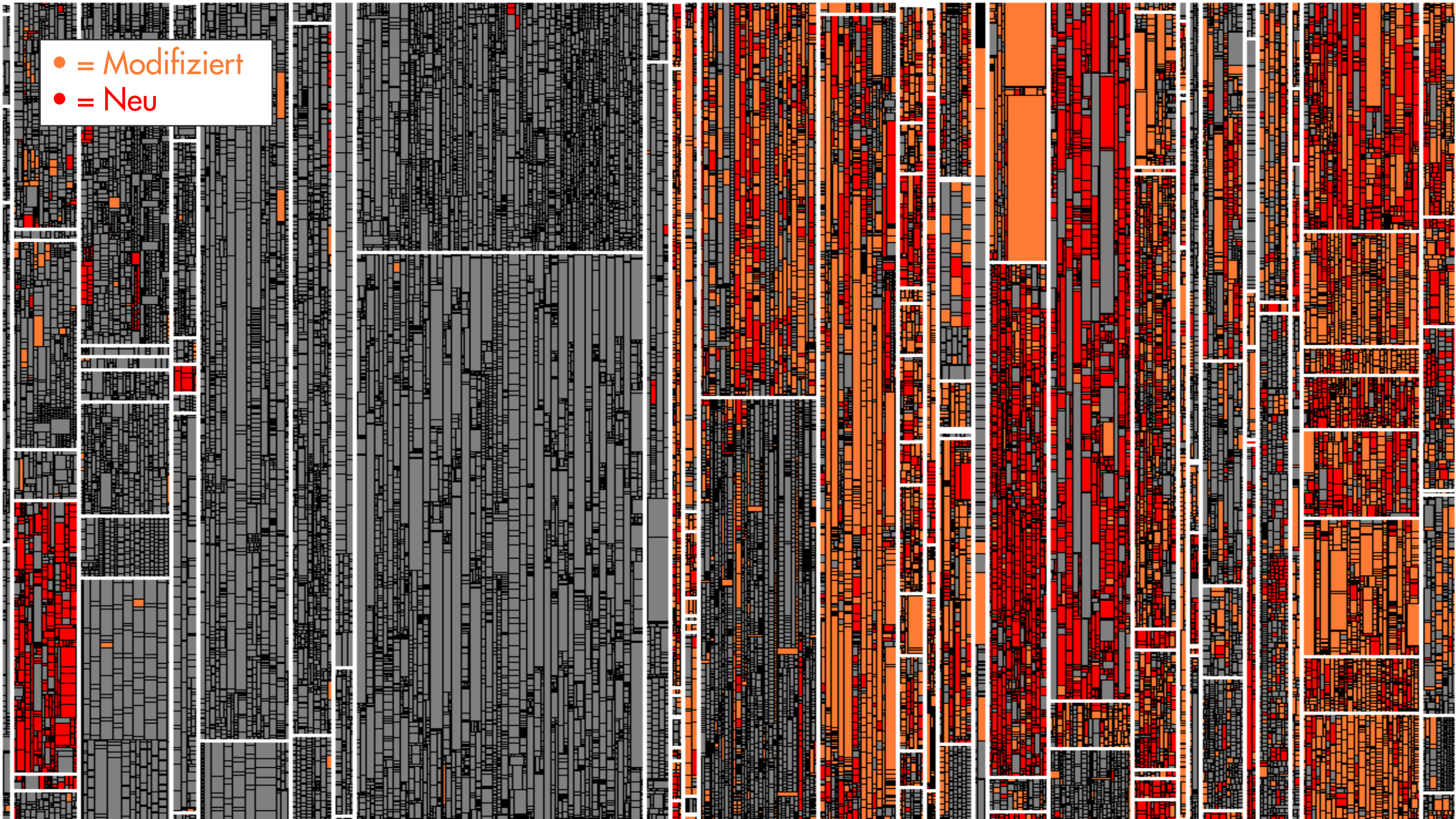
Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

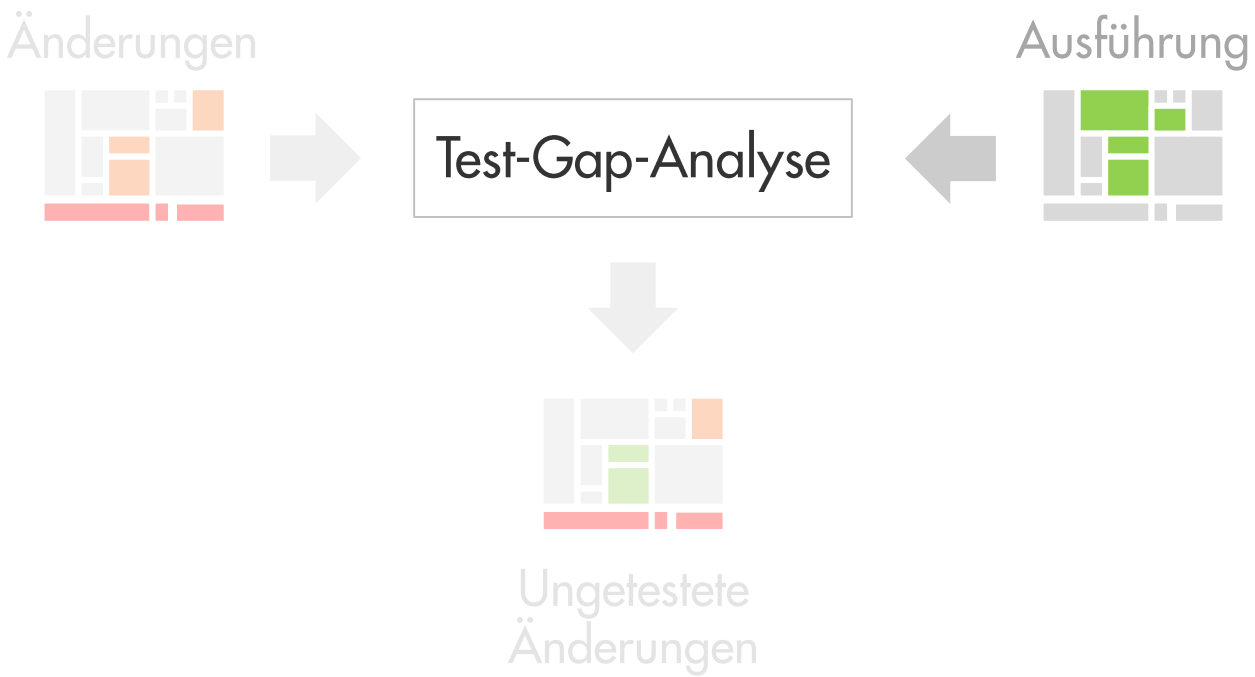

Änderungen
Ausführung
Test-Gap-Analyse
Ungetestete
Änderungen

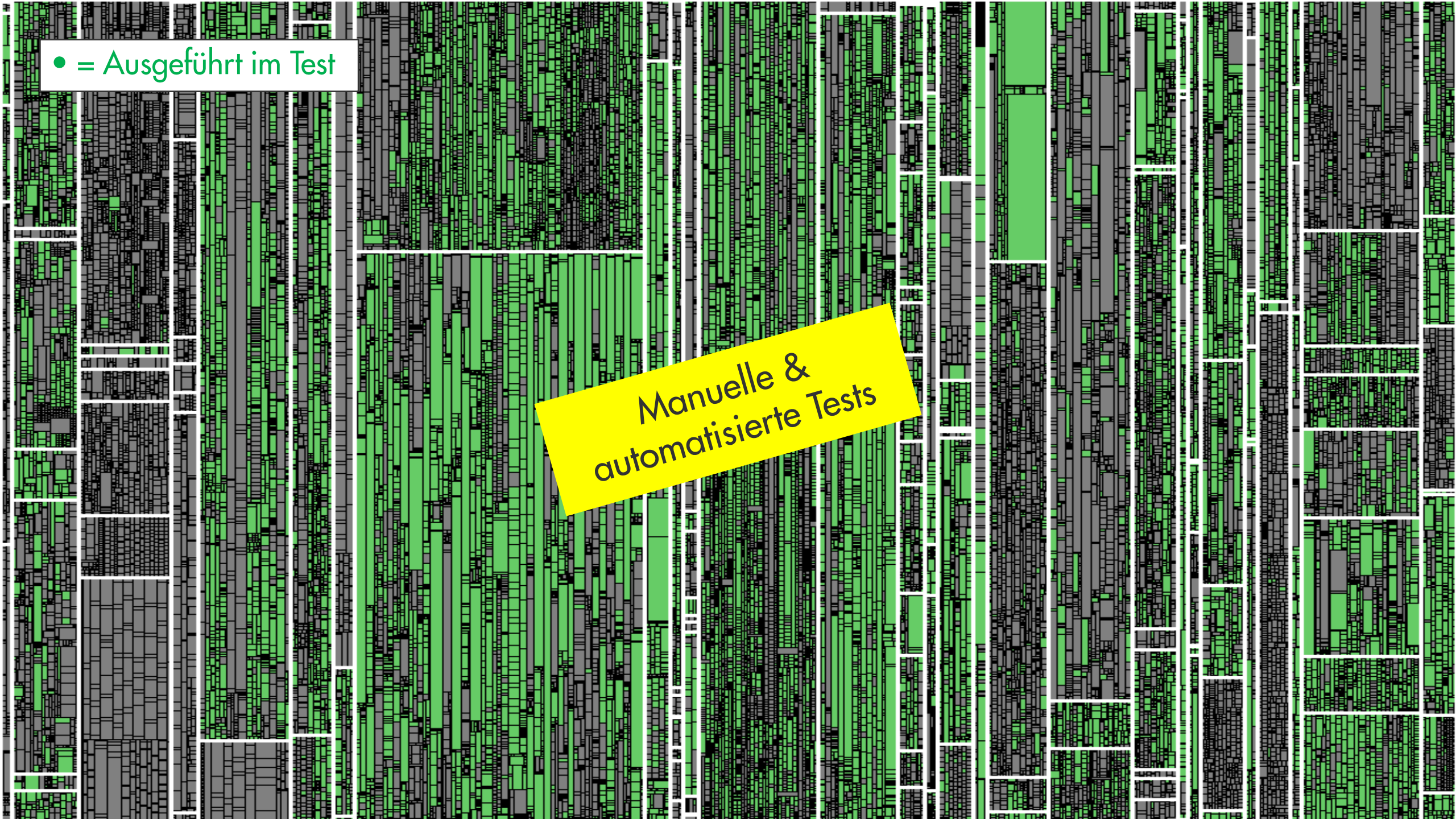

GUI.Dialogs
Authentication
UI Controls
GUI.Base
Data Validation

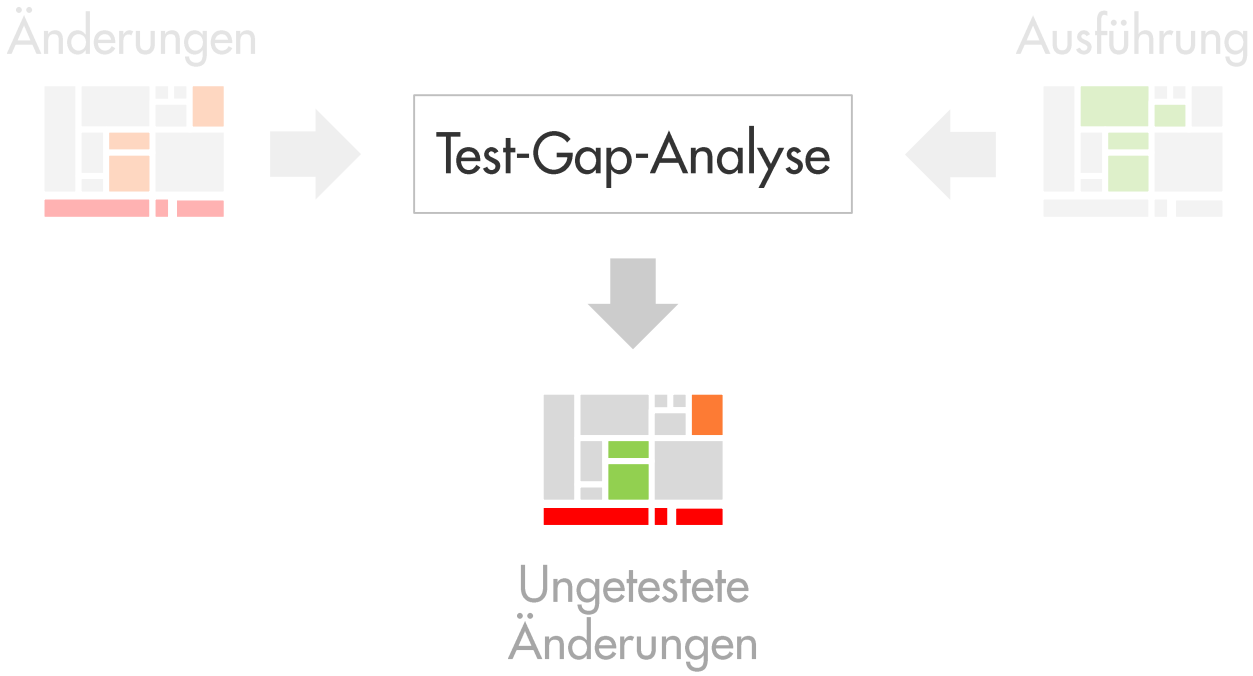


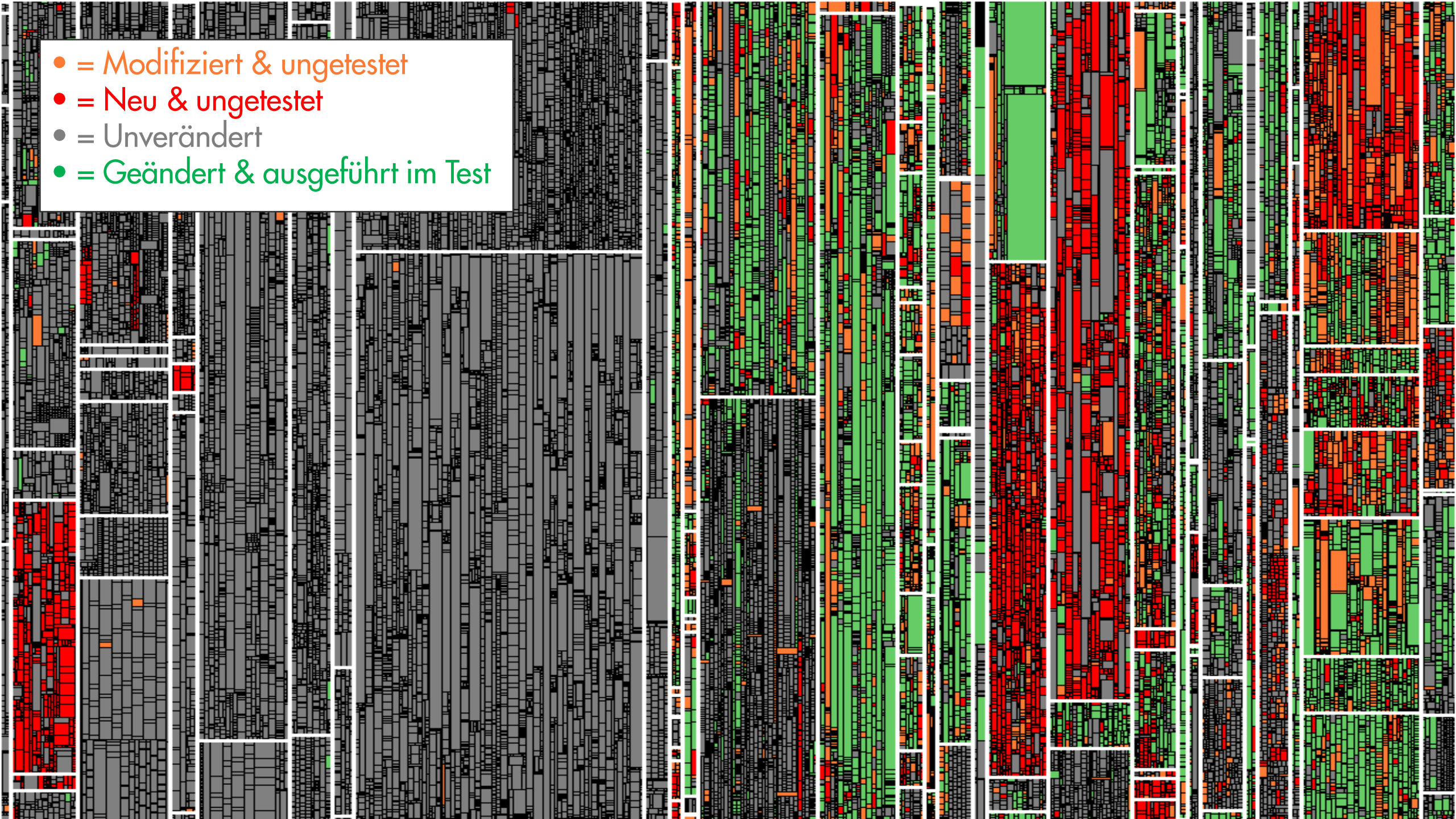

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete
Änderungen

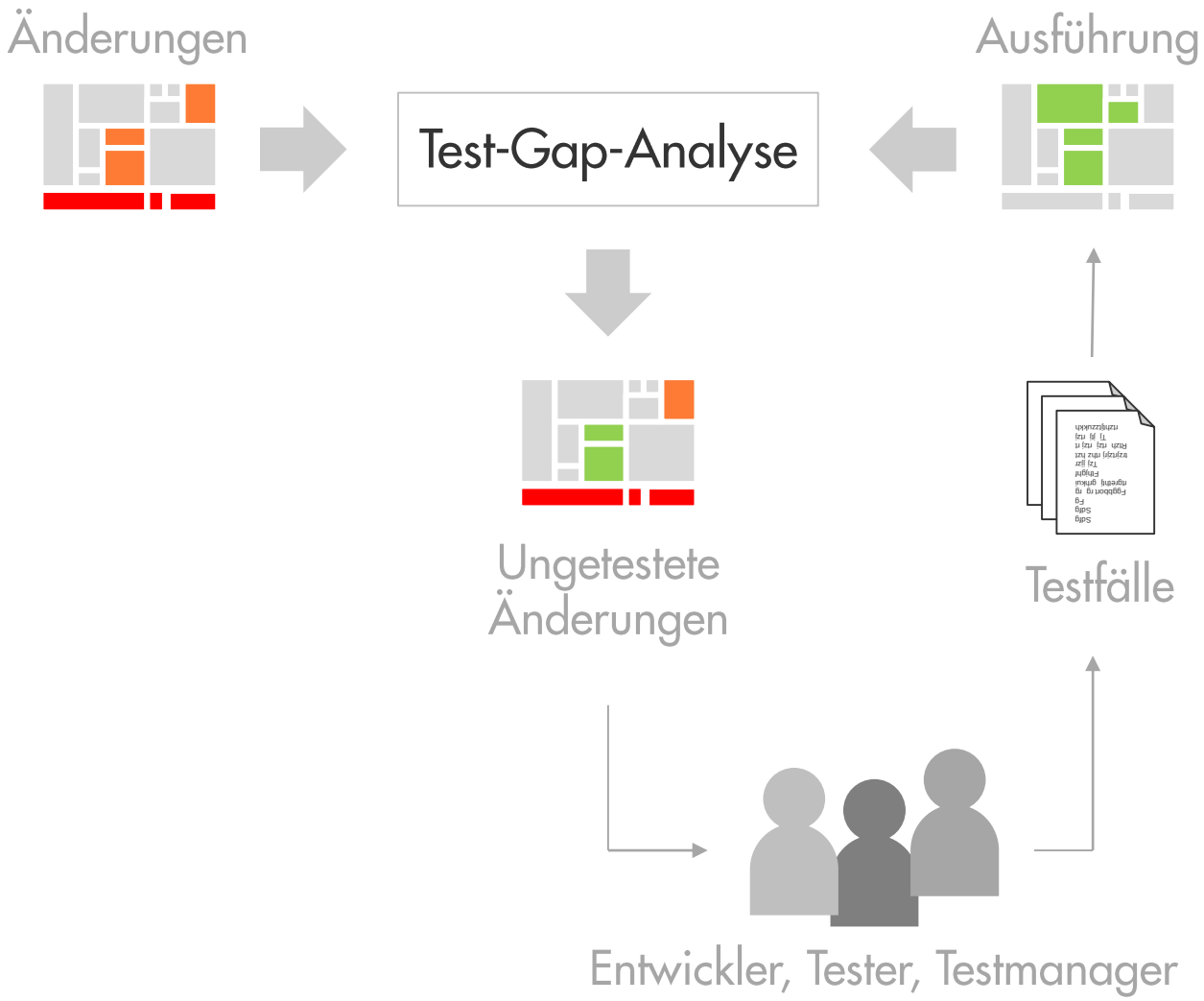

• = Modifiziert
• = Neu

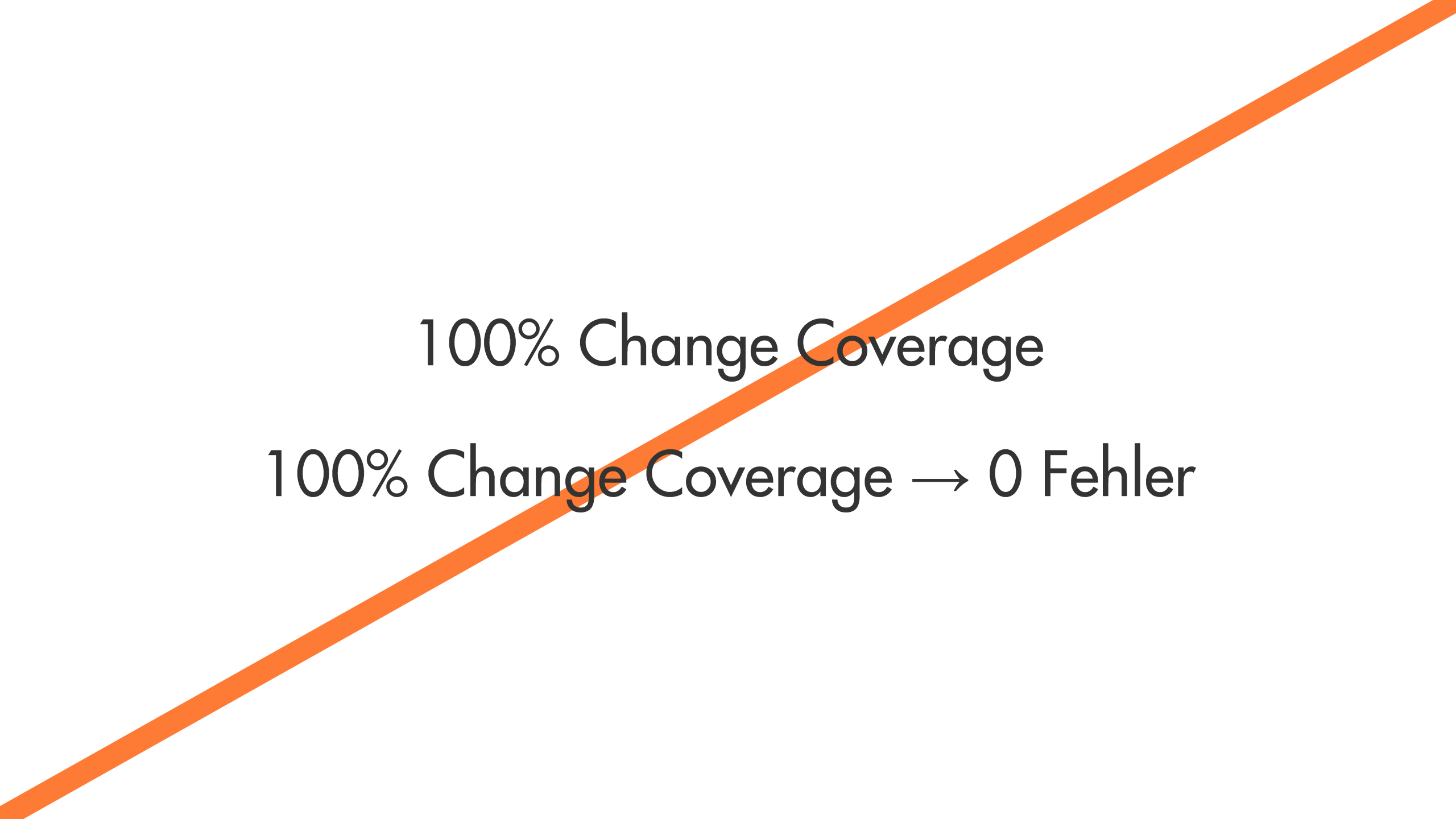

Änderungen
Ausführung
Test-Gap-Analyse
Ungetestete Änderungen

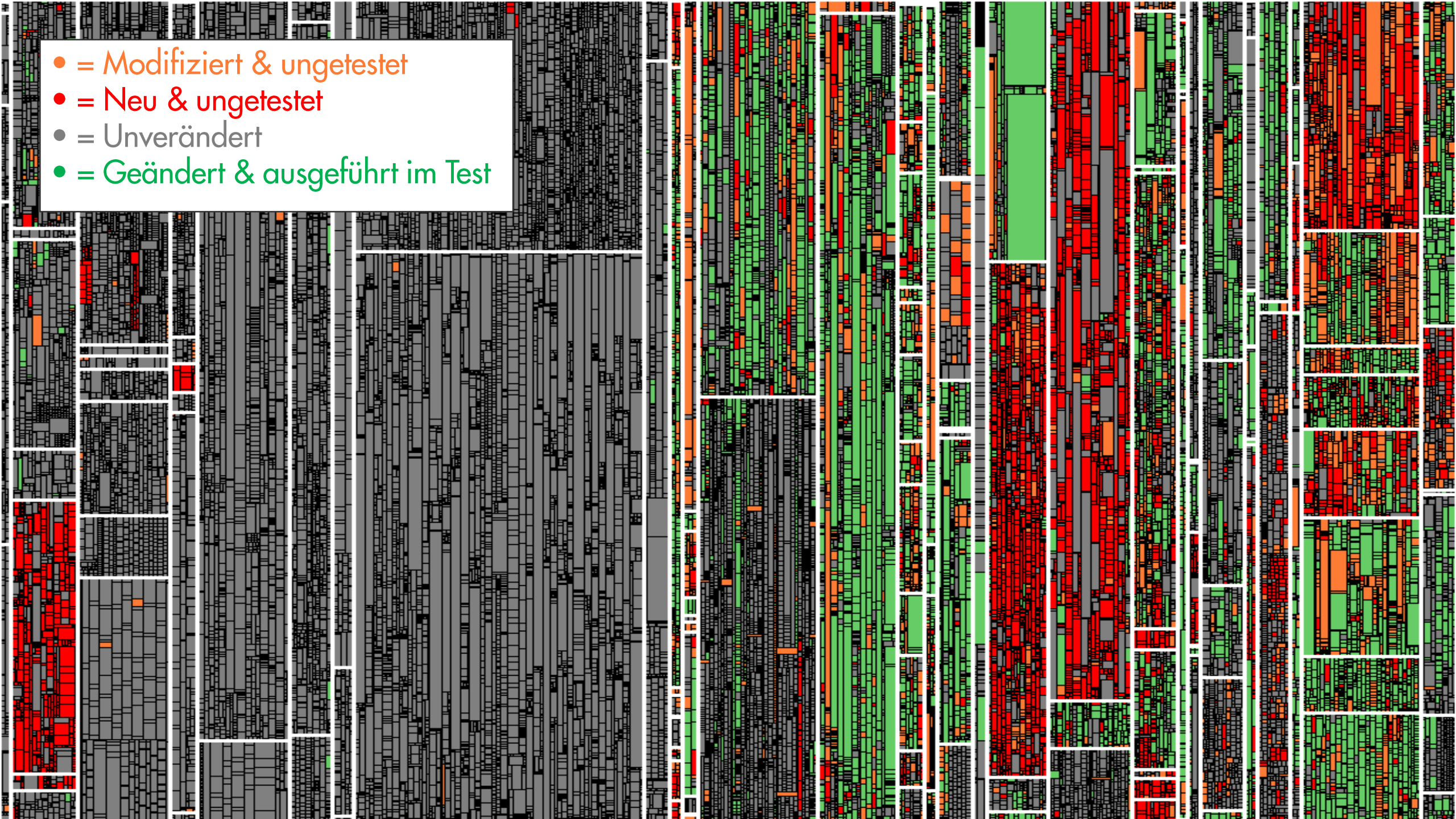

• = Ausgeführt im Test
Manuelle &
automatisierte Tests

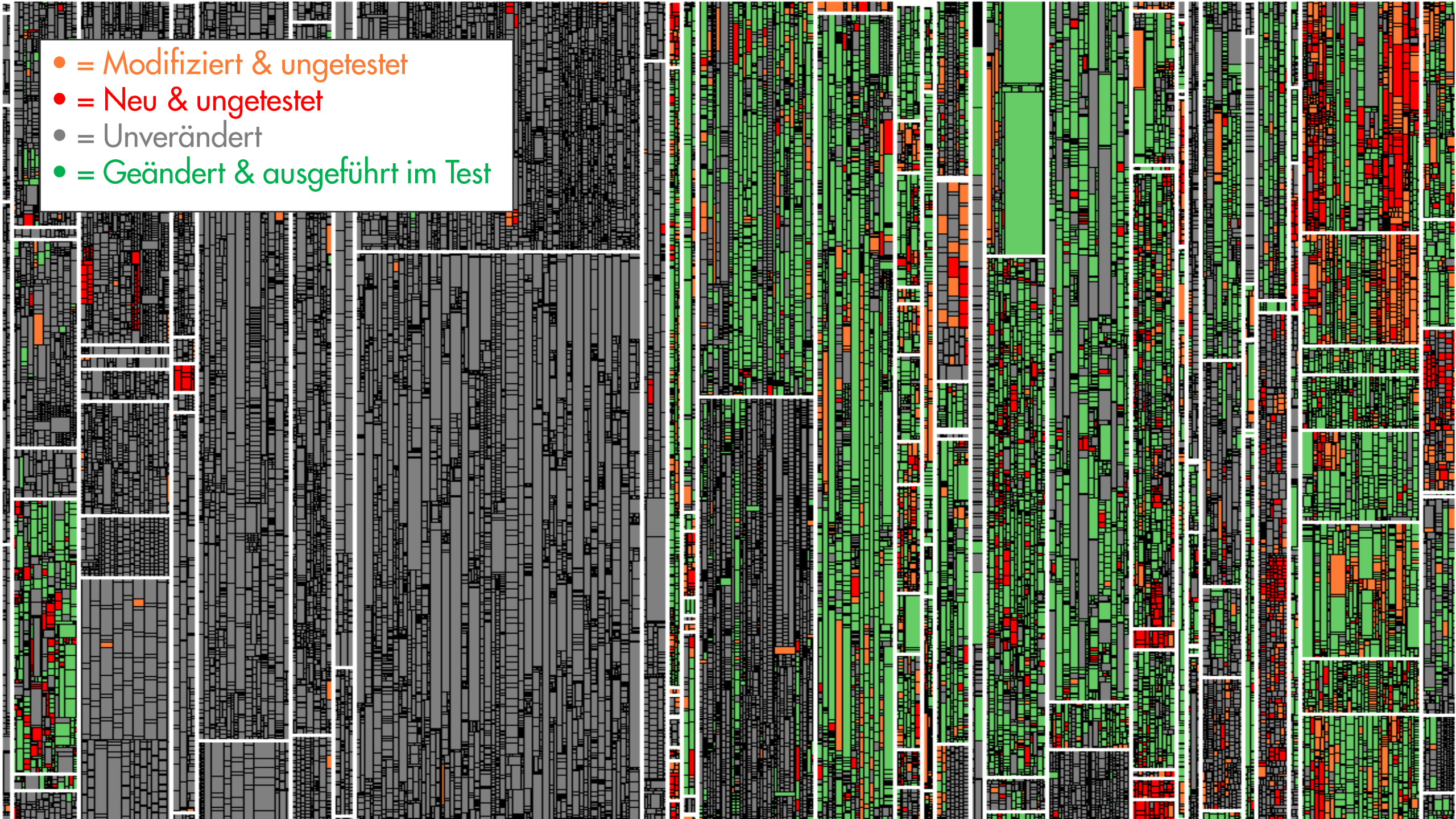

Änderungen
Ausführung
Test-Gap-Analyse
Ungetestete
Änderungen


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

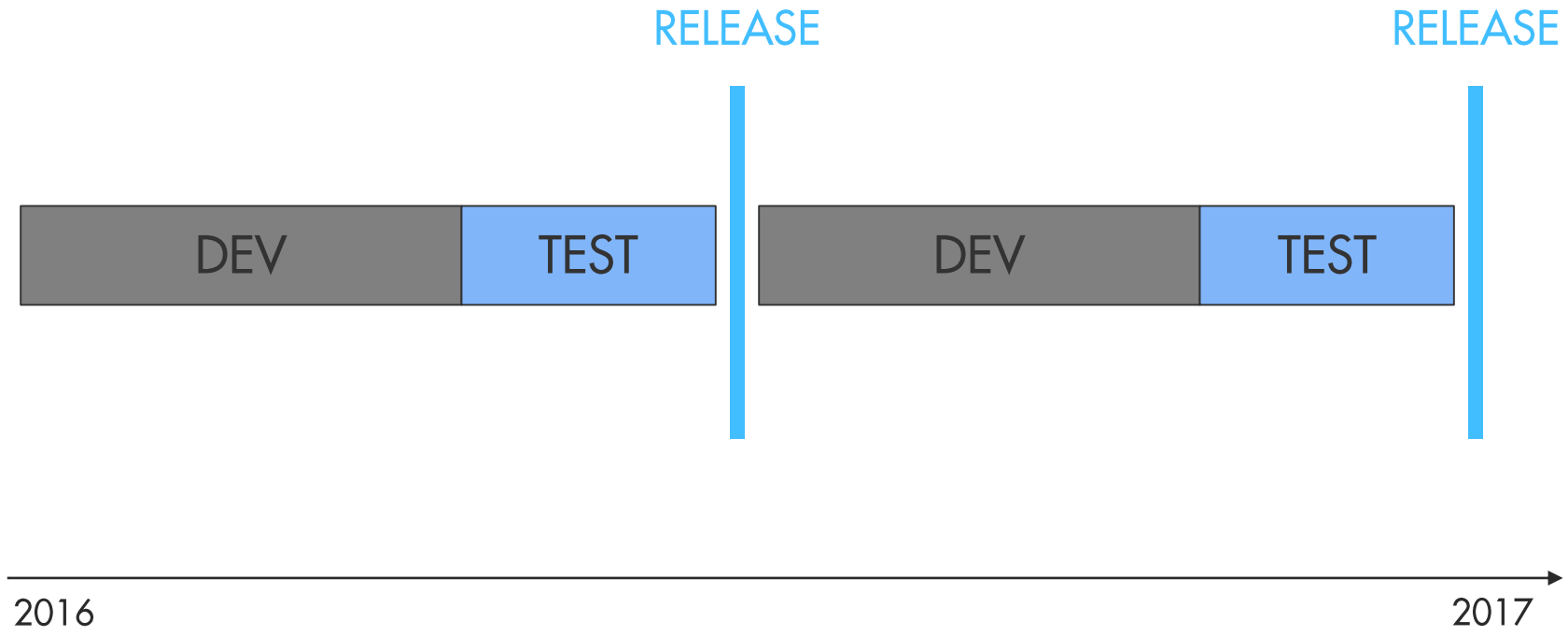

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete Änderungen
Testfälle
Entwickler, Tester, Testmanager


100% Change Coverage
100% Change Coverage → 0 Fehler

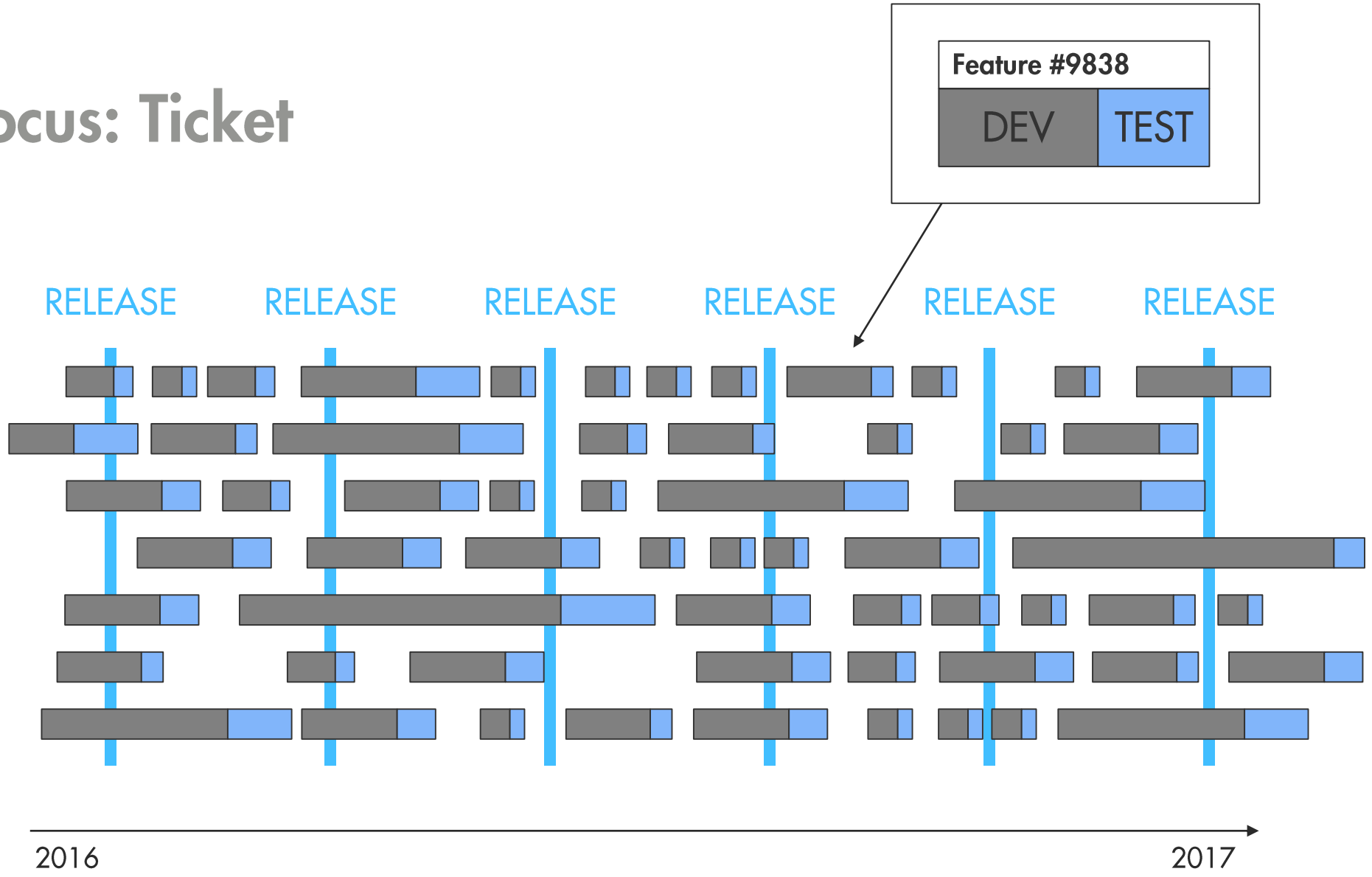

• = Modifiziert & ungetestet
• = Neu & ungetestet
• = Unverändert
• = Geändert & ausgeführt im Test

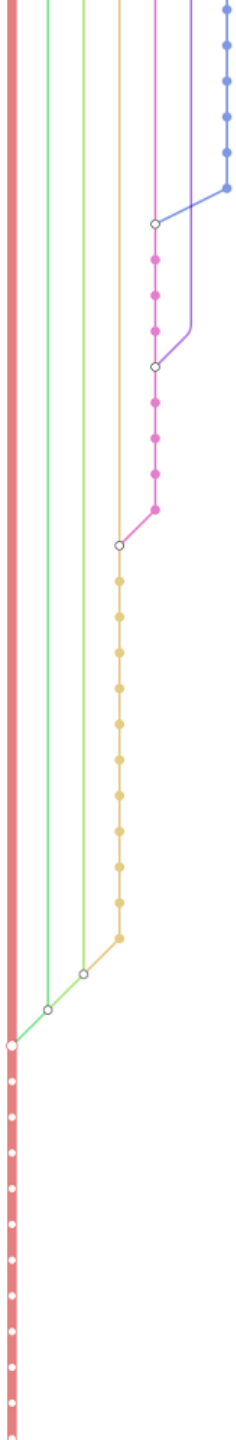

• = Modifiziert & ungetestet
• = Neu & ungetestet
• = Unverändert
• = Geändert & ausgeführt im Test

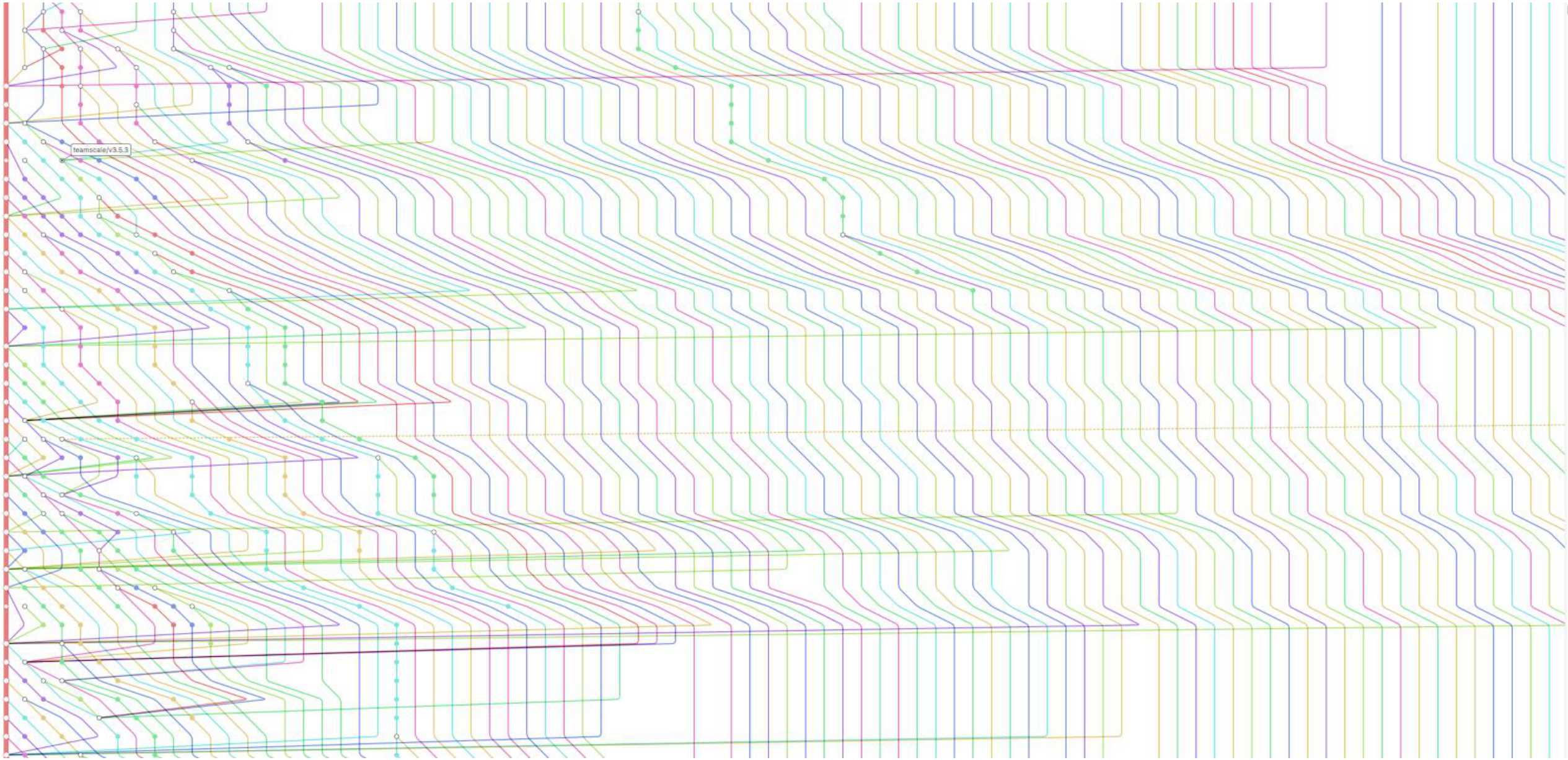
Test Focus: Release

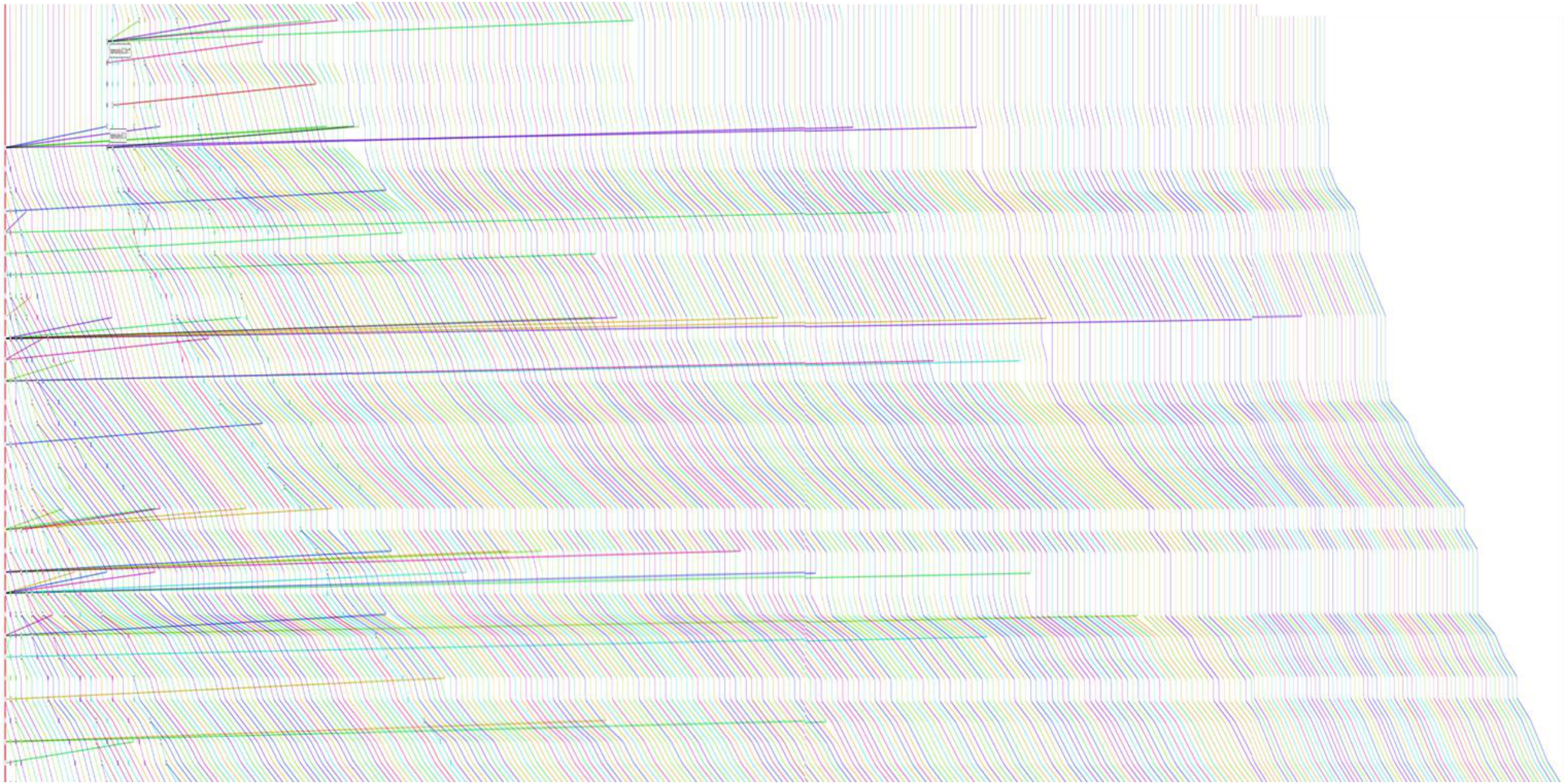


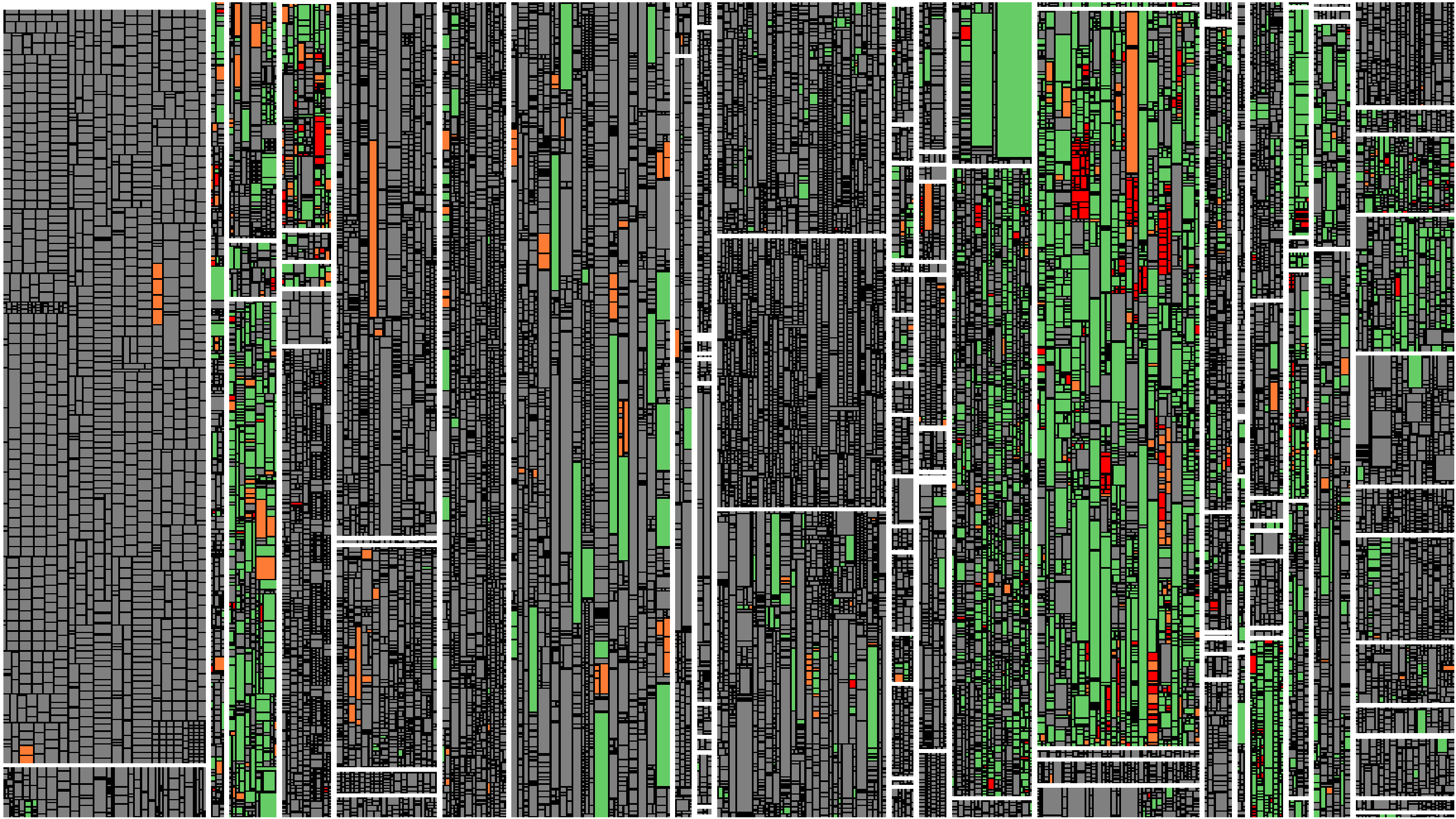
Test Focus: Ticket

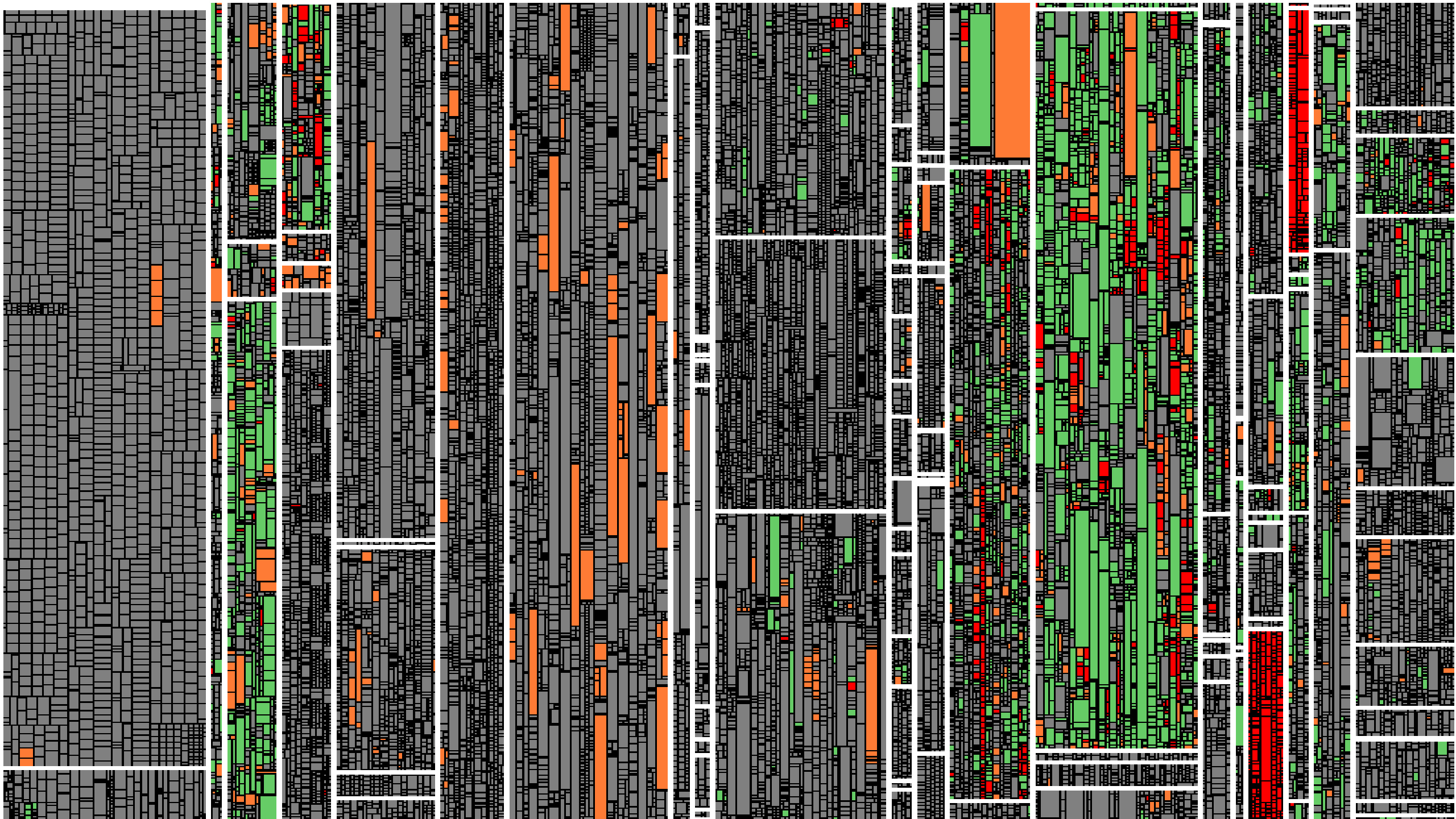





teamscale/v3.5.3





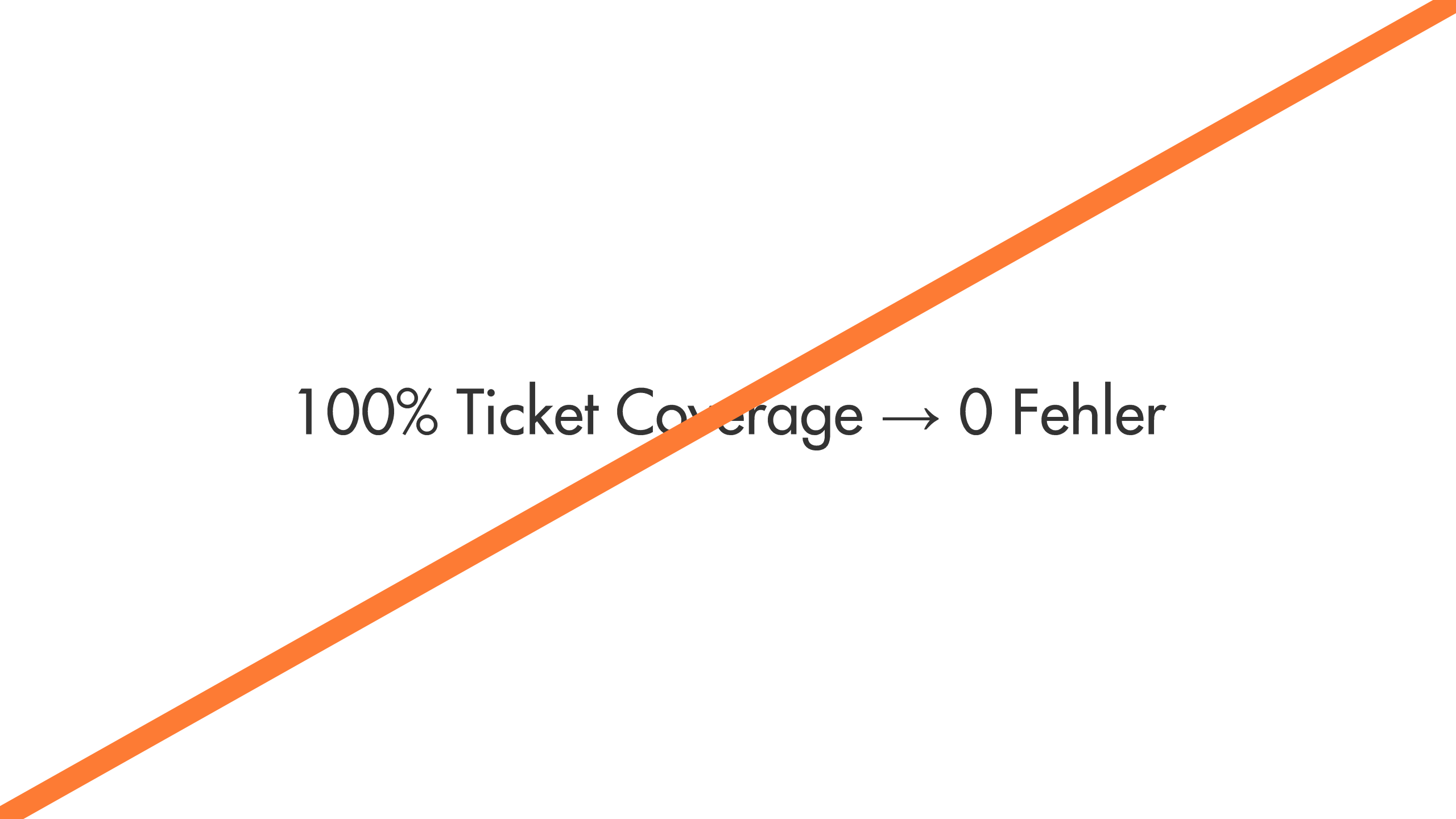



RELEASE
RELEASE
RELEASE
RELEASE
RELEASE
RELEASE
2016
2017

Ticket Coverage

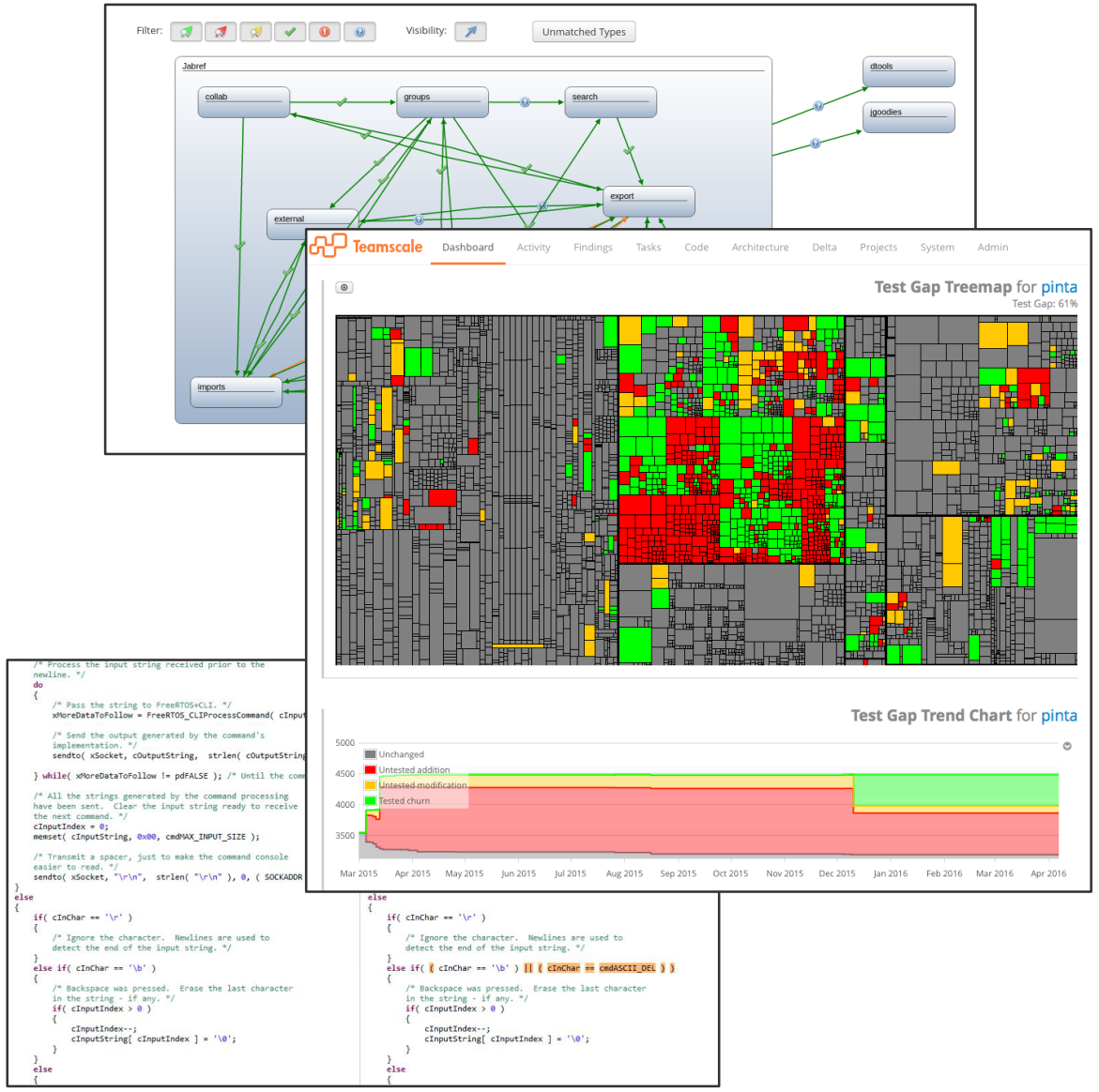
Ticket	Im Issue Tracker verwaltete Unit of Work. Bug, Issue, Change Request, User Story, …
Ticket Code	Code, der bei der Implementierung eines Tickets angefasst (geändert oder neu geschrieben) wurde.
Ticket Coverage	Anteil des Ticket Codes, der im Test zur Ausführung kam.

Number	Subject	Ticket Coverage
TS-4928	Icons for buttons in Dashboard Edit UI	0.0%
TS-5227	See full revision history of (deleted) files	20.0%
TS-5241	Support inconsistent clones	74.0%
TS-5253	Storing file unit size in chunks	100.0%
TS-5291	Provide means to filter/not report LSL findings for JS constructors (in our code)	0.0%
TS-5433	Case-insensitive file name support for IDE Plugins	7.0%
TS-5469	Adding/removing metrics to chart widget changes colors of other metrics	33.3%
TS-5557	Ratio metric is shown in percentage format but boundary values are interpreted as float	14.2%
TS-5666	IDE VS > VS Client should show visual feedback for finding location	0.8%
TS-5667	IDE VS > VS Client should jump to the tracked finding position on selecting afinding from the finding window.	40.0%
TS-5730	Provide visual feedback / hide menus for files not mapped to teamscale in VS-IDE	100.0%
TS-5834	Beautify start page of architecture perspective	100.0%
TS-6088	Allow to inspect clones by clone class	90.0%
TS-6099	Show 'potentially outdated' nightly finding more defensively	50.0%
TS-6183	Refactor project mappings?	75.0%
TS-6211	Several improvements to Teamscale IDE	66.7%
TS-6213	Allow to filter displayed findings to files with local modifications	55.0%
TS-6292	Connector validation should also check other fields, like include patterns	0.0%
TS-6299	Support deep comment analysis (category metrics, findings) also for JavaScript	11.0%


100% Ticket Coverage → 0 Fehler

Studienobjekt: Teamscale

- 750 kLOC Code (Java, JavaScript)
- 11 Jahre Historie im VCS
- >70k Commits
- 30 aktive Entwickler
- Code Peer Reviews



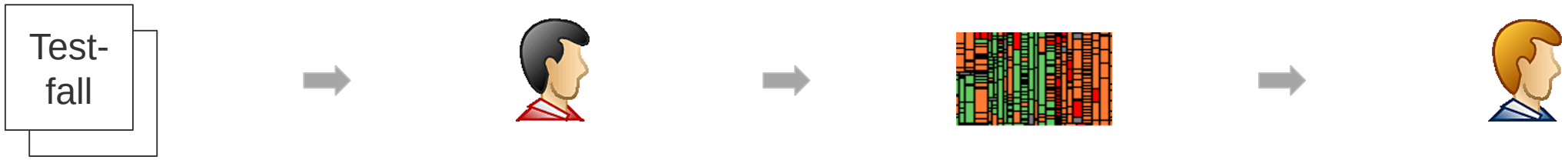
Forschungsfragen der empirischen Studie

1) Finden Tester mittels Ticket Coverage relevante Test-Gaps bei strukturierten Tests?

2) Finden Entwickler mittels Ticket Coverage relevante Test-Gaps bei explorativen Tests?

Study Design

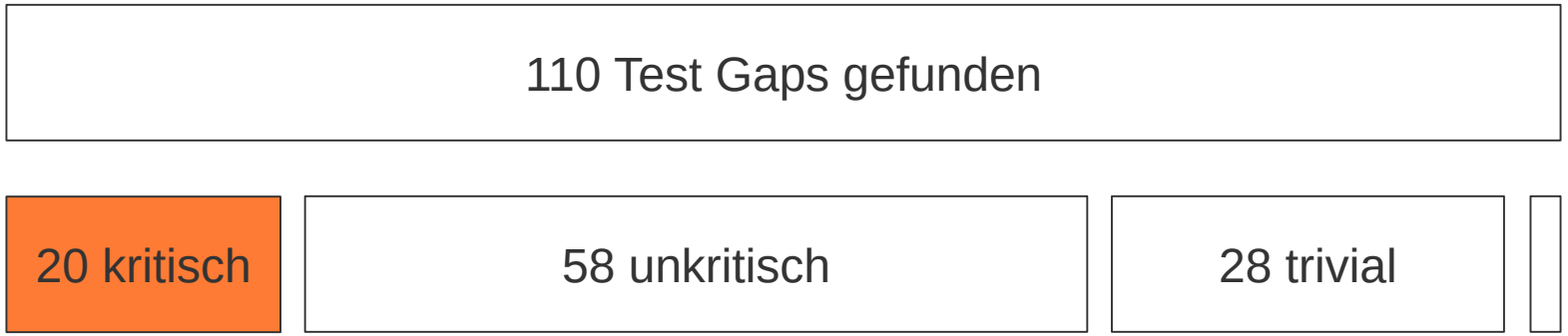
Finden Tester mittels Ticket Coverage relevante Test-Gaps bei strukturierten Tests?



- 54 Tickets analysiert
- < 20 Monate alt
- <= 3 pro Entwickler

Ticket Coverage: Putting Test Coverage into Context, Rott, Niedermayr, Juergens, Pagano, 2017 IEEE/ACM 8th International Workshop on Emerging Trends in Software Metrics (WETSoM)

Ergebnis



Information in Tickets (und daraus abgeleiteten Testfällen) oft unvollständig.
Irrelevante Test-Gaps haben zu Verbesserung der Analyse beigetragen.

Empirische Studie (2)

Finden Entwickler mittels Ticket Coverage relevante Test-Gaps bei explorativen Tests?

Affected methods (22)

Ticket Coverage: 86%

Class	Method	Line Coverage	Uncovered Lines	Change type
ProjectCreationService	retrieveProjectConfig	100%	0	changed
ProjectCreationService	fieldChangeRequiresReAnalysis	100%	0	added
ProjectService	elementUpdateQuery	100%	0	added
ProjectReanalysisService	process	100%	0	changed
ProjectCreator	refreshProject	100%	0	added
ProjectCreationService	nullOrToString	66%	1	added
ProjectCreationService	findConnectorByName	75%	1	added
ProjectCreationService	connectorRequiresReAnalysis	82%	3	added
ProjectCreationService	processPutRequest	84%	3	changed
ProjectCreationService	validateProjectConfiguration	80%	3	changed
ProjectCreationService	projectReAnalysisRequired	61%	5	added
ProjectCreationService	connectorsRequireReAnalysis	50%	7	added
ConfigOptionDescriptorBase	ConfigOptionDescriptorBase	100%	0	changed
NamingConventionConfiguration	NamingRegexOption	100%	0	changed
ProjectService	ProjectUpdateResult	100%	0	added

Empirische Studie (2)

Finden Entwickler mittels Ticket Coverage relevante Test-Gaps bei explorativen Tests?

95 Methoden analysiert. 30 enthielten Test-Gaps. 23 davon relevant.

Für Entwickler sind Ticket & Code-Änderungen unzureichend für vollständigen Test.

Test Accompanying Calculation of Test Gaps for Java, Florian Dreier, Guided Research @ TU München, 2017

Fazit

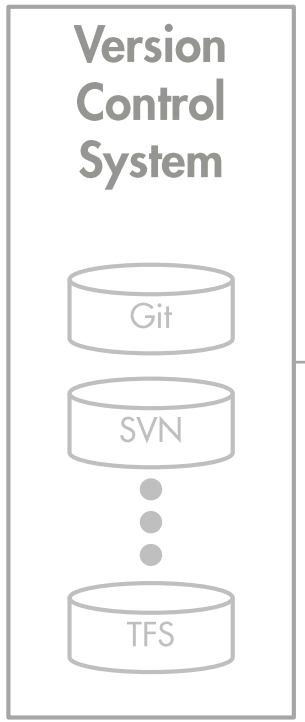
1) Finden Tester mittels Ticket Coverage relevante Test-Gaps bei strukturierten Tests?

JA: Information in Tickets (und daraus abgeleiteten Testfällen) oft unvollständig.

2) Finden Entwickler mittels Ticket Coverage relevante Test-Gaps bei explorativen Tests?

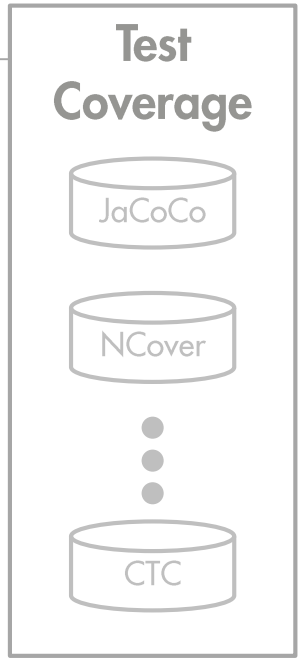
JA: Selbst für Entwickler sind Ticket & Code-Änderungen unzureichend für vollständigen Test.

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#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



Ticket Coverage


```
public static synchronized void log(String methodIdentifier) {
    if (s_mode == LoggingMode.TESTING) {
        s_testingLog.add(methodIdentifier);
    } else if (s_mode == LoggingMode.FRAMING) {
        s_framingLog.add(methodIdentifier);
    } else {
        throw new IllegalStateException();
    }
}

public static synchronized Set<String> getFramingLog() {
    return s_framingLog;
}

public static synchronized Set<String> getTestingLog() {
    return s_testingLog;
}

public static synchronized void reset() {
    s_testingLog.clear();
    s_framingLog.clear();

    setMode(LoggingMode.FRAMING);
}

public static synchronized void setMode(LoggingMode newMode) {
    s_mode = newMode;
}
```

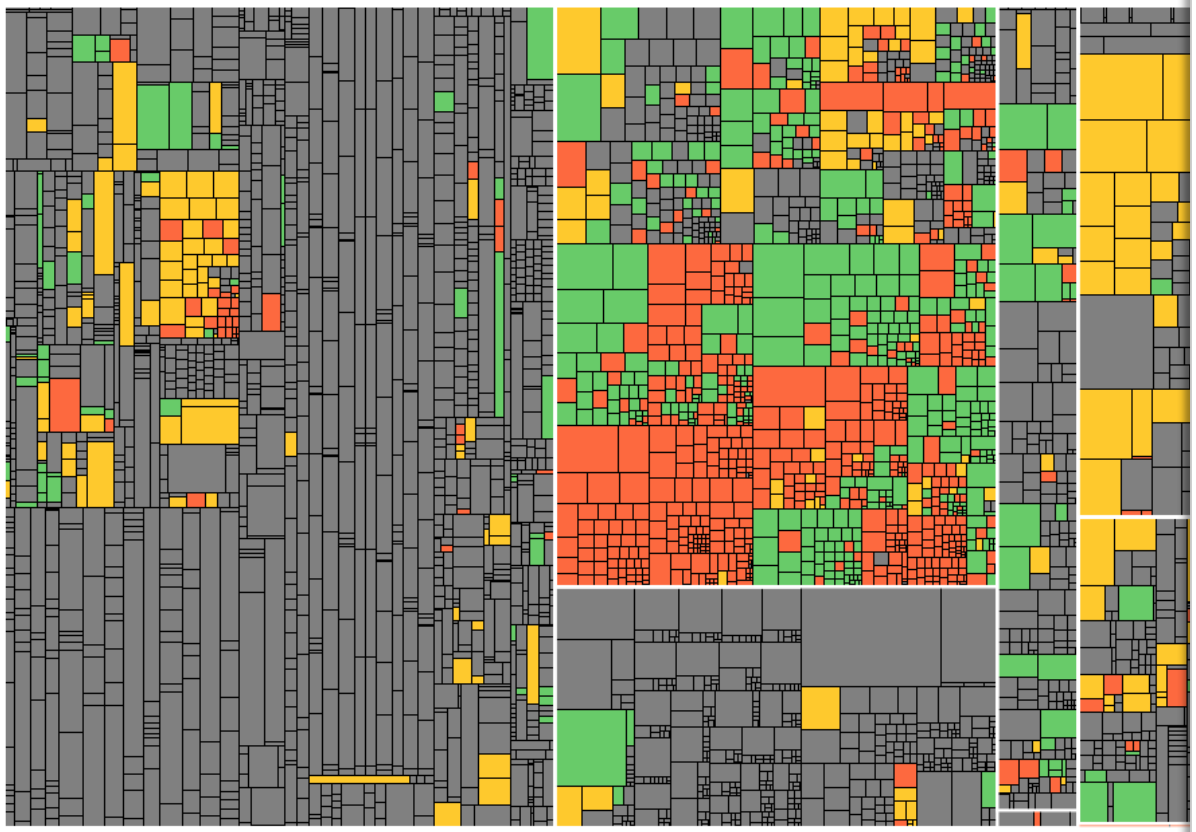


ABAP
Ada
C#
C/C++
Cobol
Delphi
Fortran
Groovy
Gosu
HANA SQLScript
HANA Views
IEC 61131-3 ST
Java
JavaScript
Magik
Matlab
Open CL
OScript
PHP
PL/SQL
Python
Rust
SQLScript
Simulink/StateFlow
Swift
TypeScript
Visual Basic .NET
Xtend

Teamscale | Dashboard | Activity | Findings | Metrics | Issues | Tasks | Audit | Architecture | Delta | More

Branch: master **Timetravel:** Current Dashboard

Feb 01 2015 00



Teamscale | Dashboard | Activity | Findings | Metrics | Issues | Tasks | More | Pinta | Search, e.g. files...

Issue details:

1438022: Improve how selection changes are detected. (**»Closed«**)

Created by
Cameron White
on Mar 17 2015 07:35

Assigned to
Cameron White

Open in bugtracker

Last updated on Jul 26 2015 08:35

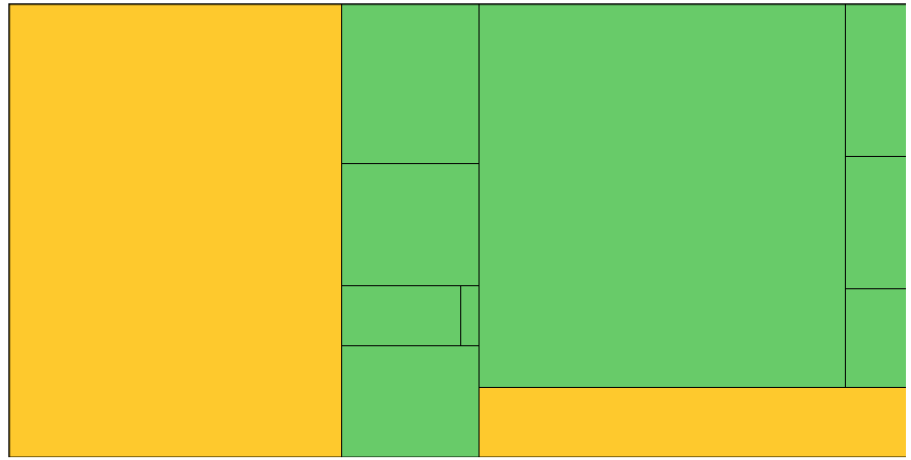
Selection changes are currently not handled properly.

Affected files (6)

Path	Change type (aggregated)
Pinta.Core/Classes/Document.cs	changed
Pinta.Core/Classes/DocumentSelection.cs	changed
Pinta.Core/Classes/SelectionModeHandler.cs	changed
Pinta.Core/HistoryItems/SelectionHistoryItem.cs	changed
Pinta.Core/Managers/ActionManager.cs	changed
Pinta.Core/Managers/WorkspaceManager.cs	changed

Test Gap Treemap

Jul 26 2015 04:23 – Now | Test Gap: 18.18%

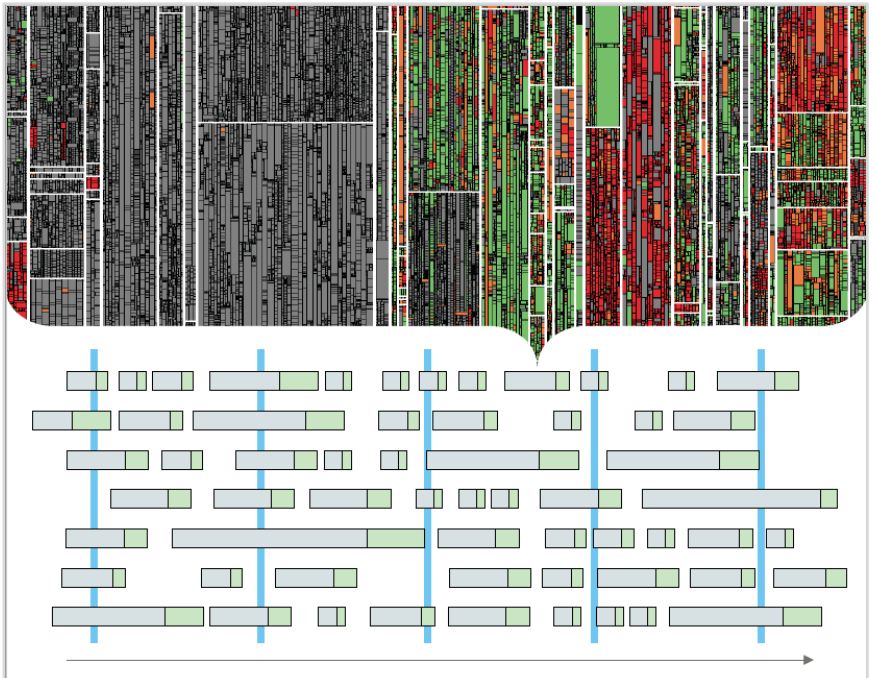


Associated repository changes (1):

Fix #1438022 - Improve how selection changes are detected.
by Cameron White in revision 3f4b4b105ee6c777d5ce999f664049aeb0f3f002 in branch master (GitHub)

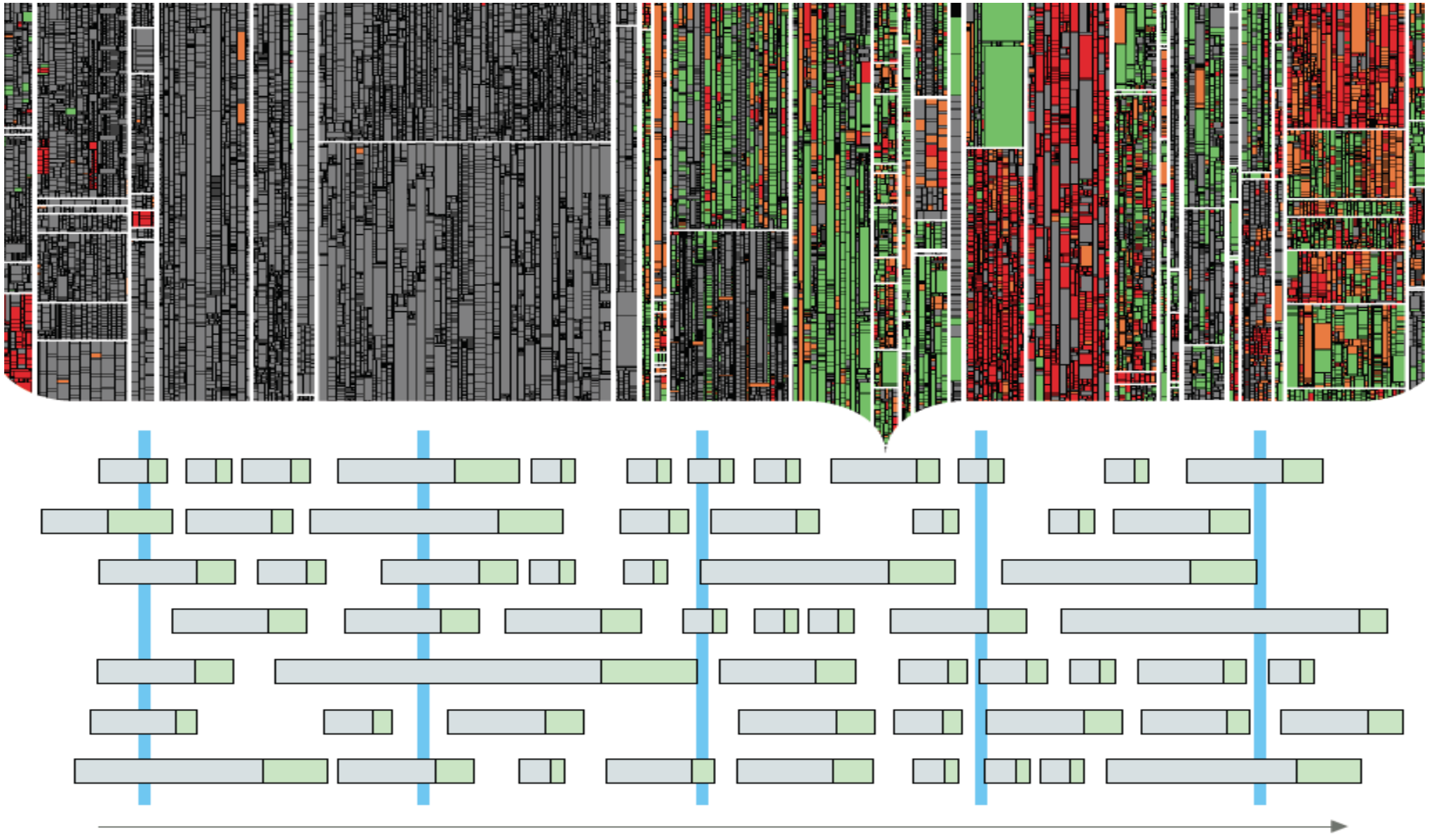
Jul 26 2015
04:23

Files: 6 changed
Findings: 6 10 1



Immer kürzere Testphasen? Mit Ticket Coverage verhindern, dass wichtige Features ungetestet bleiben

CQSE



Fazit

Kürzere Release-Zyklen führen zu Parallelisierung in Entwicklung und Test.
Dadurch steigt das Risiko ungetesteter Änderungen.

Ticket Coverage hilft, wichtige Änderungen zuverlässig zu testen.

Wir unterstützen gerne bei der Evaluierung und Einführung.

Kontakt



Dr. Dennis Pagano · pagano@cqse.eu · +49 159 04062957
Dr. Elmar Jürgens · juergens@cqse.eu · +49 179 675 3863
Fabian Streitel · streitel@cqse.eu · +49 159 04046270

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

CQSE