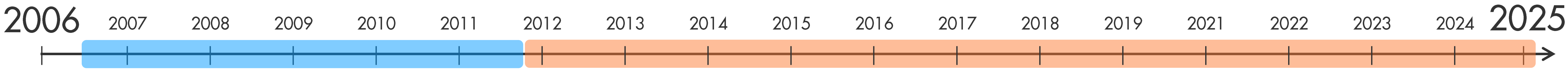
Welche Test-Gaps bergen das größte Risiko?

Roman Haas
Dr. Elmar Jürgens

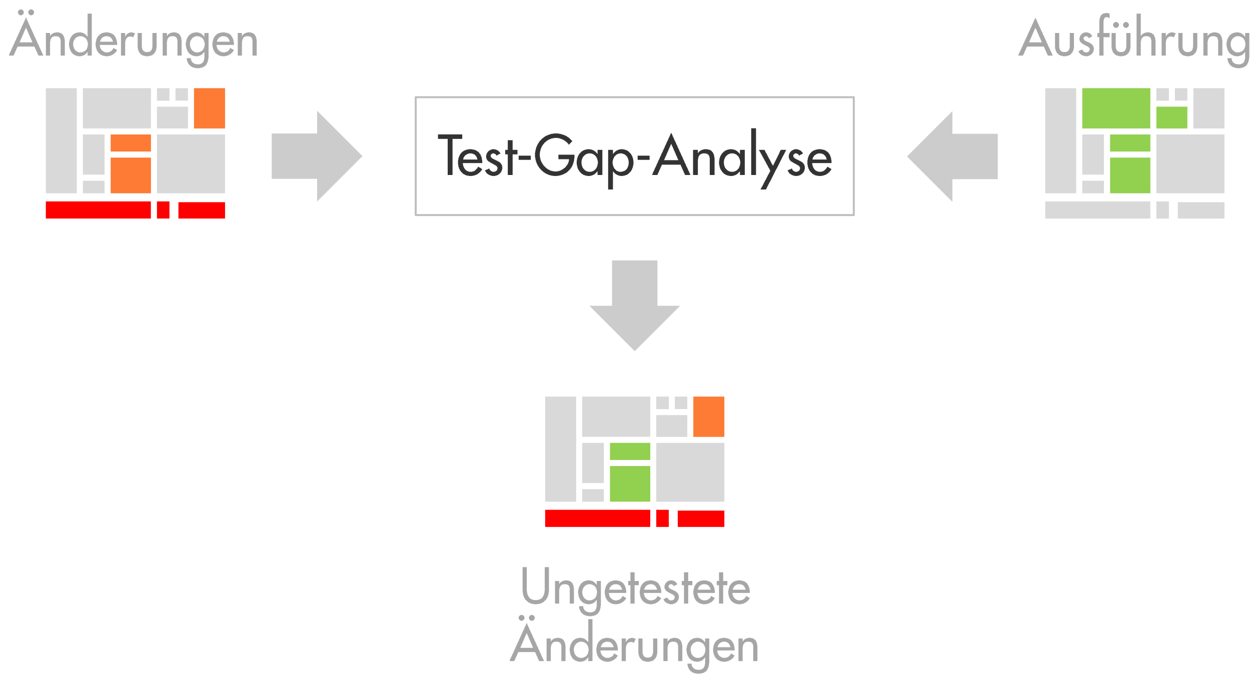
CQSE

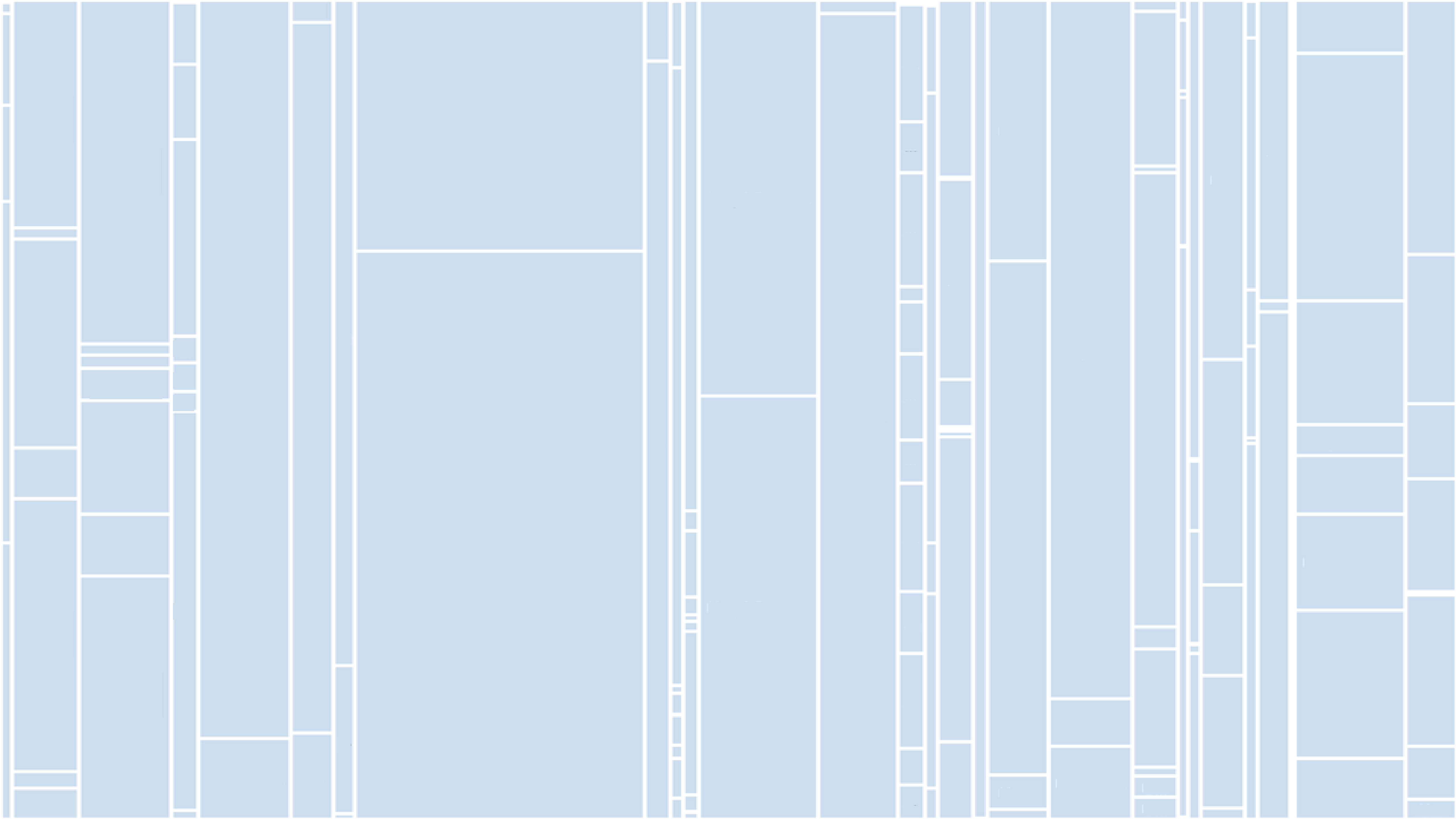
TUM

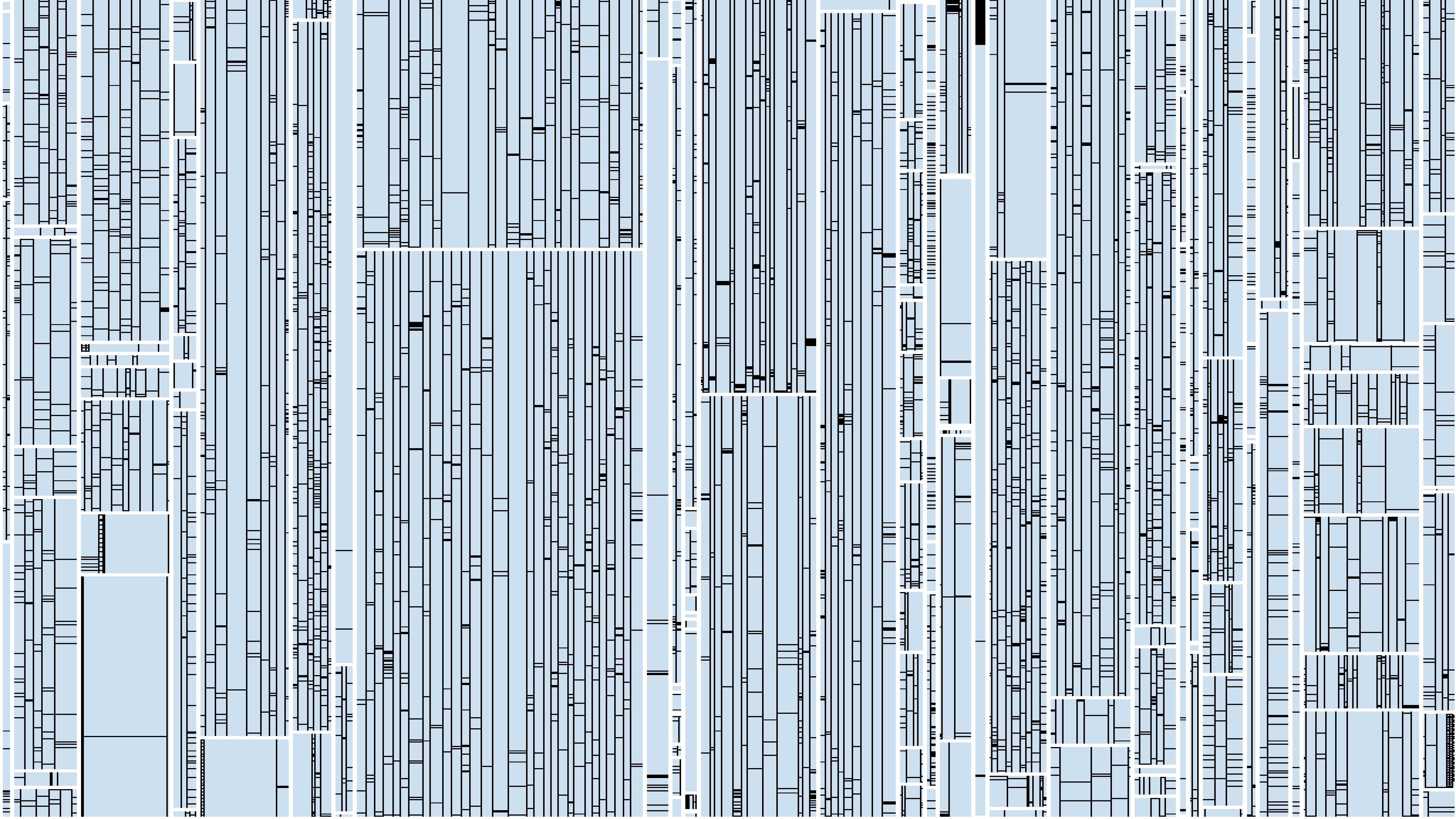
CQSE

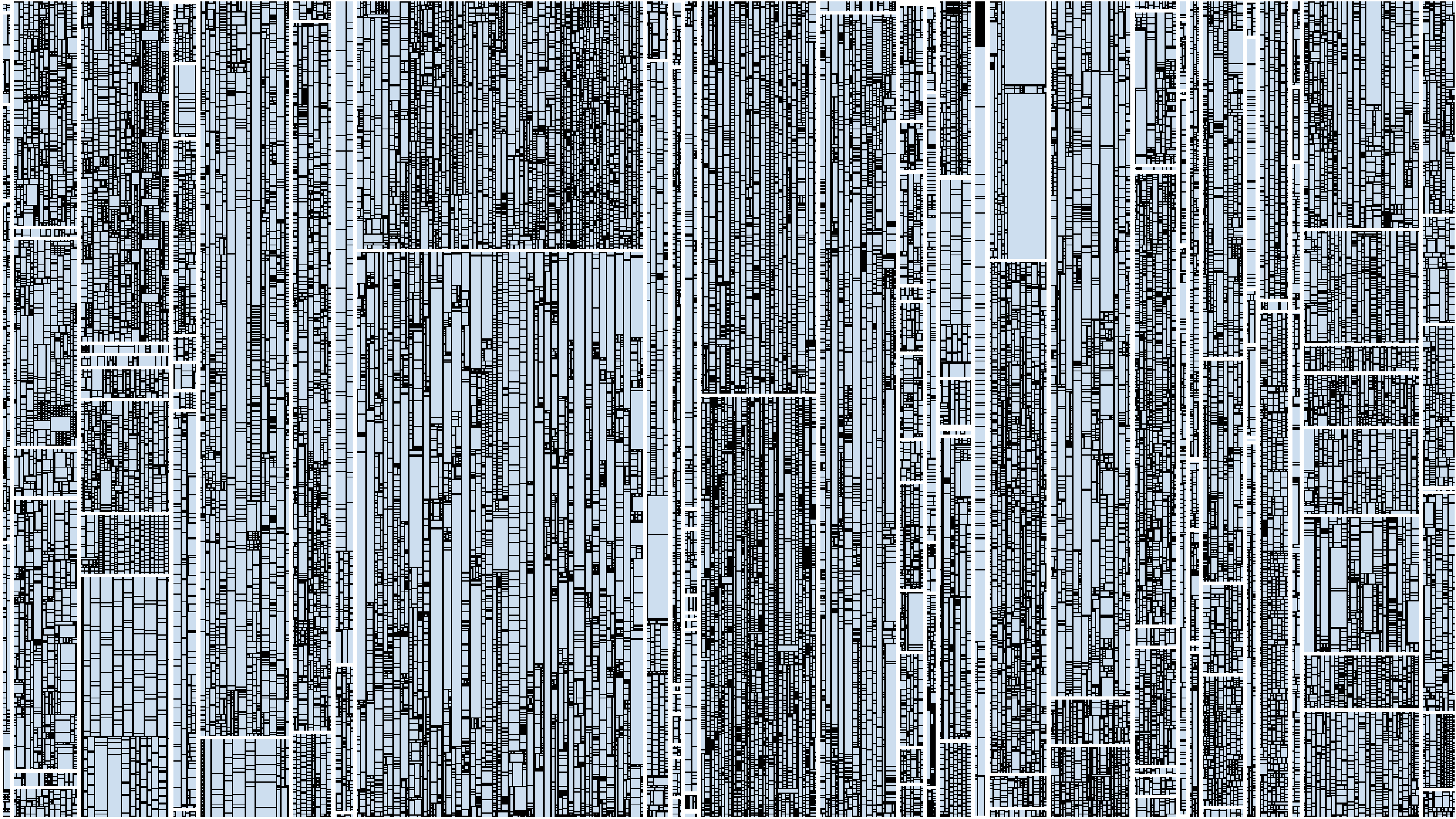


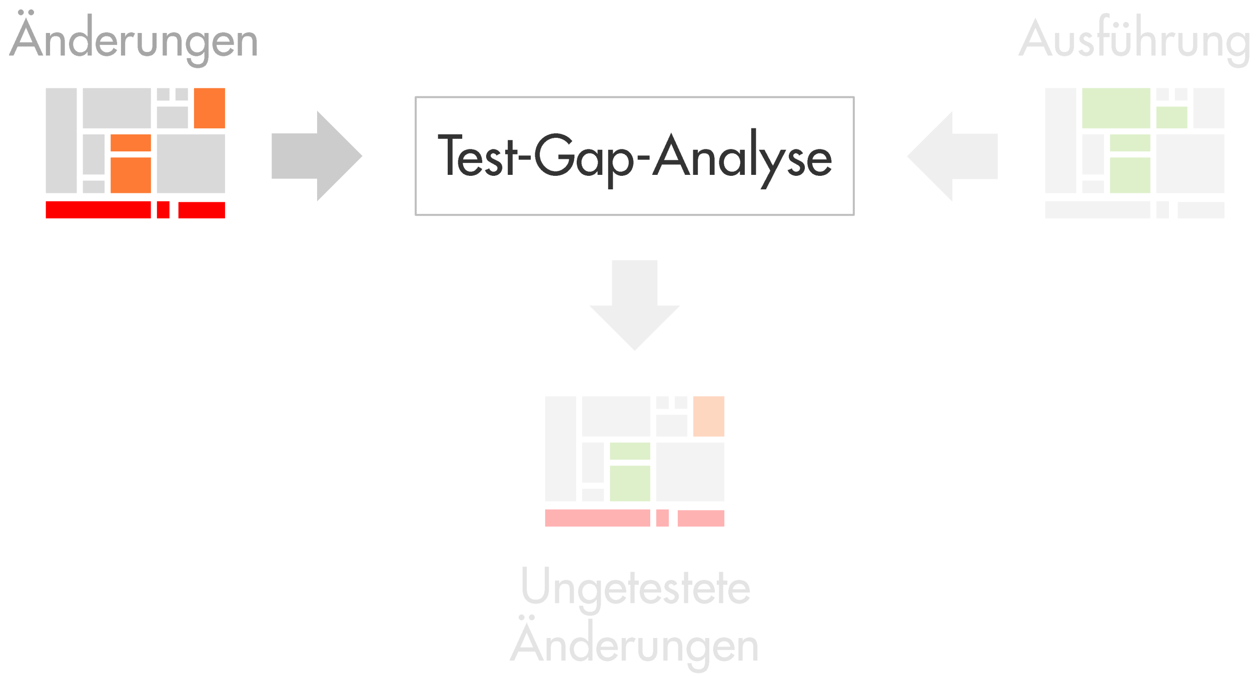
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2021
2022
2023
2024
2025

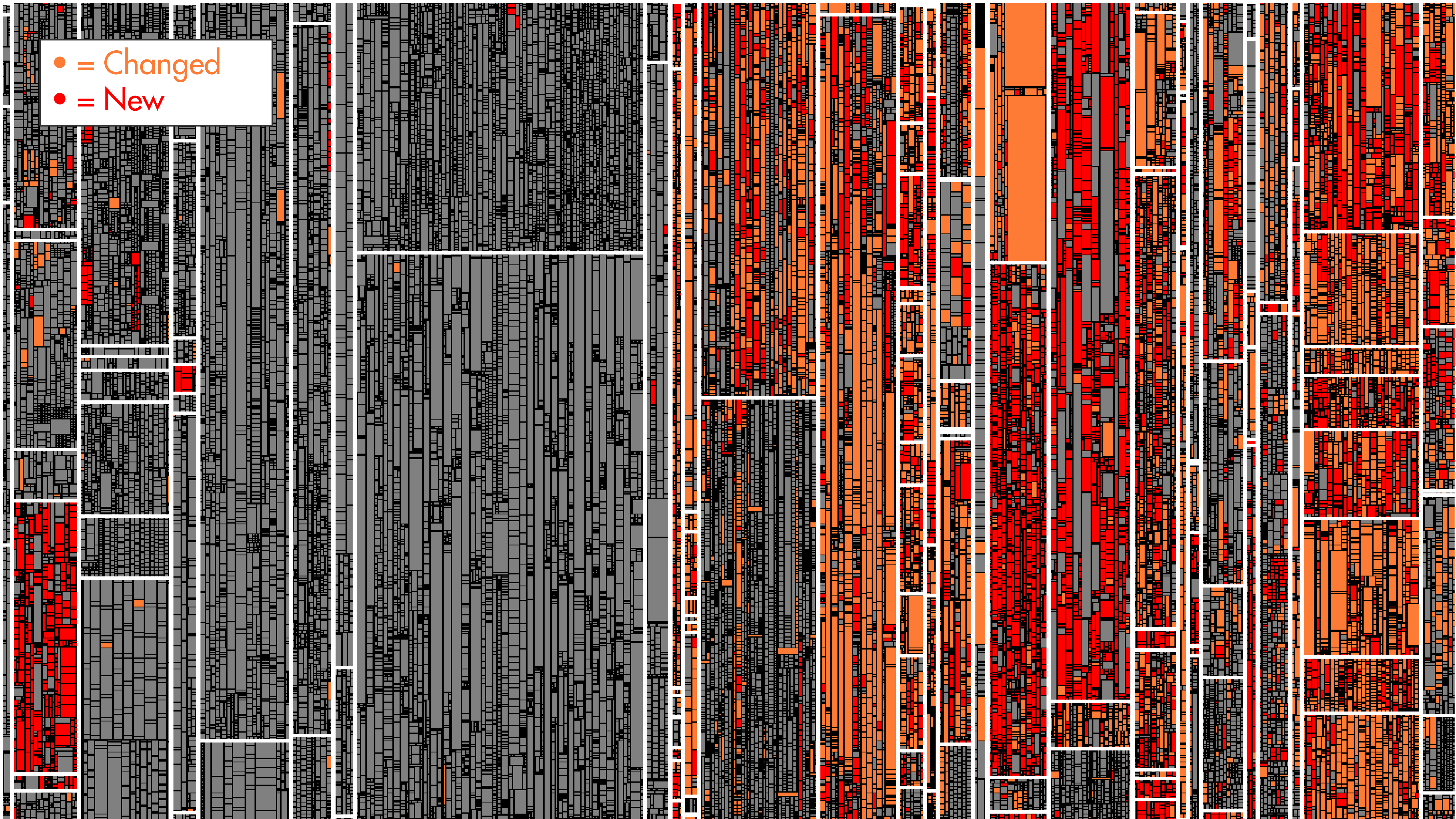

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete
Änderungen

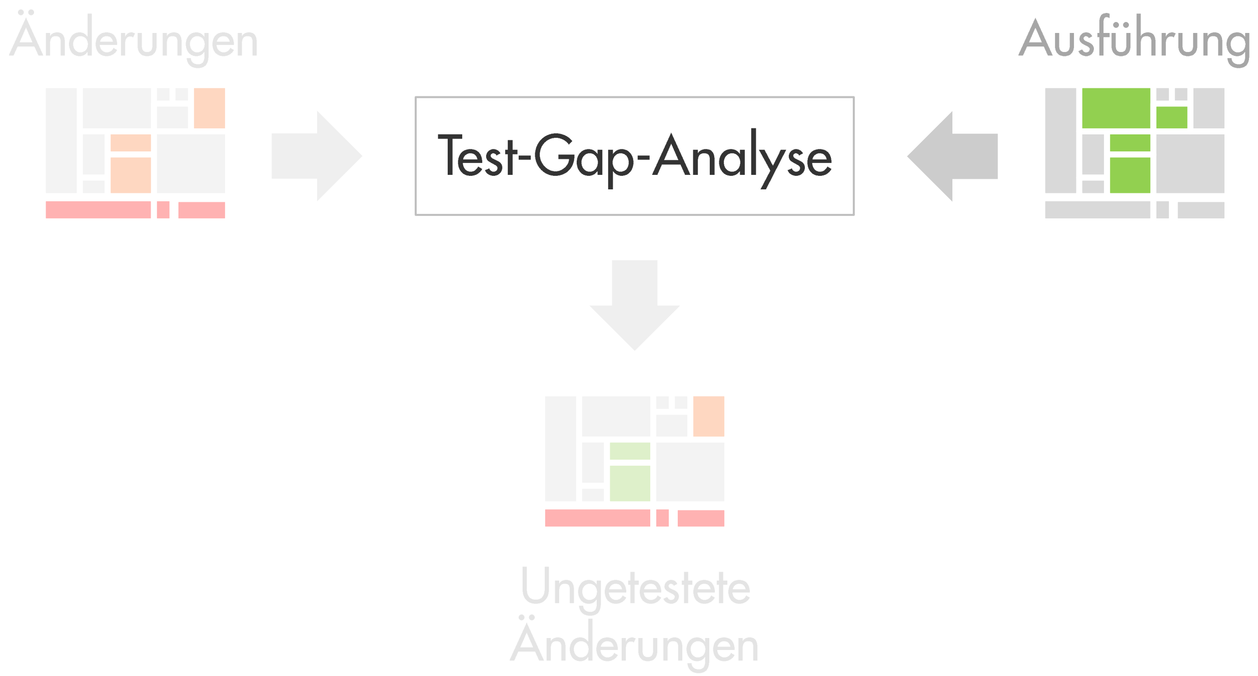


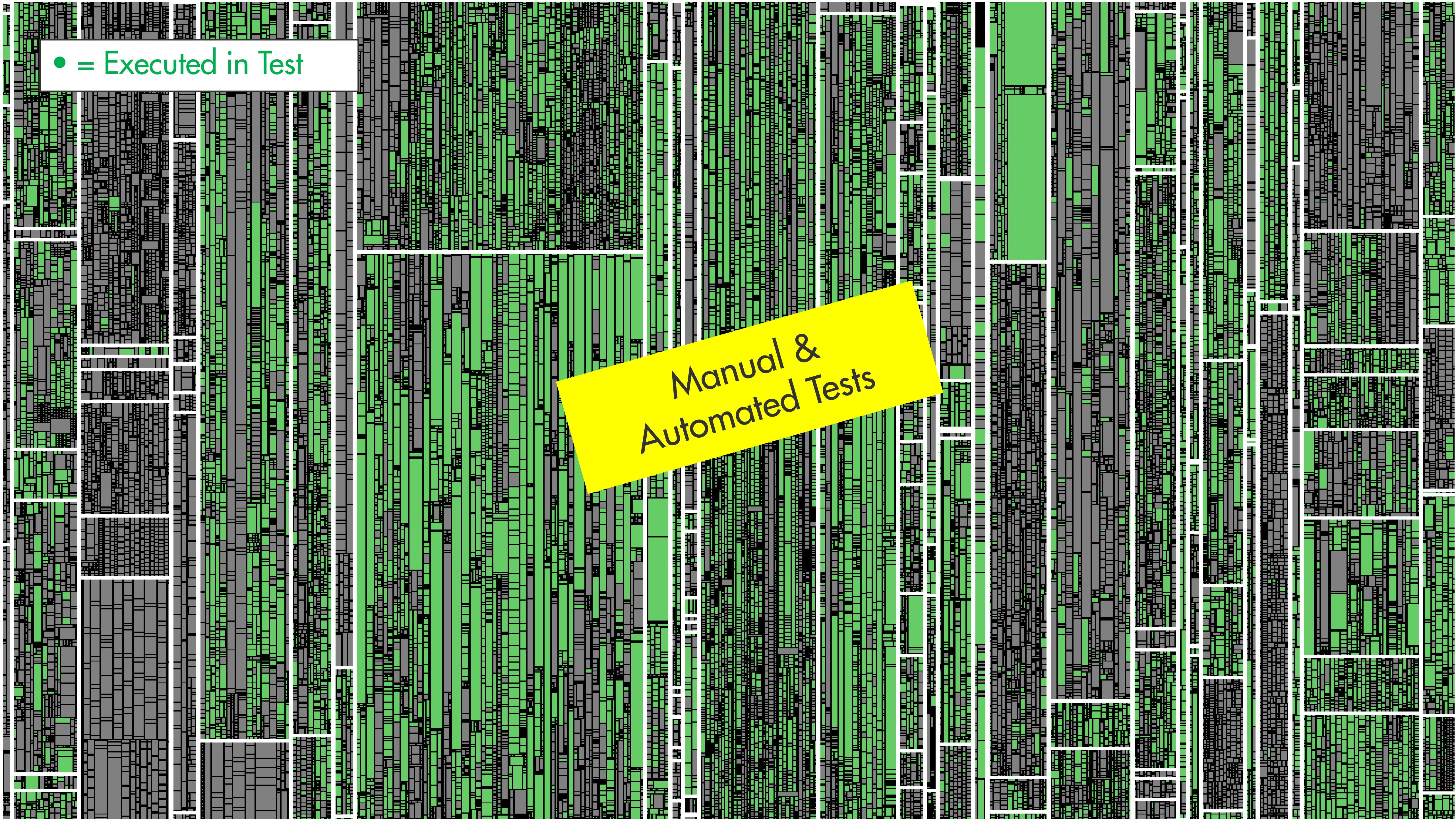


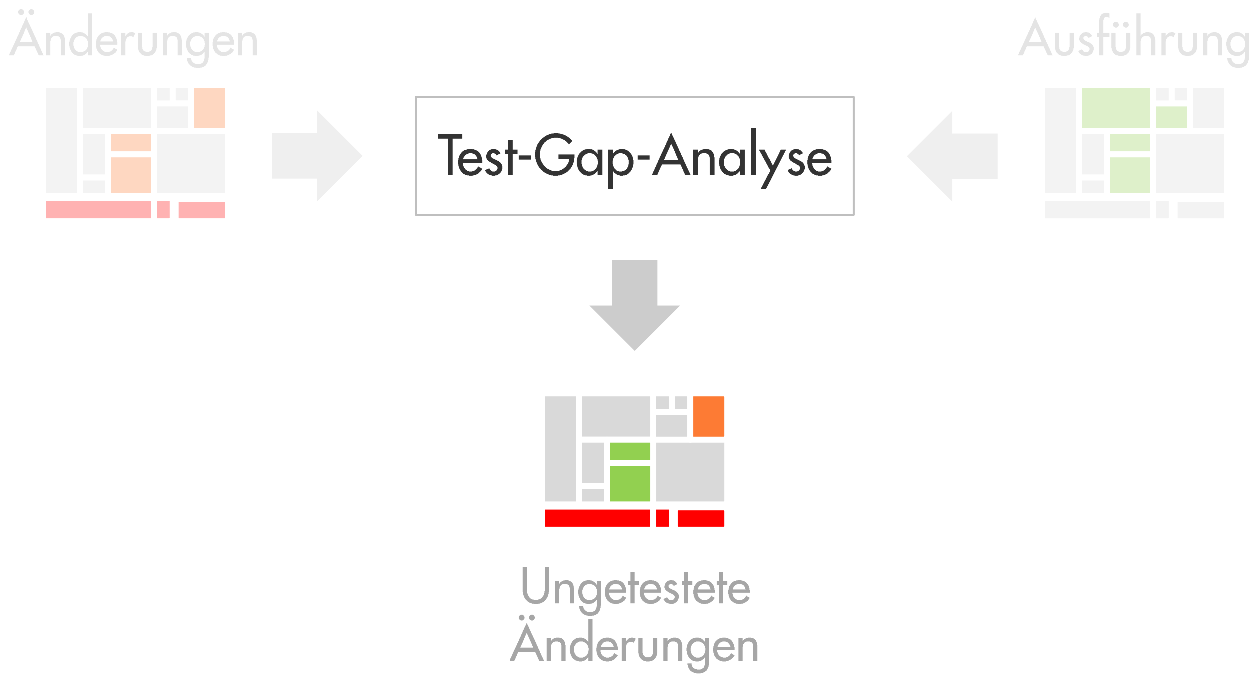


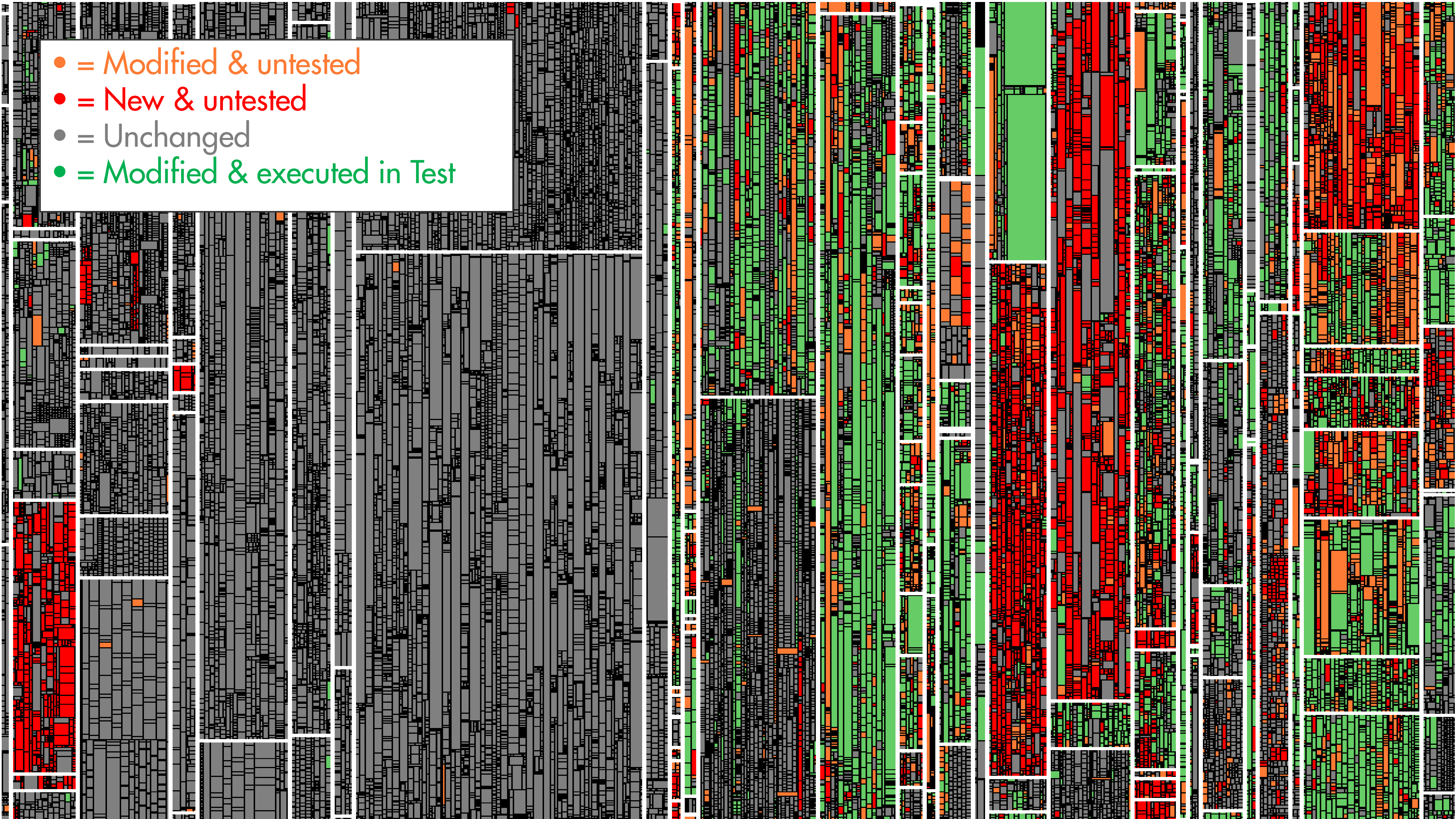

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete
Änderungen

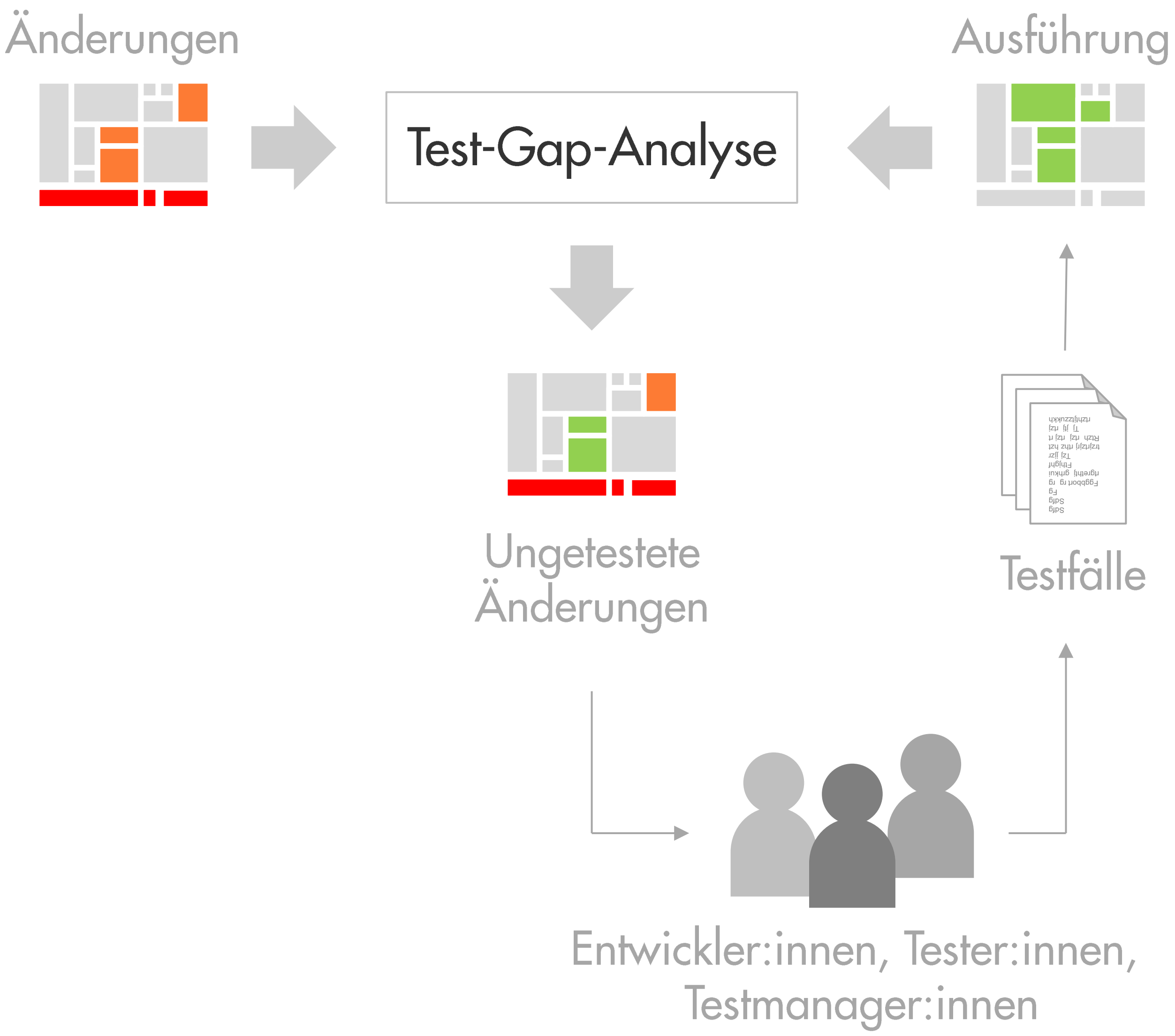

= Changed
= New


Änderungen
Ausführung
Test-Gap-Analyse
Ungetestete
Änderungen


● = Executed in Test
Manual & Automated Tests

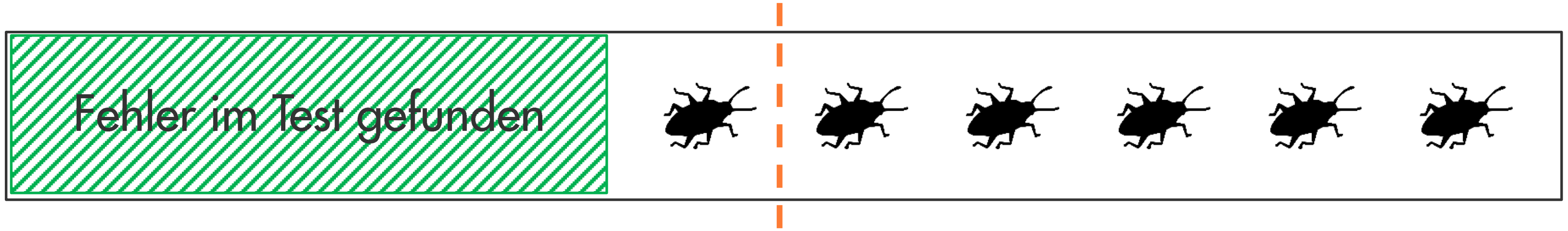

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete Änderungen


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


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

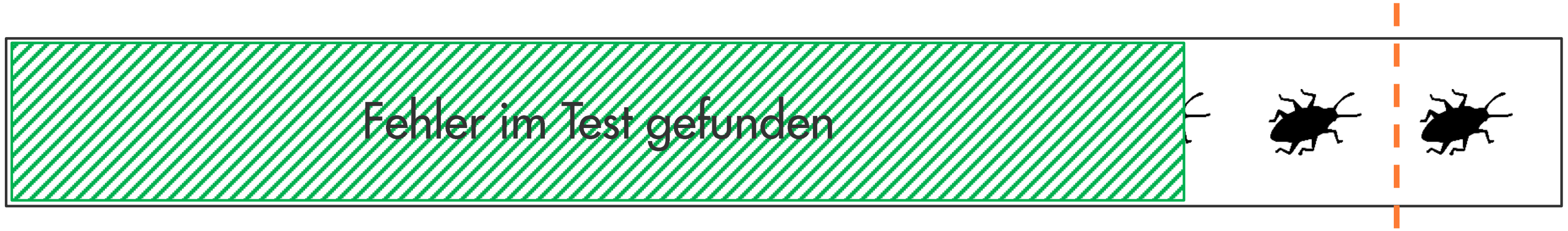
Ohne Test-Gap-Analyse

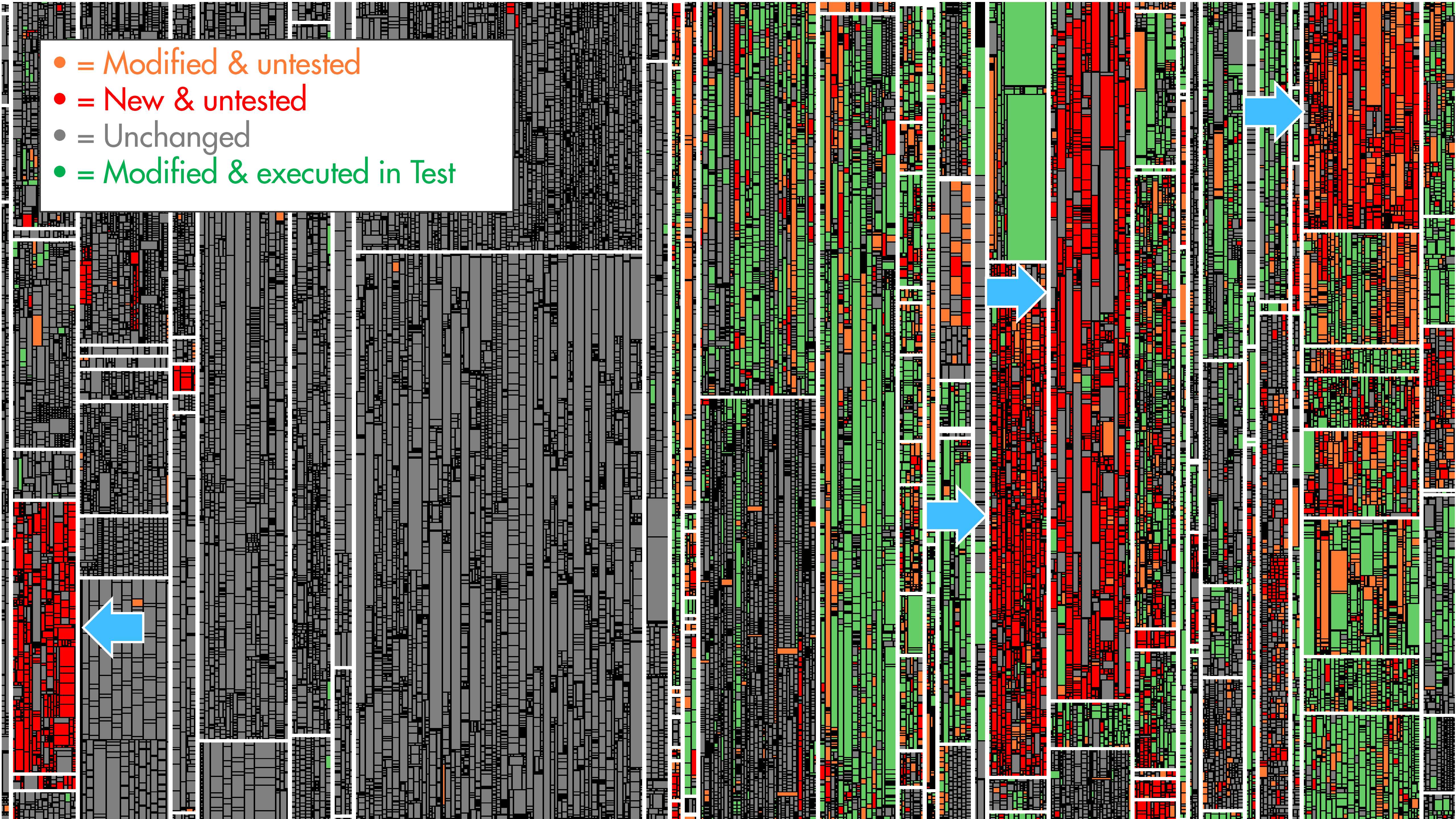
$$\%\text{Restfehler} = 60\%$$



Mit Test-Gap-Analyse

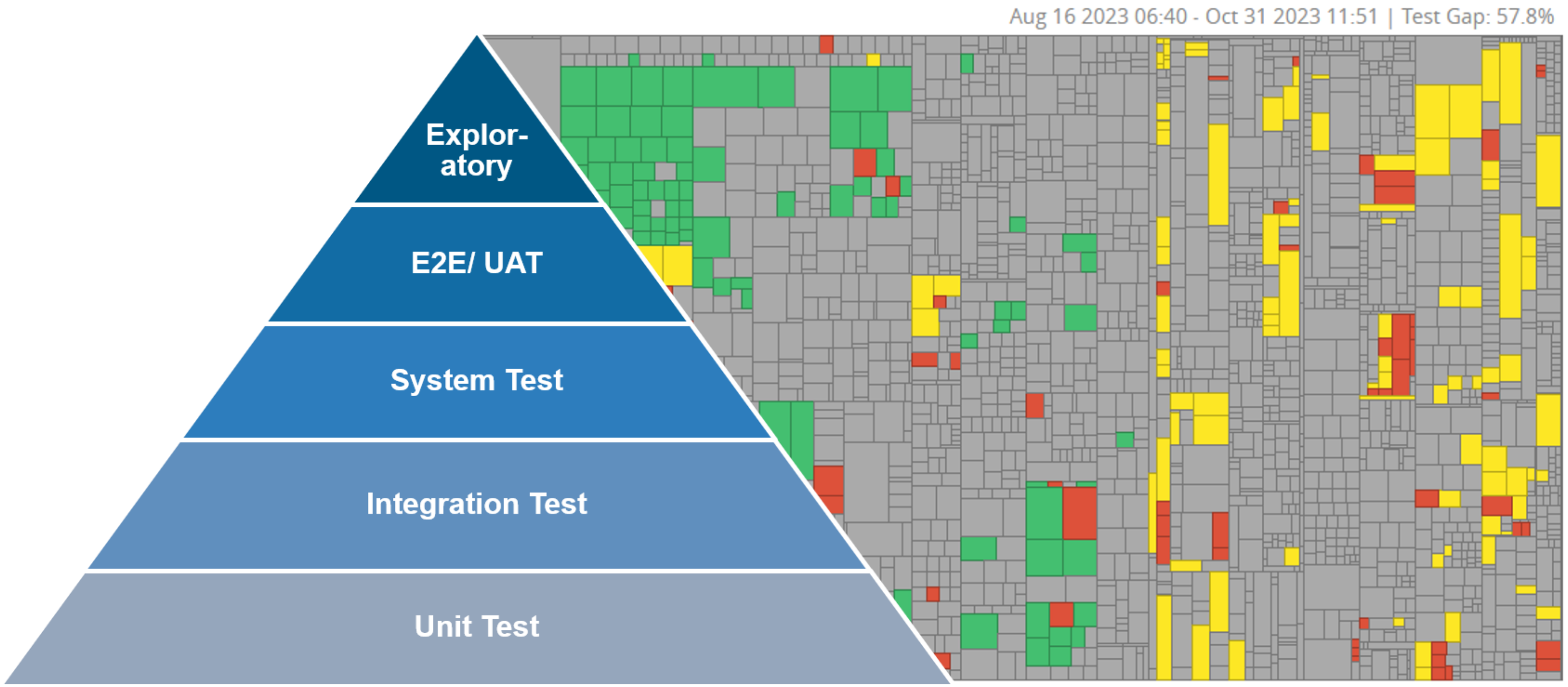
$$\%\text{Restfehler} = 28\%$$





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

Testabdeckung Ergebnisse der ersten Messung (1/2)

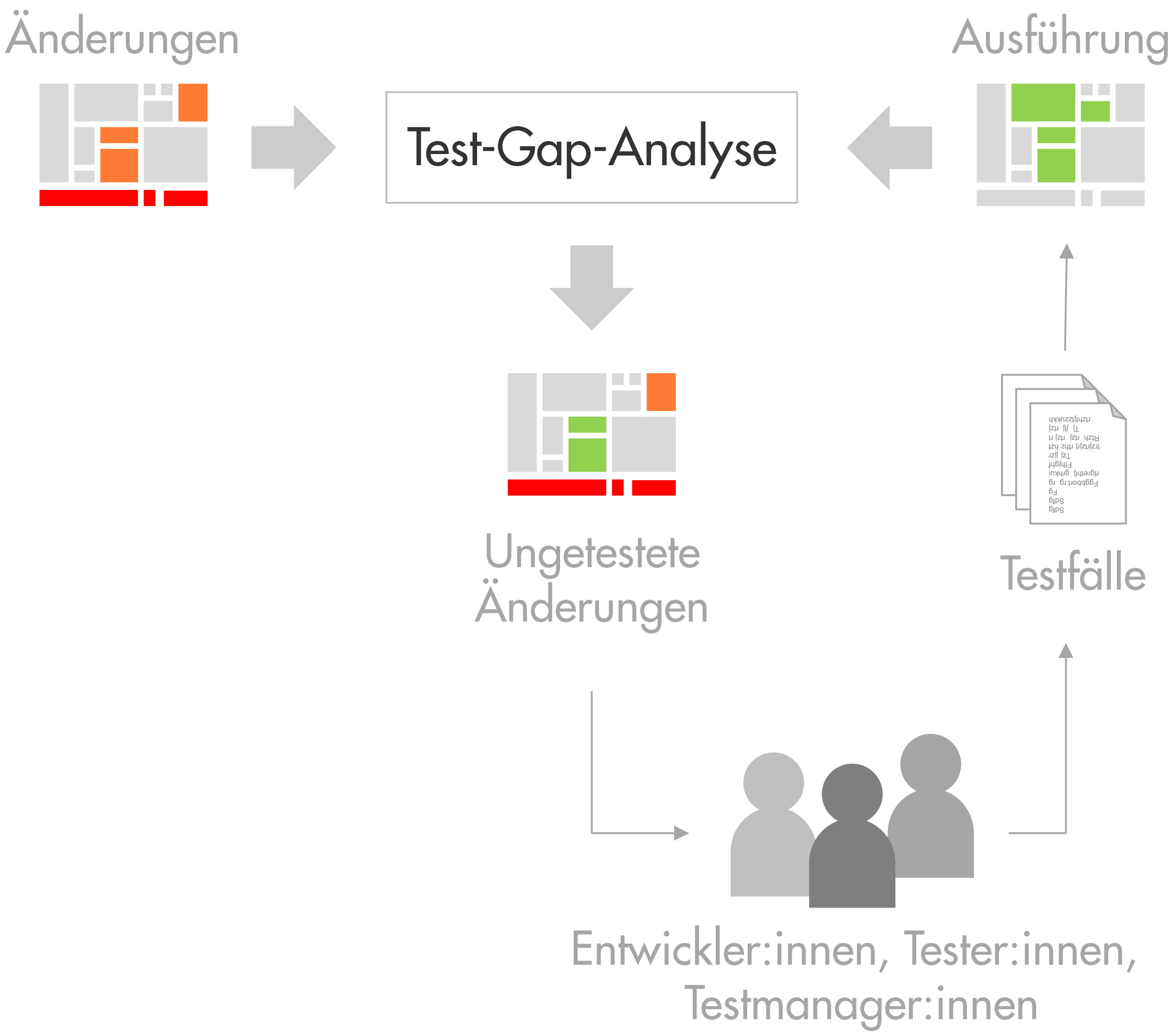


57,8% Test Gap

Messungs-
zeitraum
2,5 Monate

Alle
Testportfolios

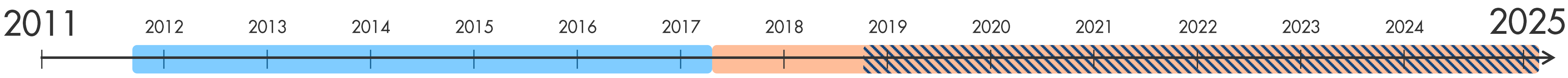
Manuelle &
Automatisierte
Tests

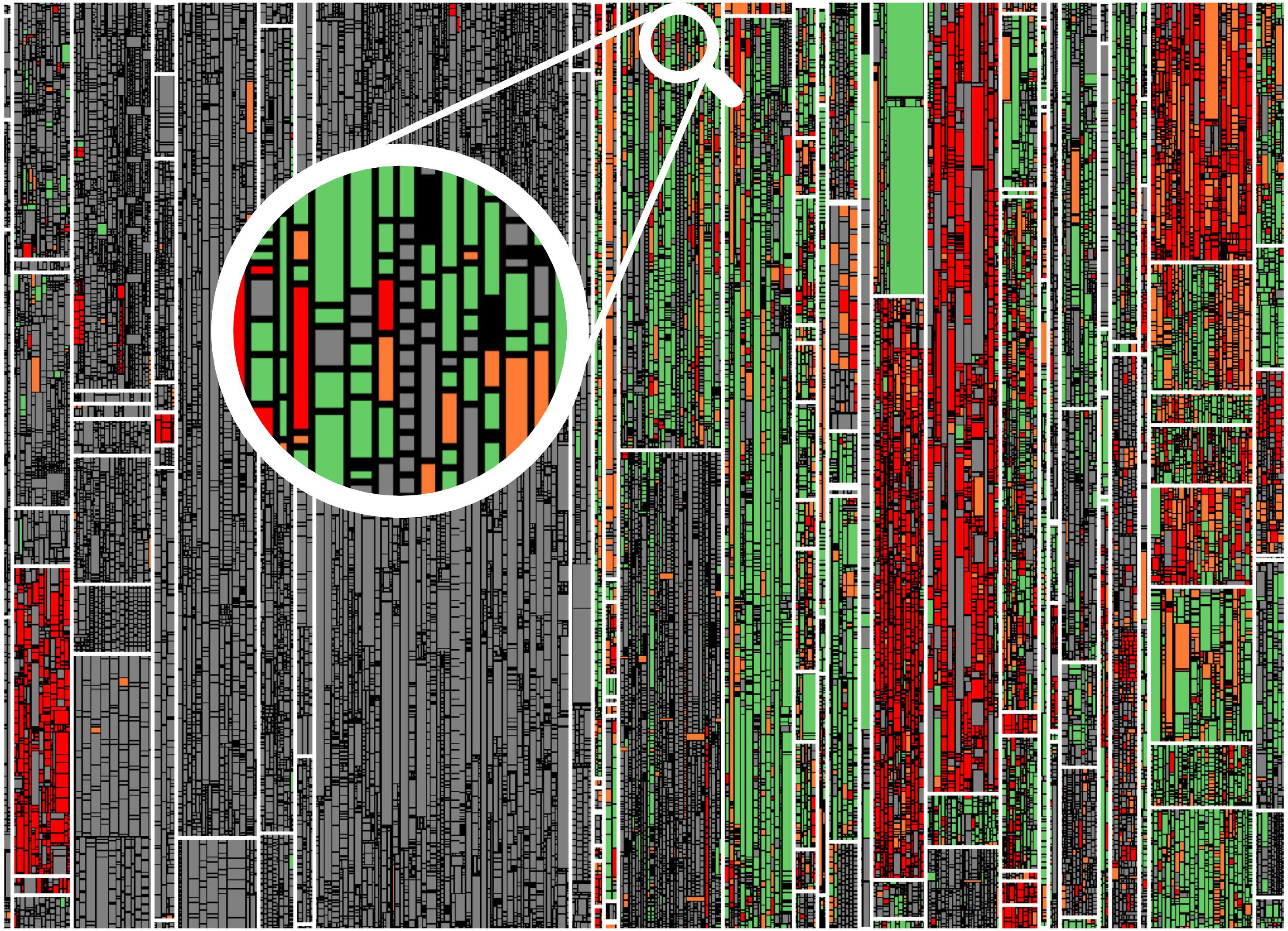

Änderungen
Test-Gap-Analyse
Ausführung
Ungetestete Änderungen
Testfälle
Entwickler:innen, Tester:innen, Testmanager:innen

?

CQSE

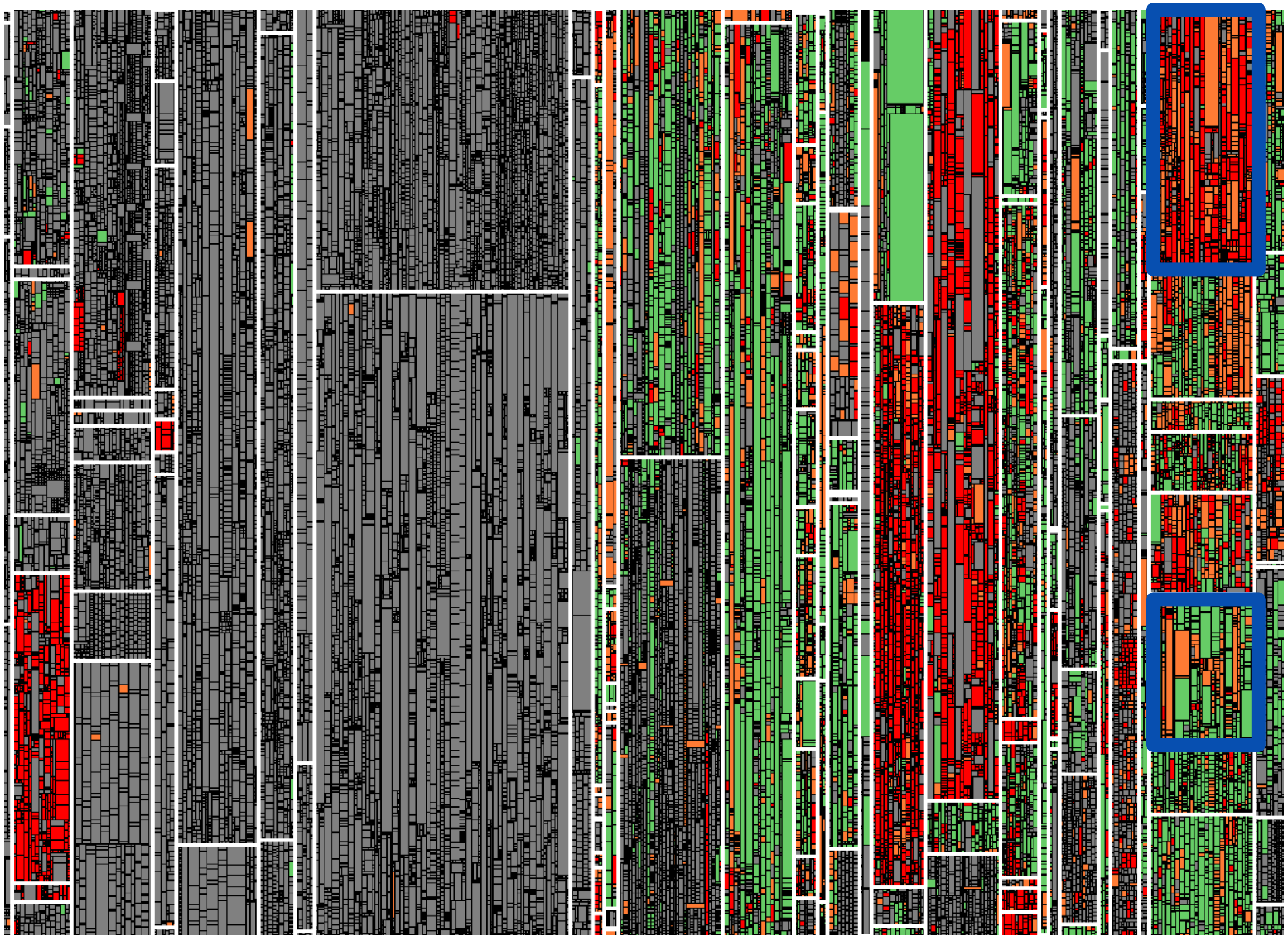
UNIVERSITÄT
DES
SAARLANDES





Welche Test-Gaps sind relevant und müssen geschlossen werden?

Test Manager



Welche Test-Gaps sind relevant und müssen geschlossen werden?

Hohe Fehlerwahr-scheinlichkeit

Hoher potentieller Schaden

Risikobasierte Priorisierung von Test-Gaps

Prioritizing Test Gaps by Risk in Industrial Practice: An Automated Approach and Multi-Method Study

Roman Haas, Michael Sailer, Mitchell Joblin, Elmar Juergens, and Sven Apel

Abstract*—*Context. **Untested code changes, called *test gaps*, pose a significant risk for software projects. Since test gaps increase the probability of defects, managing test gaps and their individual risk is important, especially for rapidly changing software systems.**

Objective. **This study aims at gaining an understanding of test gaps in industrial practice establishing criteria for precise prioritization of test gaps by their risk, informing practitioners that need to manage, review, and act on larger sets of test gaps.**

Method. **We propose an automated approach for prioritizing test gaps based on key risk criteria. By means of an analysis of 31 historical test gap reviews from 8 industrial software systems of our industrial partners Munich Re and LV 1871, and by conducting semi-structured interviews with the 6 quality engineers that authored the historical test gap reviews, we validate the transferability of the identified risk criteria, such as code criticality and complexity metrics.**

Results. **Our automated approach exhibits a ranking performance equivalent to expert assessments, in that test gaps labelled as risky in historical test gap reviews are highly ranked, on average, on the 23rd percentile. In some scenarios, our automated ranking system even outpaces expert assessments, especially for test gaps in central code—for non-developers an opaque code property.**

Conclusion. **This research underscores the industrial need of test gap risk estimation techniques to assist test management and quality assurance teams in identifying and addressing critical test gaps. Our multi-method study shows that even a lightweight prioritization approach helps practitioners to identify high-risk test gaps efficiently and to filter out low-risk test gaps.**

***Index Terms*—Software Testing, Test Gap Analysis, Risk-based Testing**

I. Introduction

FUNCTIONAL correctness is crucial for the success and acceptance of a software product. A solid testing process is imperative to uncover defects before they are deployed in the field. Since resources are limited, especially for large software heuristic search or machine learning [1]. Even though a large variety of studies has been conducted in this area, the results are often not generalizable [2], and the approaches perform poorly in real-world settings [3], [4]. As a consequence, they are rarely applied in practice [5], [6], with notable exceptions, though [7].

Addressing the notorious issues of defect prediction of our partners in industry (in particular, Munich Re and LV 1871), we strive for an approach that is viable in practice and matched the needs of our industry partners: a *prioritization* of *test gaps* by their *risk*. A *test gap* is a method, function, or module that has been modified during a specific period of time (e.g., start of last development phase or iteration) and has not been executed in its most recent version during testing (e.g., automated unit test or manual acceptance test). Intuitively, defects are introduced by code changes, and defects cannot be detected if they were not tested. In this vein, the literature suggests that modified code tends to be more defect-prone [8]–[11]. For example, Eder et al. found in an industrial case study that (1) despite a structured testing process, approximately half of the changes went into production untested, and (2) that untested changes contained up to five times more defects than other parts of the system. This clearly emphasizes the value of test gap analysis in the testing process. For this reason, test gaps are taken into consideration by test management to decide whether testing is completed.[1]

Problem Statement: The number of test gaps that need to be investigated by test management and quality assurance depends on many parameters, especially on the number of code changes and the depth of testing. In practice, when test management assesses test gaps as part of test-end criteria evaluation, there are typically dozens, hundreds, or even thousands of test gaps [12] which were not covered by automated or manual test runs. More importantly, the risk of test gaps may

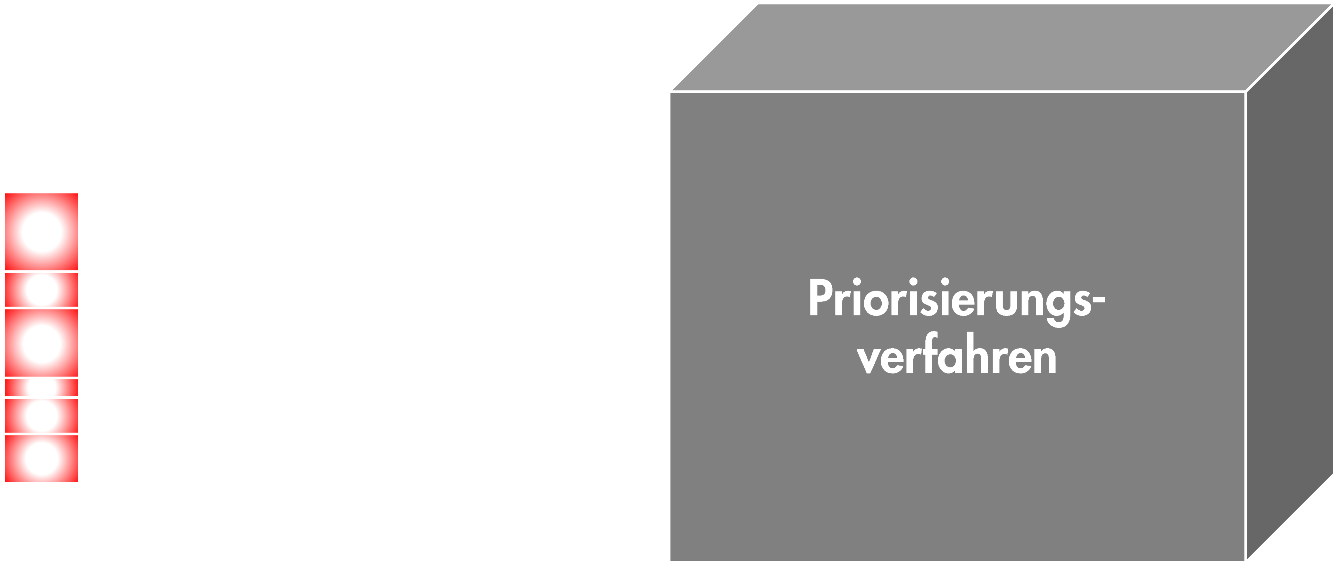
IEEE Transactions on Software Engineering

Risikobasierte Priorisierung von Test-Gaps

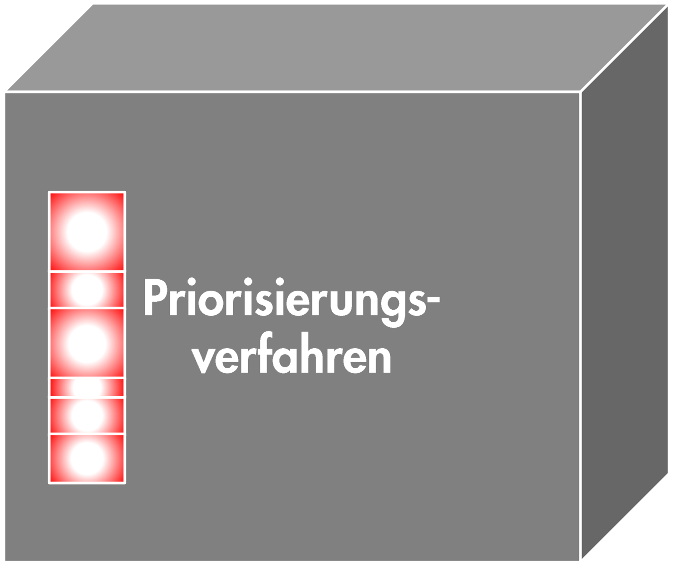
Risikobasierte Priorisierung von Test-Gaps

Risikobasierte Priorisierung von Test-Gaps

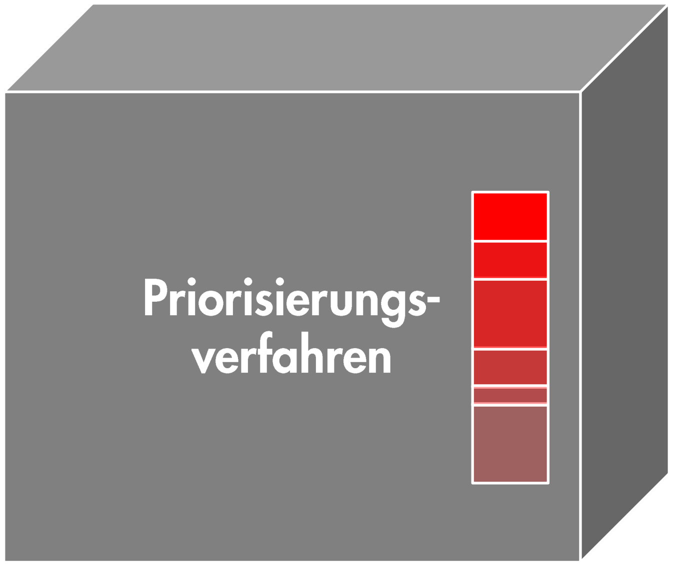
Risikobasierte Priorisierung von Test-Gaps



Risikobasierte Priorisierung von Test-Gaps



Risikobasierte Priorisierung von Test-Gaps



Risikobasierte Priorisierung von Test-Gaps

Dimensionen zur Abschätzung des Risikos

Code Kritikalität

Komplexität

Findings

Metriken zur Abschätzung des Risikos

Code Kritikalität	Komplexität	Findings
Code Zentralität Geänderte Funktionen	Länge der Ref.fktn. Geänderte LOC Komplexität der Ref.fktn. Komplexitätsänderung	Neue kritische F. Nicht behobene kritische F. Behobene kritische F. Neue normale F. Nicht normale kritische F. Behobene normale F.

Metriken zur Abschätzung des Risikos

Code Kritikalität

Komplexität

Findings

Code Zentralität
Geänderte Funktionen

Länge der Ref.fktn.
Geänderte LOC
Komplexität der Ref.fktn.
Komplexitätsänderung

Neue kritische F.
Nicht behobene kritische F.
Behobene kritische F.
Neue normale F.
Nicht normale kritische F.
Behobene normale F.

Risiko Score r_m für Test-Gap m

$$r_m = \sum_{i=1}^{k} V_m[i] \cdot W[i]$$

mit $V_m[i]$ als Metrikwerte von m
und Metrikgewichten $W[i]$

Metrik i	$W[i]$
Code Zentralität	2
Geänderte Funktionen	-1
Länge der Referenzfunktion	1
Geänderte LOC	2
Komplexität der Ref.fktn.	1
Komplexitätsänderung	2
Neue kritische Findings	2
Nicht behobene kritische F.	1
Behobene kritische Findings	-1
Neue normale Findings	4
Nicht normale kritische Findings	2
Behobene normale Findings	-2

Evaluation auf acht Industriesystemen

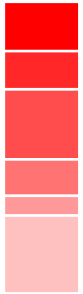
Munich RE

LV 1871

Quality Engineers

- A#foo
- B#bar

vs.



Historische Test-Gap-Reviews

2023-12: 58 test gaps. 39.2% of new or changed functions appear untested. Some test gaps appear to be of minor importance, but there are some relevant ones as well, for example:

- Function **foo** in class **A** [Link1]
- Function **bar** in class **B** [Link2]

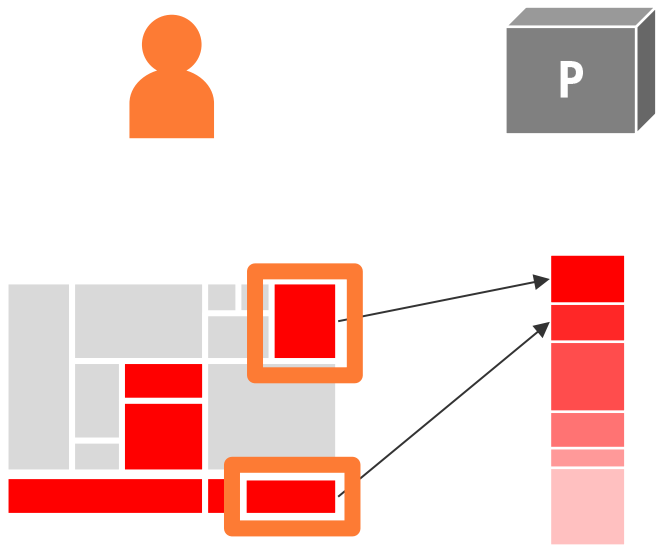
Quality Engineers

Studiensubjekte

					Test-Gaps	
Firma	**Subjekt**	**LOC**	**Sprache**	**# Reviews**	**Riskant**	**Gesamt**
Munich RE	1	1,600 K	C#	5	59	77
	2	140 K	C#	7	29	161
	3	370K	ABAP	3	21	29
	4	560 K	ABAP	4	9	32
	5	1,900 K	ABAP	4	26	53
LV 1871	6	310 K	Java	3	5	622
	7	100 K	Java	3	28	1,052
	8	150 K	Java	2	4	13

Studiendesign

Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?

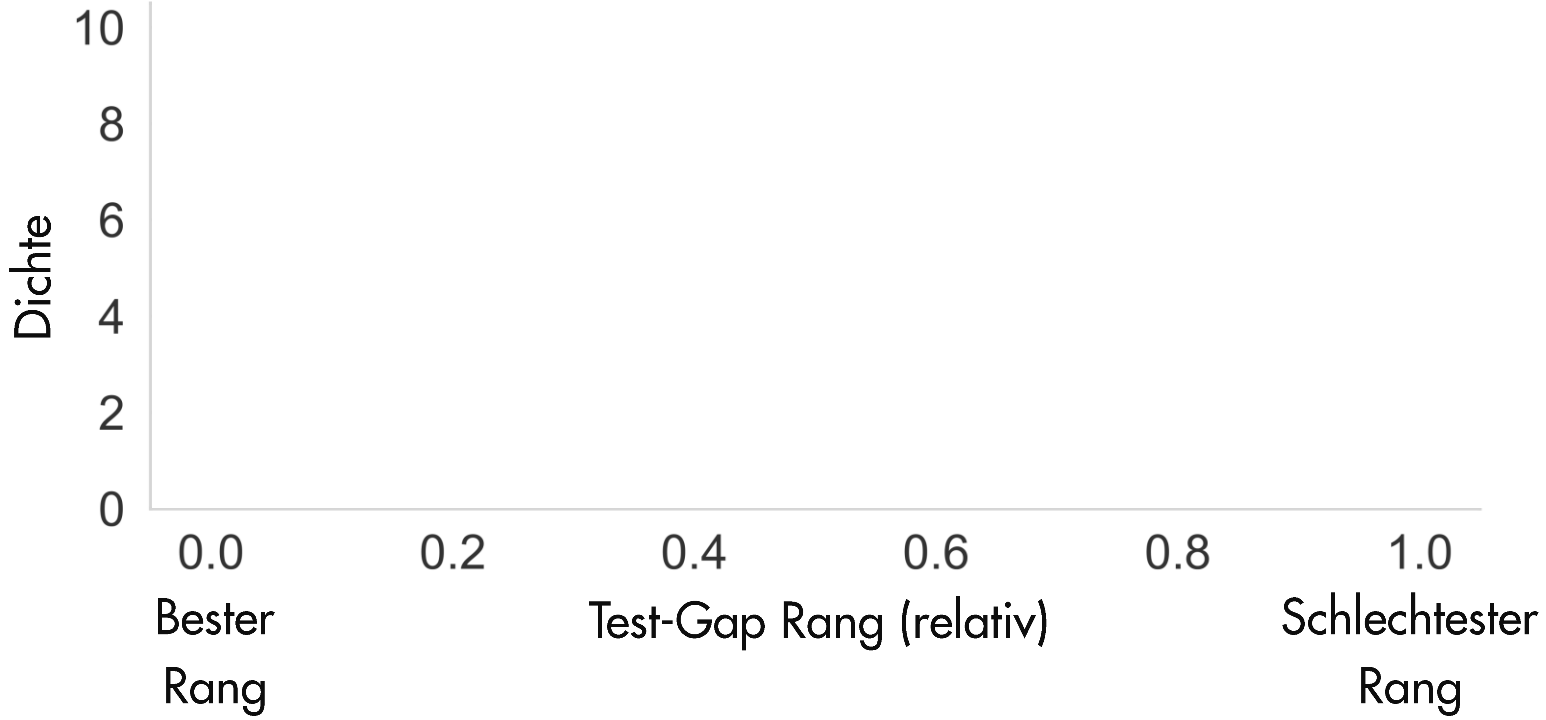


Rangvergleich von priorisierten Test-Gaps

mit von Quality Engineers als riskant gelabelten und anderen Test-Gaps

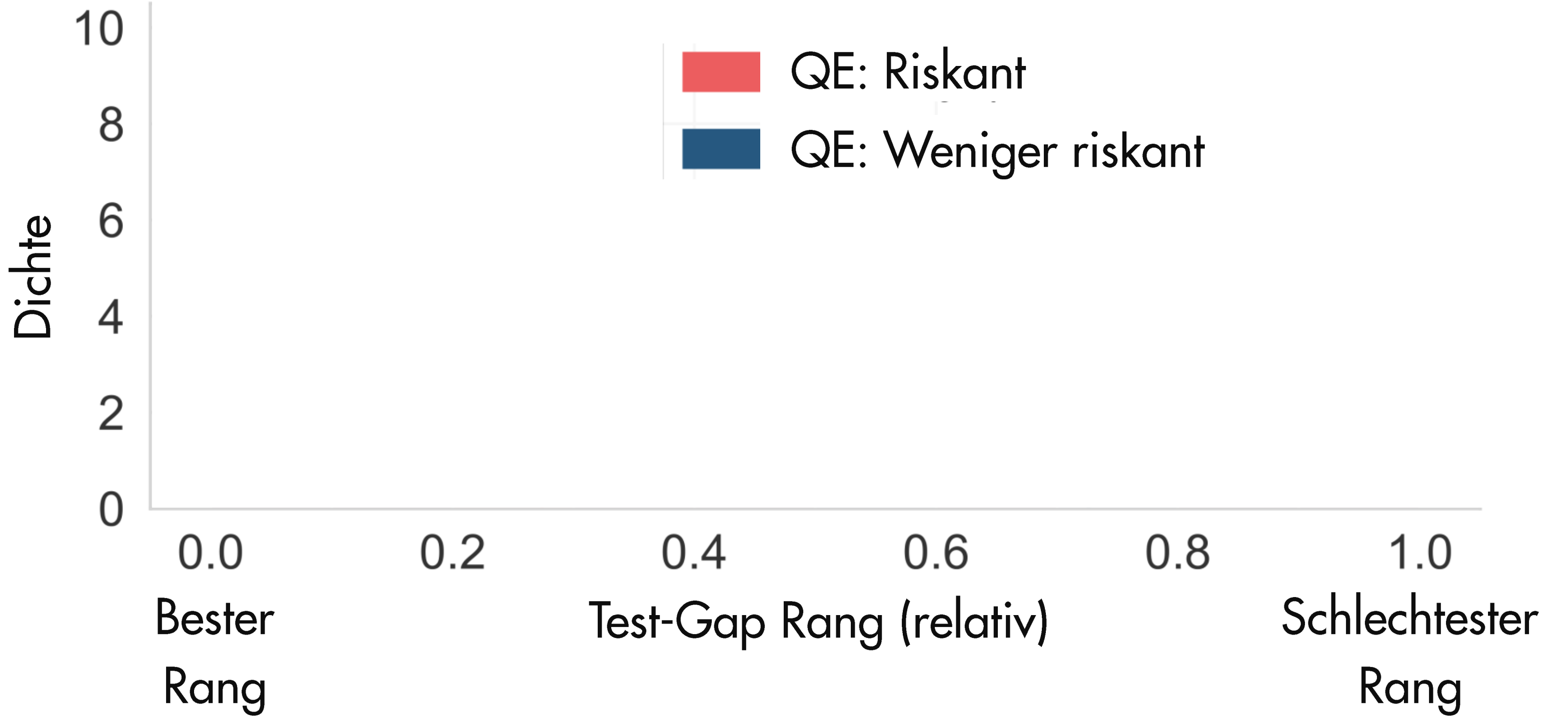
Ergebnisse

Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?



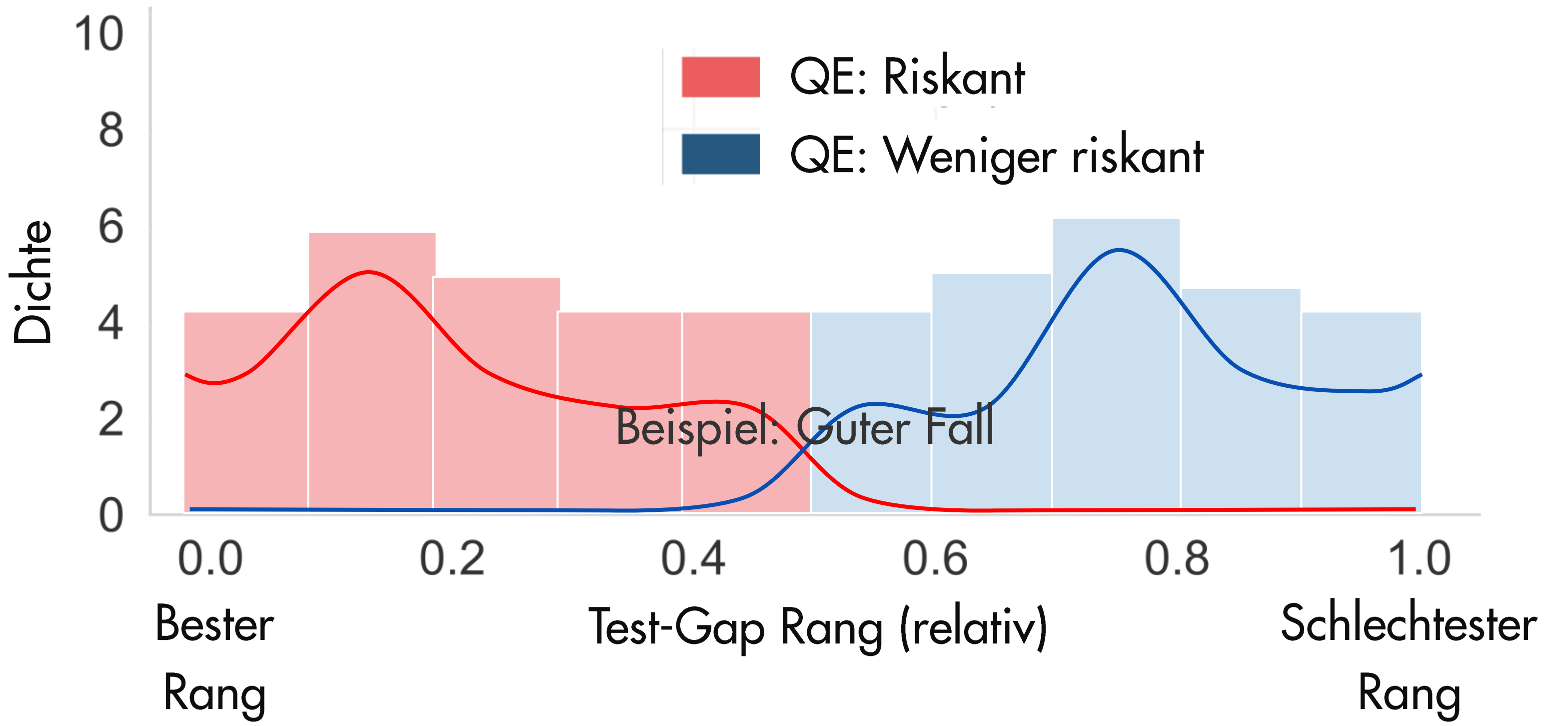
Ergebnisse

Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?



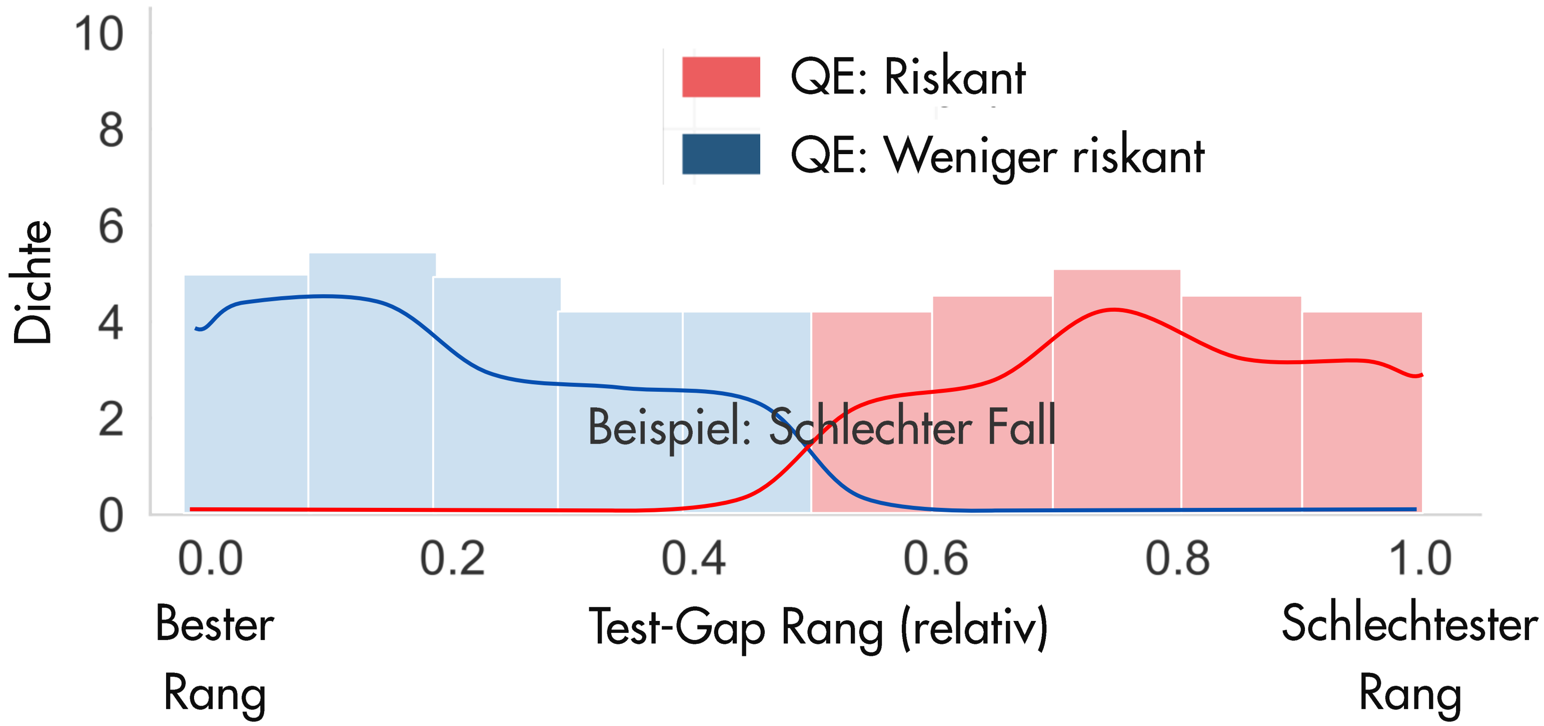
Ergebnisse

Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?



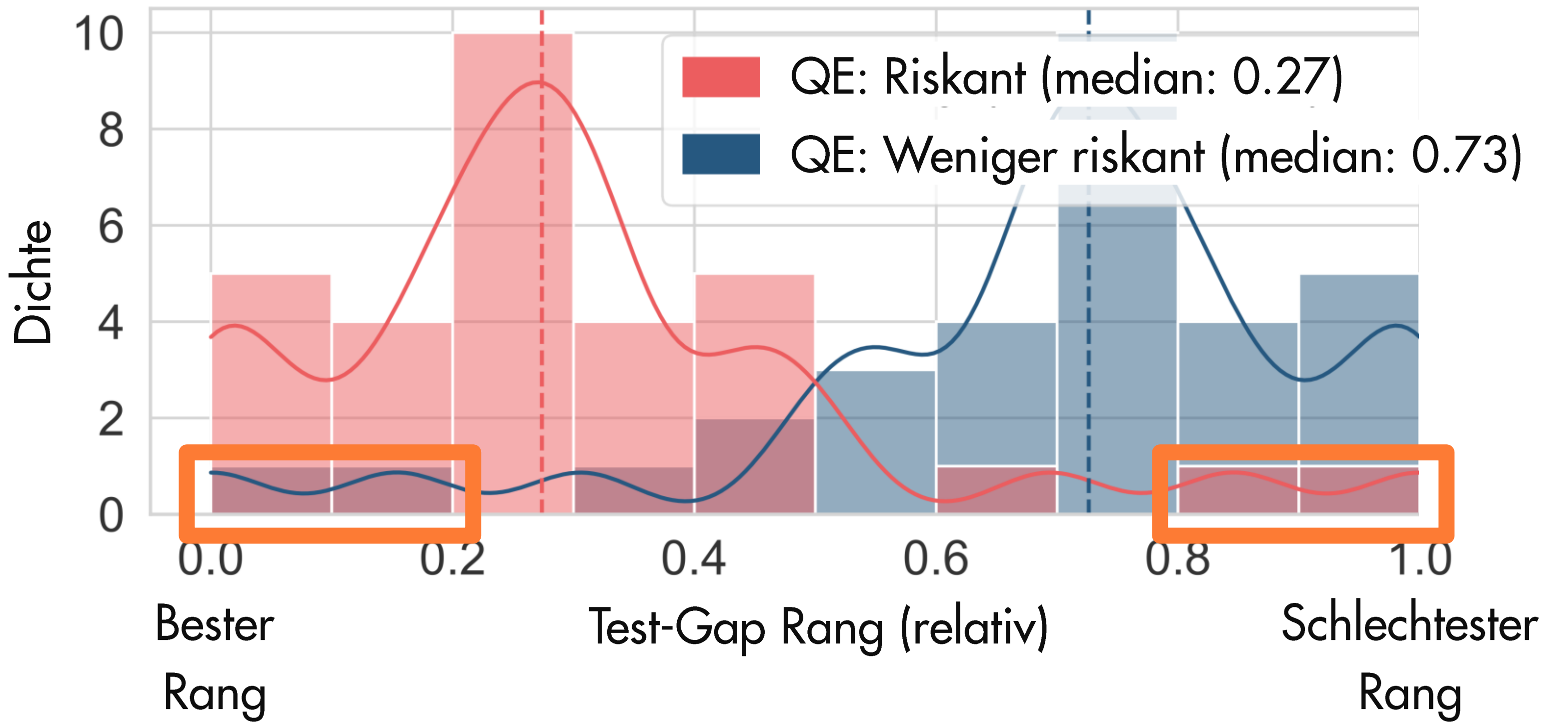
Ergebnisse

Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?



Ergebnisse

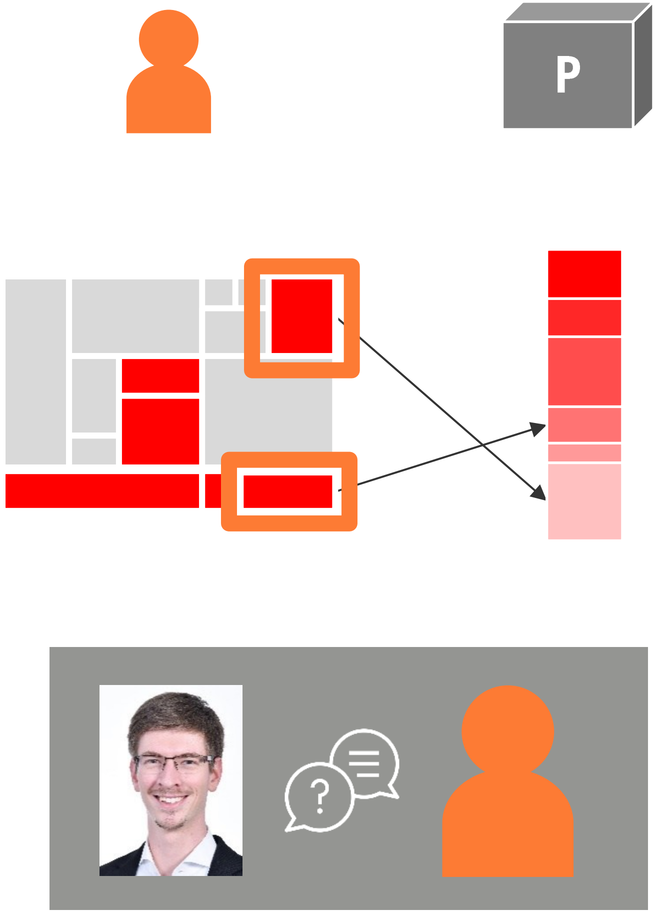
Wie schneidet unser Ansatz im Vergleich zu Risiko-Assessments von Quality Engineers ab?



- Von Quality Engineers als **riskant** markierte Test-Gaps werden im Durchschnitt deutlich **höher gerankt** als weniger riskante.

Studiendesign

Meinen Quality Engineers, dass ihnen die Test-Gap Priorisierung in ihrer täglichen Arbeit hilft, und wenn ja, wie?

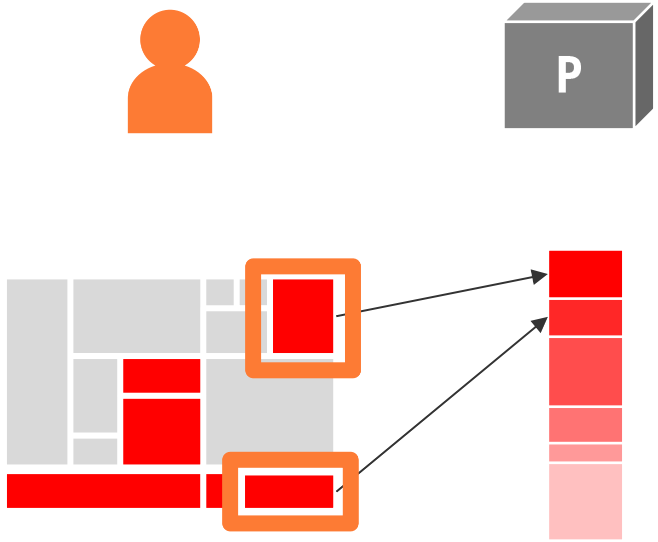


Ergebnisse

Meinen Quality Engineers, dass ihnen die Test-Gap Priorisierung in ihrer täglichen Arbeit hilft, und wenn ja, wie?

Metriken	
	■ Aussagekräftig und repräsentativ ■ Zentralitätsmetrik besonders hilfreich

Schlussfolgerung



- Die **risikobasierte Priorisierung** von Test-Gaps **funktioniert** gut
- Quality Engineers sehen **großen Mehrwert** für Test Gap Reviews

Prioritizing Test Gaps by Risk in Industrial Practice: An Automated Approach and Multi-Method Study

Roman Haas, Michael Sailer, Mitchell Joblin, Elmar Juergens, and Sven Apel

Abstract—Context. Untested code changes, called *test gaps*, pose a significant risk for software projects. Since test gaps increase the probability of defects, managing test gaps and their individual risk is important, especially for rapidly changing software systems.

Objective. This study aims at gaining an understanding of test gaps in industrial practice establishing criteria for precise prioritization of test gaps by their risk, informing practitioners that need to manage, review, and act on larger sets of test gaps.

Method. We propose an automated approach for prioritizing test gaps based on key risk criteria. By means of an analysis of 31 historical test gap reviews from 8 industrial software systems of our industrial partners Munich Re and LV 1871, and by conducting semi-structured interviews with the 6 quality engineers that authored the historical test gap reviews, we validate the transferability of the identified risk criteria, such as code criticality and complexity metrics.

Results. Our automated approach exhibits a ranking performance equivalent to expert assessments, in that test gaps labelled as risky in historical test gap reviews are highly ranked, on average, on the 23rd percentile. In some scenarios, our automated ranking system even outpaces expert assessments, especially for test gaps in central code—for non-developers an opaque code property.

Conclusion. This research underscores the industrial need of test gap risk estimation techniques to assist test management and quality assurance teams in identifying and addressing critical test gaps. Our multi-method study shows that even a lightweight prioritization approach helps practitioners to identify high-risk test gaps efficiently and to filter out low-risk test gaps.

Index Terms—Software Testing, Test Gap Analysis, Risk-based Testing

I. Introduction

FUNCTIONAL correctness is crucial for the success and acceptance of a software product. A solid testing process is imperative to uncover defects before they are deployed in the field. Since resources are limited, especially for large software heuristic search or machine learning [1]. Even though a large variety of studies has been conducted in this area, the results are often not generalizable [2], and the approaches perform poorly in real-world settings [3], [4]. As a consequence, they are rarely applied in practice [5], [6], with notable exceptions, though [7].

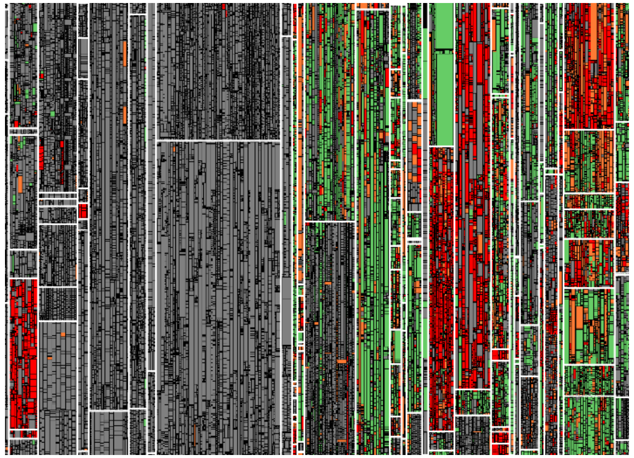
Addressing the notorious issues of defect prediction of our partners in industry (in particular, Munich Re and LV 1871), we strive for an approach that is viable in practice and matched the needs of our industry partners: a *prioritization* of *test gaps* by their *risk*. A *test gap* is a method, function, or module that has been modified during a specific period of time (e.g., start of last development phase or iteration) and has not been executed in its most recent version during testing (e.g., automated unit test or manual acceptance test). Intuitively, defects are introduced by code changes, and defects cannot be detected if they were not tested. In this vein, the literature suggests that modified code tends to be more defect-prone [8]–[11]. For example, Eder et al. found in an industrial case study that (1) despite a structured testing process, approximately half of the changes went into production untested, and (2) that untested changes contained up to five times more defects than other parts of the system. This clearly emphasizes the value of test gap analysis in the testing process. For this reason, test gaps are taken into consideration by test management to decide whether testing is completed.[1]

Problem Statement: The number of test gaps that need to be investigated by test management and quality assurance depends on many parameters, especially on the number of code changes and the depth of testing. In practice, when test management assesses test gaps as part of test-end criteria evaluation, there are typically dozens, hundreds, or even thousands of test gaps [12] which were not covered by automated or manual test runs. More importantly, the risk of test gaps may

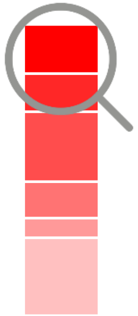
Download

IEEE Transactions on Software Engineering

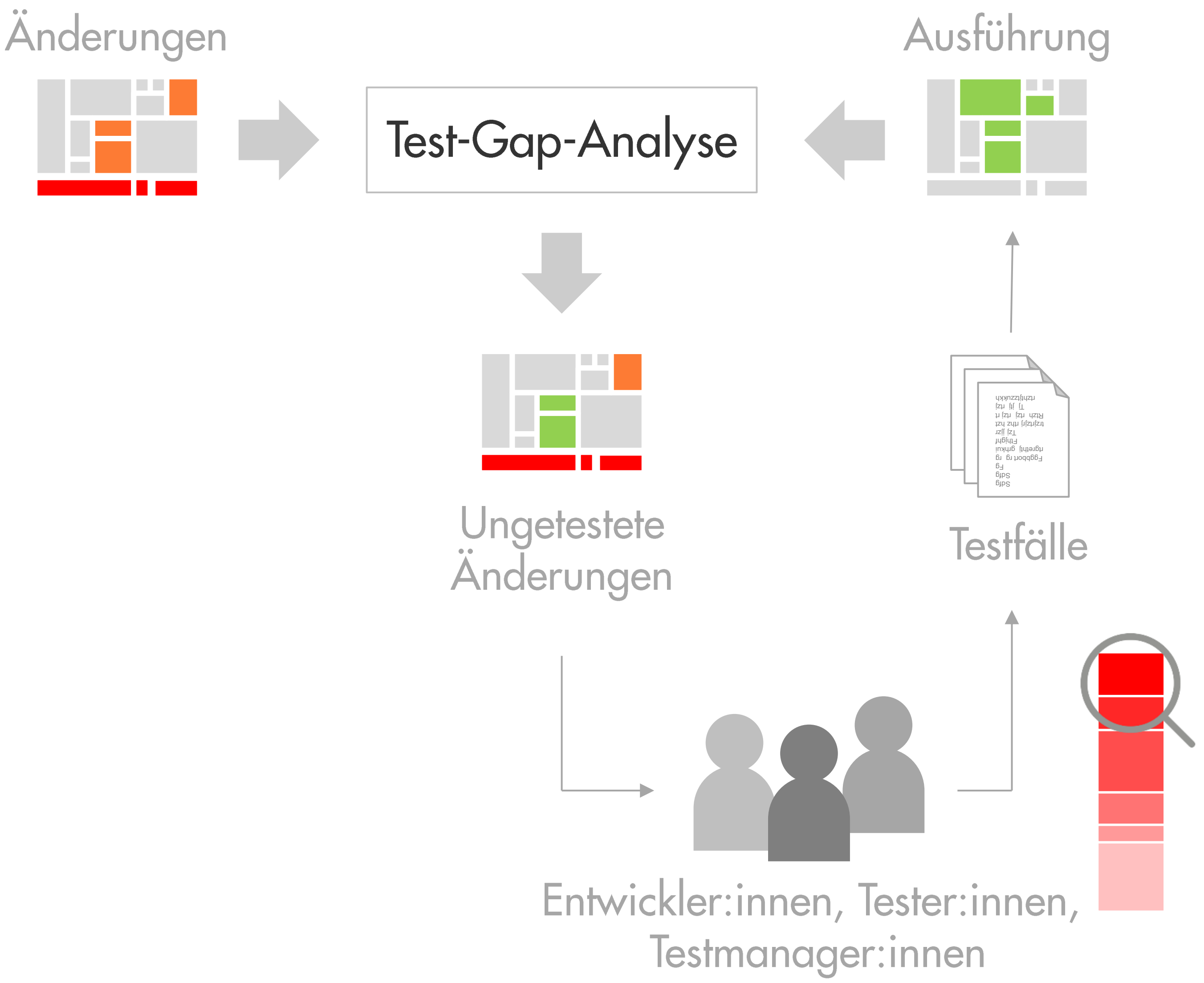
Einsatz in der Praxis



Risikobasierte Priorisierung von Test-Gaps



Quality Engineer


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

Kontakt – Wir freuen uns auf Diskussionen ☺

Link zu
OOP-Feedback

Link zu Folien

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

Roman Haas haas@cqse.eu +49 159 0458 3831

CQSE