Better Feedback Times Using Test Case Prioritization?

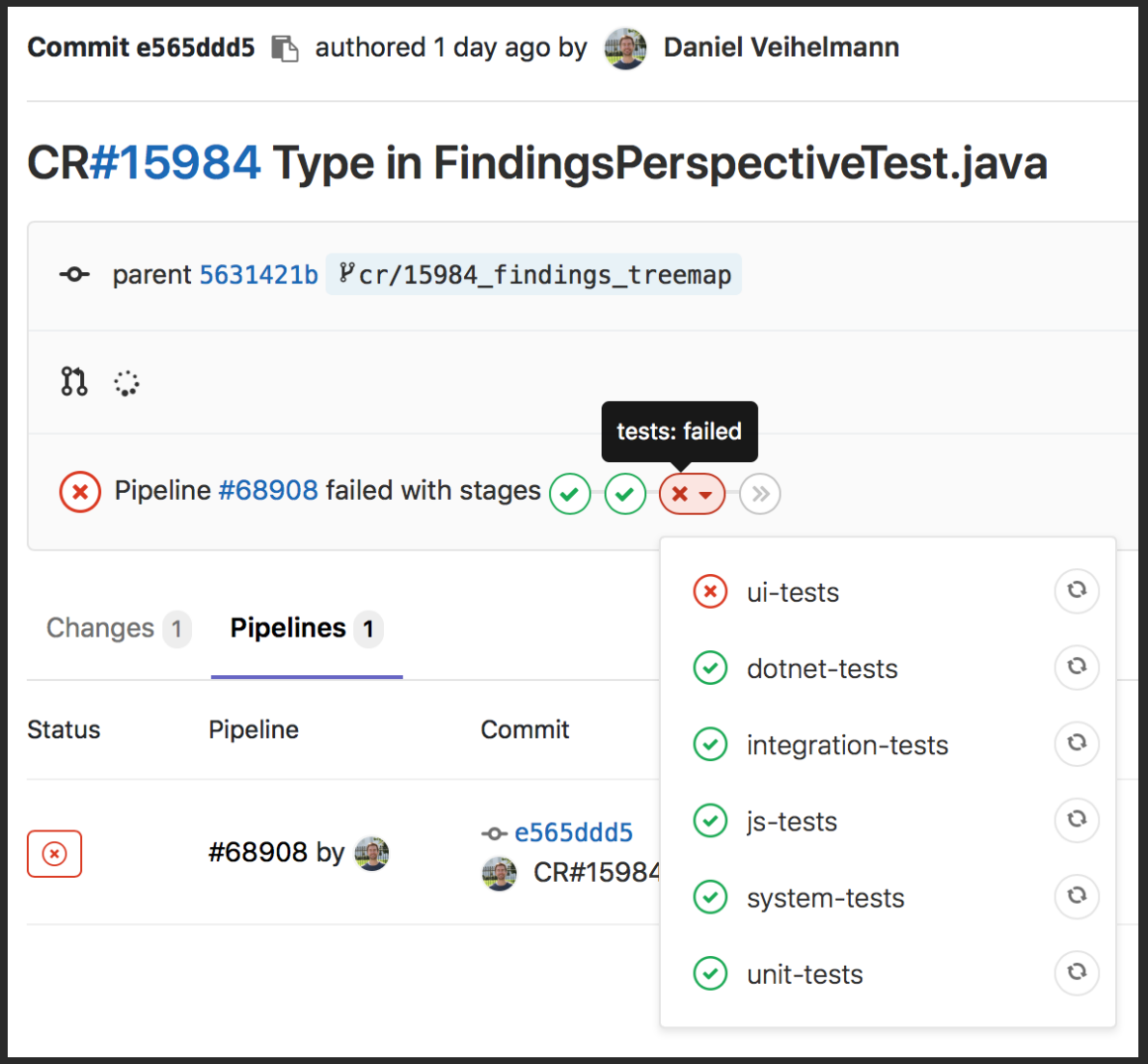
Mining Data of Past Build Failures in an Automated Benchmark

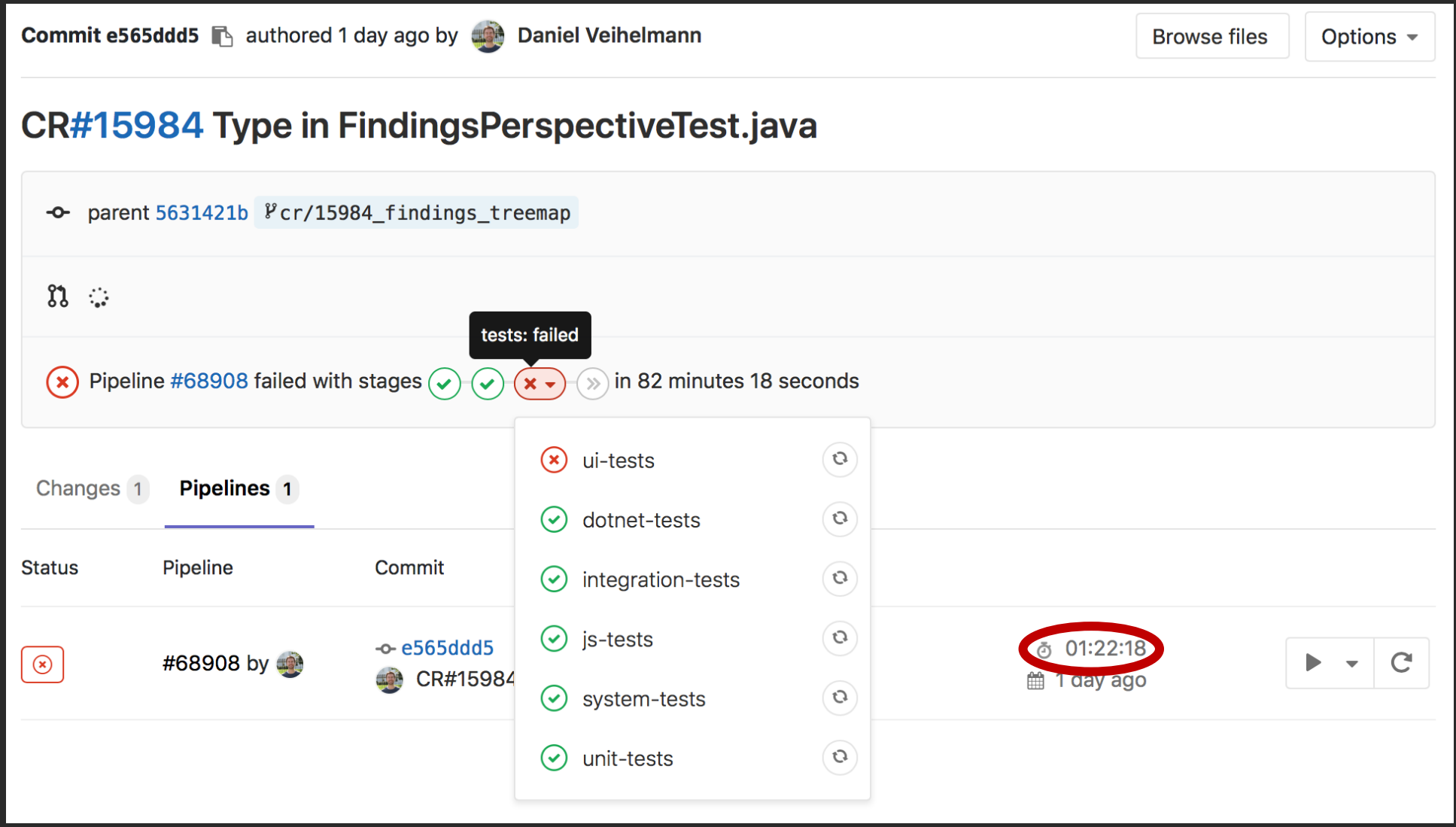
Jakob Rott
Rainer Niedermayr
Elmar Jürgens

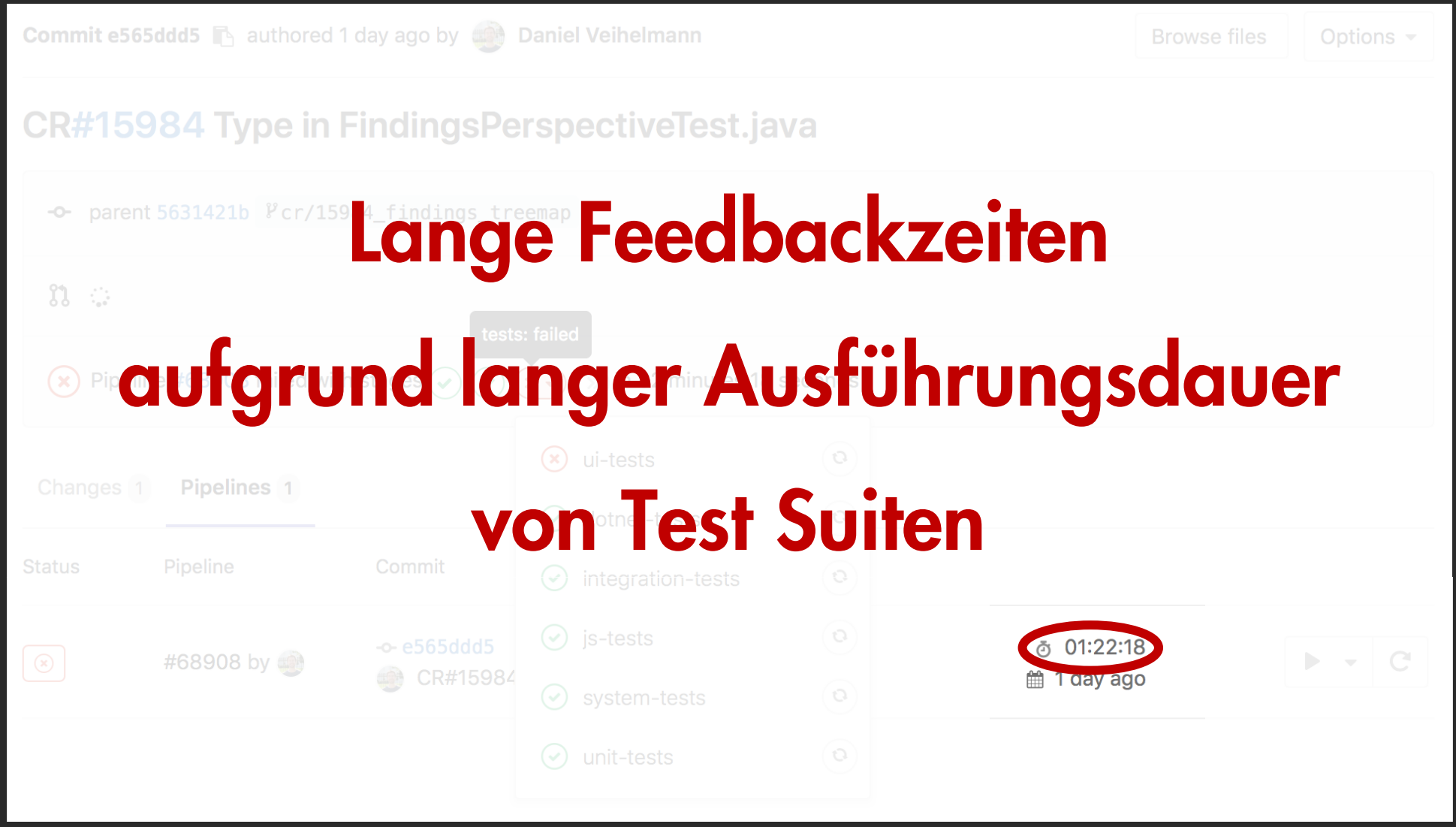
Kürzere Feedbackzyklen durch Testfallpriorisierung?

Ein Benchmark auf Grundlage vergangener Builds auf Travis CI

Jakob Rott
Rainer Niedermayr
Elmar Jürgens


Commit e565ddd5 authored 1 day ago by Daniel Veihelmann
CR#15984 Type in FindingsPerspectiveTest.java
parent 5631421b cr/15984_findings_treemap
tests: failed
Pipeline #68908 failed with stages
ui-tests
dotnet-tests
integration-tests
js-tests
system-tests
unit-tests
Changes 1
Pipelines 1
Status
Pipeline
Commit
#68908 by
e565ddd5
CR#15984


Commit e565ddd5 authored 1 day ago by Daniel Veihelmann
Browse files
Options
CR#15984 Type in FindingsPerspectiveTest.java
parent 5631421b cr/15984_findings_treemap
tests: failed
Pipeline #68908 failed with stages in 82 minutes 18 seconds
ui-tests
dotnet-tests
integration-tests
js-tests
system-tests
unit-tests
Changes 1
Pipelines 1
Status
Pipeline
Commit
#68908 by
e565ddd5
CR#15984
01:22:18
1 day ago



Lange Feedbackzeiten aufgrund langer Ausführungsdauer von Test Suiten


Lange Feedbackzeiten
aufgrund langer Ausführungsdauer
von Test Suiten

test prioritization technique

About 111,000 results (**0.08** sec)

A history-based **test prioritization technique** for regression testing in resource constrained environments

JM Kim, A Porter - … , 2002. ICSE 2002. Proceedings of the 24rd …, 2002 - ieeexplore.ieee.org

Regression testing is an expensive and frequently executed maintenance process used to revalidate modified software. To improve it, regression **test** selection (RTS) techniques strive to lower costs without overly reducing effectiveness by carefully selecting a subset of the **test** …

☆ 99 Cited by 381 Related articles All 26 versions

Selecting a cost-effective **test** case **prioritization technique**

S Elbaum, G Rothermel, S Kanduri… - Software Quality …, 2004 - Springer

Regression testing is an expensive testing process used to validate modified software and detect whether new faults have been introduced into previously tested code. To reduce the cost of regression testing, software testers may prioritize their **test** cases so that those which …

☆ 99 Cited by 233 Related articles All 16 versions

A regression **test** selection and **prioritization technique**

R Malhotra, A Kaur, Y Singh - Journal of Information Processing …, 2010 - koreascience.or.kr

Regression testing is a very costly process performed primarily as a software maintenance activity. It is the process of retesting the modified parts of the software and ensuring that no new errors have been introduced into previously tested source code due to these …

☆ 99 Cited by 34 Related articles All 4 versions »

A history-based cost-cognizant **test** case **prioritization technique** in regression testing

YC Huang, KL Peng, CY Huang - Journal of Systems and Software, 2012 - Elsevier

Software testing is typically used to verify whether the developed software product meets its requirements. From the result of software testing, developers can make an assessment about the quality or the acceptability of developed software. It is noted that during testing, the …

☆ 99 Cited by 64 Related articles All 5 versions

Test case **prioritization**: An empirical study

G Rothermel, RH Untch, C Chu… - … , 1999.(ICSM'99) …, 1999 - ieeexplore.ieee.org

… 3.5. Results An initial indication of how each **prioritization technique** affected a **test** suite's rate of detection can be determined from Figure 4, which presents boxplots of the APFD val- ues of the 9 categories of prioritized **test** suites for each program and an all-program total …

☆ 99 Cited by 505 Related articles All 14 versions

Search algorithms for regression **test** case **prioritization**

Z Li, M Harman, RM Hierons - IEEE Transactions on software …, 2007 - ieeexplore.ieee.org

… prioritized **test** cases according to the criterion of "increasing cost per additional coverage." In [20], Srivastava and Thiagarajan studied a **prioritization technique** that was based on the changes that have been made to the program and focused on the objective function of …

☆ 99 Cited by 601 Related articles All 29 versions

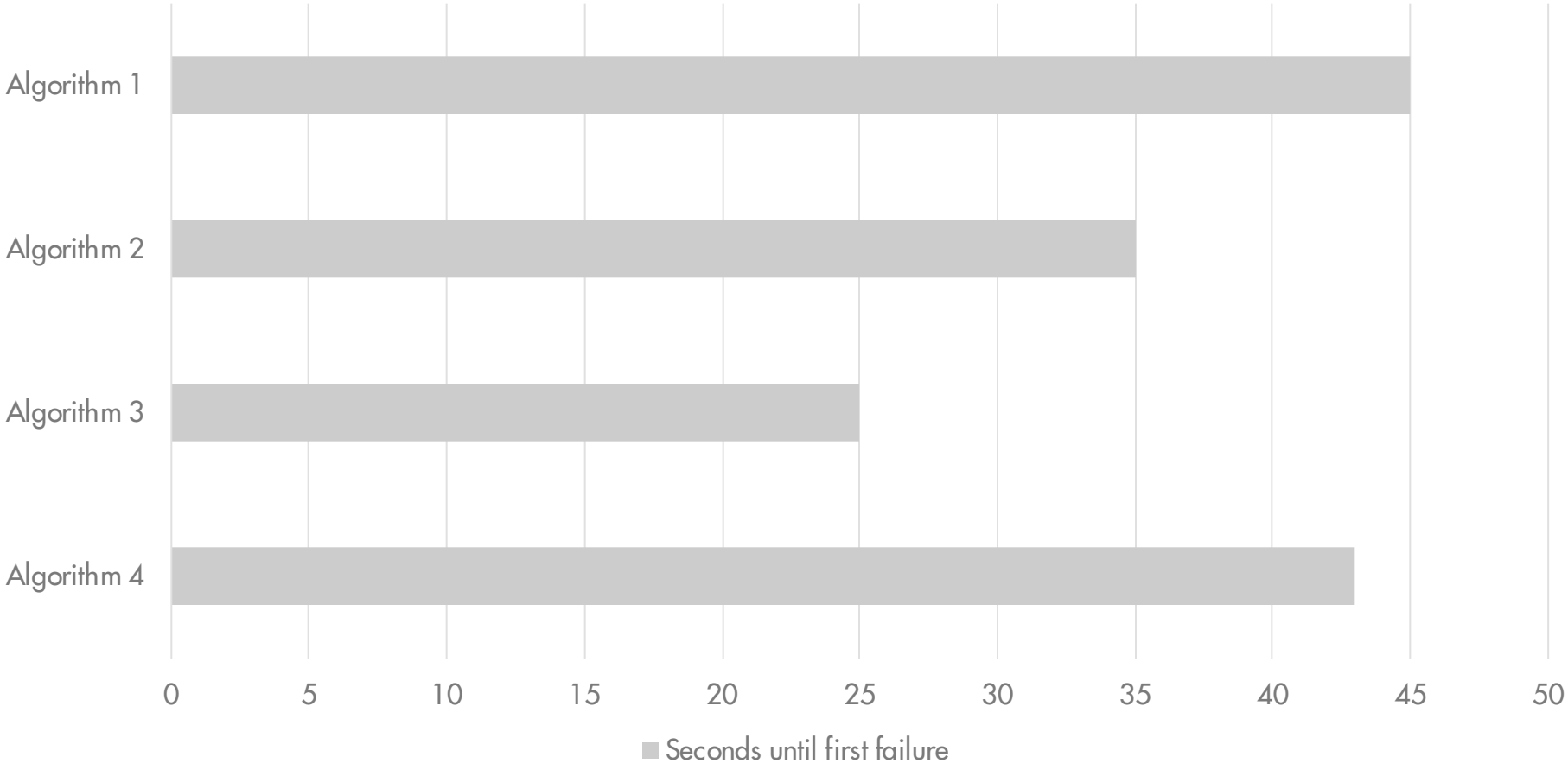
Timeaware **test** suite **prioritization**

KR Walcott, ML Soffa, GM Kapfhammer… - Proceedings of the 2006 …, 2006 - dl.acm.org

… mitted to a version control repository. This paper presents a regression **test prioritization technique** that uses a genetic algorithm to reorder **test** suites in light of testing time con- straints. Experiment results indicate that our prioritiza …

☆ 99 Cited by 329 Related articles All 26 versions

Performanz?

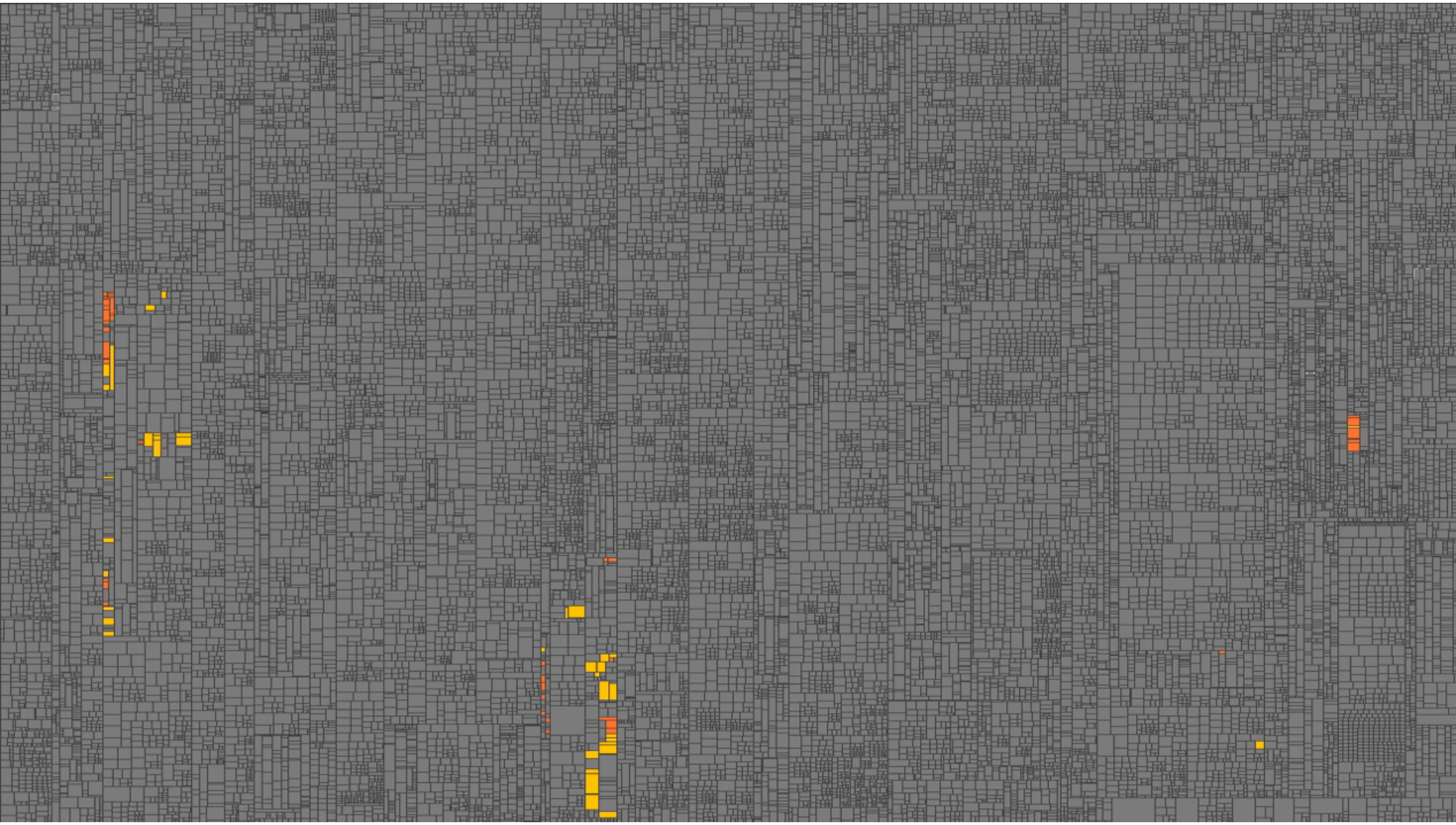

Algorithm 1
Algorithm 2
Algorithm 3
Algorithm 4
0
5
10
15
20
25
30
35
40
45
50
Seconds until first failure

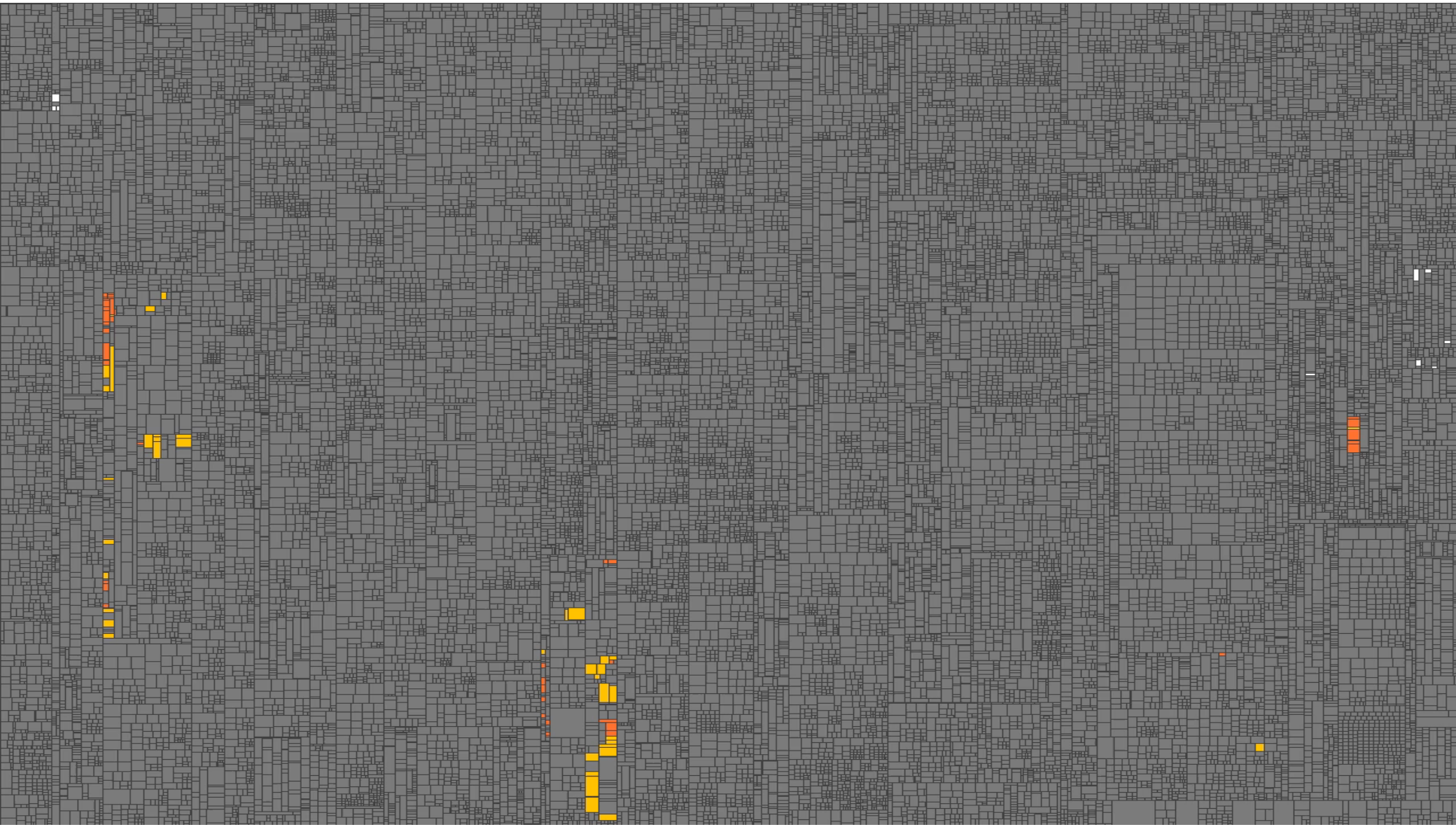
1 AUFBAU DES BENCHMARKS

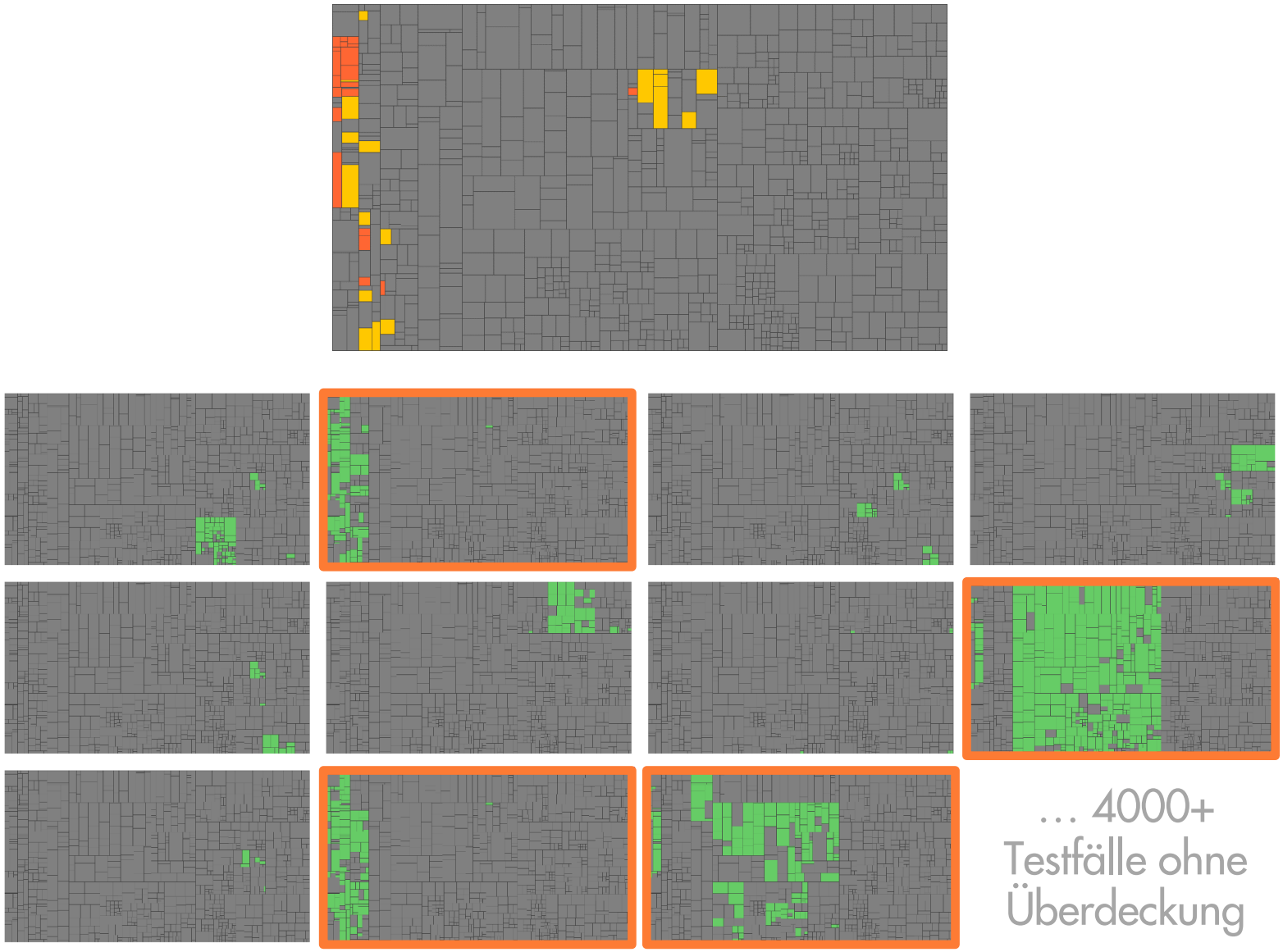
2 STUDIE

Algorithmus F3D

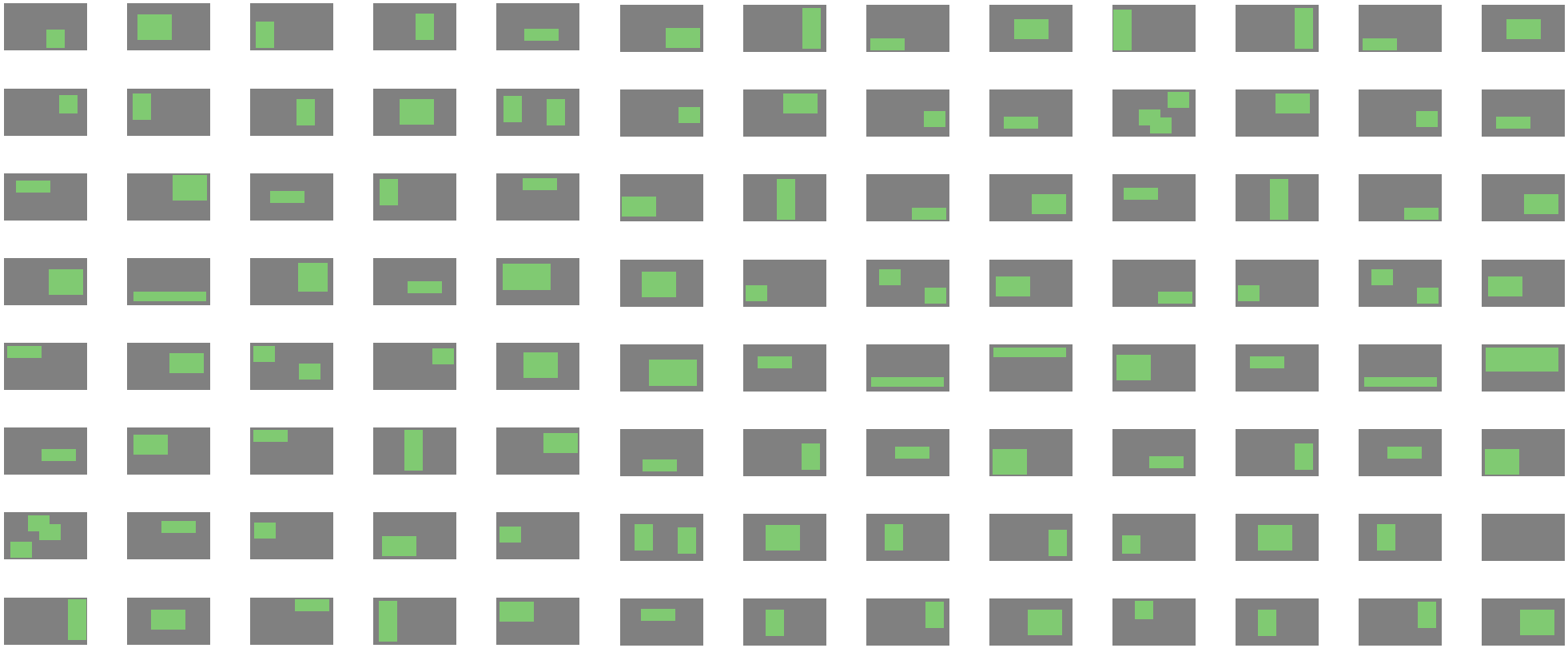
Exkurs



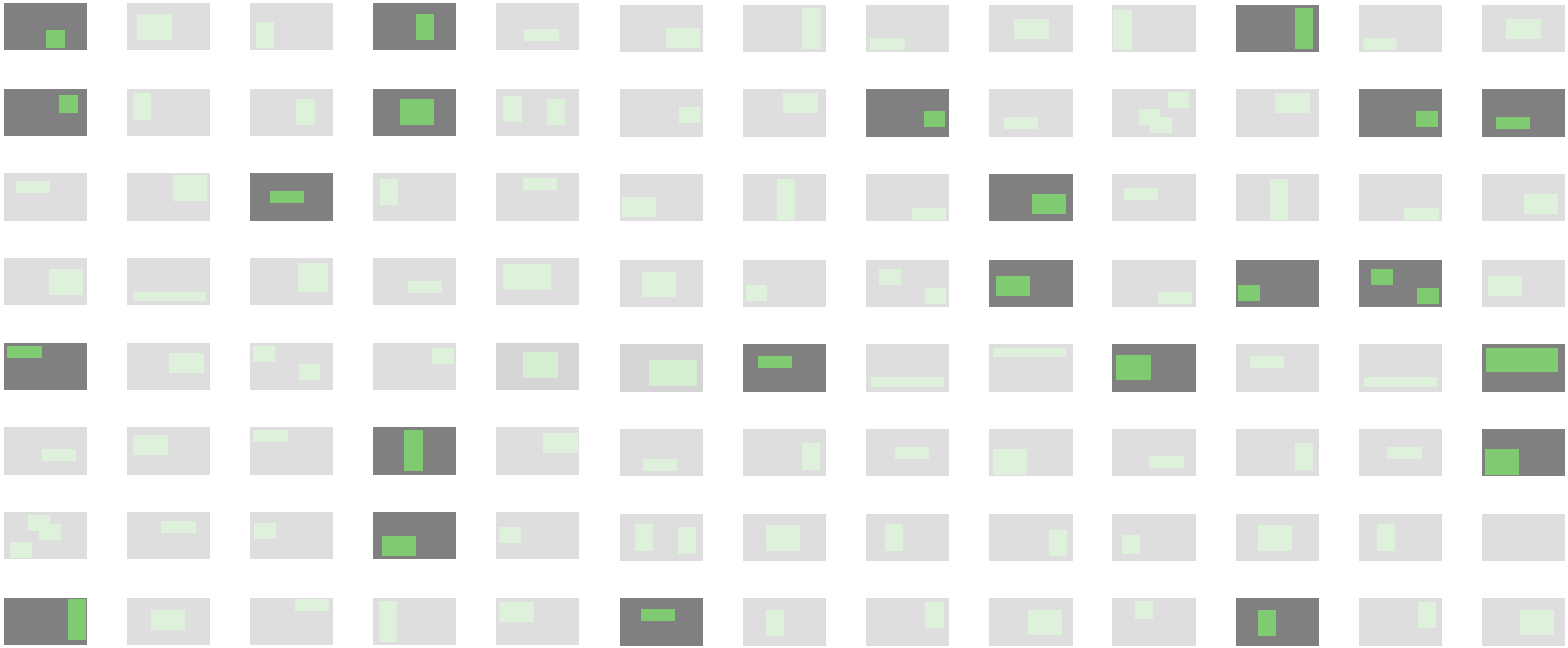




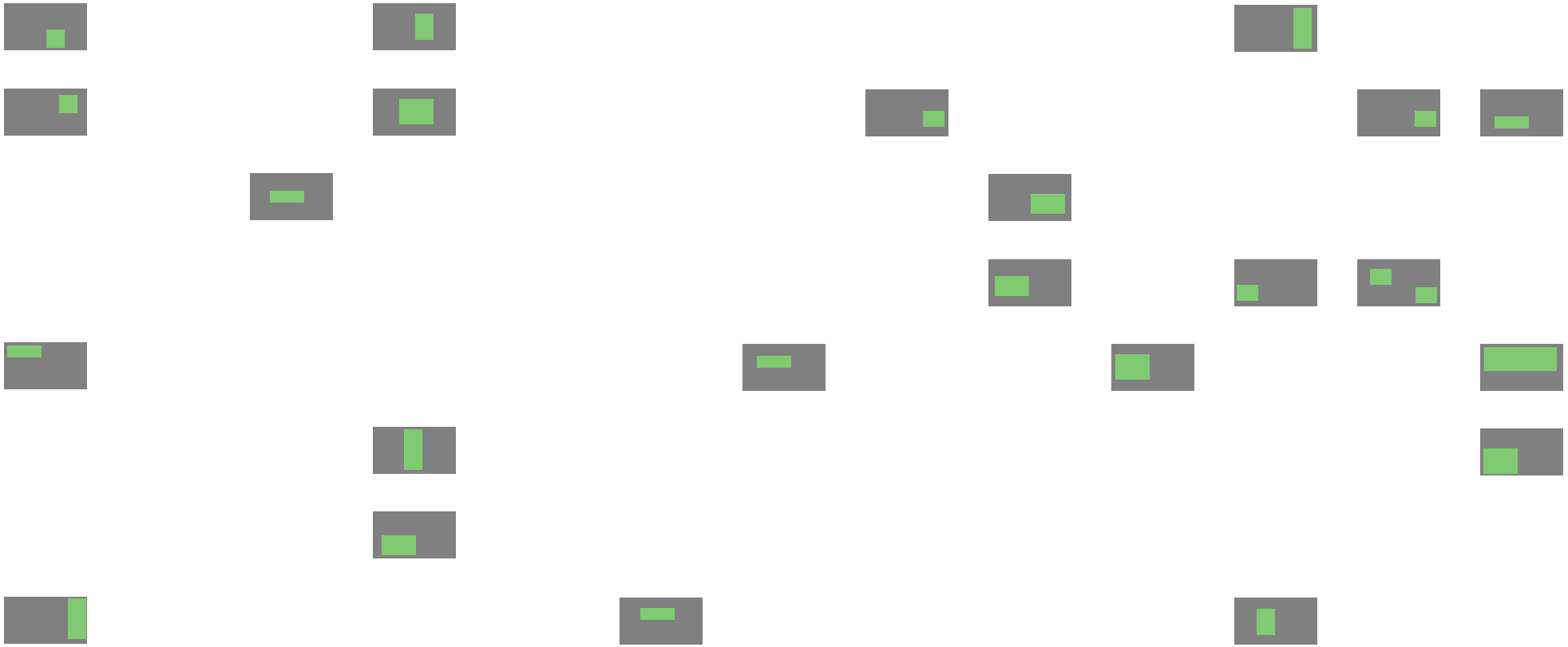
… 4000+
Testfälle ohne
Überdeckung



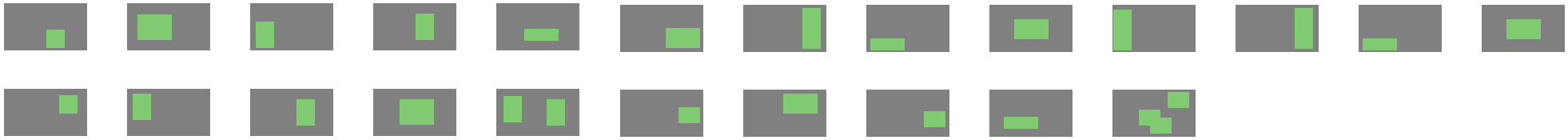
Schritt 1: Selektion betroffener Testfälle



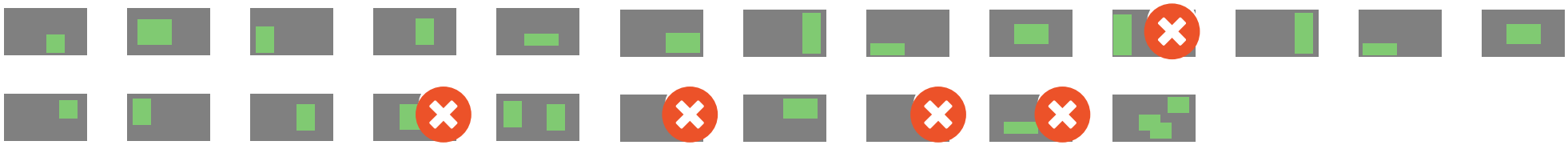
Schritt 1: Selektion betroffener Testfälle



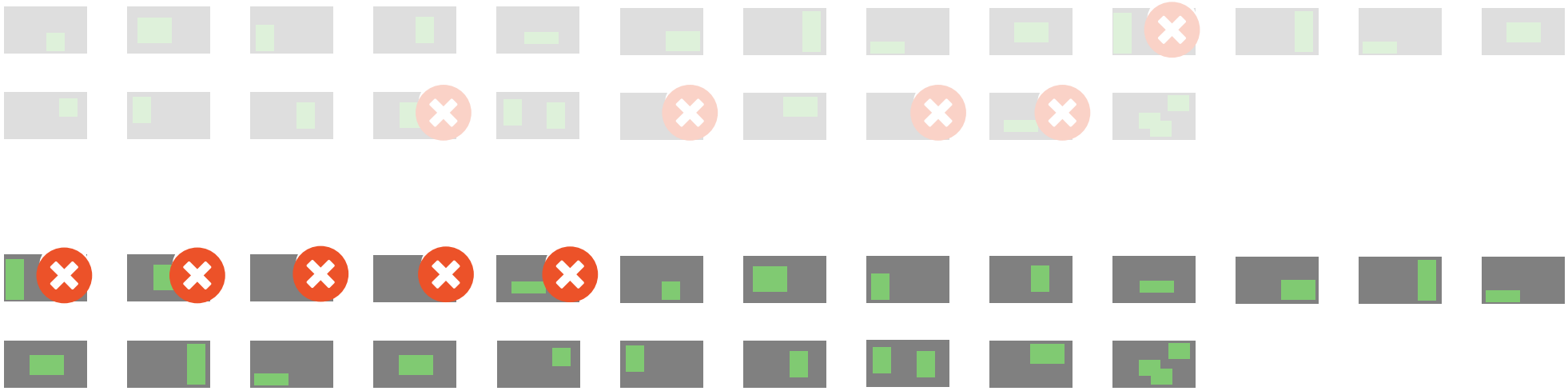
Schritt 1: Selektion betroffener Testfälle



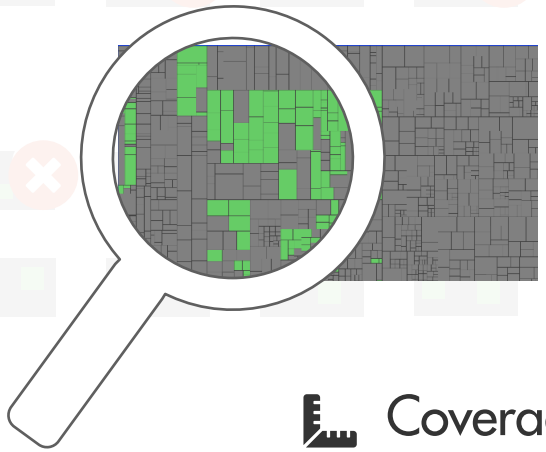
Schritt 2: Priorisierung selektierter Testfälle



Schritt 2: Priorisierung selektierter Testfälle



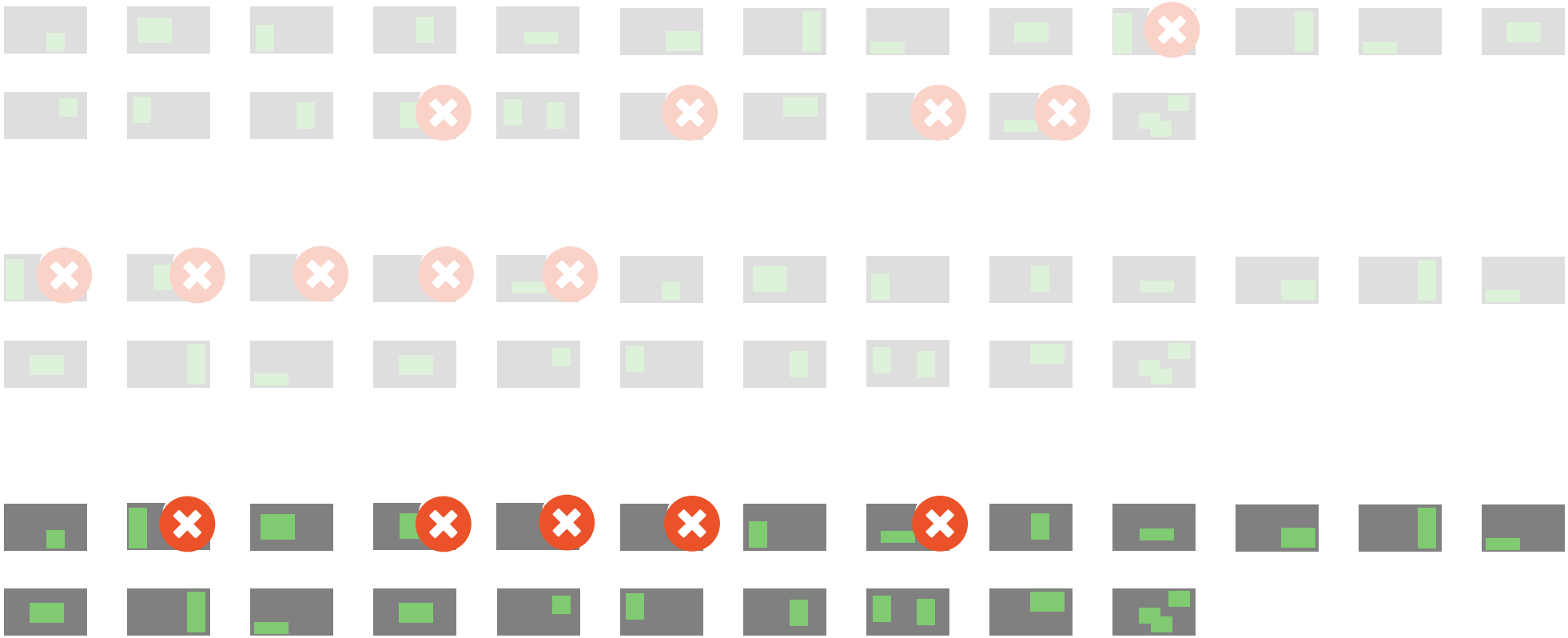
Schritt 2: Priorisierung selektierter Testfälle

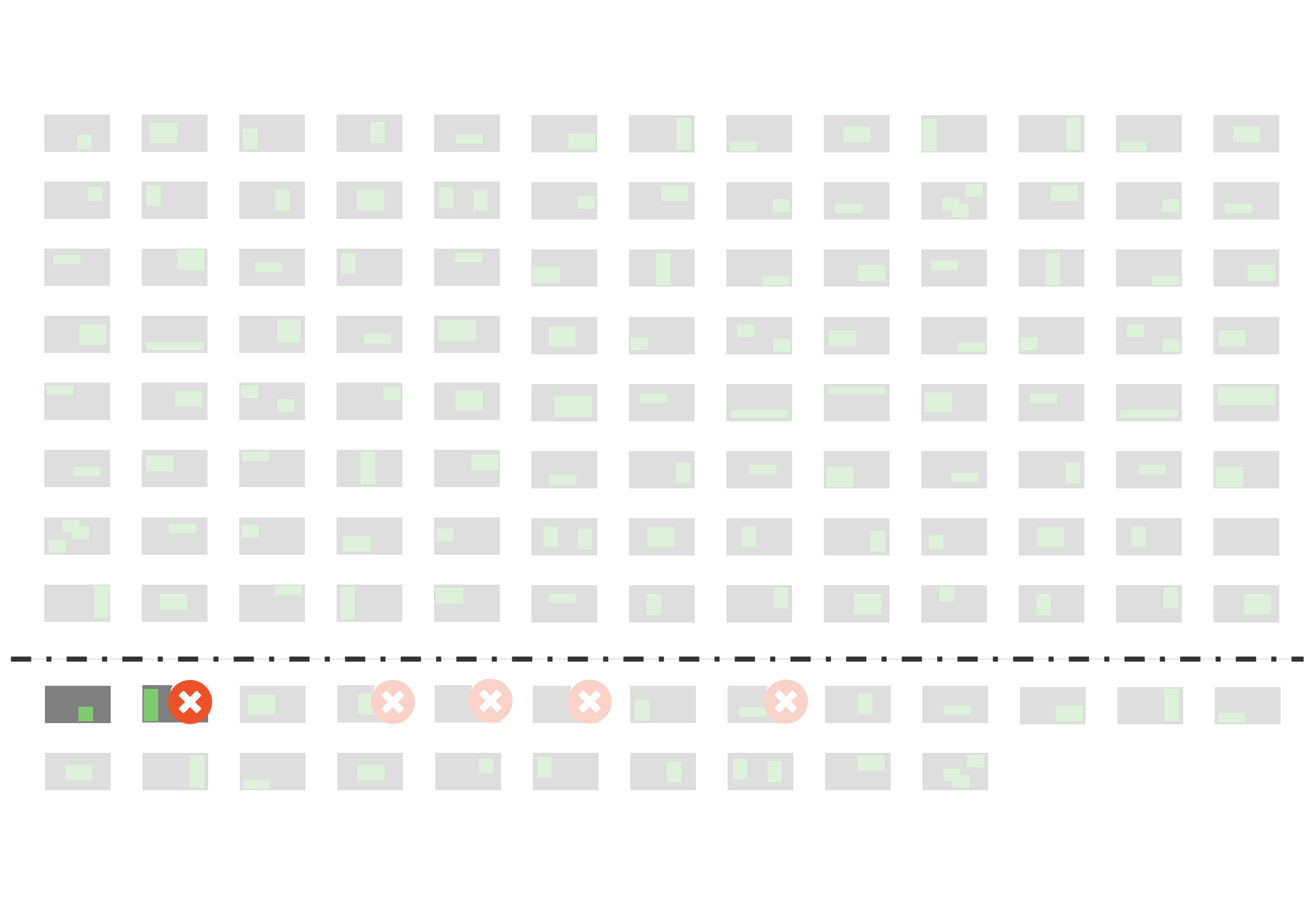


Coverage of changes

Execution time

Schritt 2: Priorisierung selektierter Testfälle

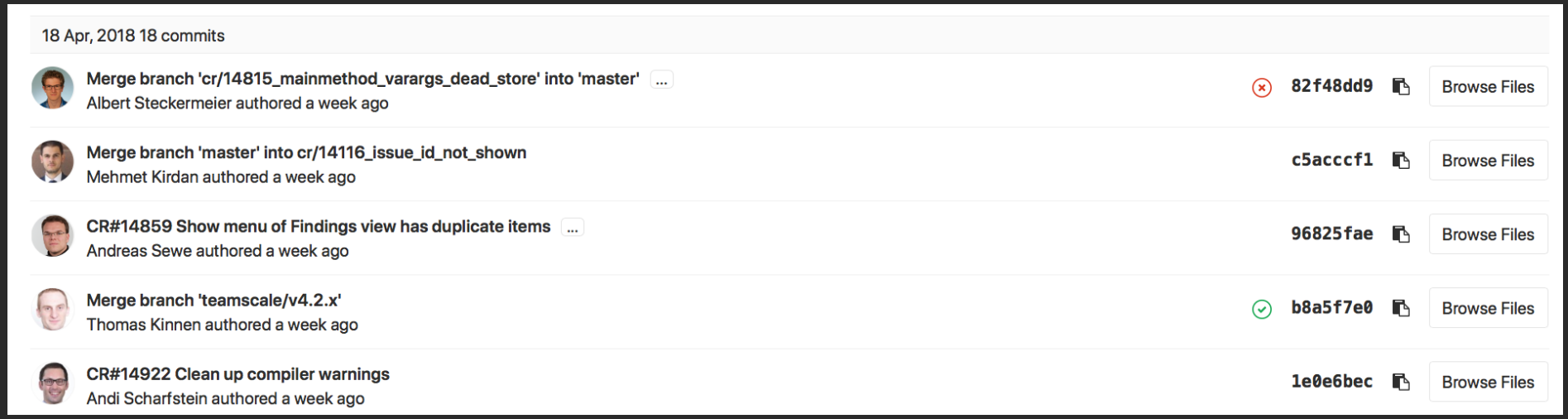




1 AUFBAU DES BENCHMARKS

2 STUDIE

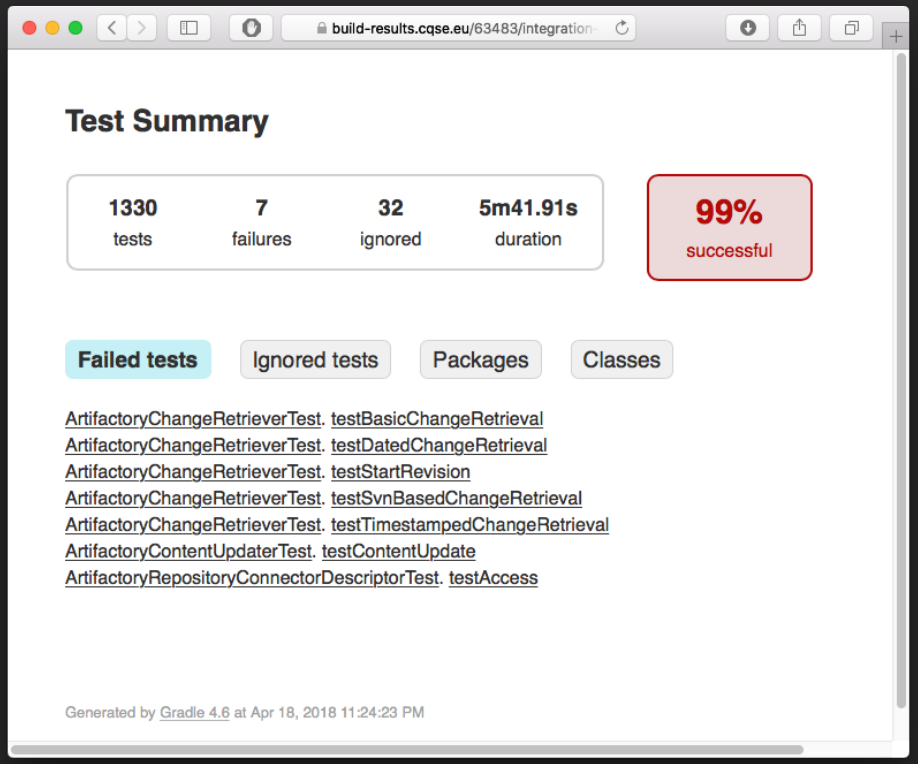
Veranschaulichung Benchmark



Veranschaulichung Benchmark

Fehlschlagende Builds

Fehlschlagende Tests

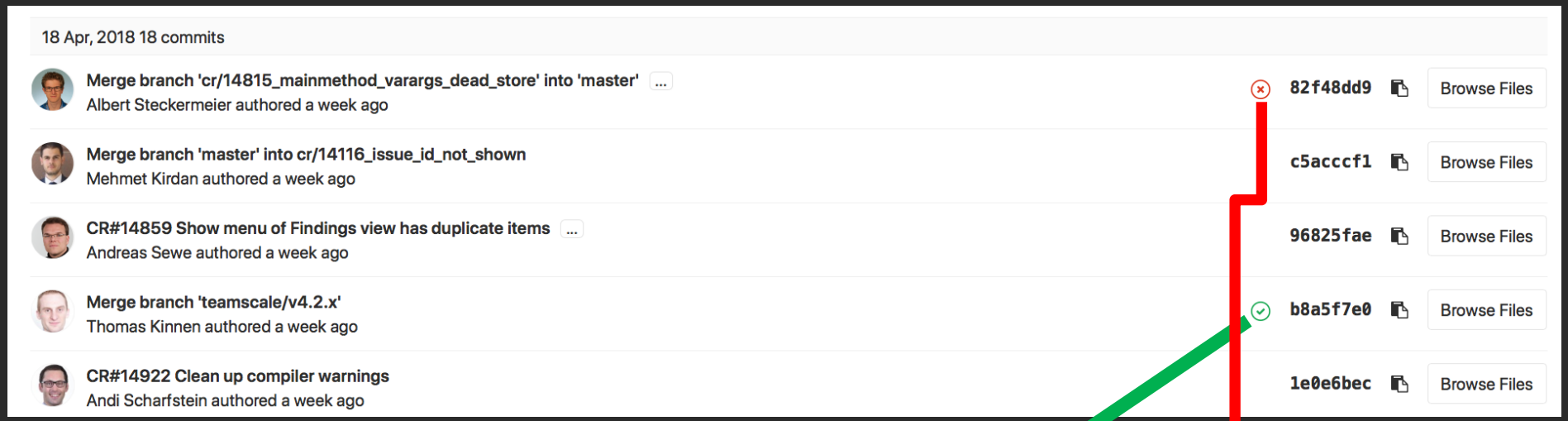


Veranschaulichung Benchmark

Fehlschlagende Builds

Fehlschlagende Tests

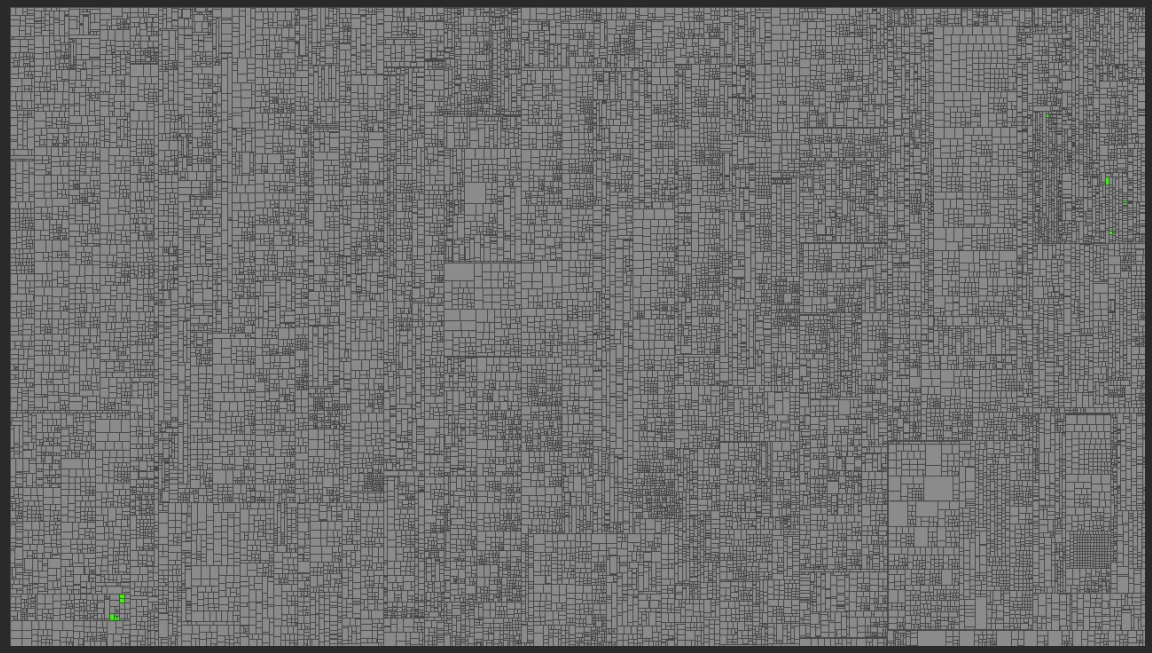
Berechnung Time to Failure



Vorbereitung Eingabedaten

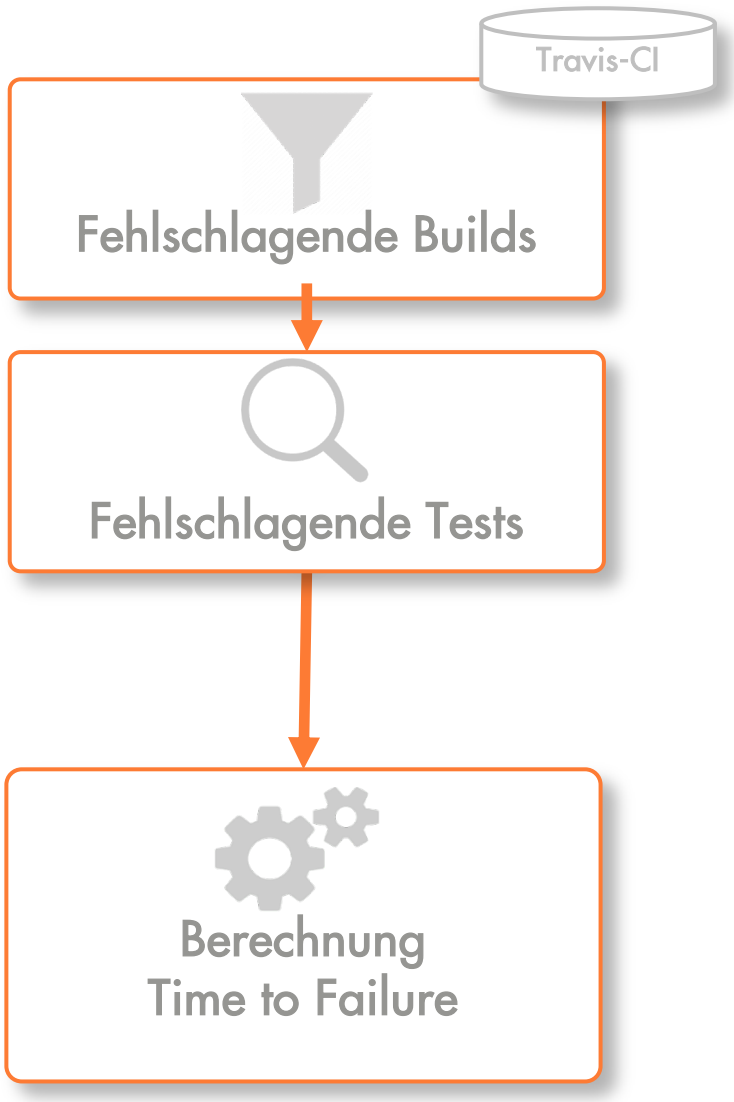
Codedownload (`b8a5f7e0` & `82f48dd9`)

Veranschaulichung Benchmark

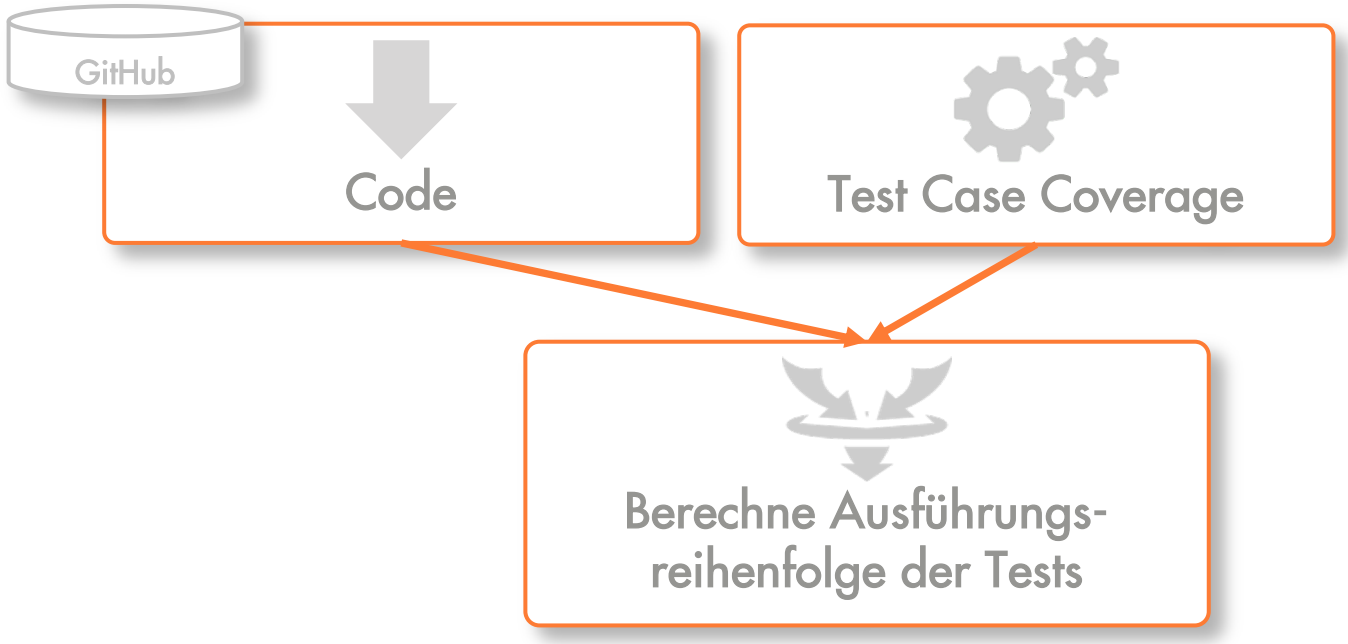


Vorbereitung Eingabedaten

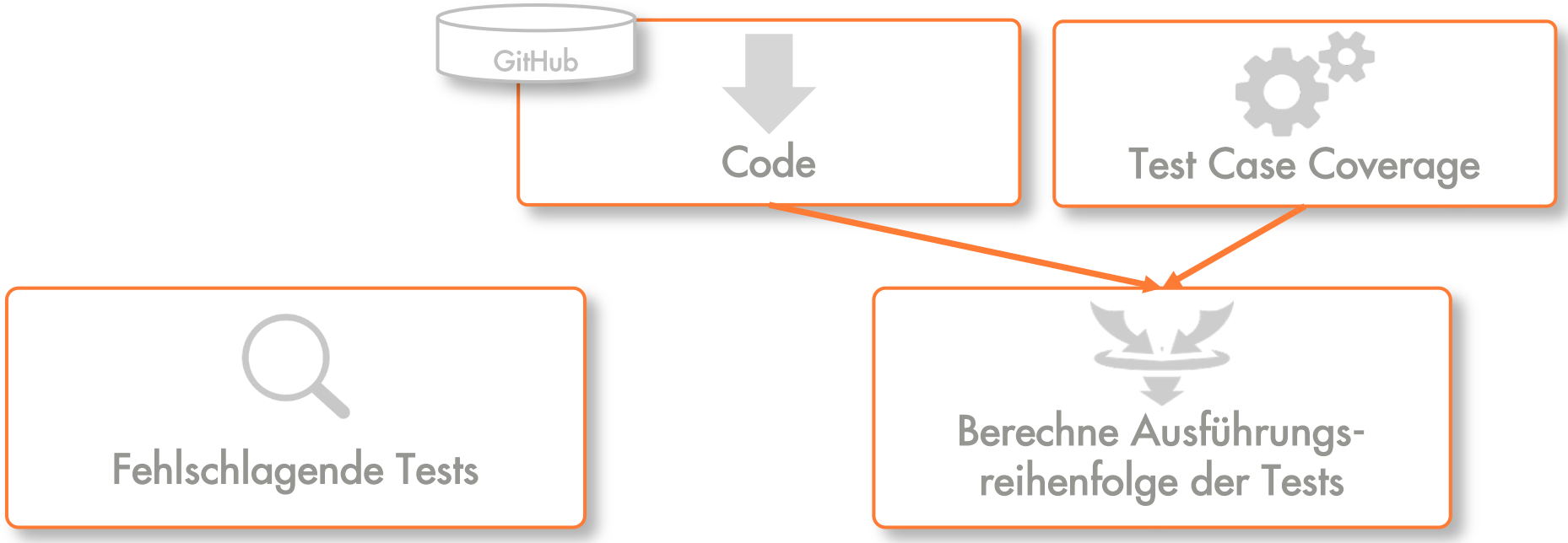
Testfallspezifische Coverage


Travis-CI
Fehlschlagende Builds
Fehlschlagende Tests
Berechnung
Time to Failure

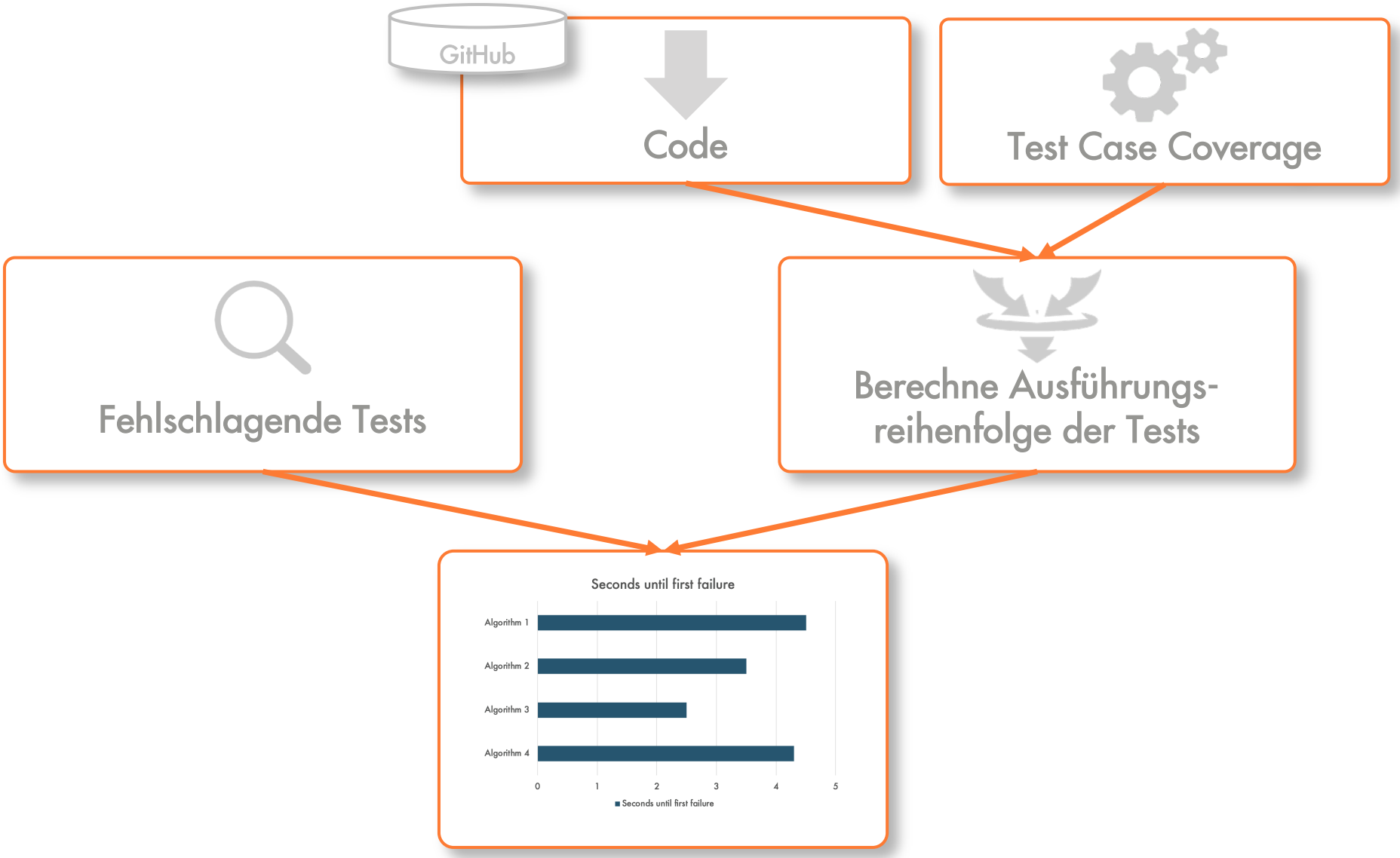
Berechnung Time to Failure



Berechnung Time to Failure



Berechnung Time to Failure



1 AUFBAU DES BENCHMARKS

2 STUDIE

Forschungsfragen

- RQ1
  Mit welchem Priorisierungsalgorithmus wird der erste Testfehlschlag am schnellsten aufgedeckt?
- RQ2
  Wie viel Zeit spart man sich durch die Anwendung von Algorithmus F3D im Vergleich zu einer randomisierten Testausführung?

Im Benchmark: 7 Priorisierungsalgorithmen

- F3D
  In Teamscales *Test Impact Analyse* eingesetzt (leicht abgewandelt)
- DUR
  Geordnet nach Dauer der Testausführung (aufsteigend)
- LPS
  Geordnet nach der Geschwindigkeit in Zeilen pro Sekunde (absteigend)
- TNC
  Ordnung nach Anzahl der pro Testfall aufgerufenen Methoden (absteigend)

7 Priorisierungsalgorithmen (fort.)

- TNC
  Ordnung nach Anzahl der pro Testfall aufgerufenen Methoden (absteigend)
- ANC
  Wie TNC, wobei bereits aufgerufene Methoden nicht zählen
- DIS
  Geordnet nach der String Distanz der Testfälle
- RDM
  Zufällige Ausführungsreihenfolge

Studienobjekte

- 437 Builds von insgesamt 31 Projekten

Project	kLOC	Src kLOC	Test kLOC	#builds ↓	#tests
graphhopper	72.7	48.0	24.7	102	1214
jmeter-plugins	85.3	47.0	38.4	75	1568
biojava	276.2	235.2	40.9	56	976
dropwizard	58.6	28.0	30.7	38	1164
graylog2-server	140.2	105.7	34.4	36	1208
jackson-databind	159.2	100.0	59.1	20	1586
logback	98.8	54.0	44.8	19	885
⋮	⋮	⋮	⋮	⋮	⋮
∅	**294**	**249**	**45**	**14**	**1098**

Forschungsfragen

- RQ1
  Mit welchem Priorisierungsalgorithmus wird der erste Testfehlschlag am schnellsten aufgedeckt?
- RQ2
  Wie viel Zeit spart man sich durch die Anwendung von Algorithmus F3D im Vergleich zu einer randomisierten Testausführung?

Ergebnis – RQ 1

Am schnellsten zum ersten Testfehlschlag

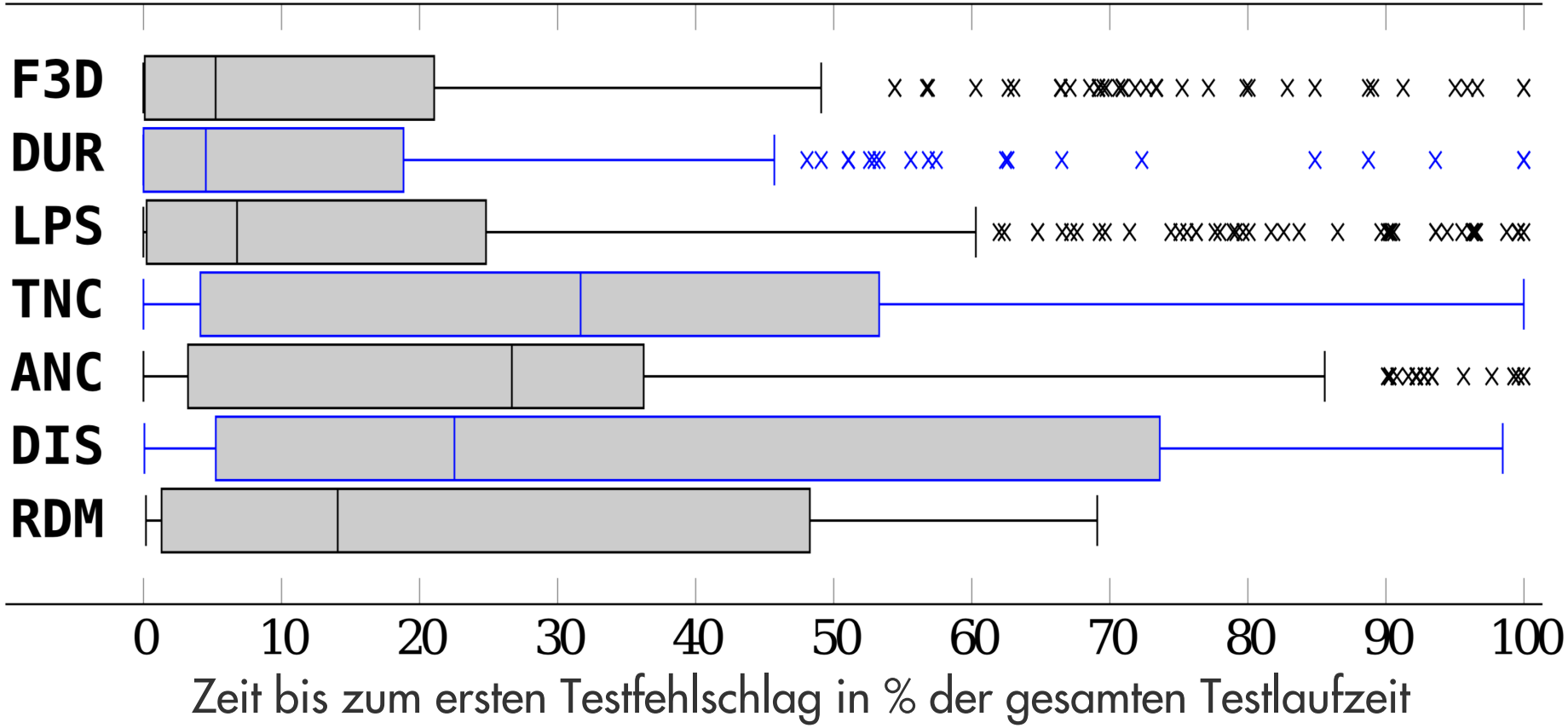
- Paarweiser Vergleich

  Lies: Algorithmus in Zeile ist schneller als Algorithmus in Spalte

	F3D	DUR	LPS	TNC	ANC	DIS	RDM	
F3D			✓	✓	✓	✓	✓	5 wins
DUR			✓	✓	✓	✓	✓	5 wins
LPS				✓	✓	✓		3 wins
TNC								
ANC				✓		✓		2 wins
DIS								
RDM				✓	✓	✓		3 wins
	0 miss	0 miss	2 misses	5 misses	4 misses	5 misses	2 misses	

Ergebnis – RQ 1

Am schnellsten zum ersten Testfehlschlag



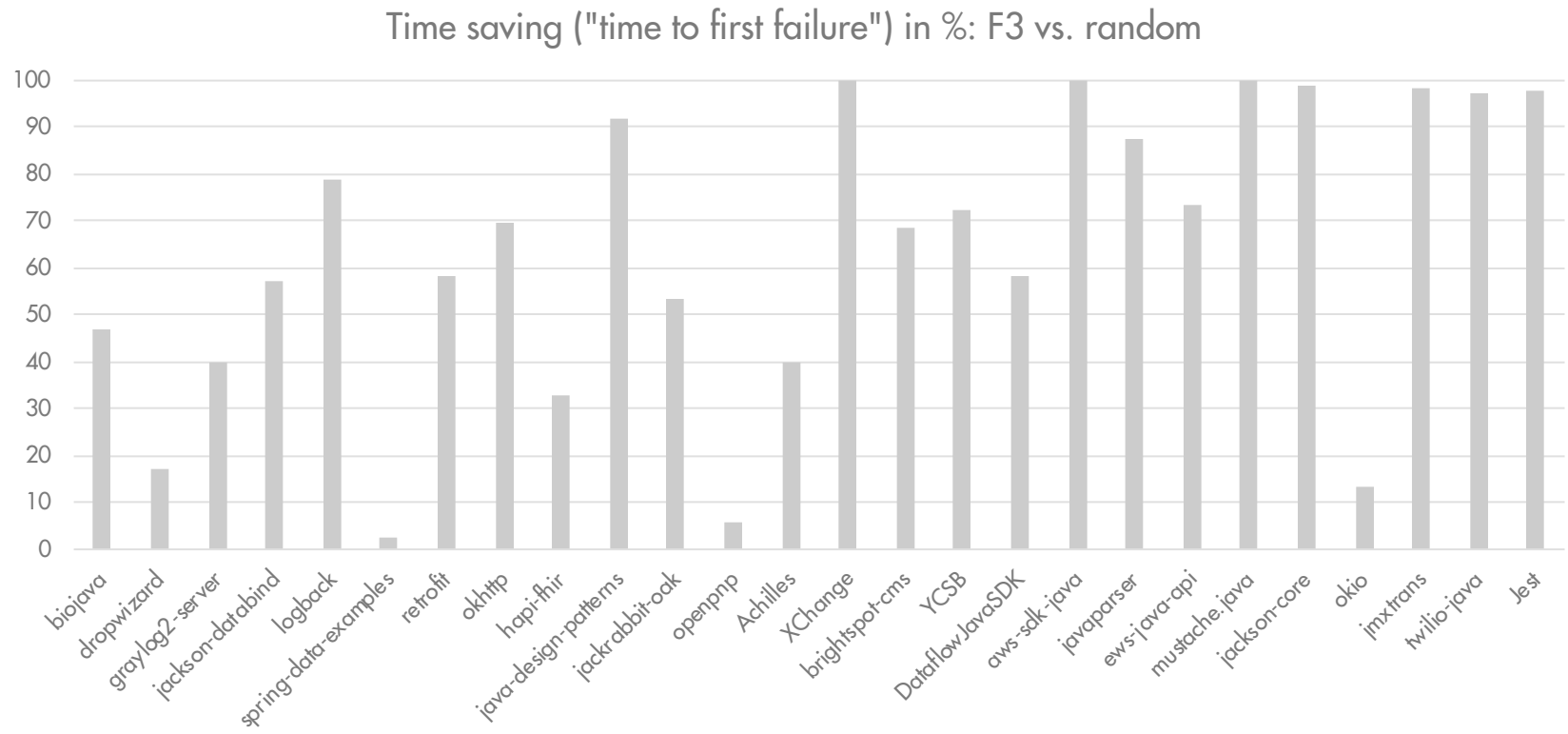
Forschungsfragen

- RQ1
  Mit welchem Priorisierungsalgorithmus wird der erste Testfehlschlag am schnellsten aufgedeckt?
- RQ2
  Wie viel Zeit spart man sich durch die Anwendung von Algorithmus F3D im Vergleich zu einer randomisierten Testausführung?

Ergebnis – RQ 2

F3D im Vergleich zur zufälligen Reihenfolge

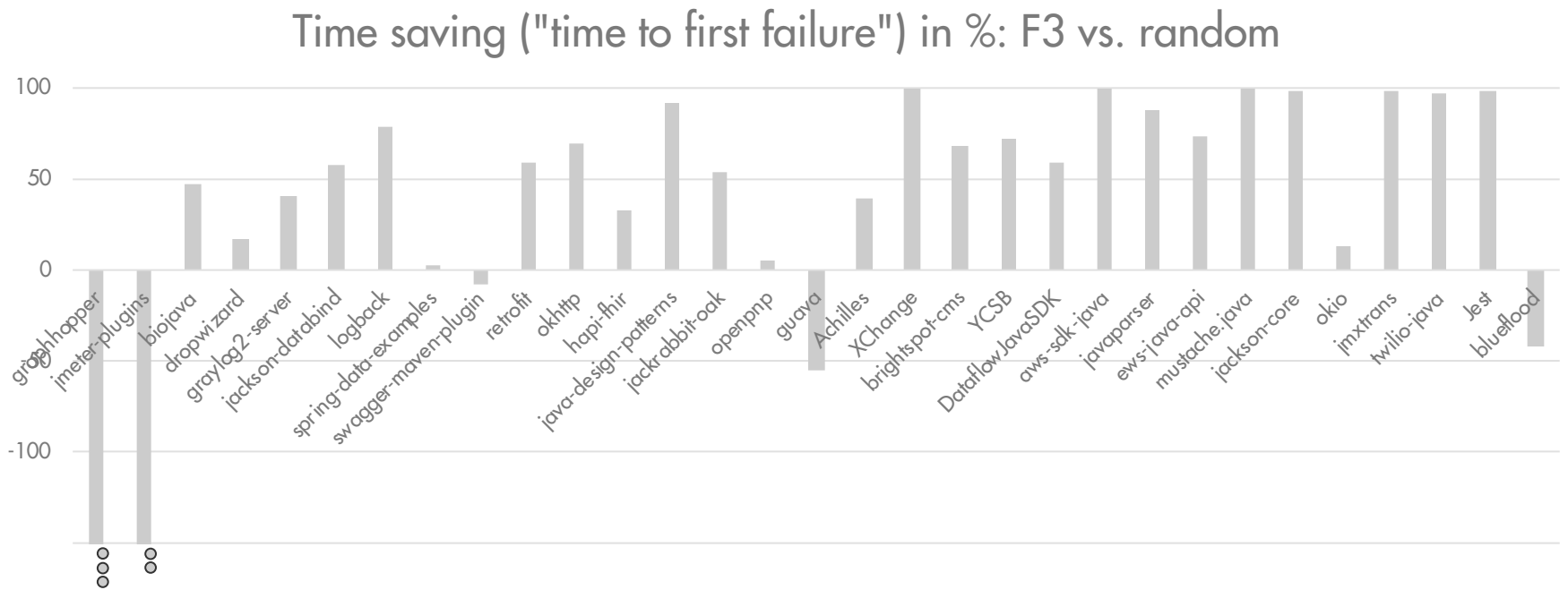
- Zeitersparnis in 26 von 31 Projekten
- Durchschnittliche Ersparnis dabei: 63,8%



Ergebnis – RQ 2

F3D im Vergleich zur zufälligen Reihenfolge

- Zeitersparnis in 26 von 31 Projekten
- Über alle 31 Projekte: Median der Zeitersparnis 58%
- Ausreißer: Testlaufzeit um das 7,5-fache erhöht
- In Ausreißerprojekt schlugen oft sehr viele Tests fehl (>100)



Einschränkung der Validität

- 75% der untersuchten Builds stammen von nur sechs unterschiedlichen Projekten
- Berechnung der *Time to Failure* erfolgte theoretisch, Tests wurden nicht in der vorgeschlagenen Reihenfolge ausgeführt

Zusammenfassung und Ausblick

- Mit F3D und DUR erreichte man den ersten fehlschlagenden Test am schnellsten
- F3D spart in vielen Projekten > 50% Testlaufzeit verglichen zur zufälligen Testreihenfolge

- Mehr (industrielle) Projekte in Benchmark
- Anwendung des Benchmarks erleichtern
- Fokus auf Builds in denen Testfehlschläge tatsächlich auf Codeänderungen rückführbar sind

Kürzere Feedbackzyklen durch Testfallpriorisierung?

Ein Benchmark auf Grundlage vergangener Builds auf Travis CI

Jakob Rott
Rainer Niedermayr
Elmar Jürgens