Erfahrungen aus 10 Jahren

Test-Gap-Analyse im Praxiseinsatz

CQSE

Dr. Sven Amann

Agenda

- Teil 1: Grundlagen der Test-Gap-Analyse
- Teil 2: Herausforderungen bei der Einführung
- Teil 3: Kosten-Nutzen-Berechnung

Teil 1

Grundlagen der Test-Gap-Analyse

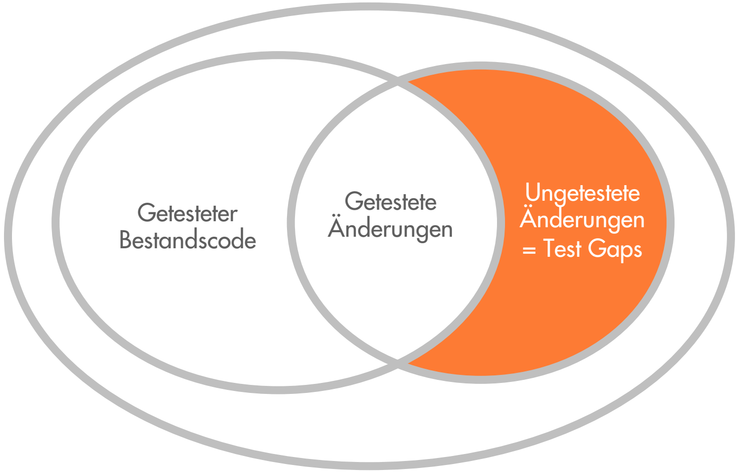
Stellen Sie sich vor, Sie sind dafür verantwortlich, dass alle Codeänderungen »*ausreichend*« getestet werden…

Wo treten Fehler in Produktion auf?

Studie: C# System

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

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



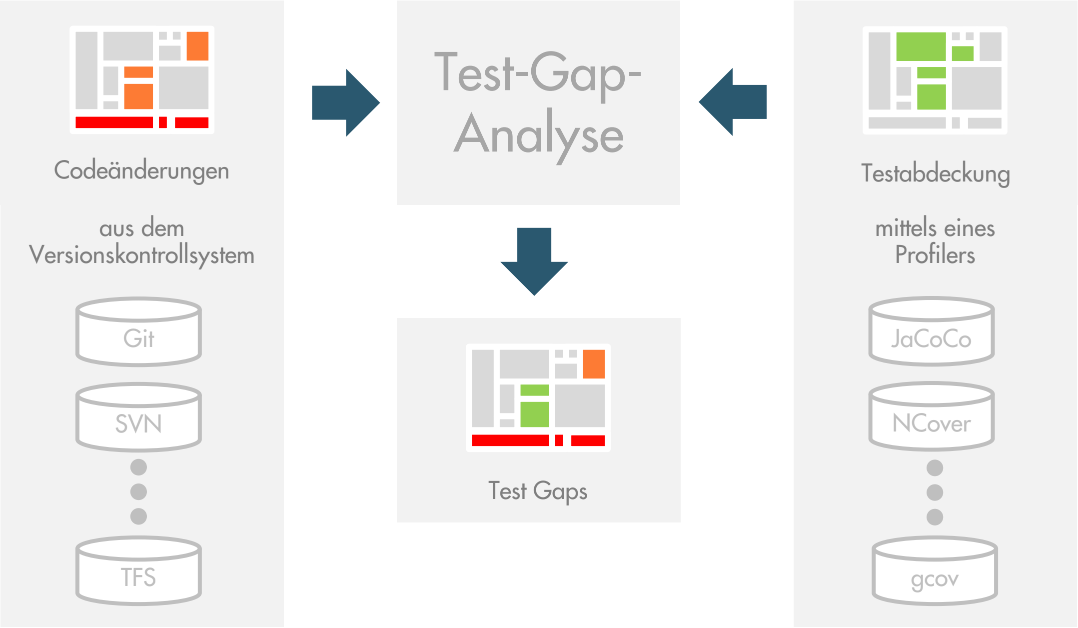
Feldfehlerwahrscheinlichkeit 5x höher für ungetestete Änderungen!

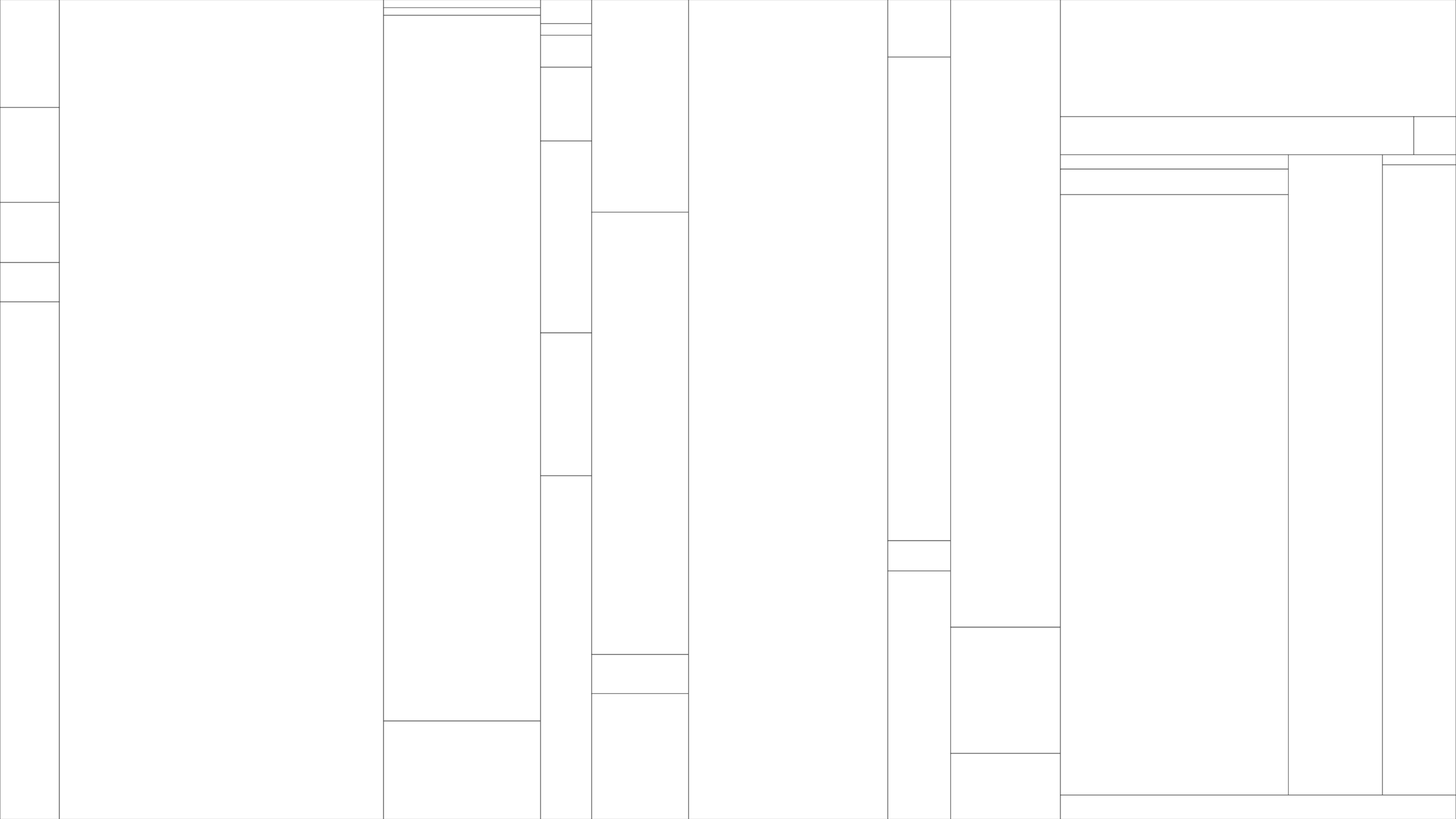
Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

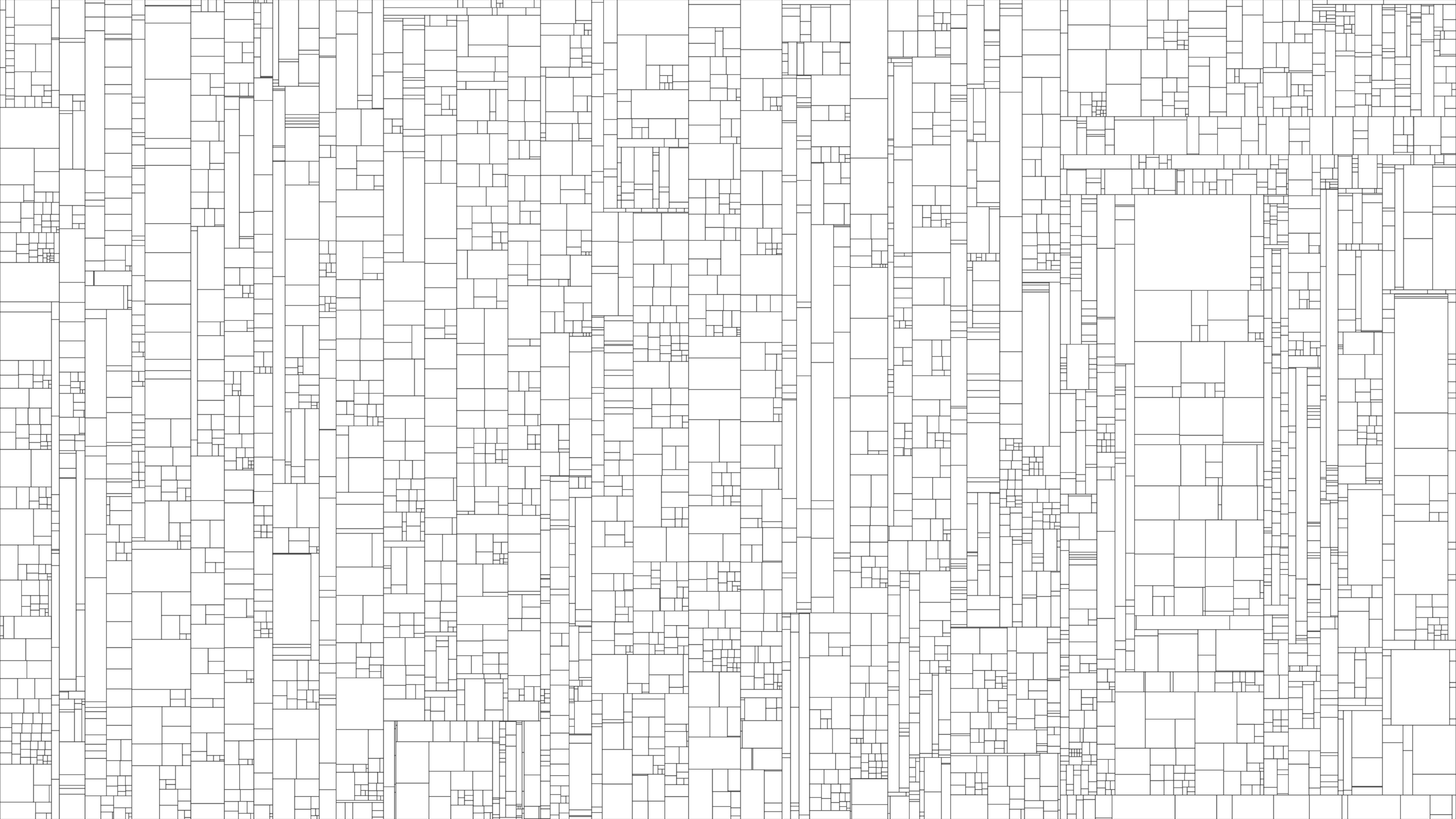
Ziel

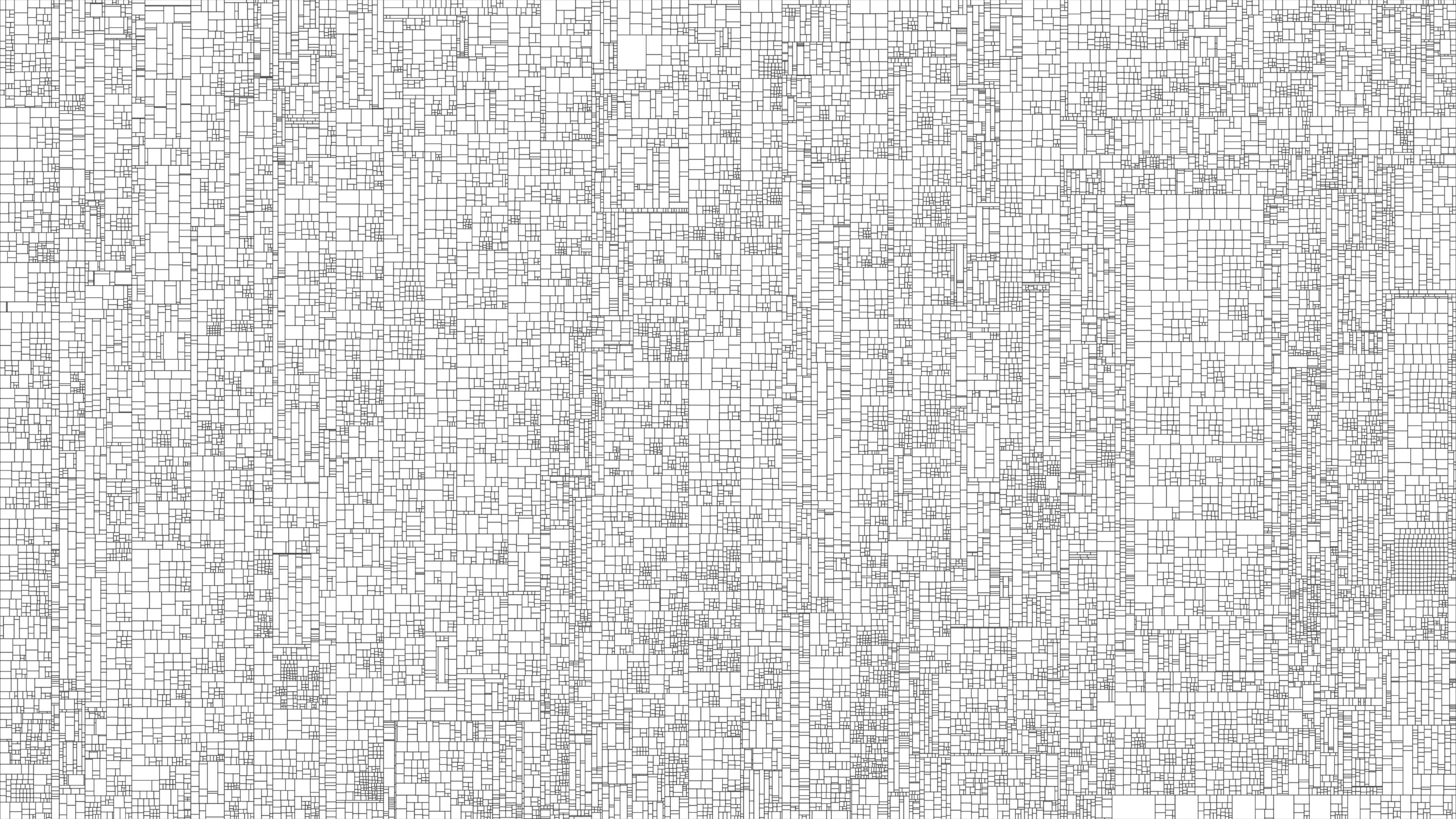
Finde die ungetesteten Änderungen (= Test Gaps)

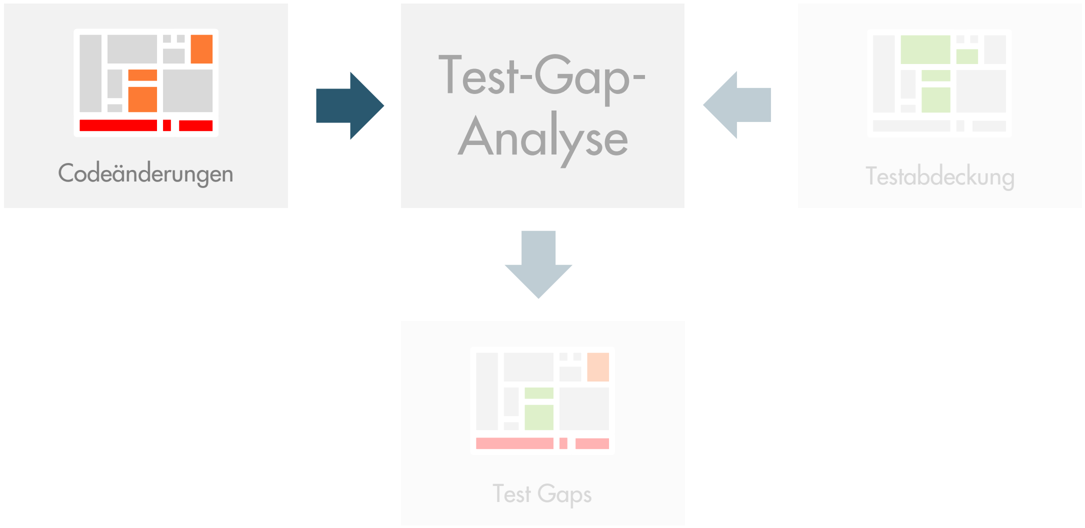
Weil Fehler im geänderten, ungetesteten Code sehr viel wahrscheinlicher sind als anderswo

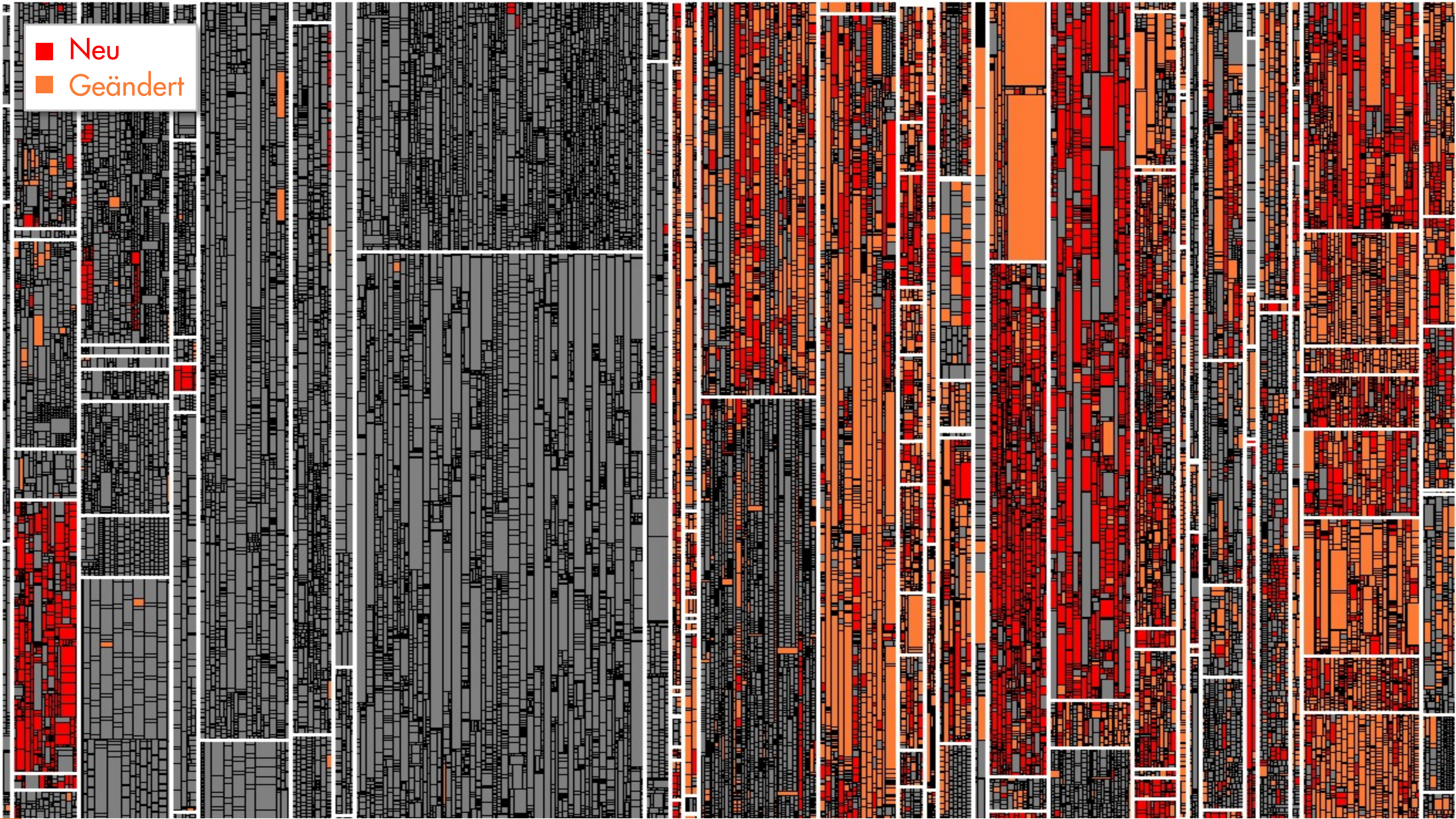

Codeänderungen
aus dem Versionskontrollsystem
Git
SVN
TFS
Test-Gap-Analyse
Test Gaps
Testabdeckung
mittels eines Profilers
JaCoCo
NCover
gcov

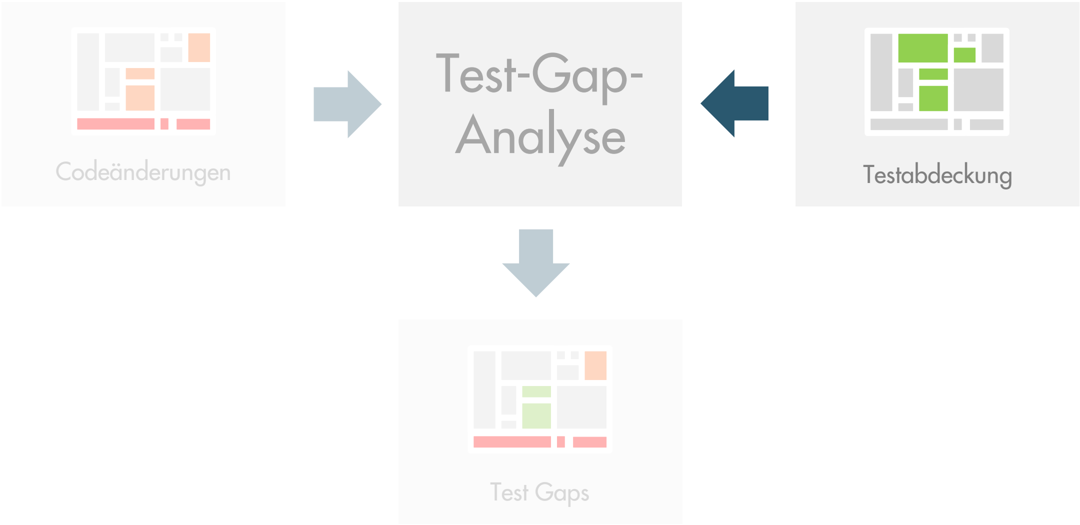


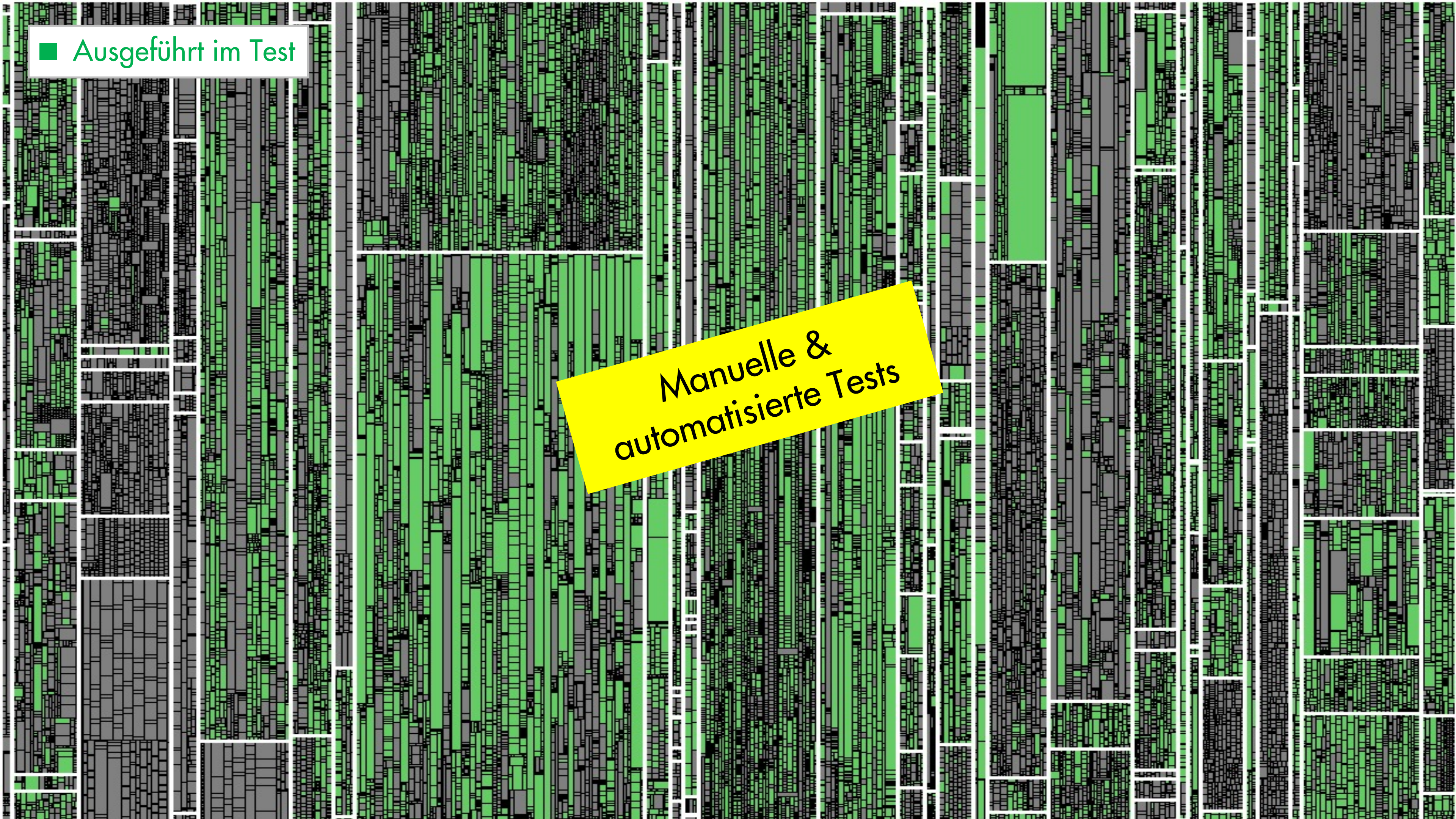


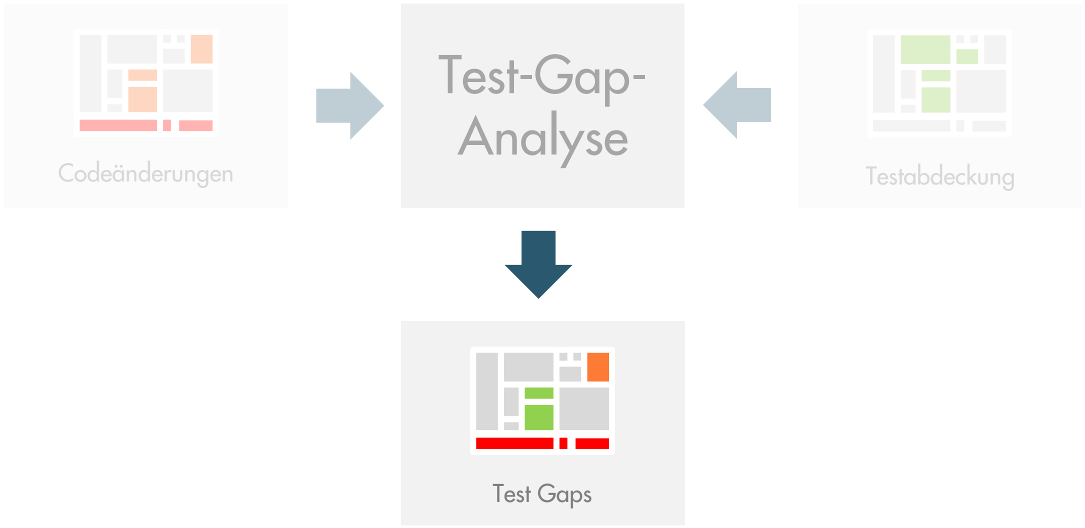


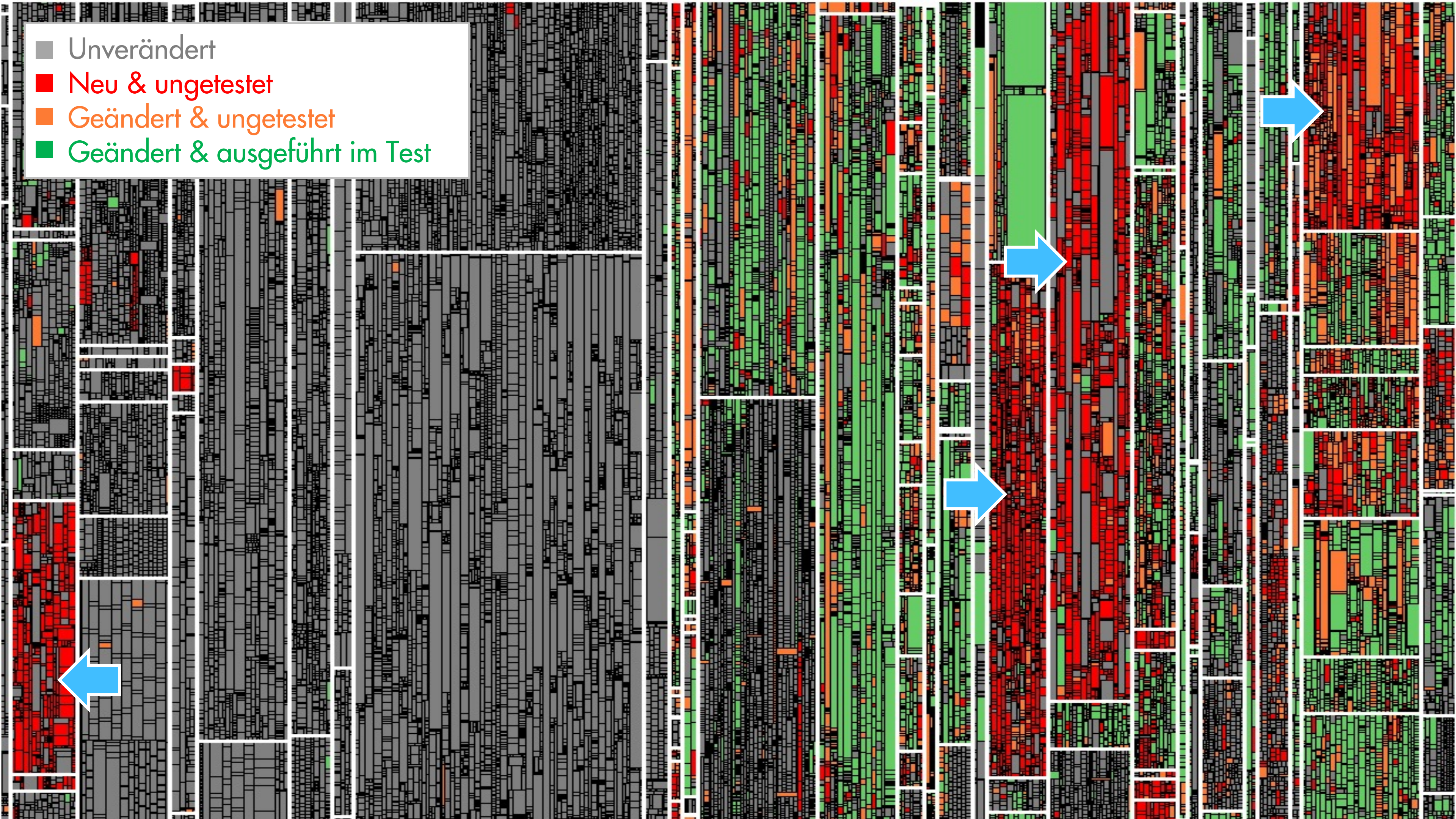

Codeänderungen
Test-Gap-Analyse
Testabdeckung
Test Gaps

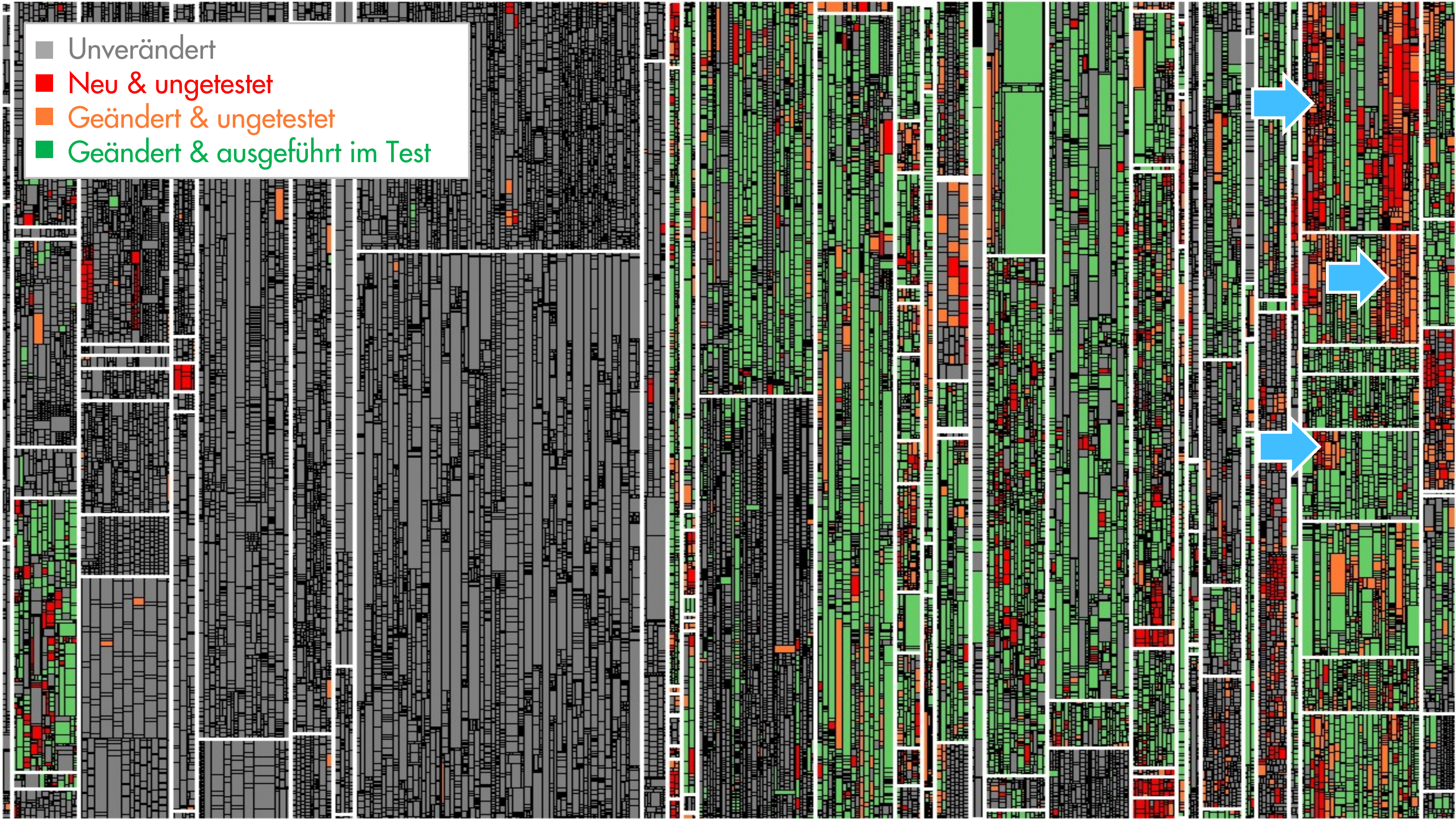

Neu
Geändert


Codeänderungen
Test-Gap-Analyse
Testabdeckung
Test Gaps

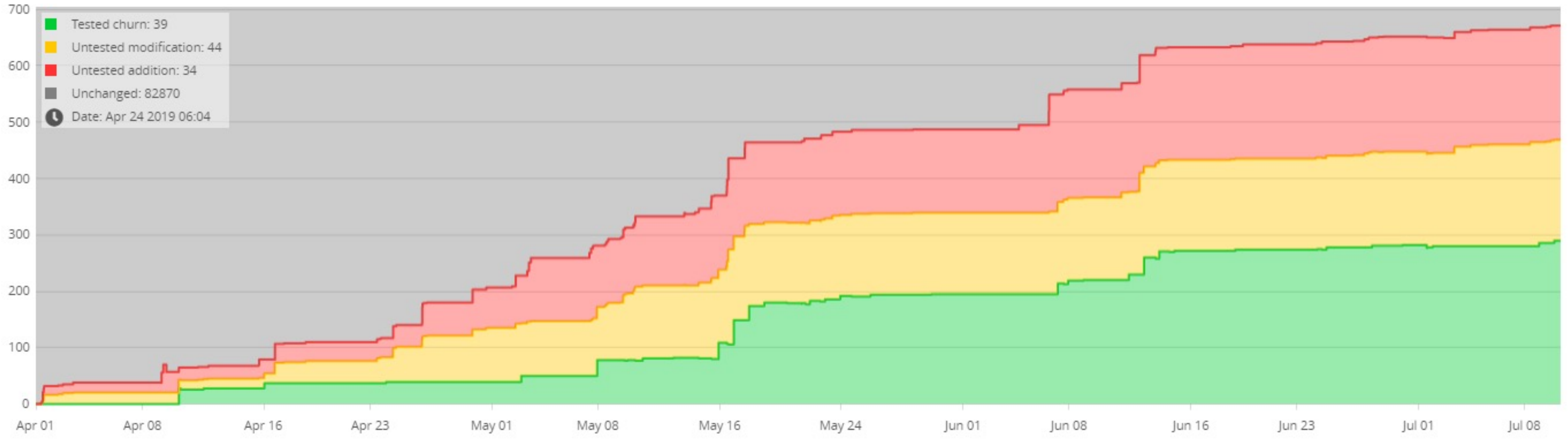

Ausgeführt im Test
Manuelle &
automatisierte Tests


Codeänderungen
Test-Gap-Analyse
Testabdeckung
Test Gaps

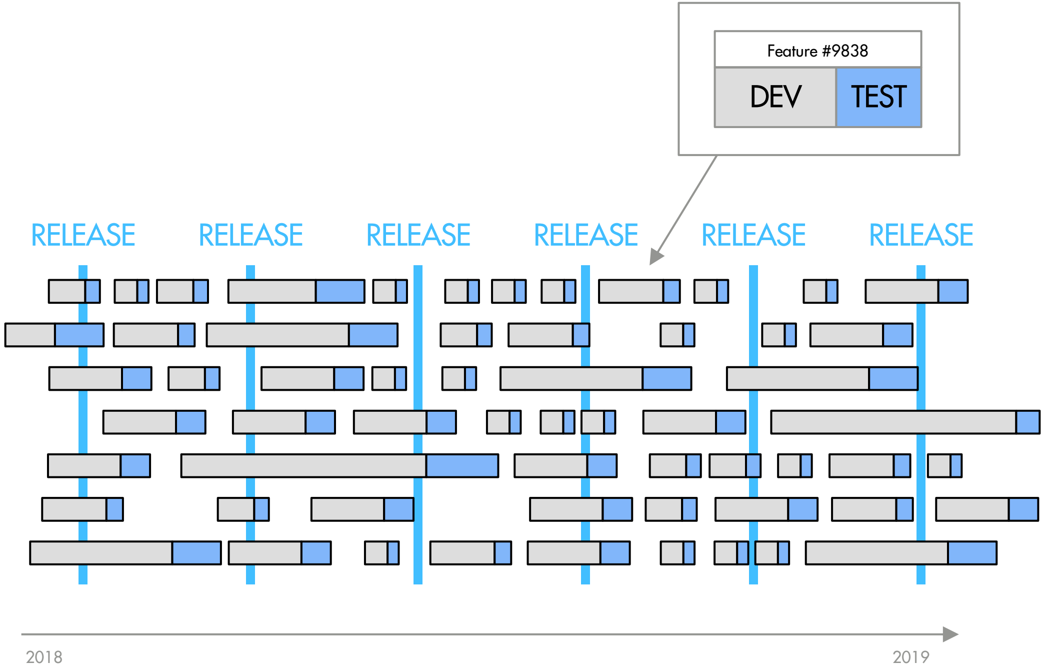

Unverändert
Neu & ungetestet
Geändert & ungetestet
Geändert & ausgeführt im Test


Unverändert
Neu & ungetestet
Geändert & ungetestet
Geändert & ausgeführt im Test

Zeitlicher Verlauf: Trends



Entwicklungsbegleitender Test



Done **Issue TS-23282 - clang-tidy causes SIGSEGV on C++ project (rewrite clang-tidy integration from JNI to call-in-new-process)**

Updated Aug 10 2020 11:23

Creator: Nils Kunze (on May 28 2020 12:32)

Assignee: Alexander von Rhein

project	Type	Priority	Resolution	Fix Version	Component
TS	Bug	High	Green	Teamscale 6.1	Backend
Labels	**Affected Version**	**Customer**	**Customer Issue**	**Dev Squad**	**Epic Name**
long-runner	6.0 RC3	[redacted]		Denali	

Freshdesk URL	Merge Request	PDash Task	QA-Contact
	https://git.cqse.eu/cqse/teamscale/-/merge_requests/8246	#4887	wilhelm

Description

Our clang-tidy integration can lead to Teamscale crashes because the clang-tidy tool sometimes (non-deterministic) causes segfault errors.
Since we execute clang-tidy via JNI in the same process as Teamscale, this segfault tears Teamscale down.

The concrete segfault appears in clang-tidy 9.0.2 (which we integrate currently) and has probably been fixed in clang-tidy 10.0.0. https://github.com/llvm/llvm-project/commit/f28972facc1fce9589feab9803e3e8cfad01891c#diff-

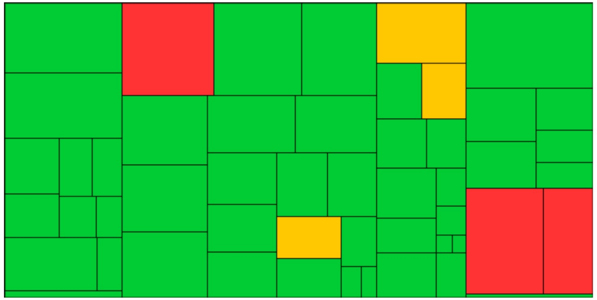
read more

Affected files 1046

Test Gaps

Auto-select issue branch ?   Auto-selected: cr/23282_reimplement_clang_tidy_integration

Jun 16 2020 13:47–Now | Test Gap: 100%   **Coverage sources:** All



Findings 12 1 1

Commits 44

Issues: Bug Fix Day 9.06.20

Auto-select issue branch (Automatically selected)

All issues **Coverage sources:** All

Found **210 issues** matching your query

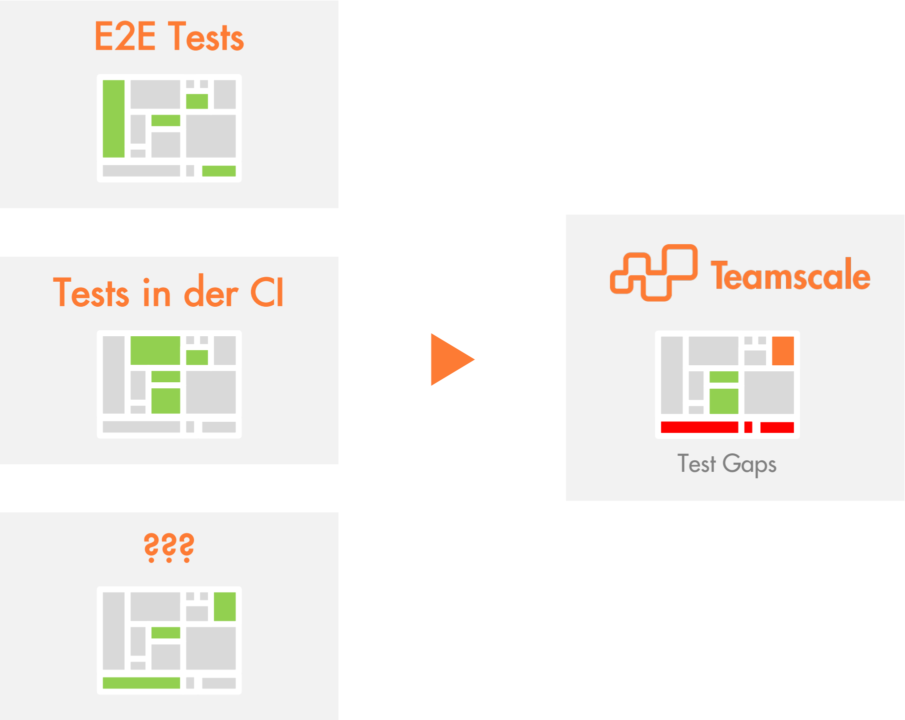
Test Gap over all matching issues: 34%

ID	Subject		# Changes	Test Gap
TS-23445	GitChangeRetriever stuck in branch labeling for 10-15 minutes	Done	11	0%
TS-23460	TestImpactSynchronizer still runs OOM	Done	47	4%
TS-23547	Slow analysis progress due to long labeling	Done	8	13%
TS-23501	Security: XML External Entity vulnerability in architecture uploads	Done	7	29%
TS-23599	Potentially swallowed exception in AnalysisReportPersister	Discarded	3	33%
TS-23576	Force Rollback UI broken	Done	3	33%
TS-23446	Python architecture analysis handles late addition of __init__.py file incorrectly	Done	3	33%
TS-23450	JIRA-Integration: Duplicated Table Rows, even for the same project	Done	2	50%
TS-23458	Audit search appears to ignore line breaks	Done	3	67%
TS-23558	External Upload view doesn't load due to JSON error	Done	30	77%

Teil 2

Herausforderungen bei der Einführung

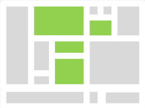
Herausforderung: Vollständiges Bild



Herausforderung: Änderung des Entwicklungsprozesses

E2E Tests

Tests in der CI



???

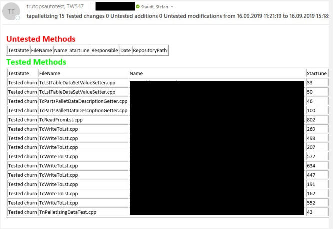
Teamscale

Test Gaps

Ergebnisse integrieren

. . .

Benachrichtigungen



trutopsautotest, TW547 [redacted] Staudt, Stefan

tapalletizing 15 Tested changes 0 Untested additions 0 Untested modifications from 16.09.2019 11:21:19 to 16.09.2019 15:18

Untested Methods

TestState	FileName	Name	StartLine	Responsible	Date	RepositoryPath

Tested Methods

TestState	FileName	Name	StartLine
Tested churn	TcLstTableDataSetValueSetter.cpp	[redacted]	33
Tested churn	TcLstTableDataSetValueSetter.cpp	[redacted]	50
Tested churn	TcPartsPalletDataDescriptionGetter.cpp	[redacted]	46
Tested churn	TcPartsPalletDataDescriptionGetter.cpp	[redacted]	100
Tested churn	TcReadFromLst.cpp	[redacted]	802
Tested churn	TcWriteToLst.cpp	[redacted]	269
Tested churn	TcWriteToLst.cpp	[redacted]	498
Tested churn	TcWriteToLst.cpp	[redacted]	207
Tested churn	TcWriteToLst.cpp	[redacted]	572
Tested churn	TcWriteToLst.cpp	[redacted]	634
Tested churn	TcWriteToLst.cpp	[redacted]	447
Tested churn	TcWriteToLst.cpp	[redacted]	191
Tested churn	TcWriteToLst.cpp	[redacted]	162
Tested churn	TcWriteToLst.cpp	[redacted]	552
Tested churn	TnPalletizingDataTest.cpp	[redacted]	43

Herausforderung: Einfluss des Profilings

Performance

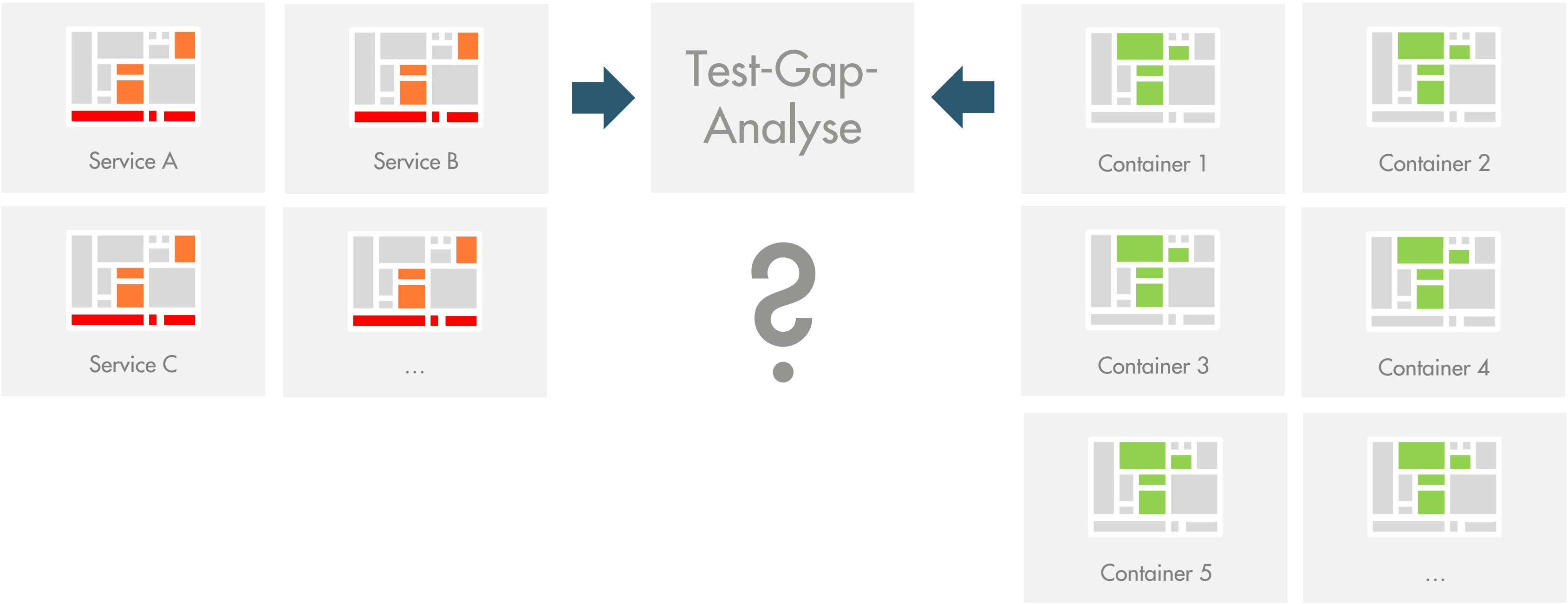
Verhalten

Profiler-Wahl

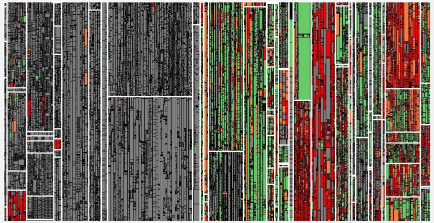
cqse.eu/tga-trumpf

Redundanz

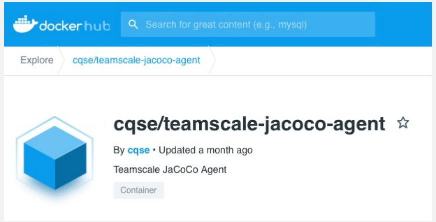
Herausforderung: Microservices



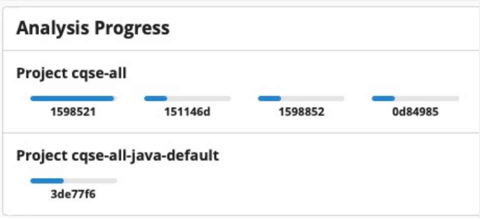
Herausforderung: Microservices



Gesamtsicht über alle Repositories



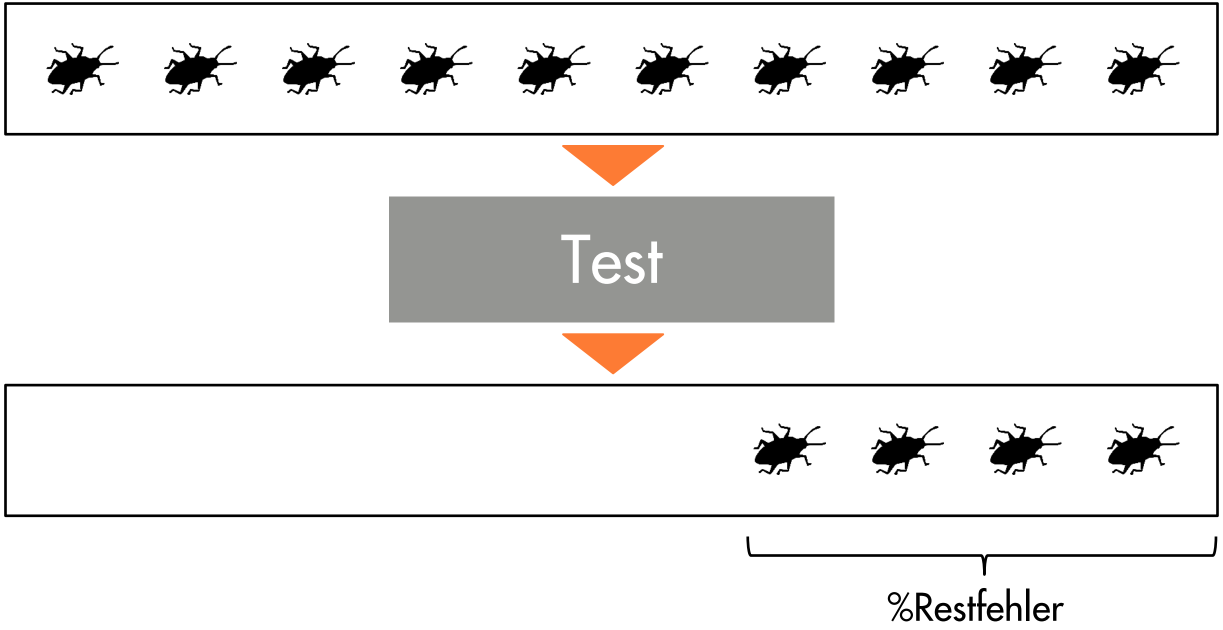
Infrastructure as Code



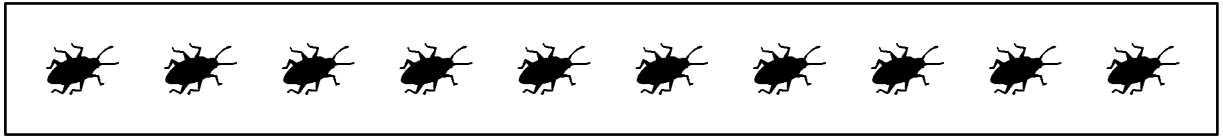
Analyse-Performance

Teil 3

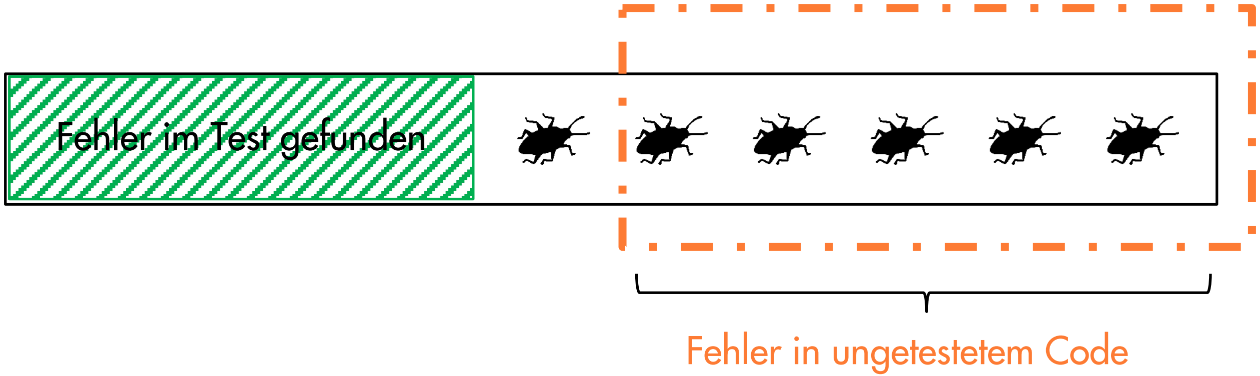
Kosten-Nutzen-Berechnung der Test-Gap-Analyse


Test
%Restfehler

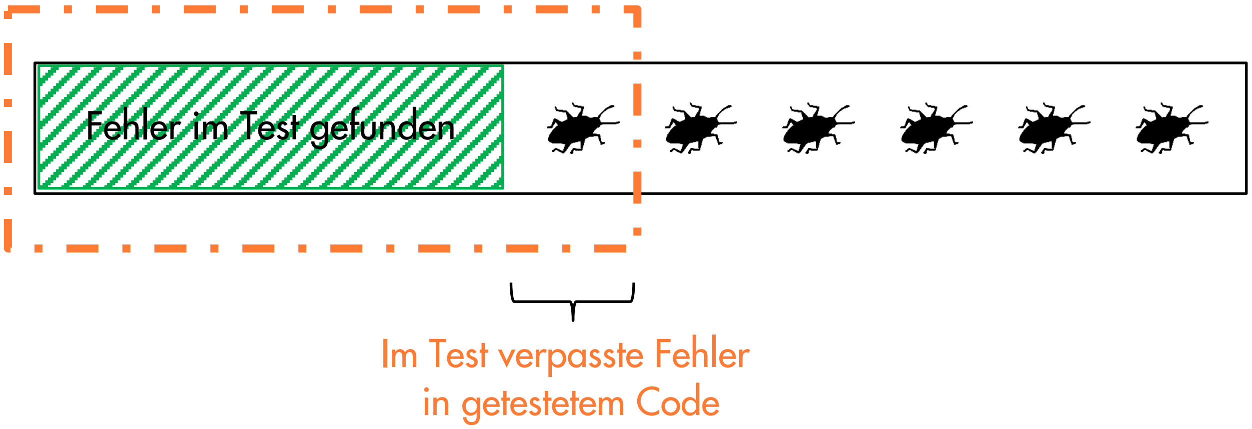
$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



$$\%\text{Restfehler} = \%\text{Getestet} * \text{Testineffektivität} + \%\text{Testgap}$$



$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



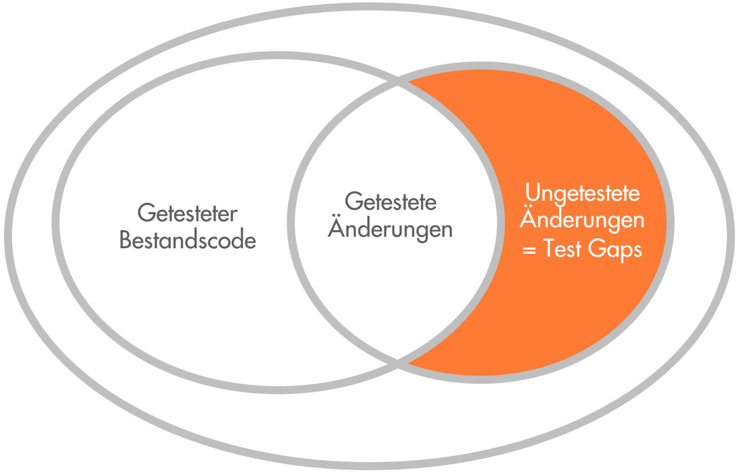
Paper published at Workshop on Automation of Software Test, 2013.

Wo treten Fehler in Produktion auf?

Studie: C# System

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

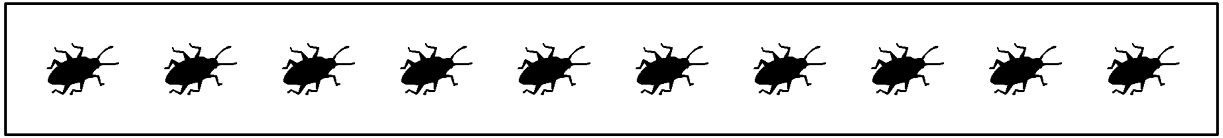
Release B:
15% Code neu/geändert,
>60% ungetestet



Feldfehlerwahrscheinlichkeit 5x höher für ungetestete Änderungen!

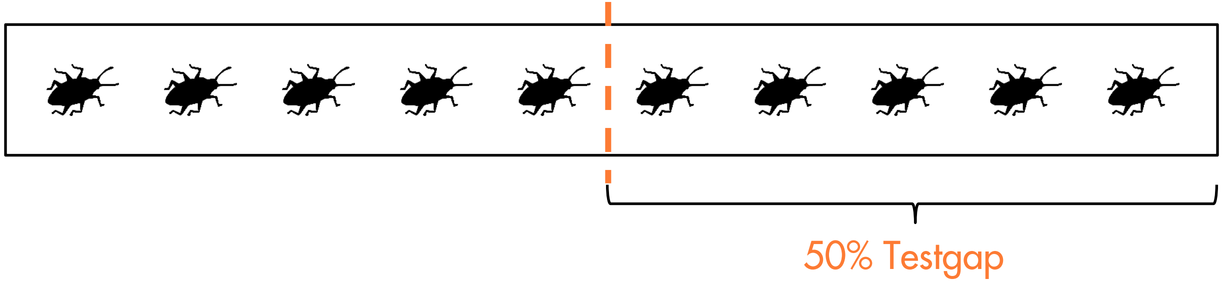
Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

$$\%\text{Restfehler} = \%\text{Getestet} * \text{Testineffektivität} + \%\text{Testgap}$$

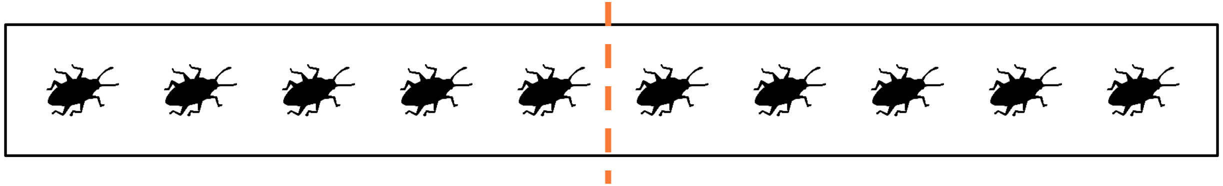


Ohne Test-Gap-Analyse

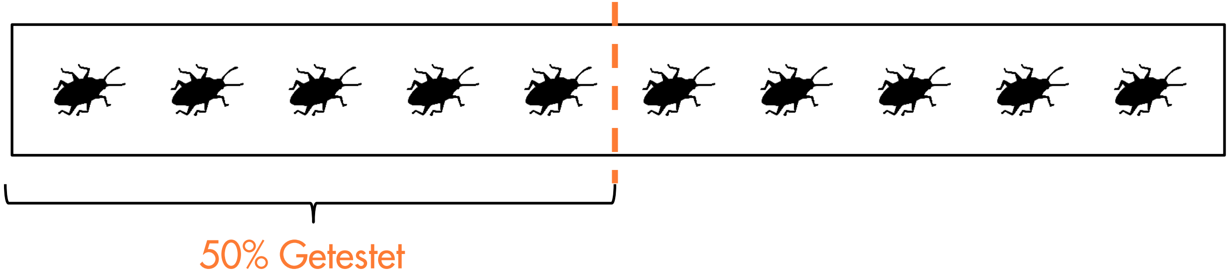
$$\%\text{Restfehler} = \%\text{Getestet} * \text{Testineffektivität} + 50\%$$



$$\%\text{Restfehler} = \%\text{Getestet} * \text{Testineffektivität} + 50\%$$

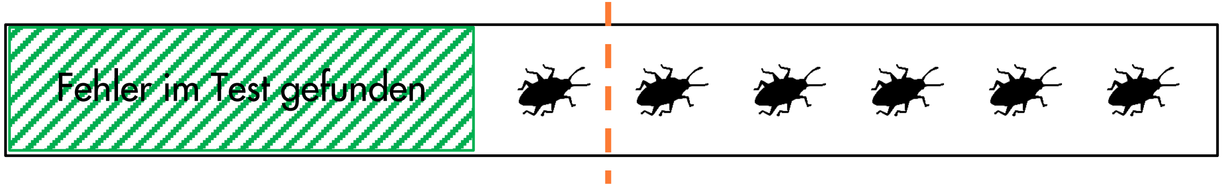


$$\%Restfehler = 50\% * Testineffektivität + 50\%$$



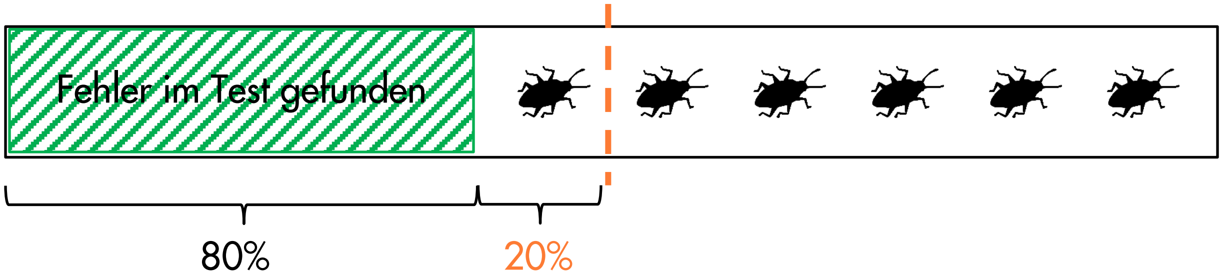
Ohne Test-Gap-Analyse

$$\%Restfehler = 50\% * Testineffektivität + 50\%$$



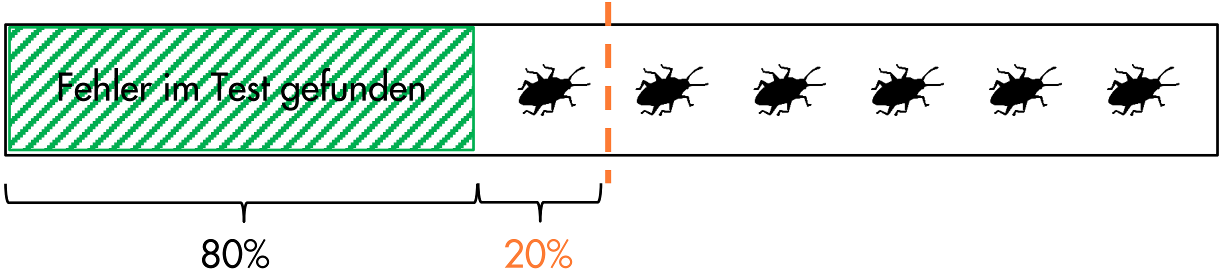
Ohne Test-Gap-Analyse

$$\%\text{Restfehler} = 50\% * \text{Testineffektivität} + 50\%$$

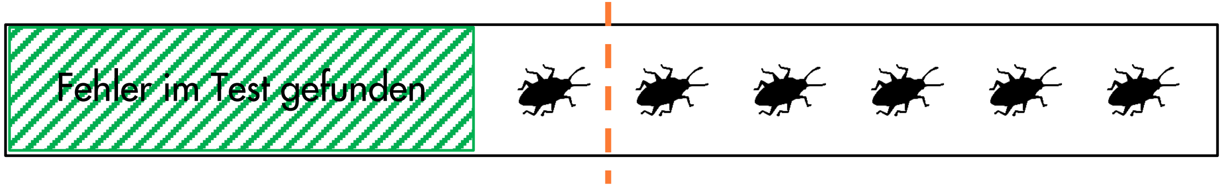


Ohne Test-Gap-Analyse

$$\%\text{Restfehler} = 50\% * 20\% + 50\%$$

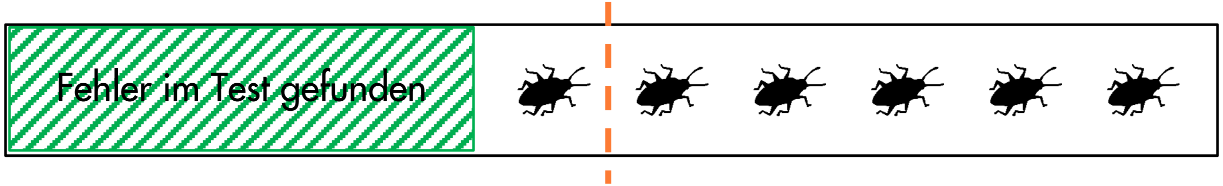


$$\%Restfehler = 10\% + 50\%$$



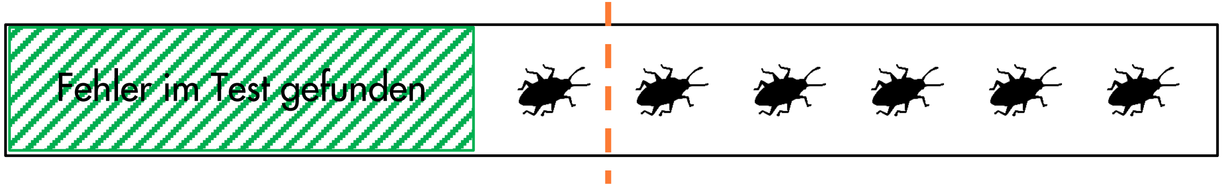
Ohne Test-Gap-Analyse

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



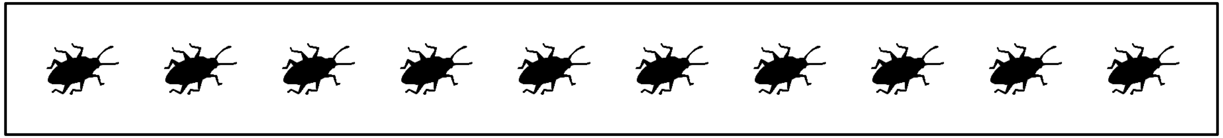
Ohne Test-Gap-Analyse

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



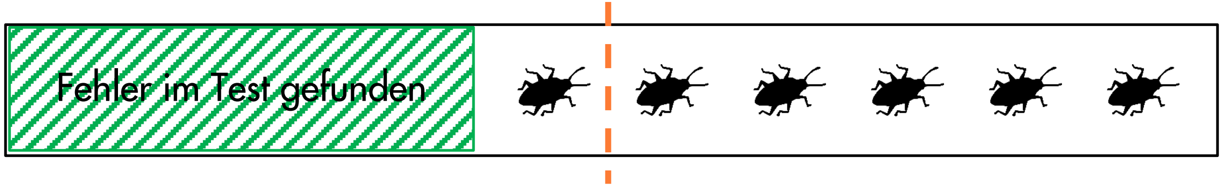
Mit Test-Gap-Analyse

$$\%\text{Restfehler} = \%\text{Getestet} * \text{Testineffektivität} + \%\text{Testgap}$$



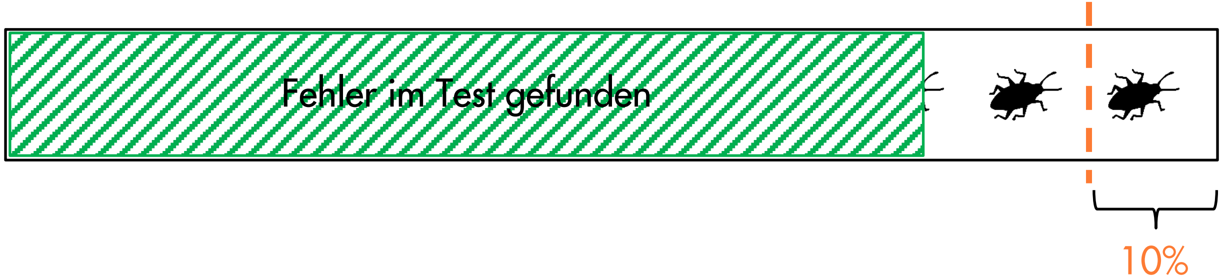
Ohne Test-Gap-Analyse

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



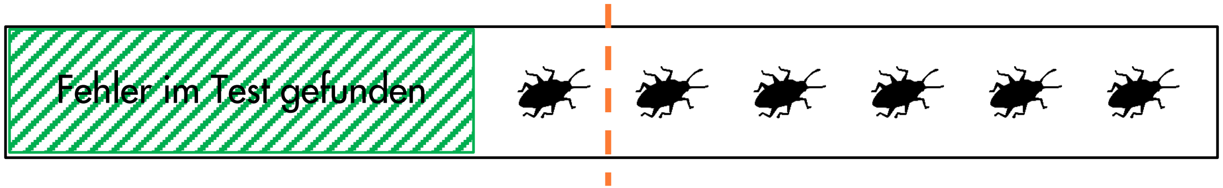
Mit Test-Gap-Analyse

$$\%\text{Restfehler} = 90\% * \text{Testineffektivität} + 10\%$$



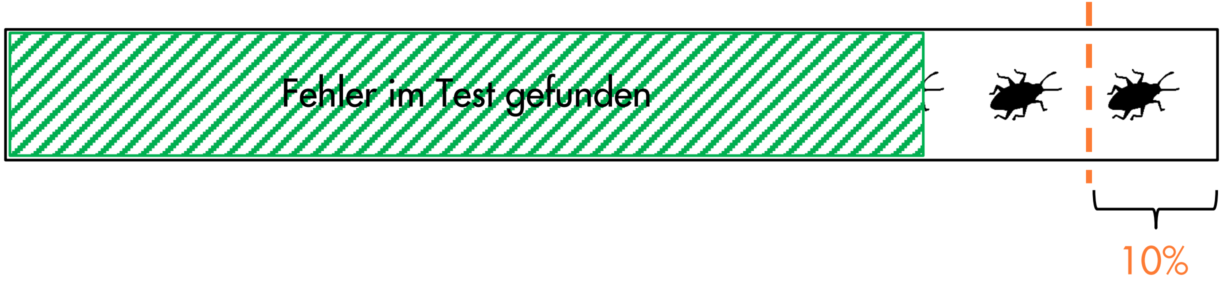
Ohne Test-Gap-Analyse

$$\%Restfehler = 60\%$$



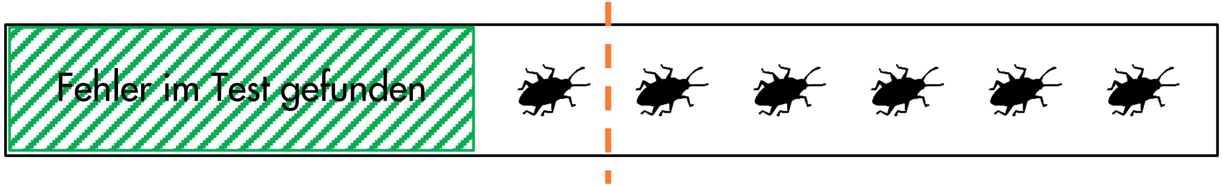
Mit Test-Gap-Analyse

$$\%Restfehler = 90\% * 20\% + 10\%$$



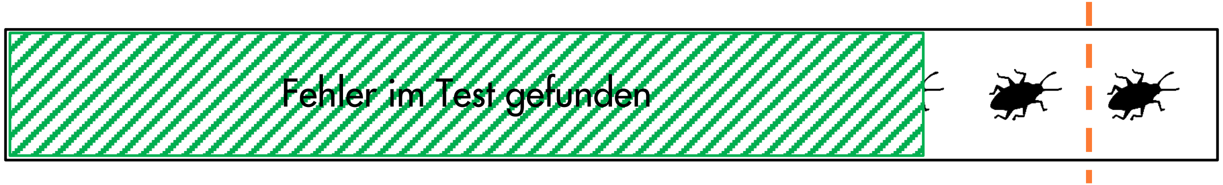
Ohne Test-Gap-Analyse

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



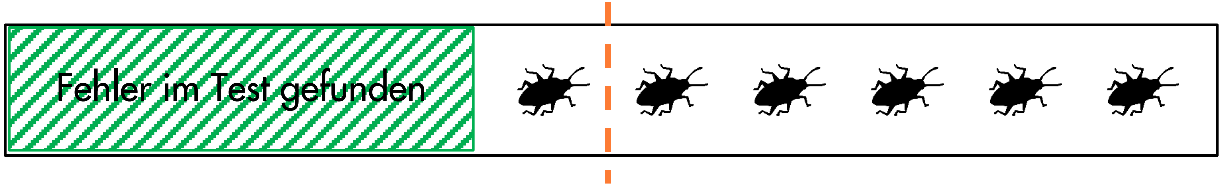
Mit Test-Gap-Analyse

$$\%\text{Restfehler} = 18\% + 10\%$$



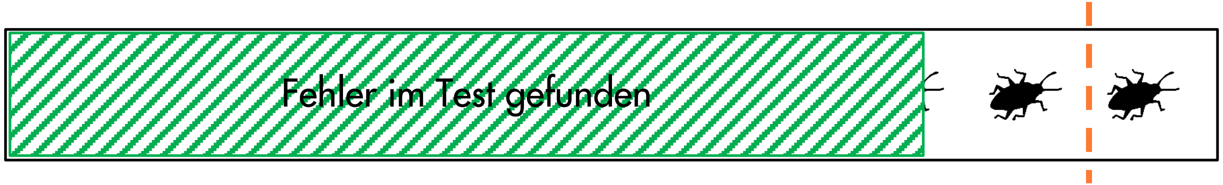
Ohne Test-Gap-Analyse

%Restfehler = 60%



Mit Test-Gap-Analyse

%Restfehler = 28%



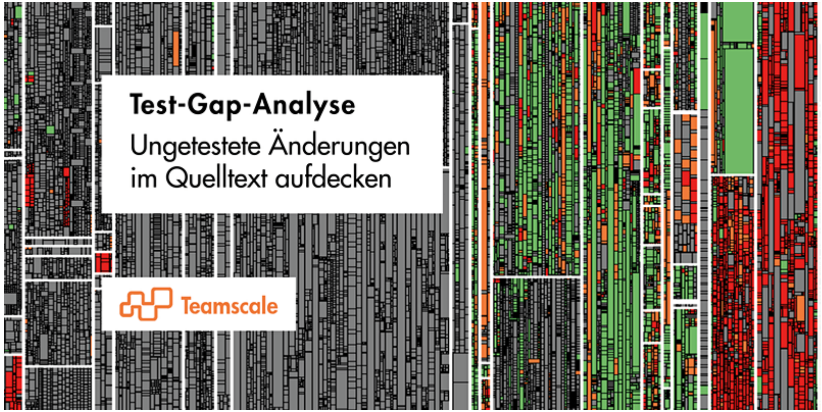
Reduzierte Feldfehler = **50%**

Test-Gap-Analyse reduziert Feldfehler in Applikationen der Munich Re um ½

Fazit

- **Sichtbarmachen** von Qualität ist essentiell
- **Werkzeuge und Prozesse** sind wichtig
- Internes **Change Management** notwendig
- Deutlicher **positiver Effekt** beobachtbar
- Am besten **gleich von Anfang** an einsetzen

Mehr Details und Live Demo im CQSE Workshop



🇩🇪 16. Februar 2022, 10:30-12:00 Uhr CEST
Registrierung: cqse.eu/tga-workshop-de-2022-02-s

🇺🇸 09. März 2022, 17:00-18:30 Uhr CEST
Registrierung: cqse.eu/tga-workshop-en-2022-03-s

Kontakt – Ich freue mich auf Diskussionen!

Dr. Sven Amann · amann@cqse.eu · +49 172 1860063

CQSE GmbH
Centa-Hafenbrädl-Str. 59
81249 München

CQSE
Continuous Quality in Software Engineering