Test-Gap-Analyse

Wie erkennen wir ungetestete Änderungen, bevor sie ins Release gehen?

Dr. Andreas Göb
XP Days Germany 2015

CQSE
Continuous Quality in Software Engineering

Über mich

Forschung

- Modellierung von Softwarequalität
- Wartbarkeit, Software-Architektur

Beratung

- Qualitäts-Bewertung & Qualitäts-Controlling
- Wirksamer Werkzeugeinsatz in Entwicklung und Test

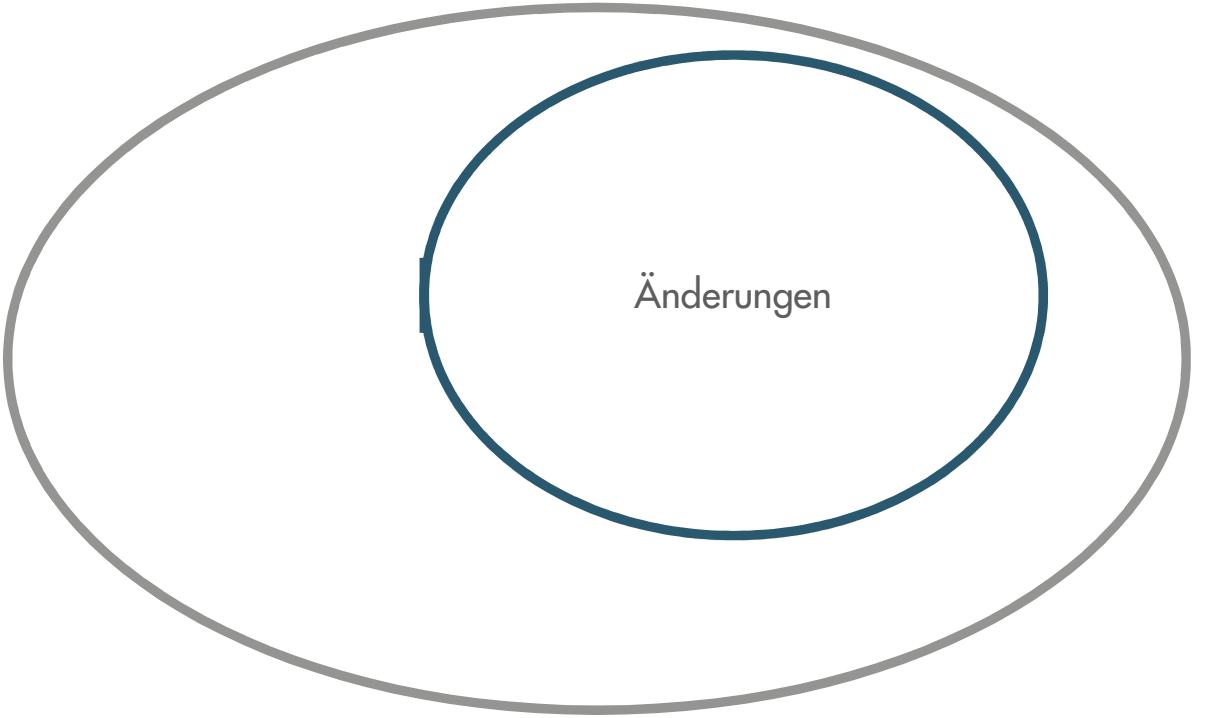
Entwicklung

- Continuous Quality Assessment Toolkit ConQAT
- >400 kLOC, Apache Lizenz, >25.000 Downloads
- Kommerzielle Erweiterung: Teamscale

Agenda

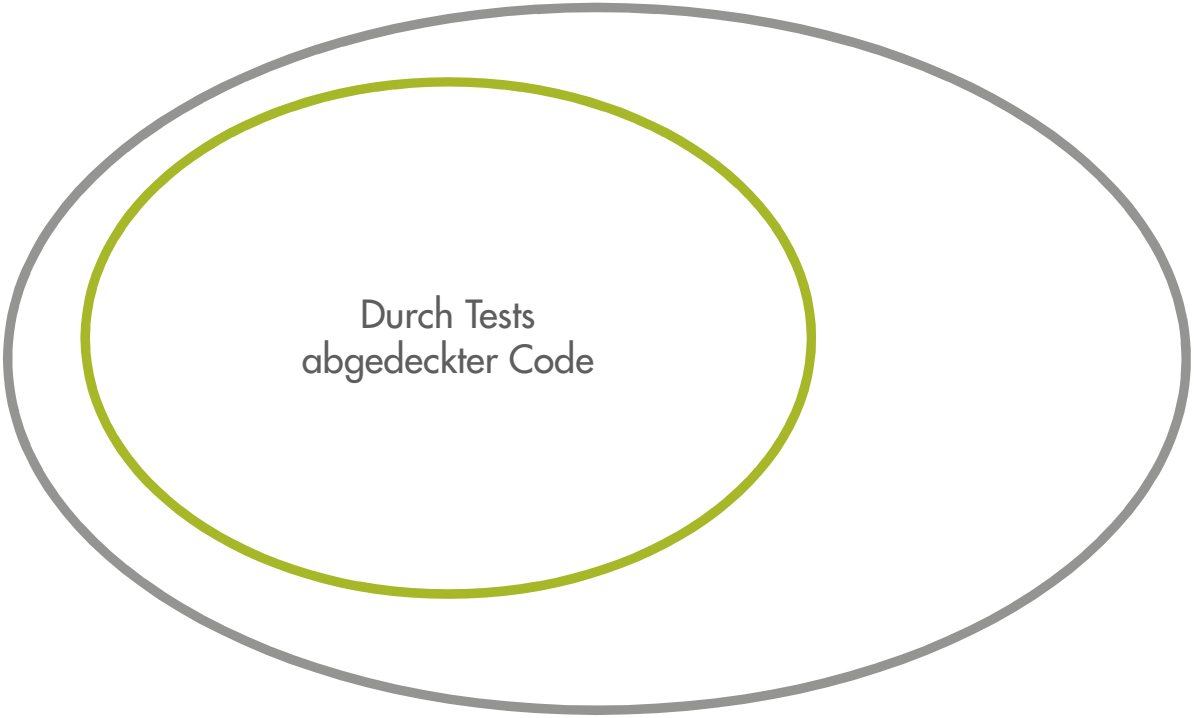
- **Hintergrund**
- Analyse und Visualisierung
- Erfahrungen aus der Praxis
- Ausblick und Diskussion

Hintergrund



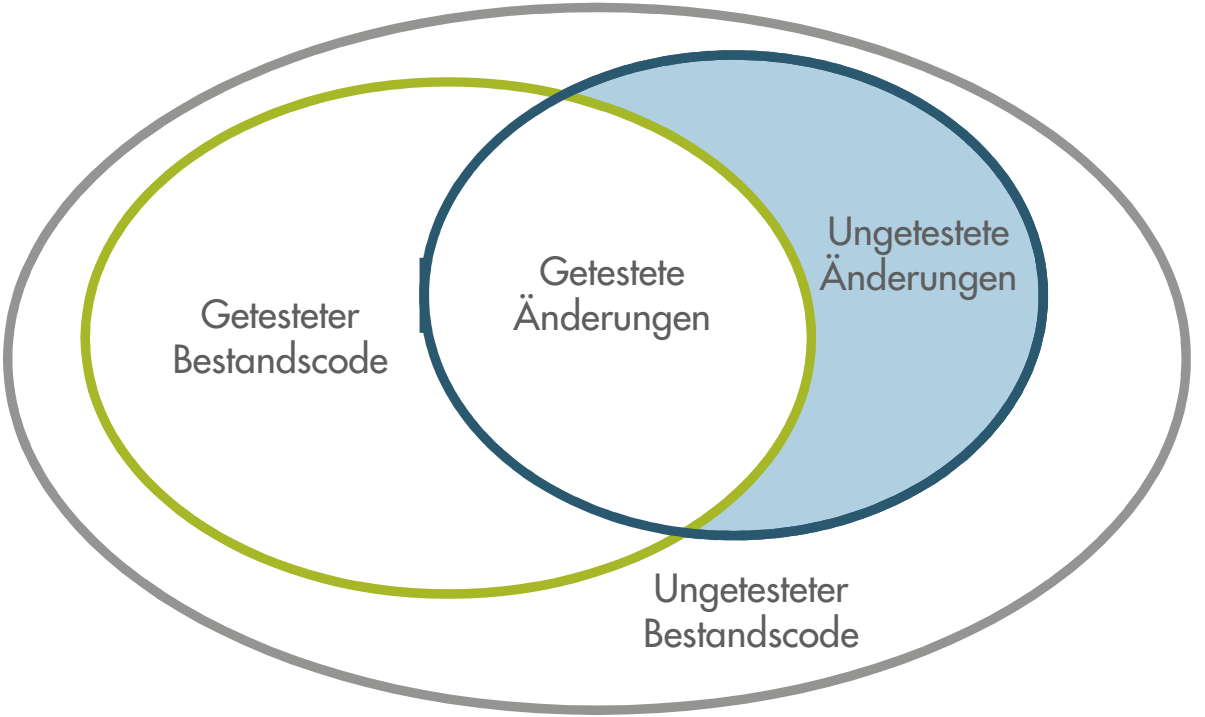
Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

Hintergrund



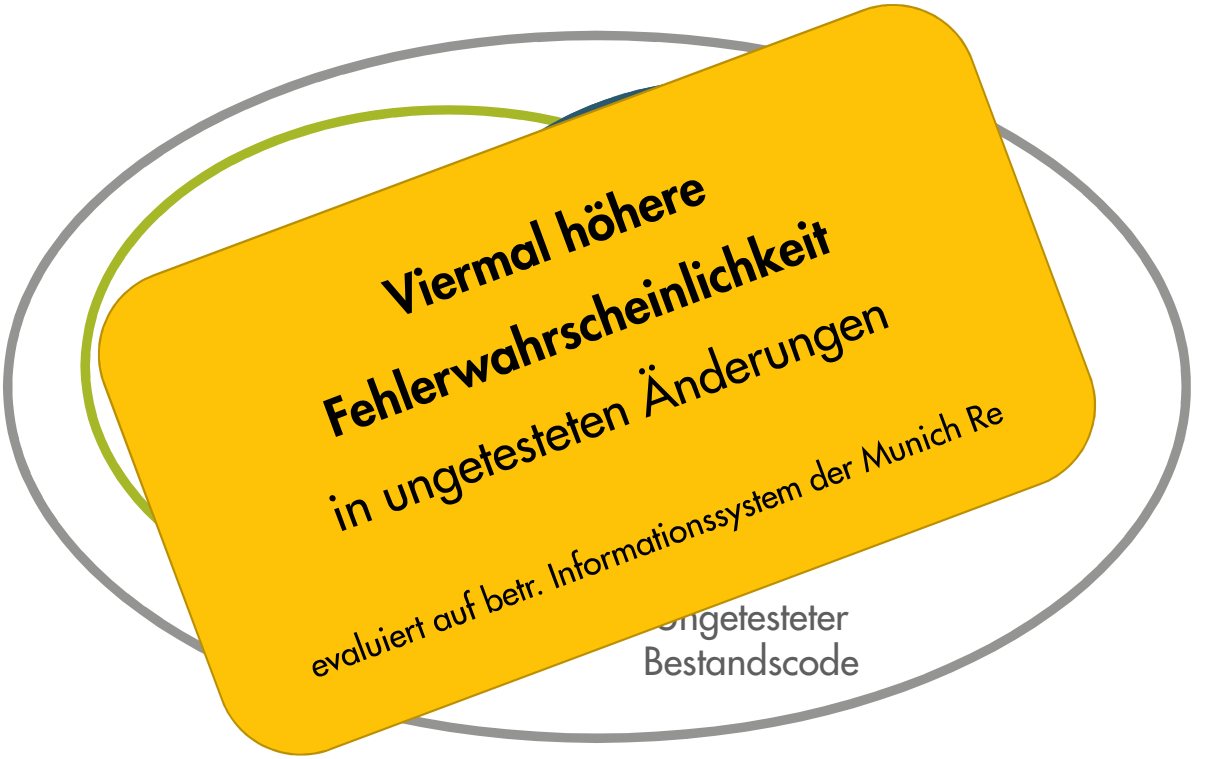
Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

Hintergrund



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

Hintergrund

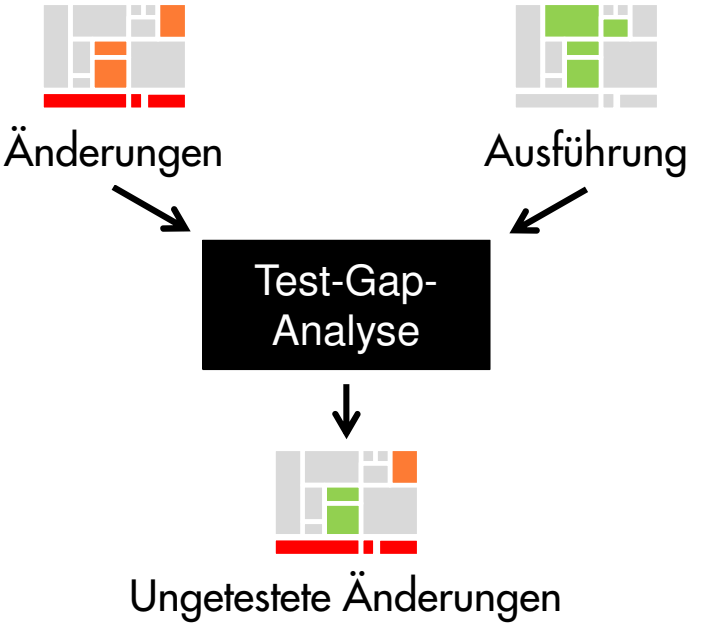


Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

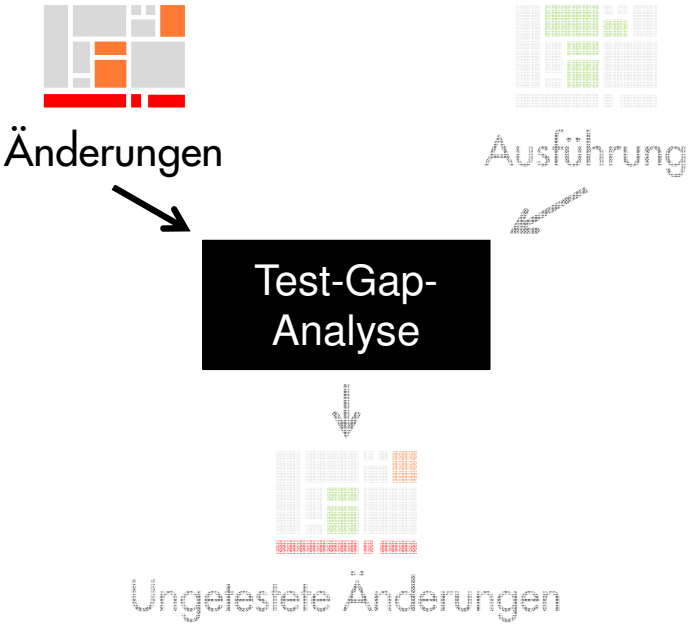
Agenda

- Hintergrund
- **Analyse und Visualisierung**
- Erfahrungen aus der Praxis
- Ausblick und Diskussion

Die Test-Gap-Analyse



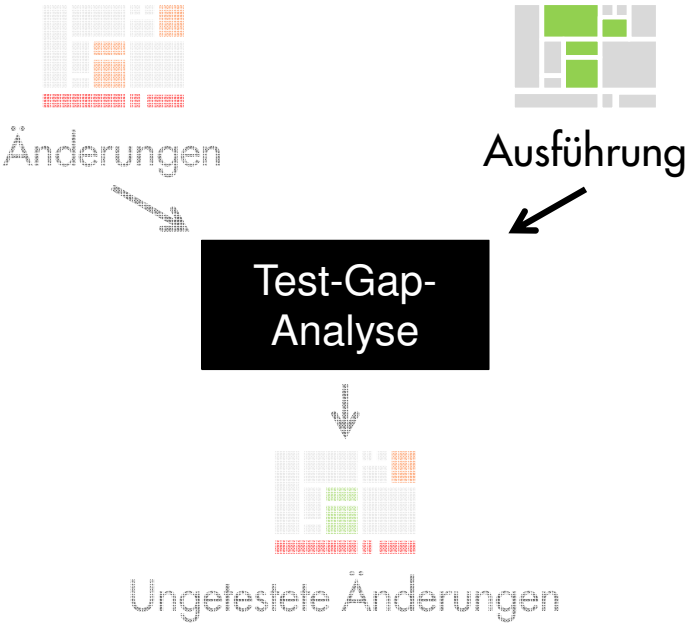
Änderungserkennung



- Gegenüber Referenzstand
- Akkumulierte Analyse

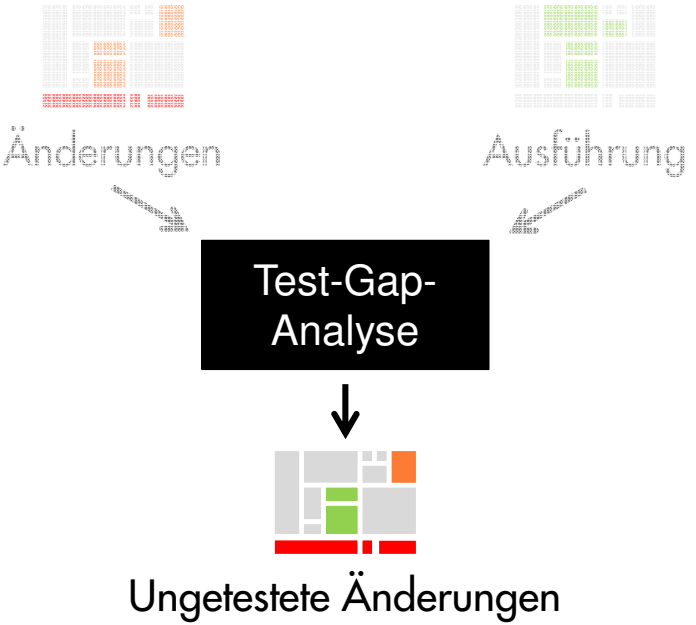
- Nur funktionale Änderungen
  - Keine Kommentare
  - Keine Umbenennungen
  - Keine Verschiebungen
  - Keine Refactorings

Auswertung der Ausführung

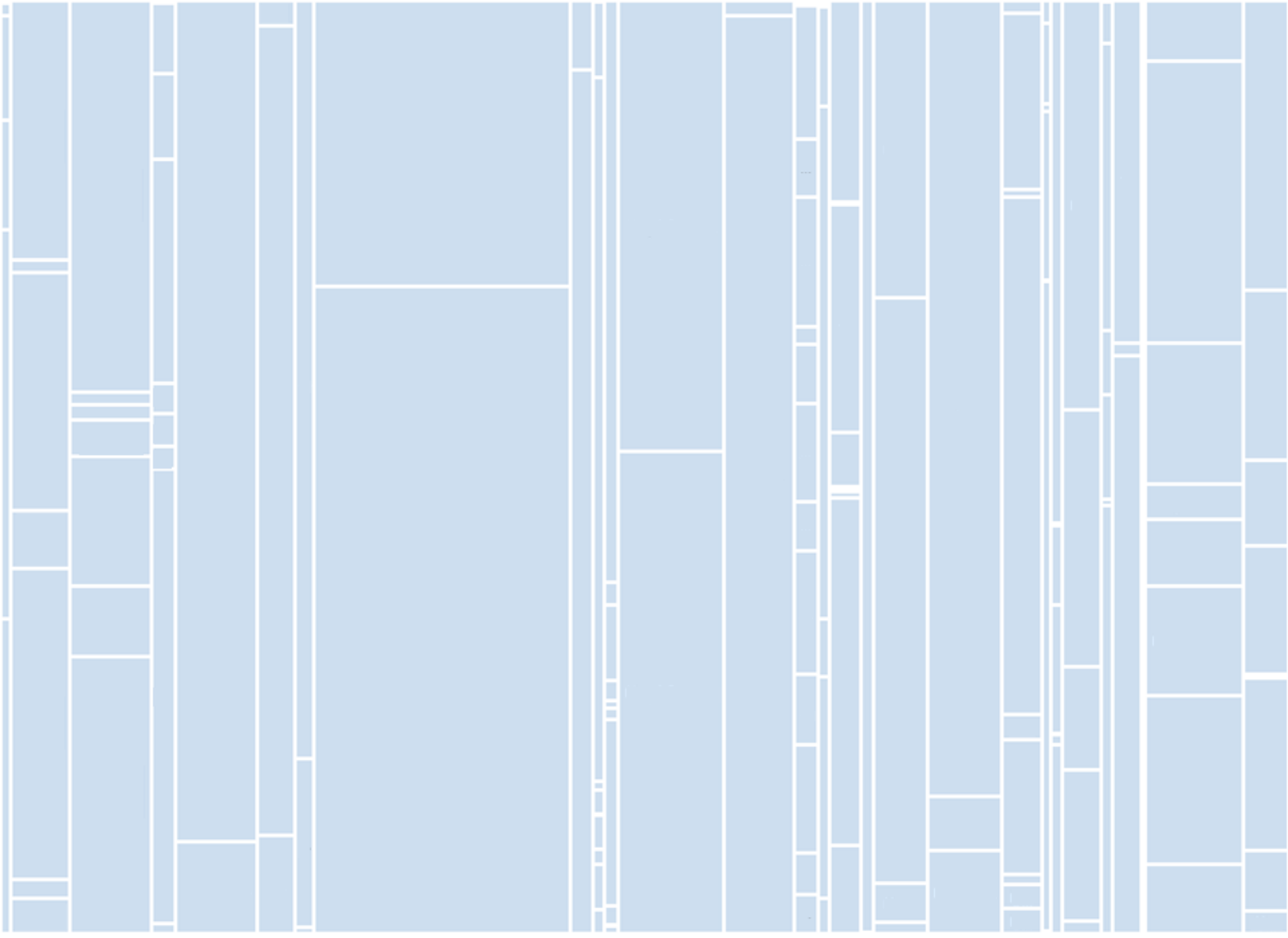


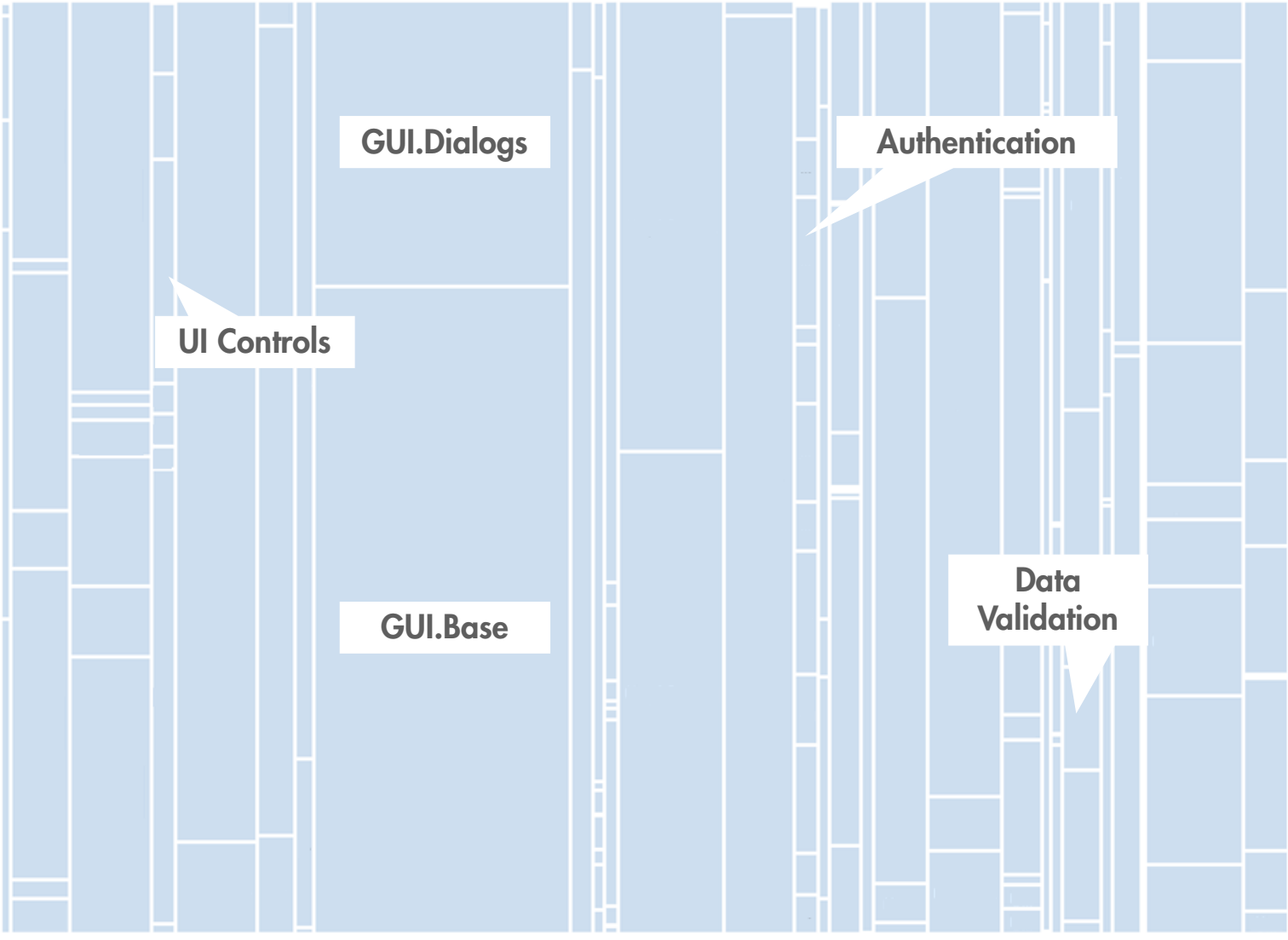
- Granularität: Methoden
- Auf Server oder Clients
- Manuelle oder Unit-Tests
- Minimaler Overhead
- Plattformspezifisch

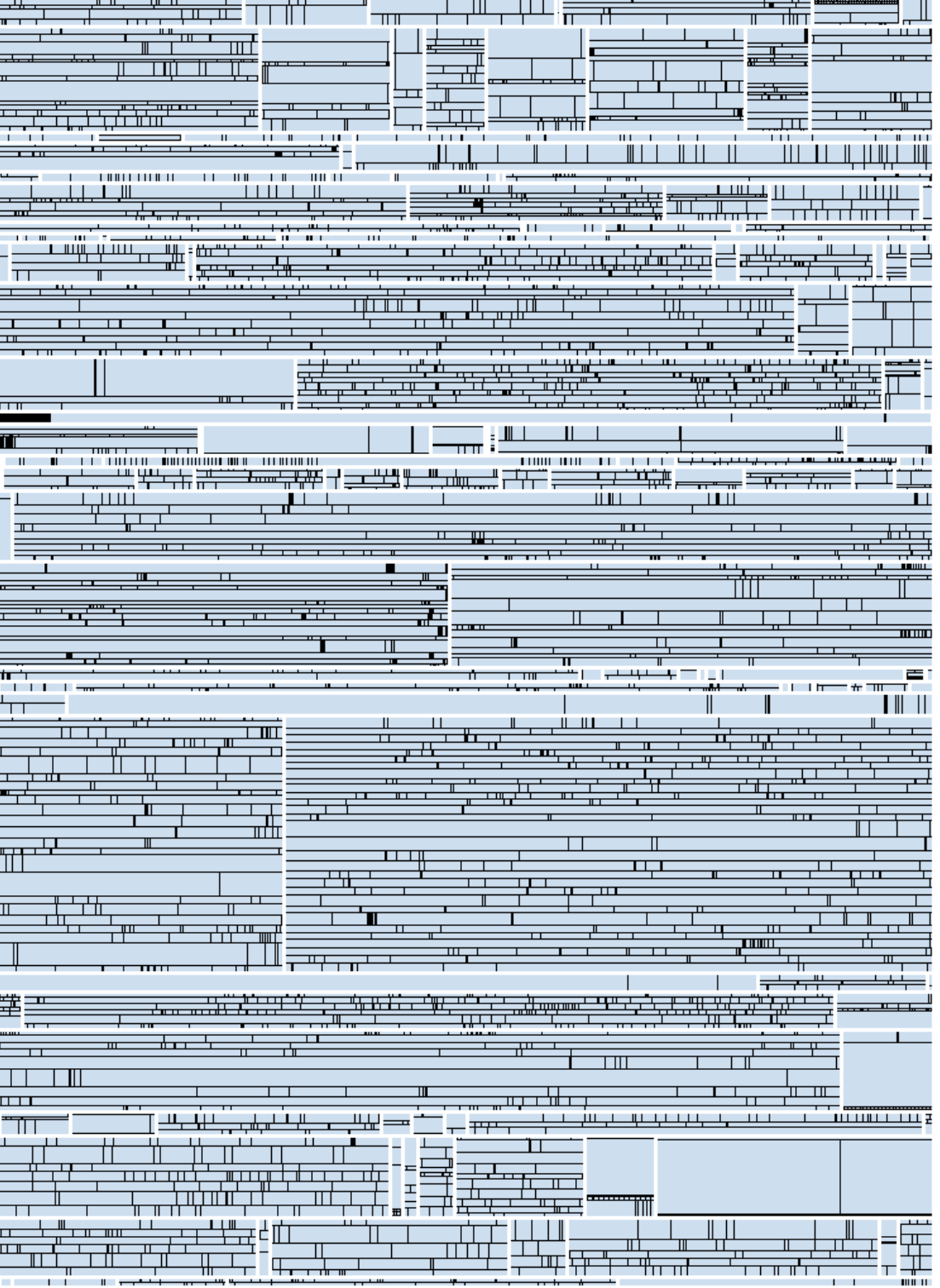
Kombination von Änderung und Ausführung



- Täglich aktualisiert
- Inkrementell verzahnt
- Pro Testumgebung
- Aggregierte Sicht



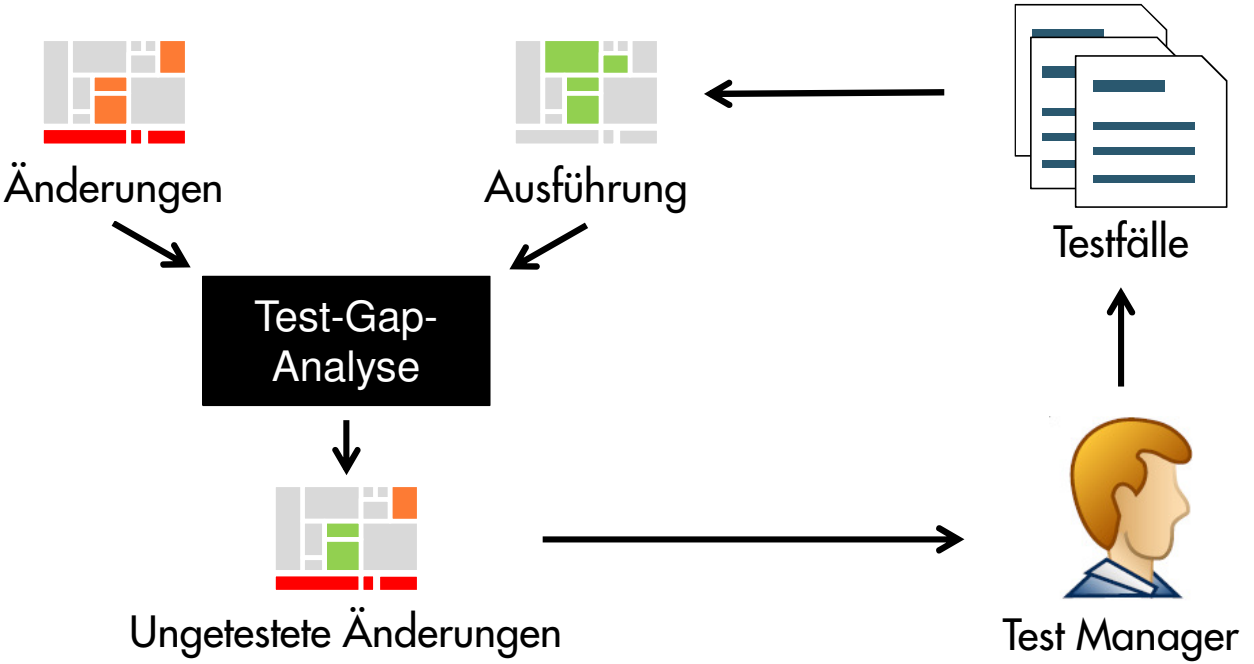

GUI.Dialogs
Authentication
UI Controls
Data
Validation
GUI.Base



Agenda

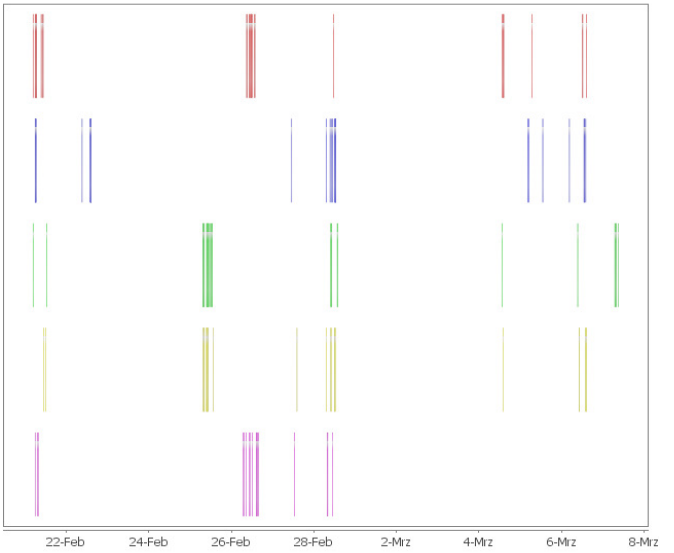
- Hintergrund
- Analyse und Visualisierung
- **Erfahrungen aus der Praxis**
- Ausblick und Diskussion

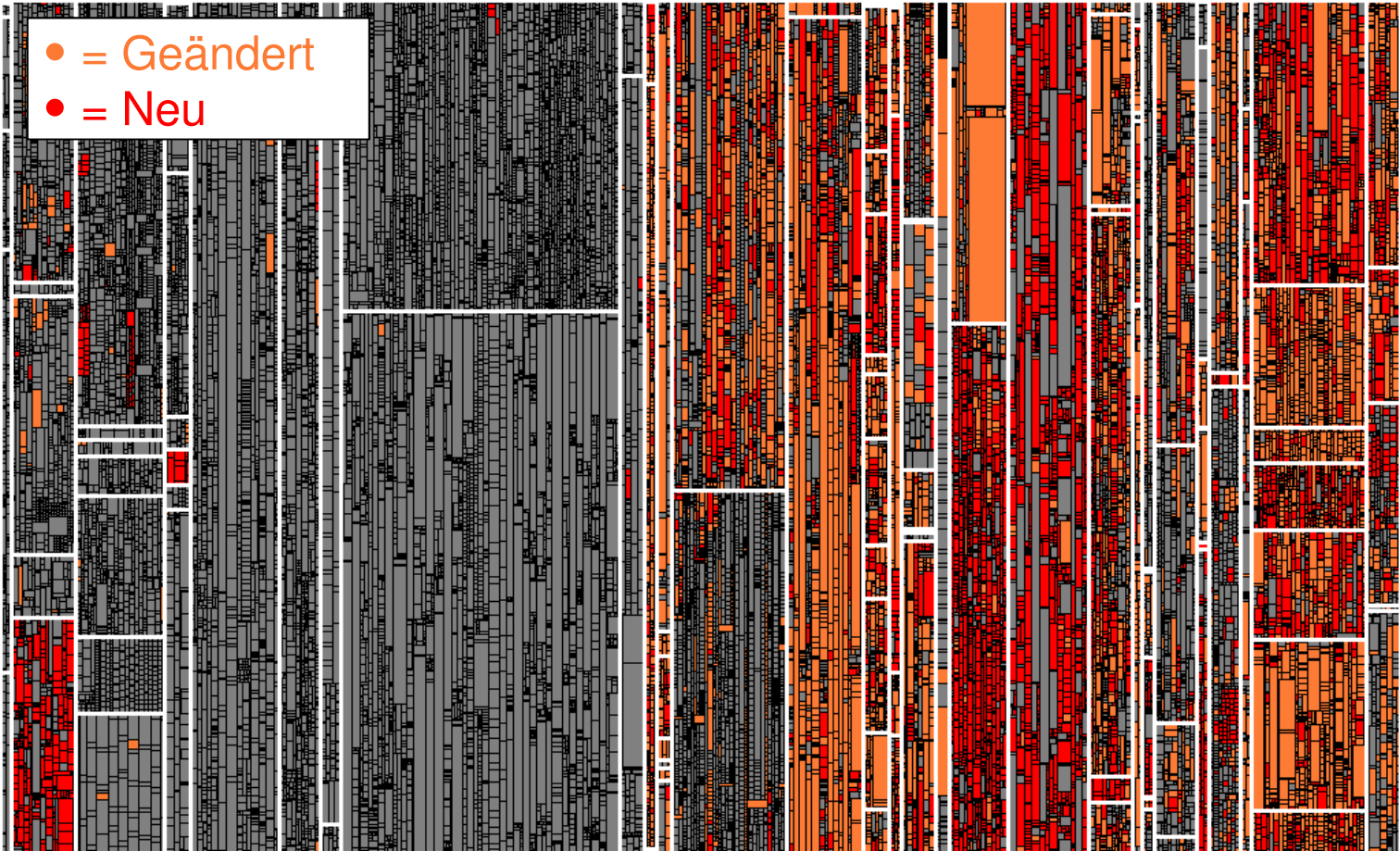
Einsatz im Testprozess

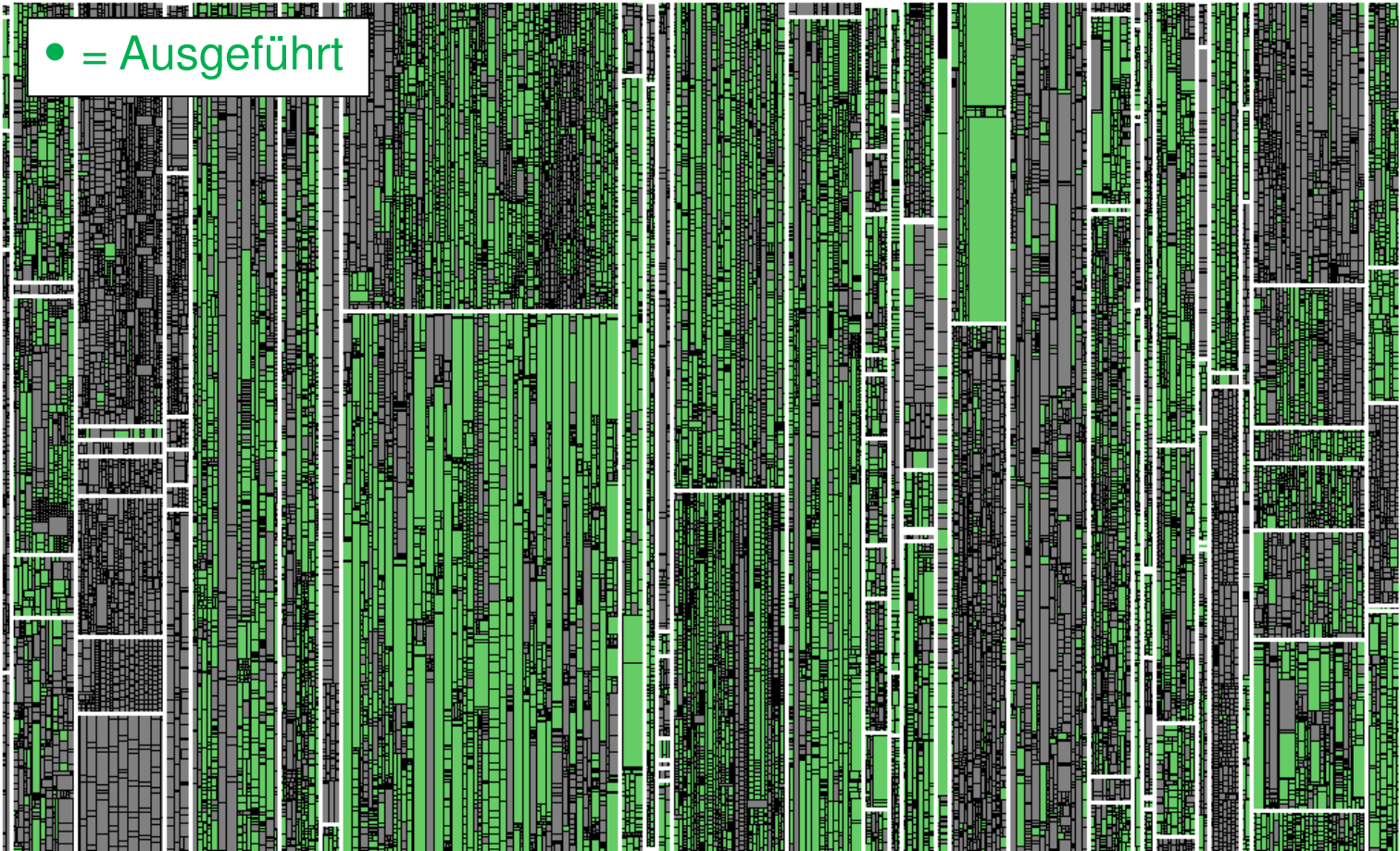


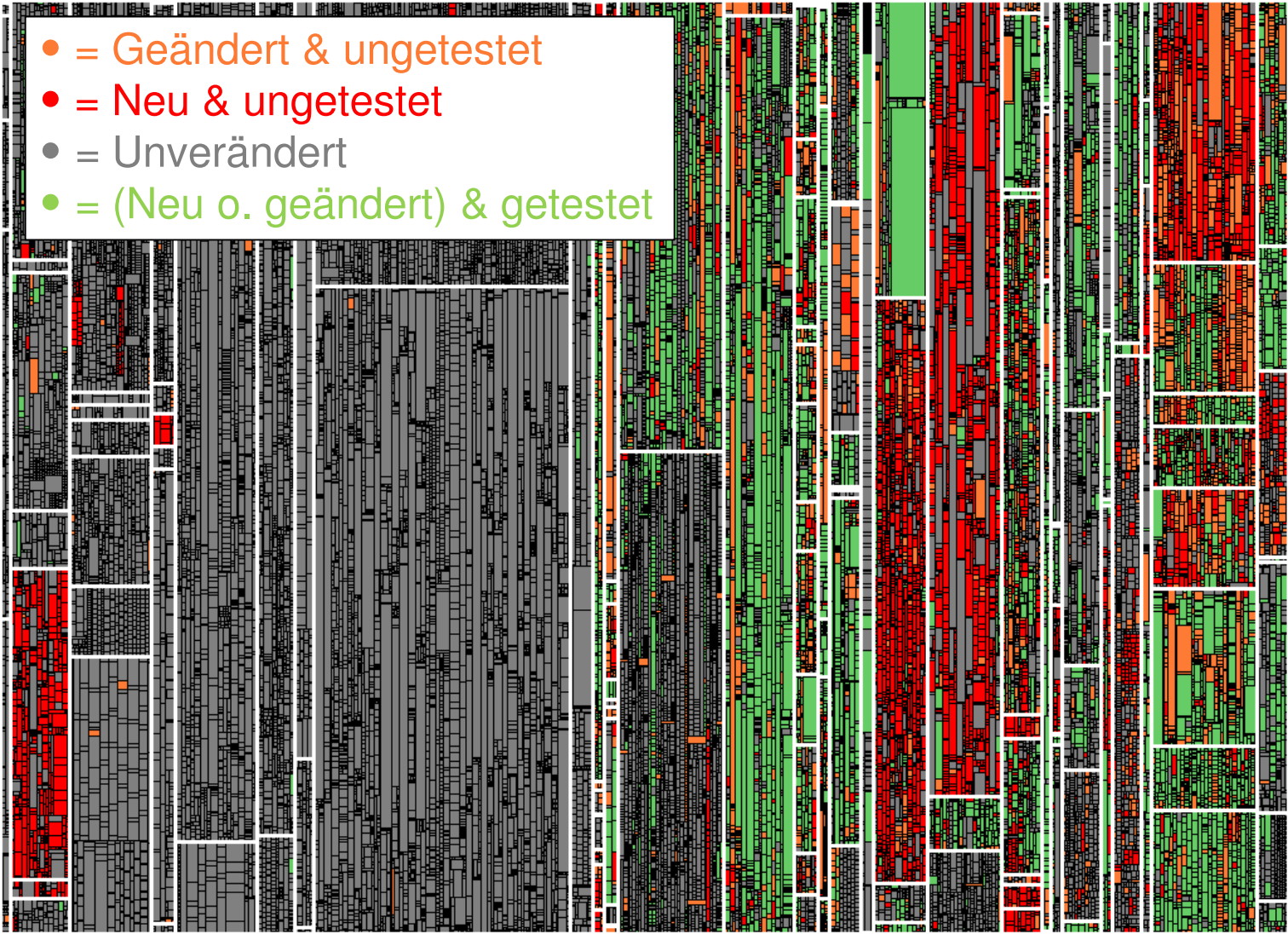
Fallstudie: Test-Factory

- 5 Tester in Indien
- 2 Wochen
- 143 Test-Sessions
- 6 Systemversionen



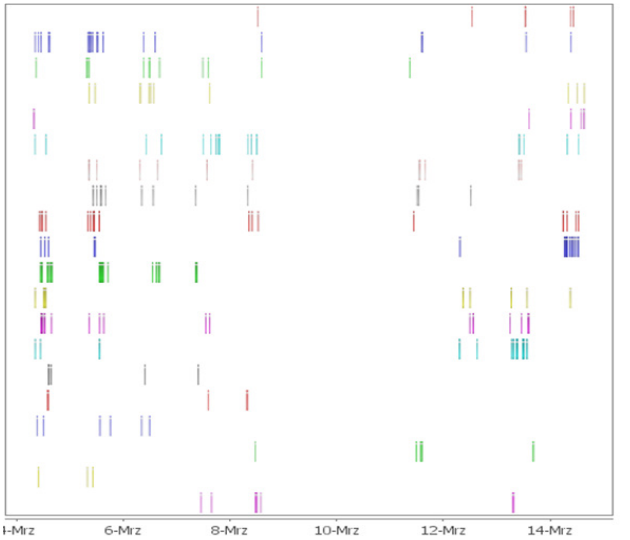

= Geändert
= Neu

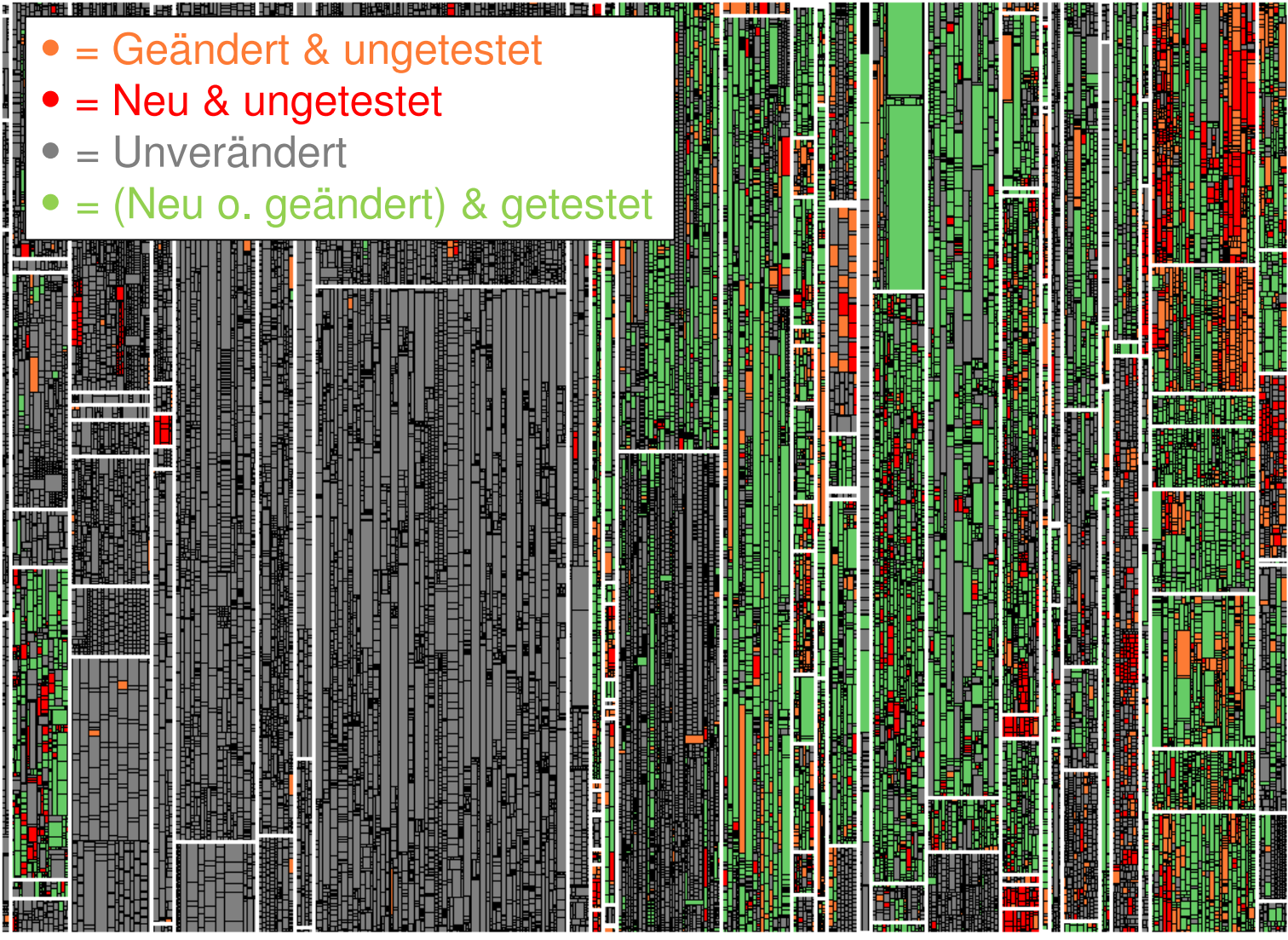

• = Ausgeführt

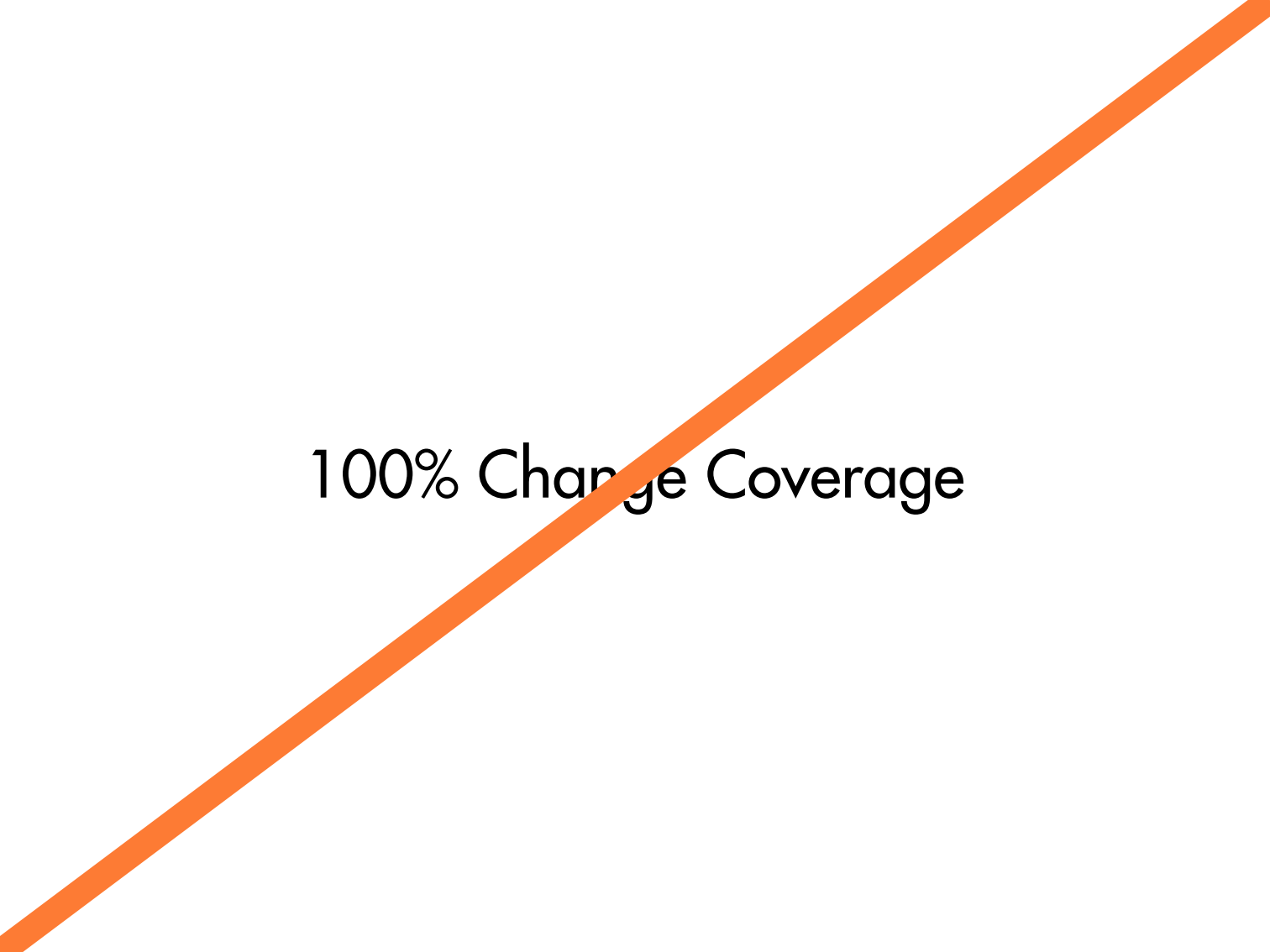

= Geändert & ungetestet
= Neu & ungetestet
= Unverändert
= (Neu o. geändert) & getestet

Fallstudie: Explorativer Test (»Trampeltest«)

- 23 Tester
  - Requirement Engineers
  - Fachbereich
  - Endnutzer
- 2 Wochen
- 307 Test-Sessions
- 8 Systemversionen




= Geändert & ungetestet
= Neu & ungetestet
= Unverändert
= (Neu o. geändert) & getestet

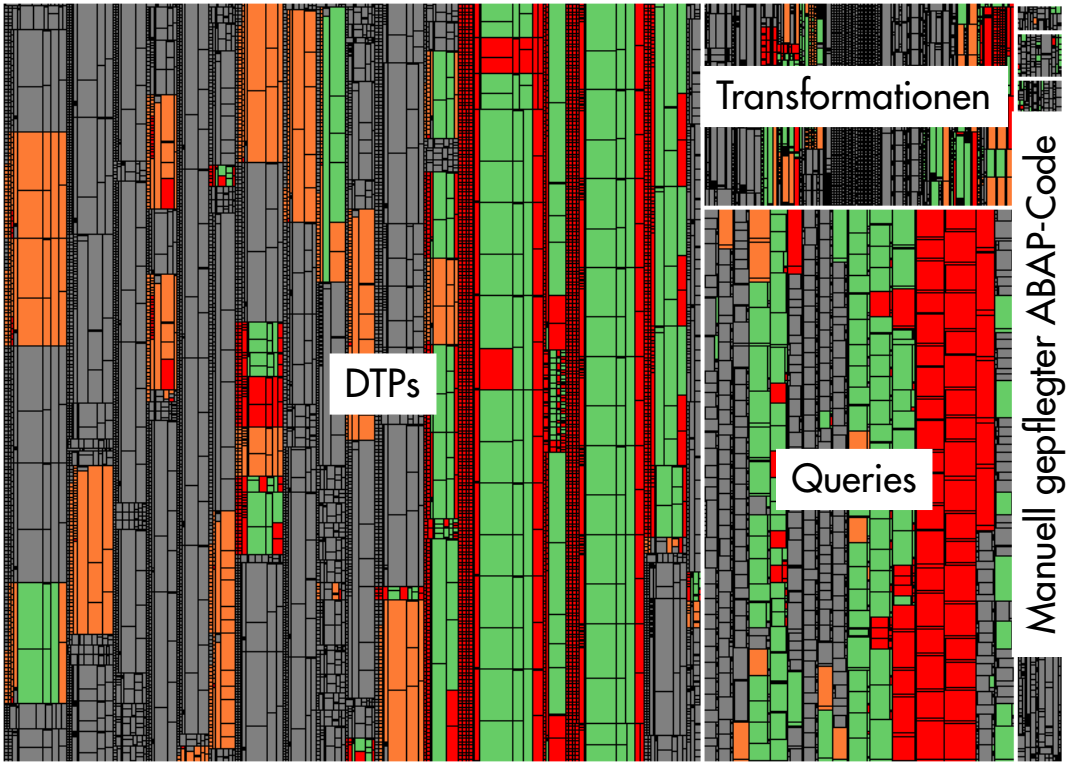

100% Change Coverage

Wie funktioniert das in der SAP-Welt?

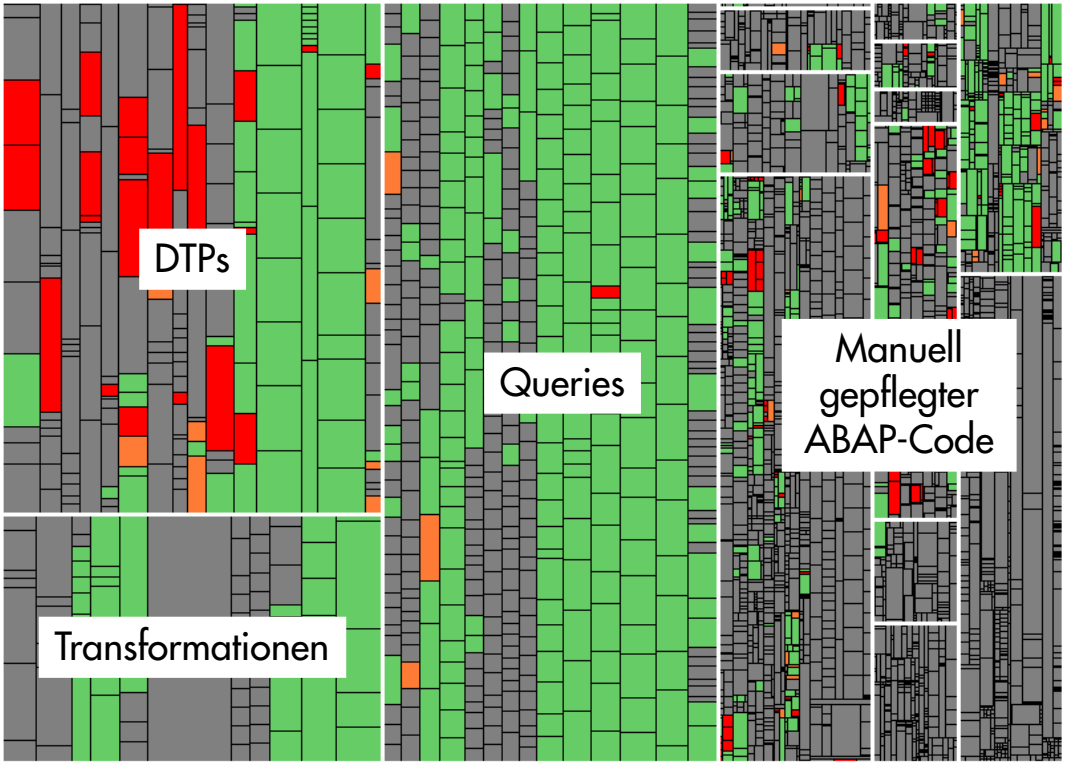
- ABAP: Analoger Ansatz zur Analyse in .NET
- Nächtlicher Export von Code und Ausführungsdaten
- Ausführungsmessung mit Bordmitteln (SCOV)

- Unterschiede in SAP BW
  - Mehr grafische Modellierung als ABAP-Entwicklung
  - ABAP-Code aus Modellen generiert
  - Bezeichner generiert → Code-Ursprung mitführen

Code-Verteilung



Code-Verteilung (skaliert und aggregiert)



Erkenntnisse

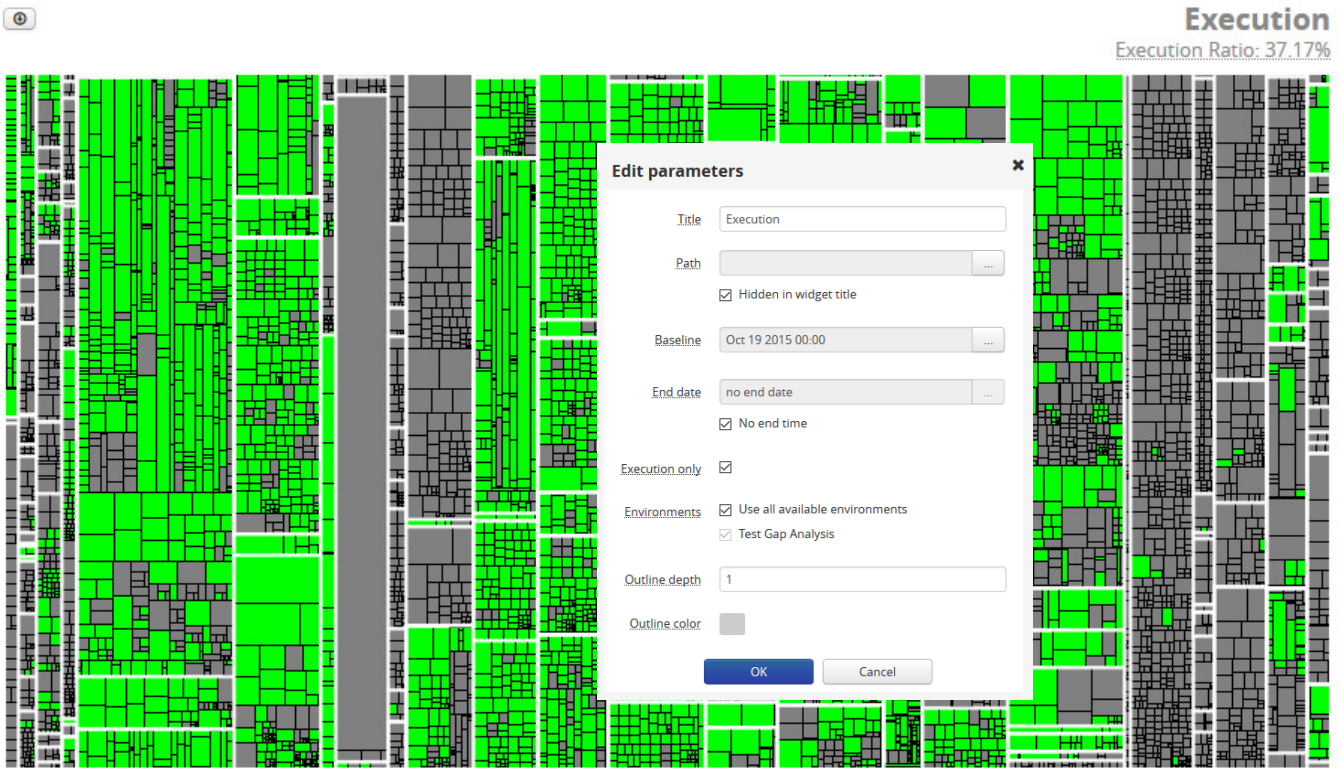
- Code-Änderungen sind guter Fehlerprädiktor
- Test-Gap-Analyse kann ungewollte Produktivsetzung ungetesteter Änderungen verhindern.

- Methode und Tool müssen individualisierbar sein
- Verschiedene Visualisierungen je nach Stakeholder
- Flexible Auswahl von Testumgebungen / Referenzversion erwünscht

Agenda

- Hintergrund
- Analyse und Visualisierung
- Erfahrungen aus der Praxis
- **Ausblick und Diskussion**

Ausblick: Test-Gap-Analyse interaktiv



Software Quality Blog

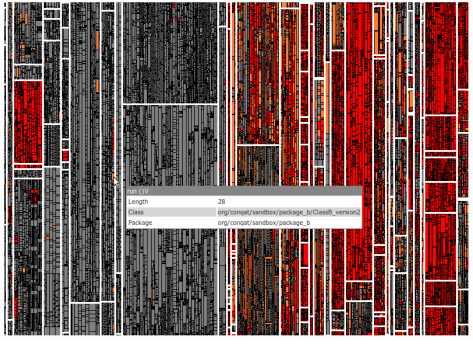
Take control of your testing process!

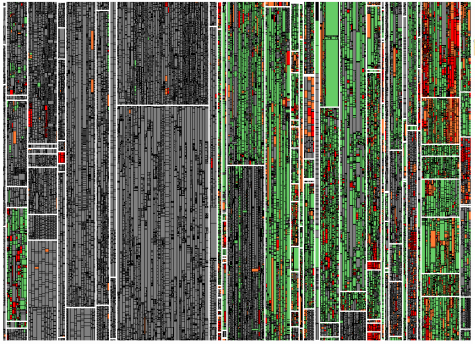
Posted on 10/08/2014 *by* Fabian Streitel

Testing is an integral part of a software product's life-cycle. Some people prefer executing their tests continuously while they develop, others have a separate testing phase before each release. The goal, however, is always the same: finding bugs. The more, the better.

Unfortunately, the time available for testing and for writing new tests is limited. At some point, you have to get on with development, ship your software and be confident it contains no serious flaws. Often, more tests exist than can be run in the available time span. This is especially true if your tests are executed manually, as is the case for many of our customers. Then the question becomes: which tests do I select to find the most bugs before the release?

Research h likely ("Did W errors thar ure that all ch his is true wheth hole test suit





https://www.cqse.eu/en/blog/take-control-of-your-testing-process/

Software Quality Blog

Testing Changes in SAP BW Applications

Posted on 04/29/2015 *by* Dr. Andreas Göb

As my colleague Fabian explained a few weeks ago, a combination of change detection and execution logging can substantially increase transparency regarding which recent changes of a software system have actually been covered by the testing process. I will not repeat all the details of the Test Gap Analysis approach here, but instead just summarize the core idea: Untested new or changed
informati
em. Therefore it makes sense to use
fy those changed but untested areas.

Several ti
In the ma
from Pyth
containin
may prov
Analysis i
ecific project

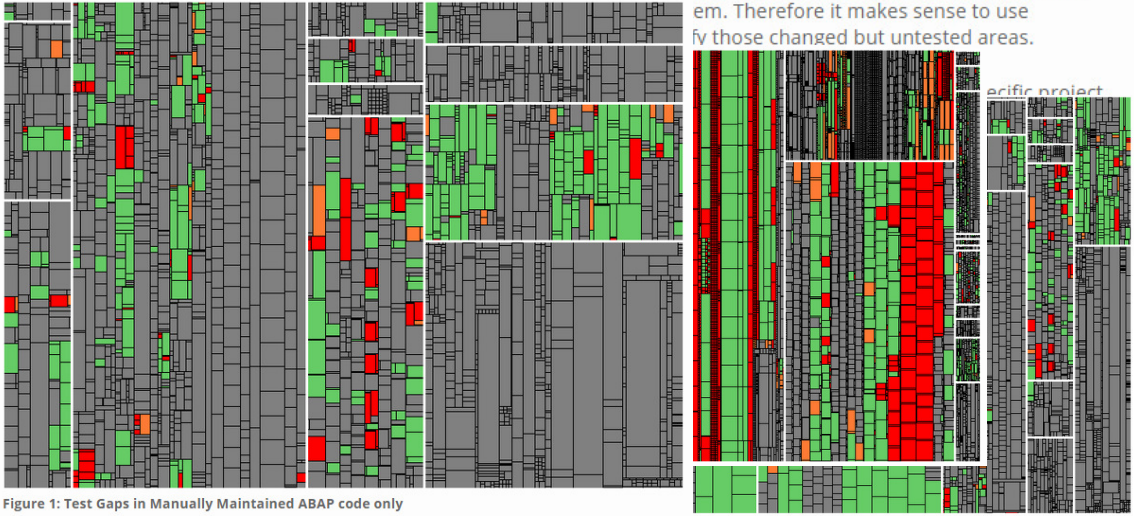


Figure 1: Test Gaps in Manually Maintained ABAP code only

https://www.cqse.eu/en/blog/testing-changes-in-sap-bw/

Kontakt

Ich freue mich auf Diskussionen, auch im Anschluss oder morgen!

Dr. Andreas Göb · goeb@cqse.eu · +49 176 101 552 25

@a_goeb
www.cqse.eu/en/blog

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München

CQSE
Continuous Quality in Software Engineering