Continuous Quality Control

Qualität trotz immer kürzerer Releasezyklen

JFS 2025

CQSE | Dr. Tobias Röhm

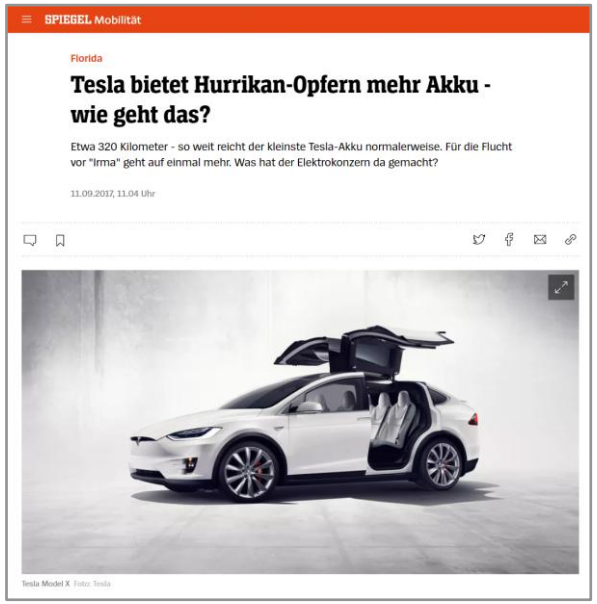
Dr. Tobias Röhm

Freelancer — 2004 - 2010

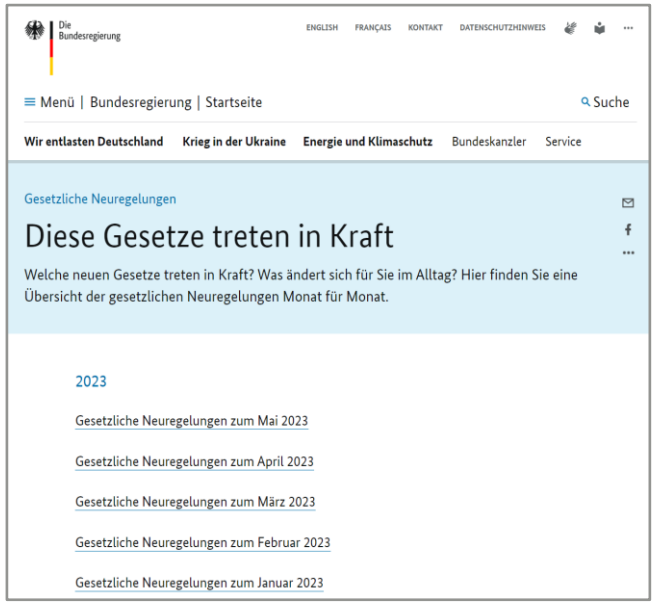
TUM — 2010 - 2015

CQSE — 2015 - Heute

Kontext



Ereignisbasierte
Over-the-Air-Updates

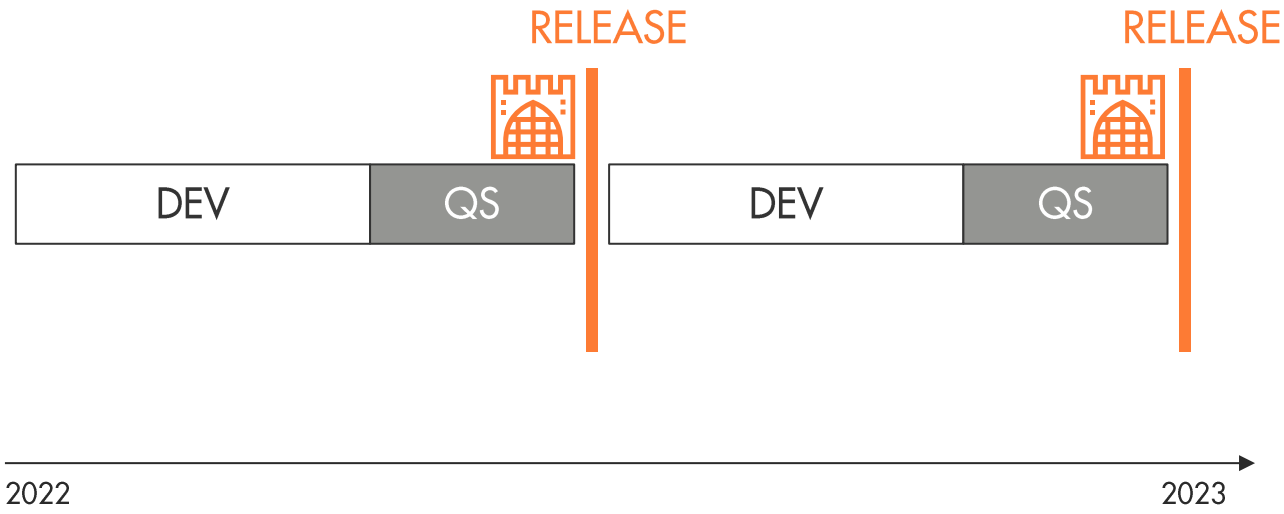


Viele, teilweise rückwirkende
Gesetzesänderungen

Zeitkritische Security-Patches

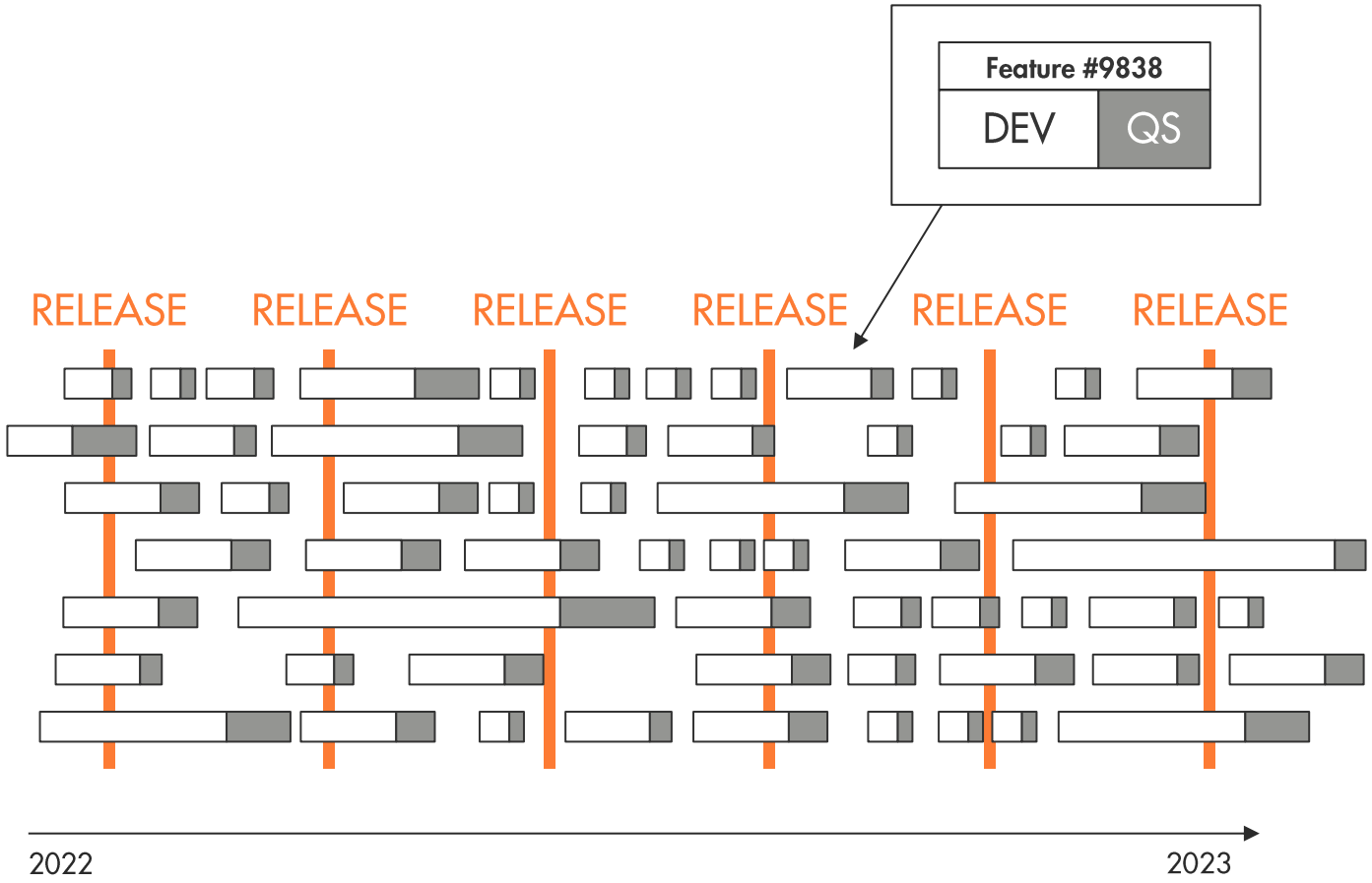
→ Software-Updates müssen jederzeit möglich sein
→ Software muss jederzeit ein releasefähiges Qualitätsniveau aufweisen

Problem: Traditionelle QS-Prozesse



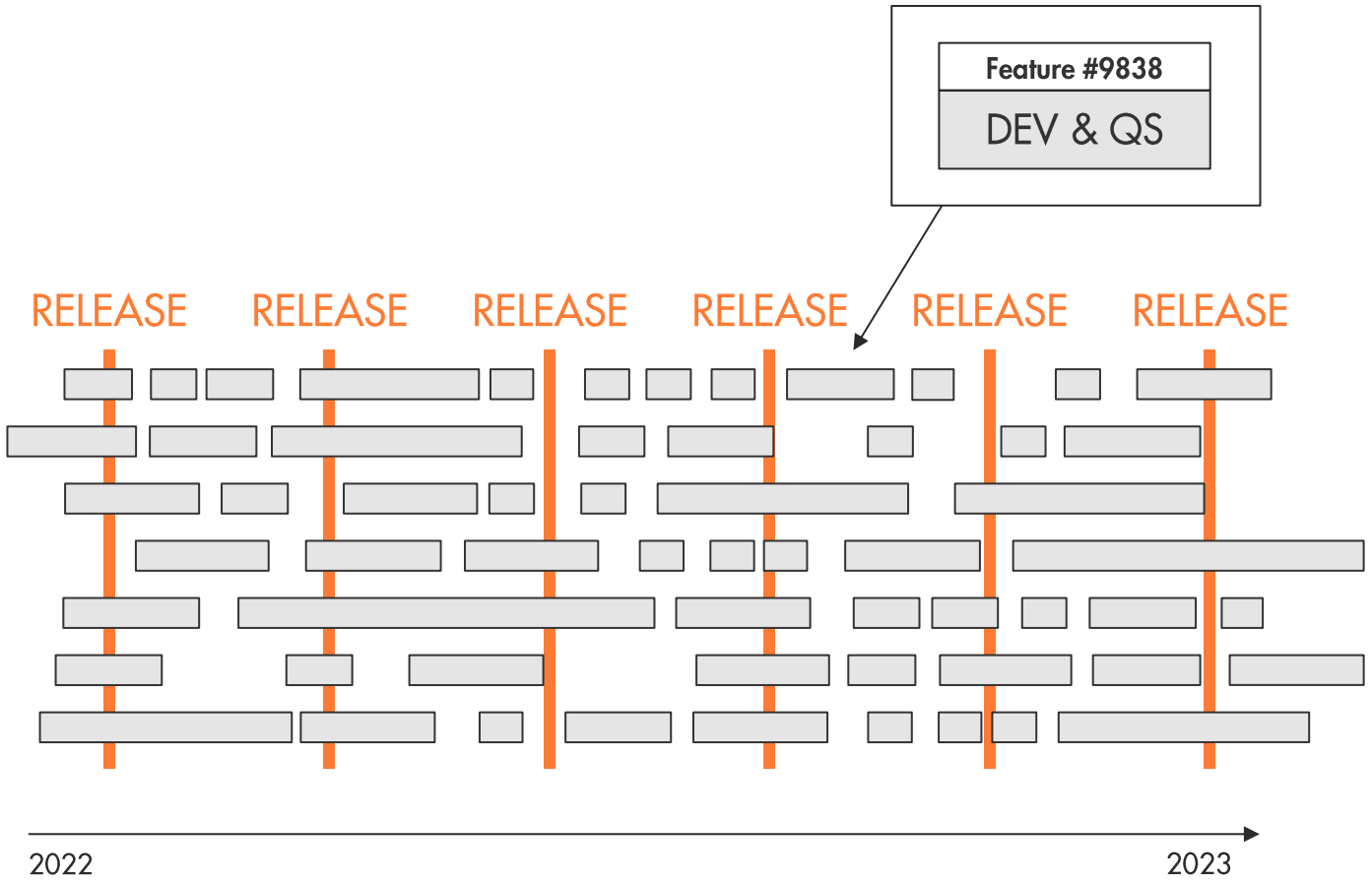
→ Traditionelle QS-Prozesse sind (zu) langsam
→ Software ist NICHT jederzeit auf einem releasefähigem Qualitätsniveau

Lösung: Continuous Quality Control (CQC)



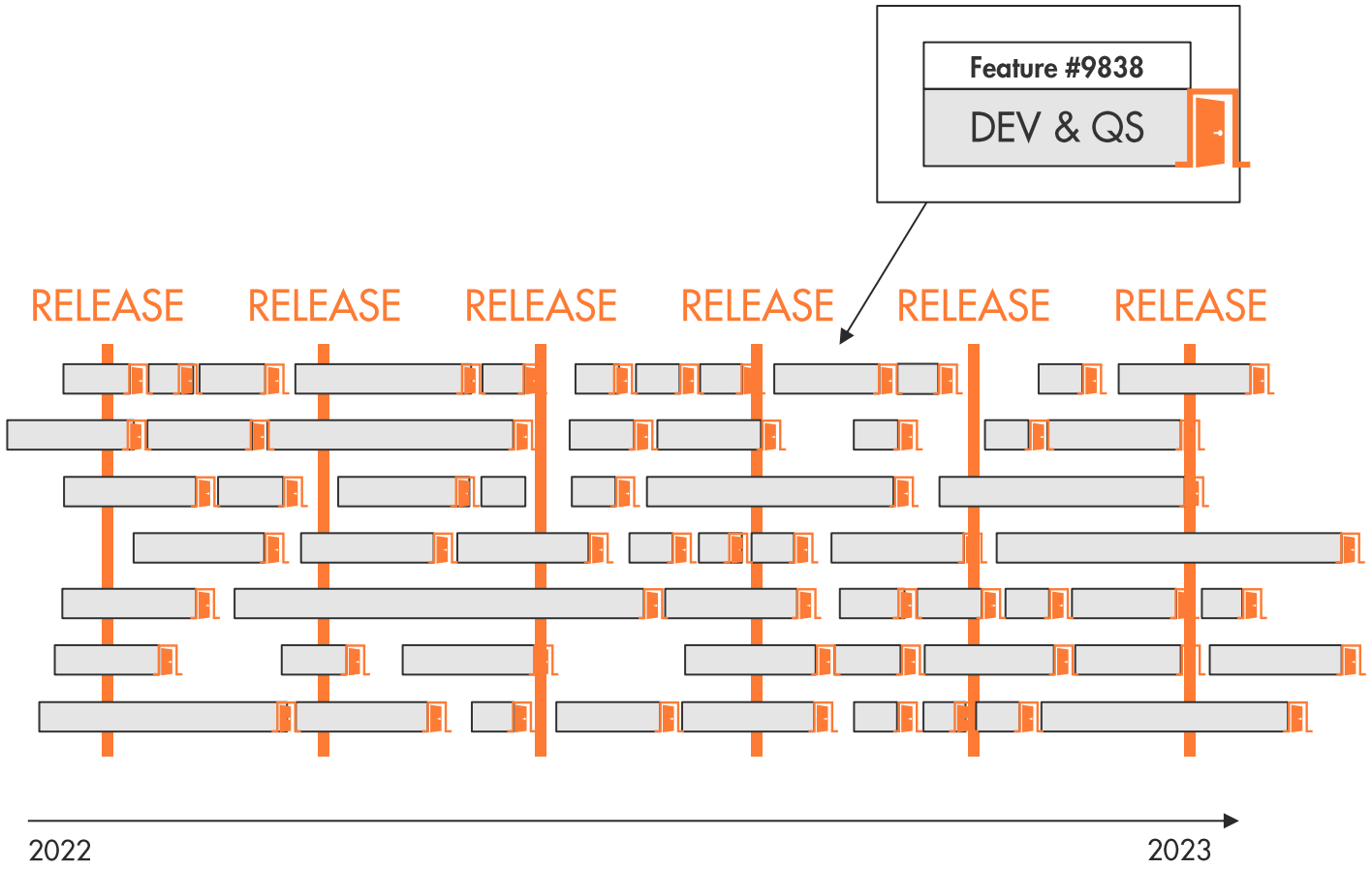
→ Qualitätssicherung auf Ticketebene um zu parallelisieren und zu beschleunigen

Lösung: Continuous Quality Control (CQC) II



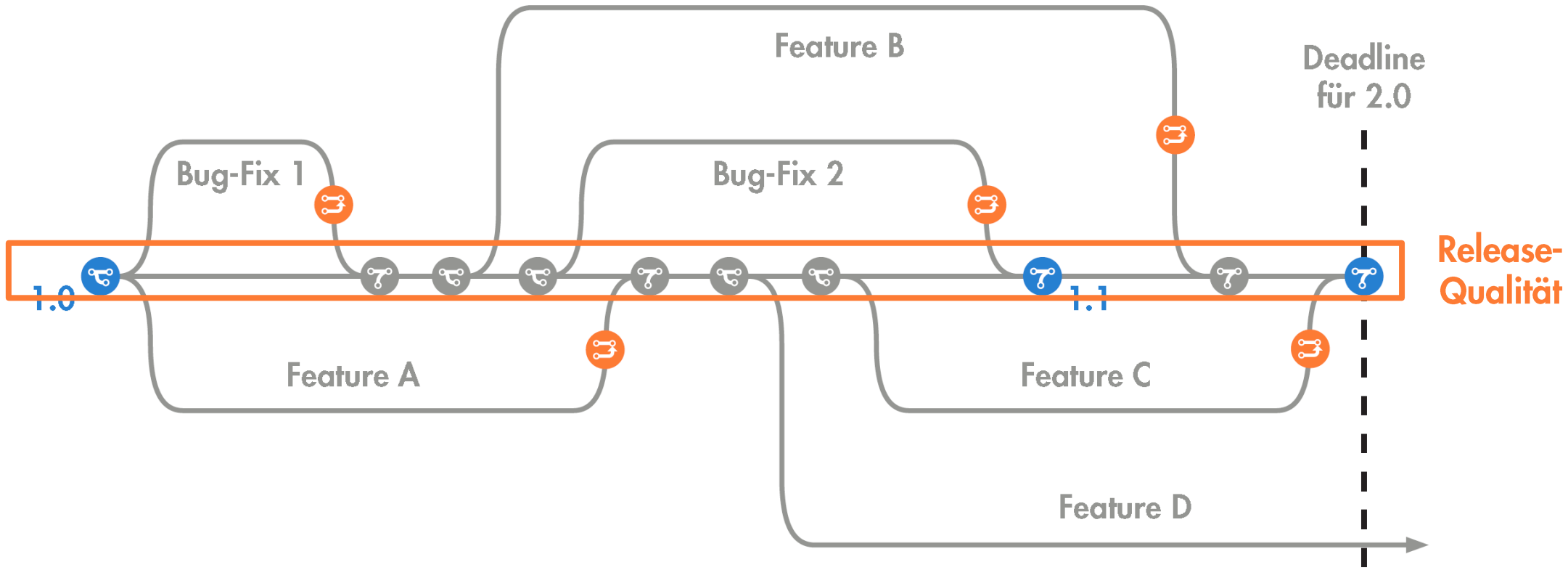
→ Verschmelzung von Entwicklung und QS um zu beschleunigen

Lösung: Continuous Quality Control (CQC) III



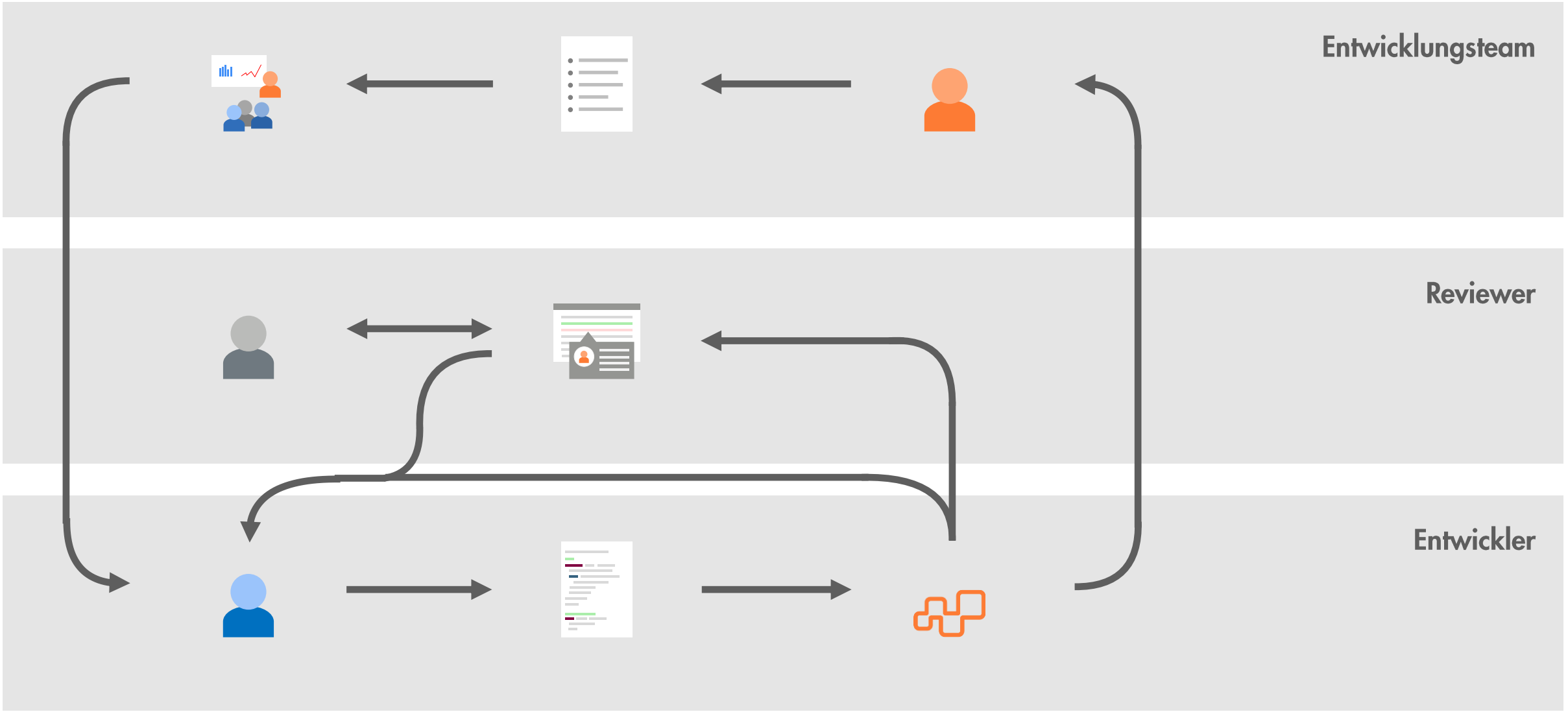
→ Qualitätstürchen als frühe, leichtgewichtige Qualitätsprüfung
→ Qualität ist jederzeit auf releasefähigem Niveau

Implementierung durch Feature Branches in Git



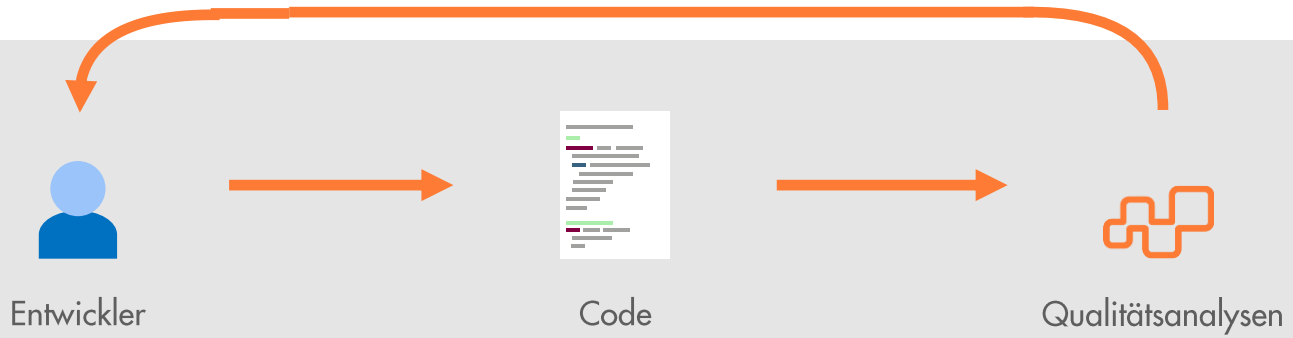
→ Merge Request als »Qualitätstürchen«

Agenda: Continuous Quality Control-Feedbackschleifen



Feedbackschleife 1:
Statische Codeanalyse hilft Entwicklern, Fehler & Q-Defizite zu vermeiden

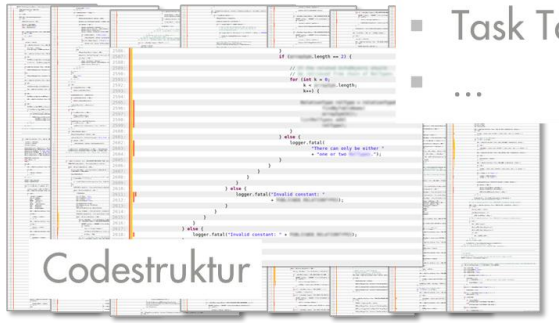
Feedbackschleife 1: Statische Codeanalyse hilft Entwicklern, Fehler & Q-Defizite zu vermeiden

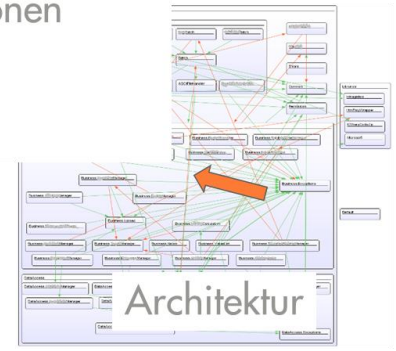


Entwickler-Feedback
Fehler & Q-Defizite vermeiden
durch schnelles Feedback

Übersicht Statische Codeanalyse

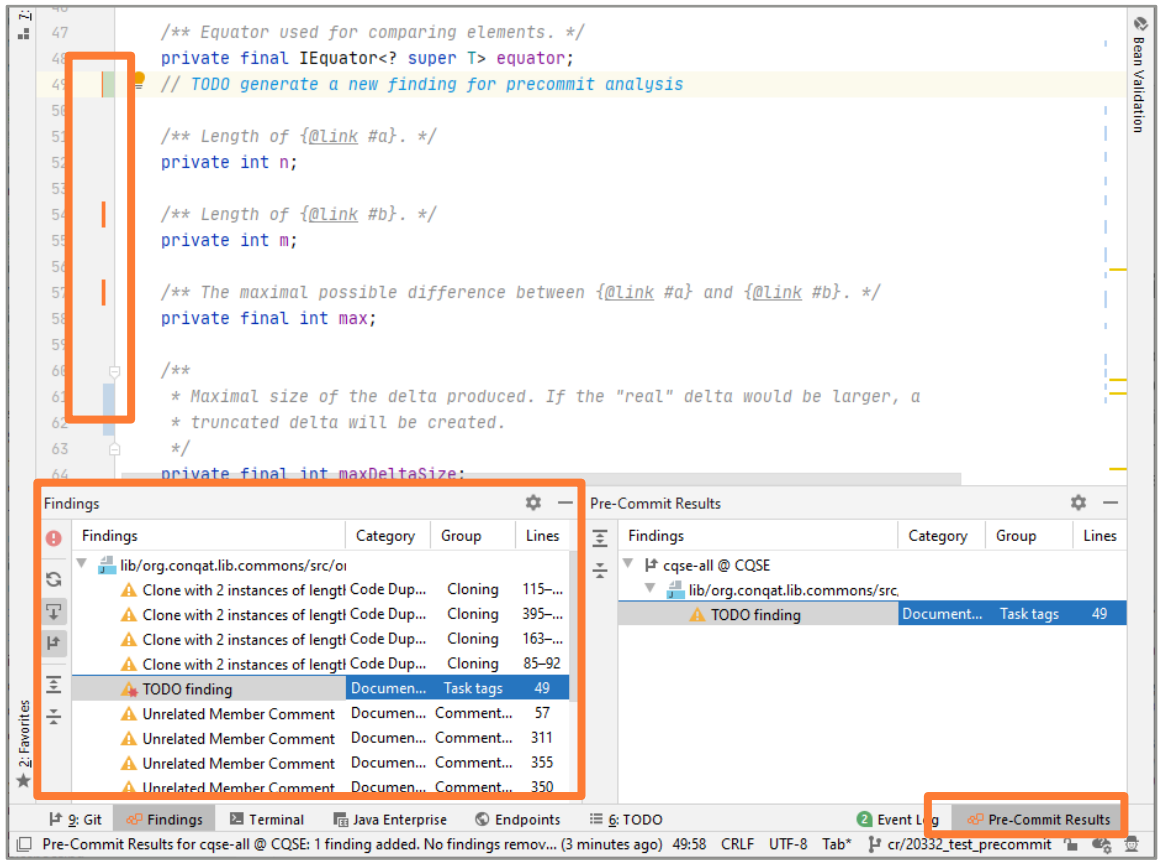
- Kommentarvollständigkeit
- Richtlinienverletzungen
- Auskommentierter Code
- Namenskonventionen
- Task Tags
- ...





→ Statische Codeanalyse schafft Transparenz bezüglich Code- & Architekturqualität

IDE-Integration & Precommit-Analyse

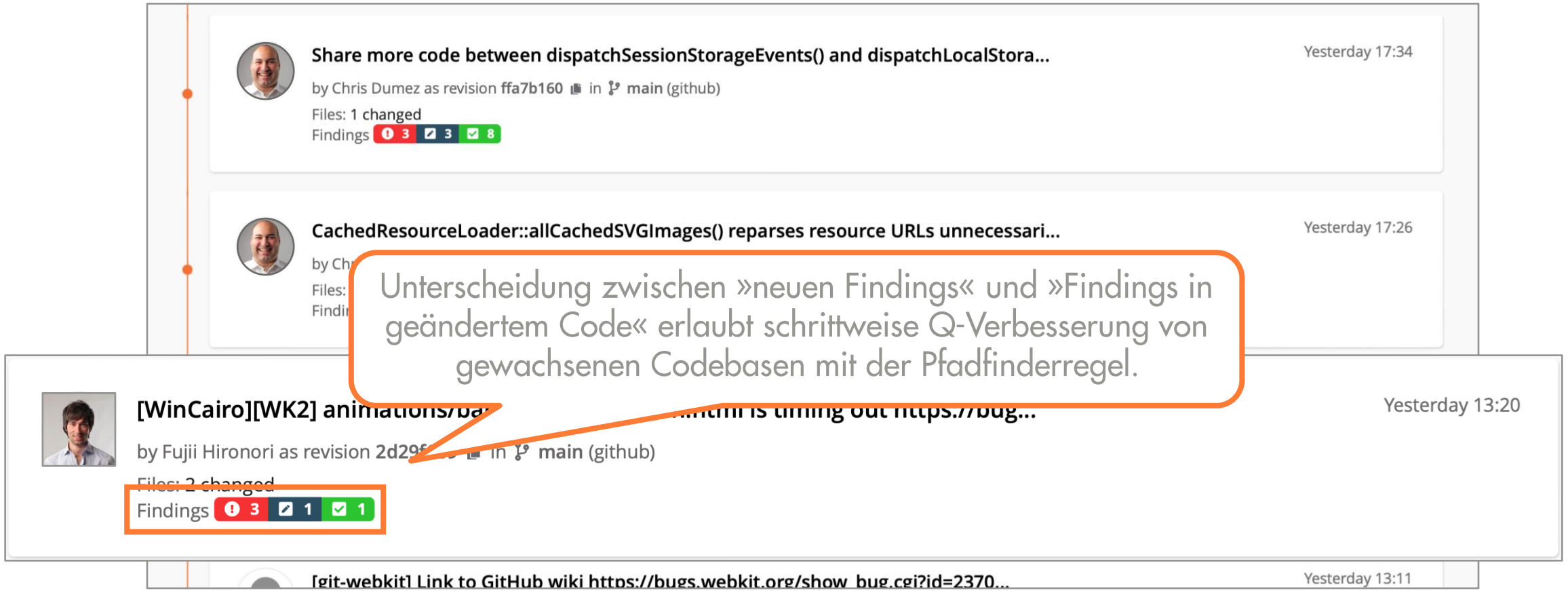


Precommit-Analyse ermöglicht Q-Feedback bereits vor VCS-Commit

Inkrementelle Analyse erlaubt schnelle Ergebnisse auch für komplexe Analysen

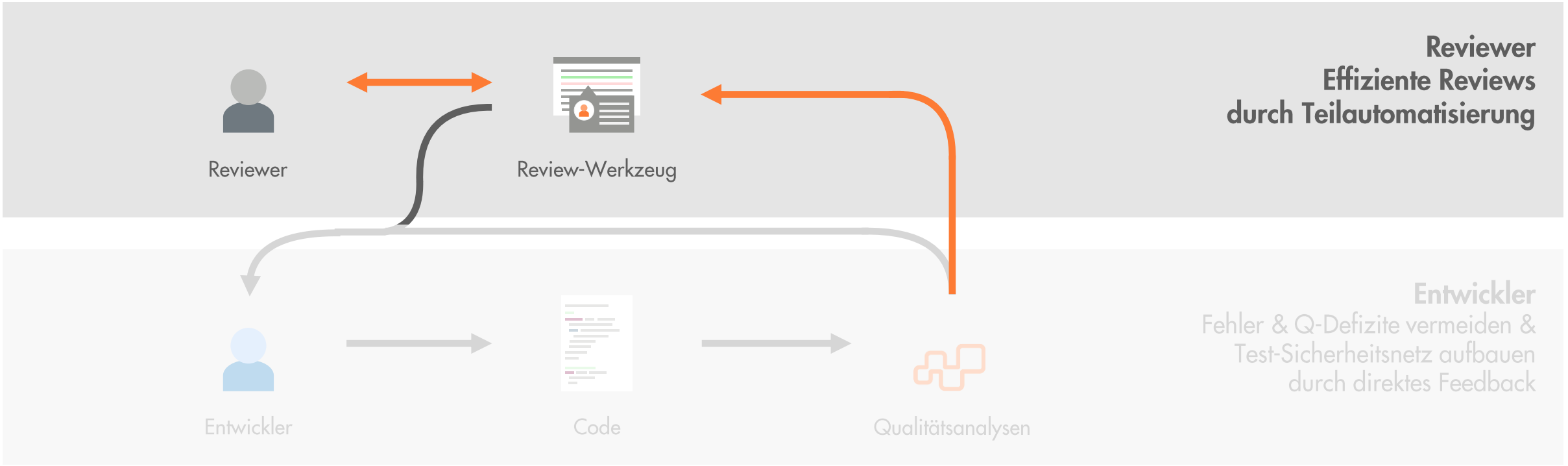
→ Feedback aus statischer Codeanalyse ist schnell, fokussiert und integriert nutzbar

Findings-Churn & Qualitätsziele für gewachsene Systeme

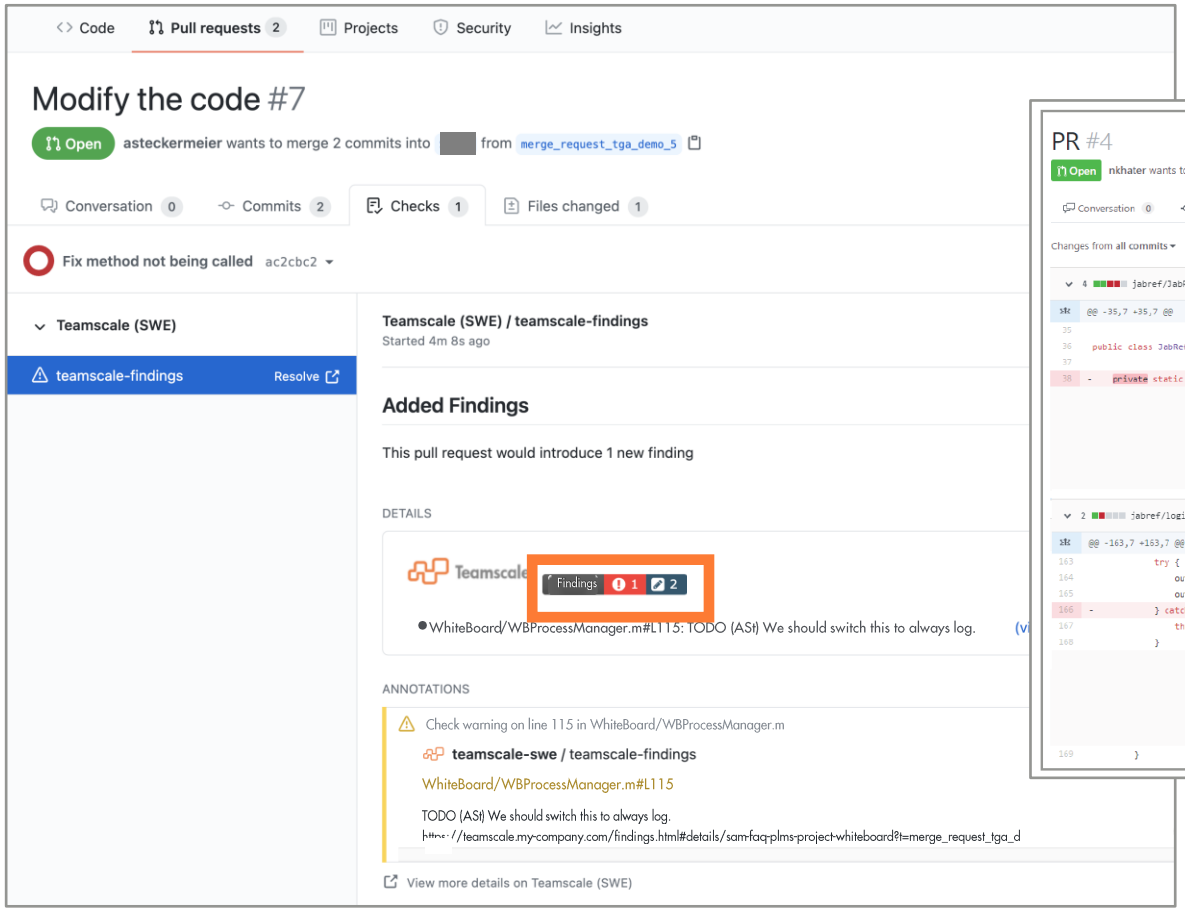


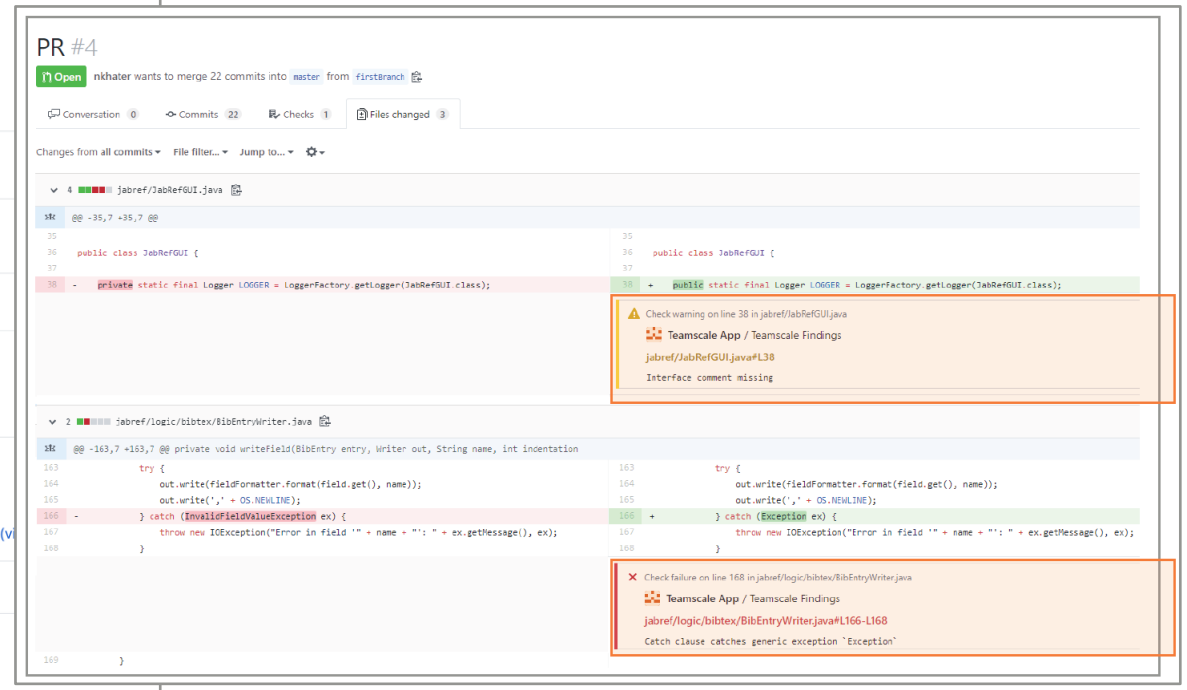
Feedbackschleife 2:
Statische Codeanalyse und Test-Gap-Analyse ermöglichen Reviewern effiziente Code Reviews

Feedbackschleife 2: Stat. Codeanalyse und Test-Gap-Analyse ermöglichen Reviewern effiziente Code Reviews



Annotation von Findings aus statischer Codeanalyse an Merge/Pull Requests

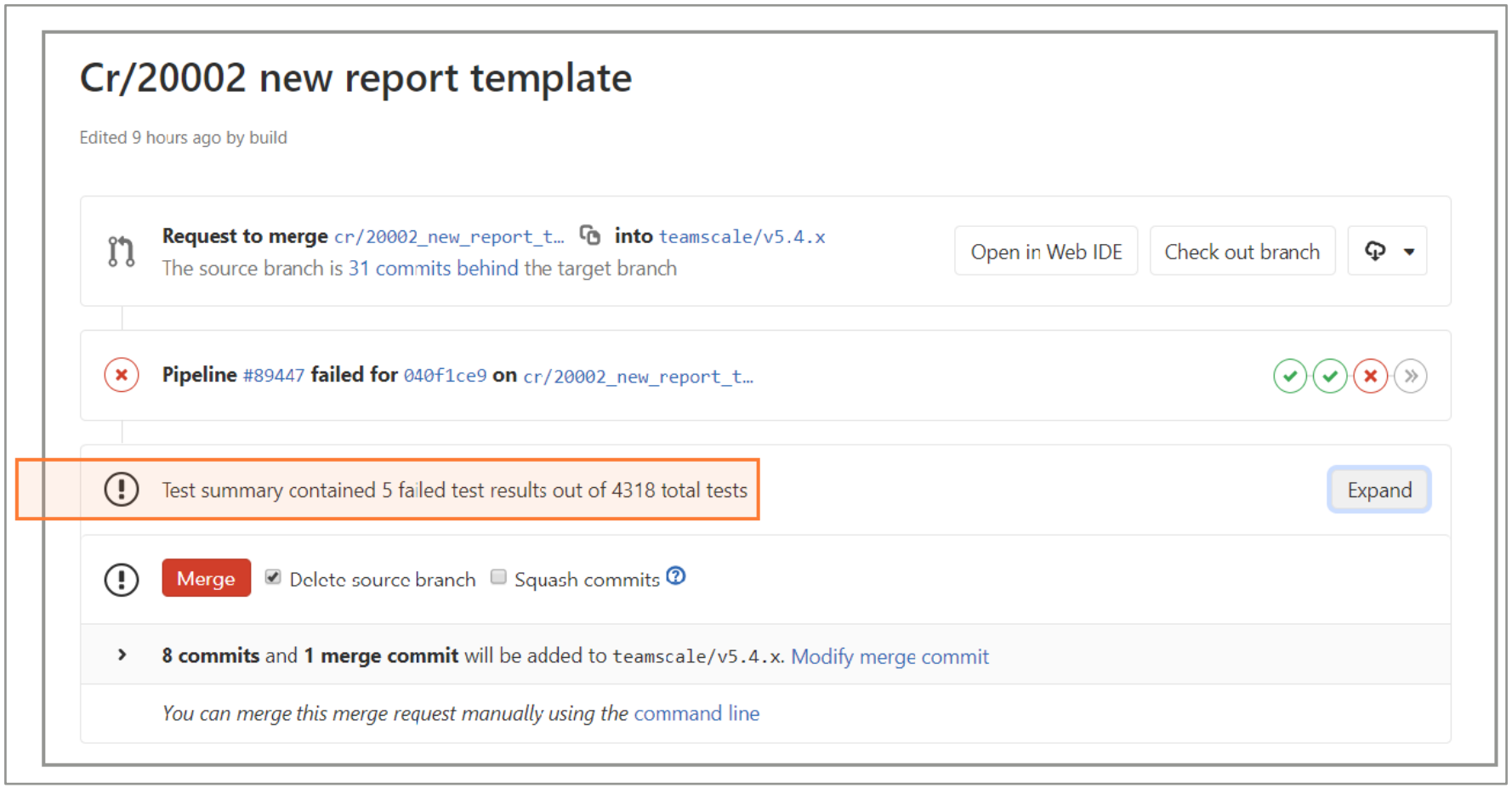




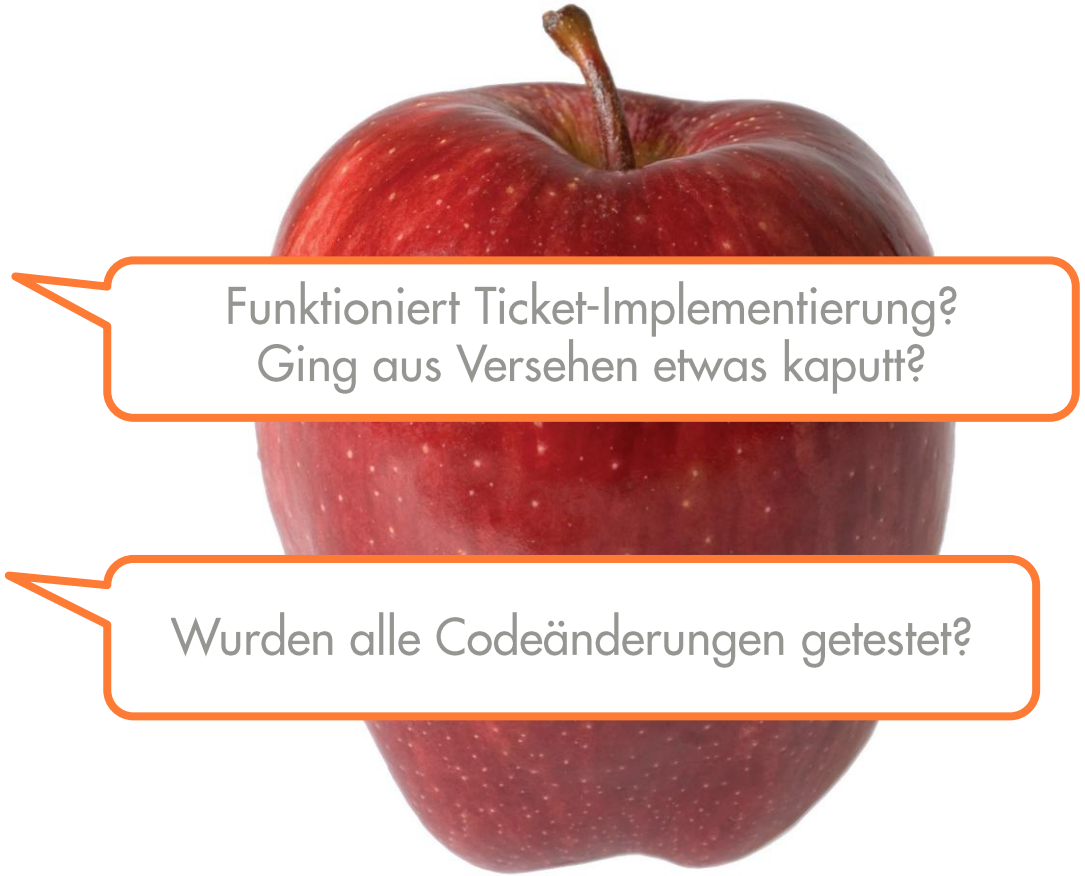
Reviewer/Test(er)

Anwendung

Annotation von Testergebnissen an Merge/Pull Requests

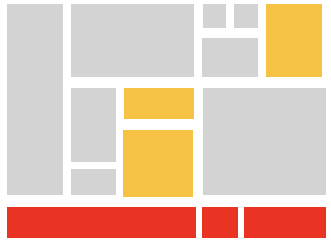


Reviewer/Test(er)

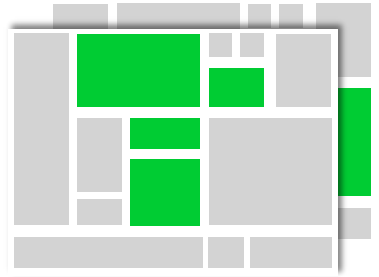


Anwendung

Test-Gap-Analyse

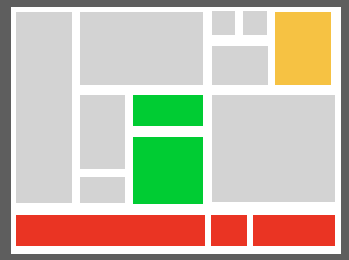


Codeänderungen



Testcoverage

Teamscale



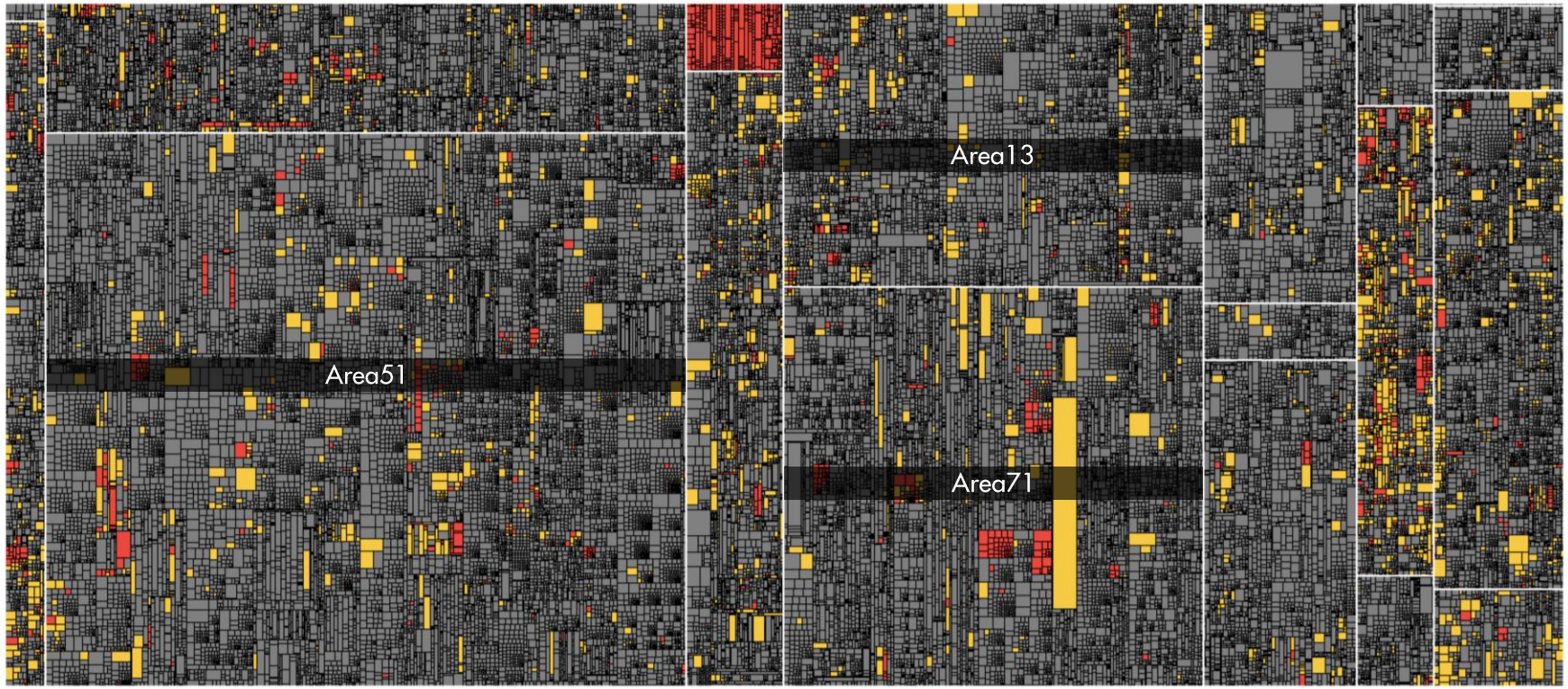
Test Gaps =
Ungetestete Codeänderungen

Test-Gap-Analyse identifiziert
ungetestete Codeänderungen.
Diese haben ein hohes Fehlerrisiko.

Did we test our changes? Assessing alignment between tests and development in practice. (Eder et al. 2013)

Codeänderungen

Churn
Jul 10 2022 - Aug 10 2022



Methoden wurden…

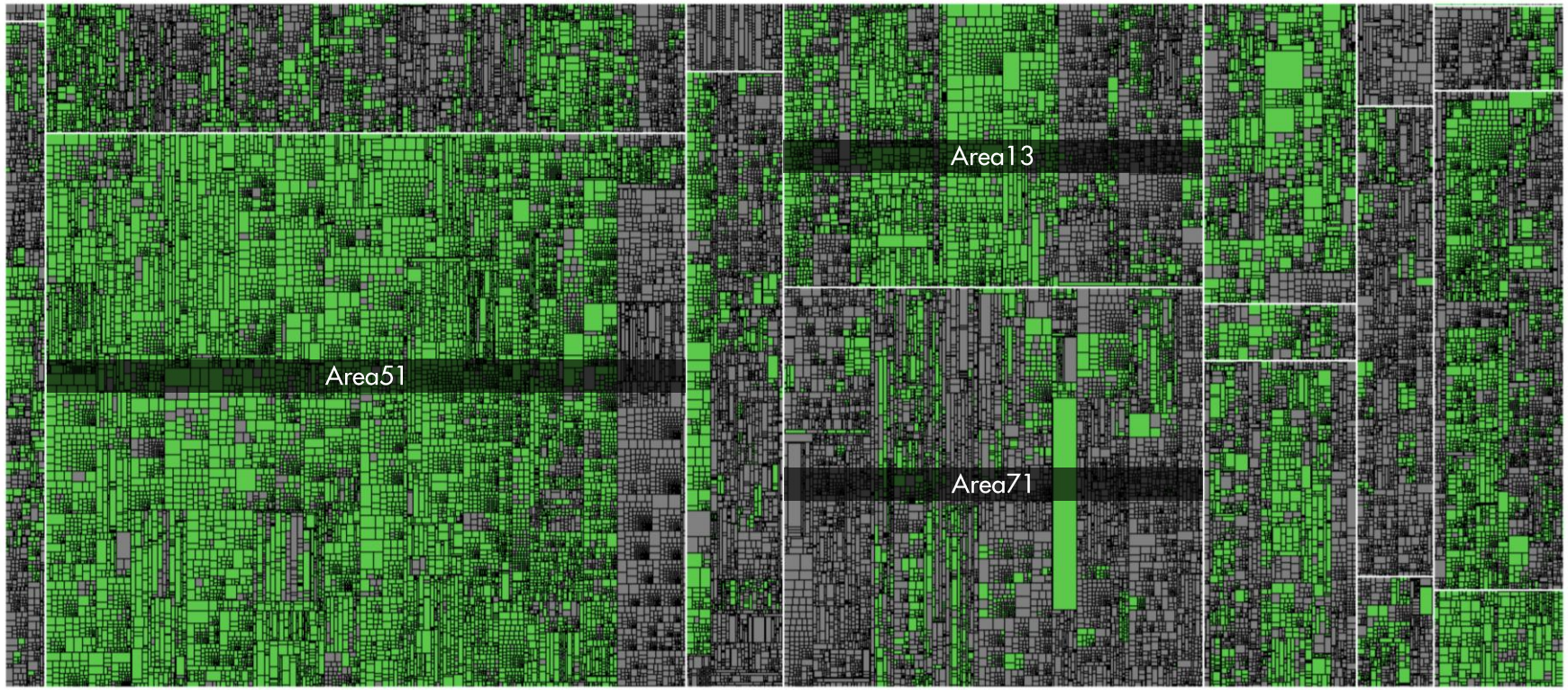
… nicht geändert

… hinzugefügt

… geändert

Testcoverage

Function Execution
Jul 10 2022 - Aug 10 2022 | Execution Ratio: 43.79%

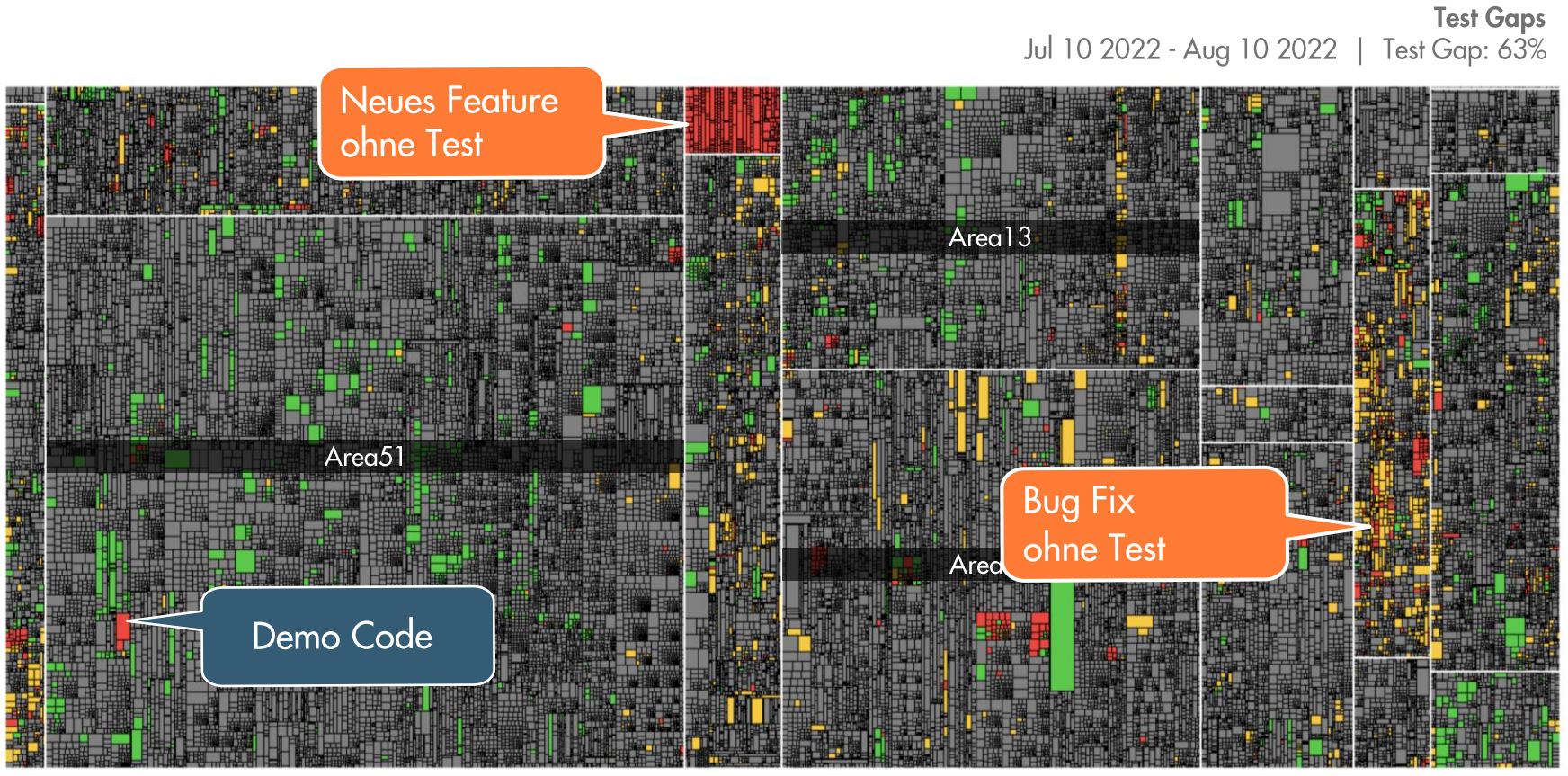


Methoden wurden…

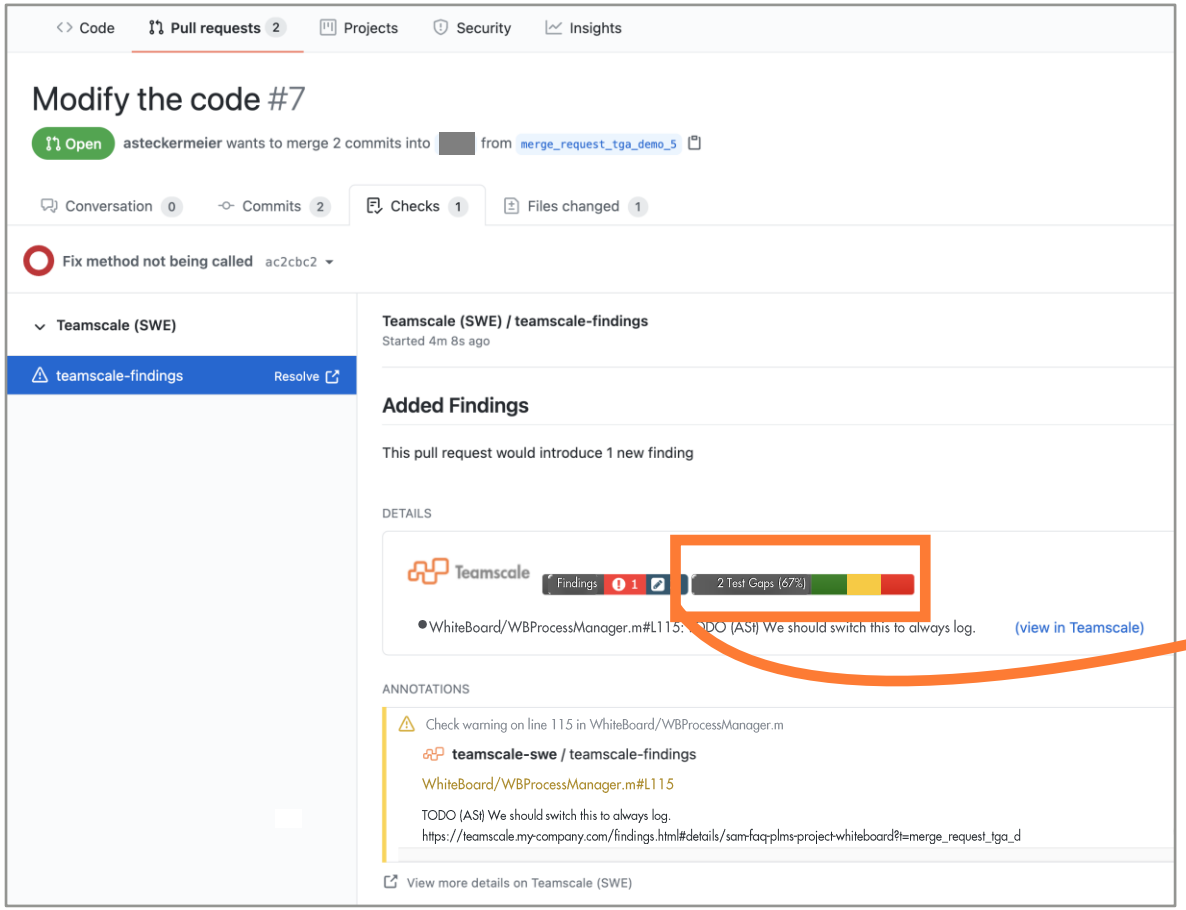
… nicht getestet

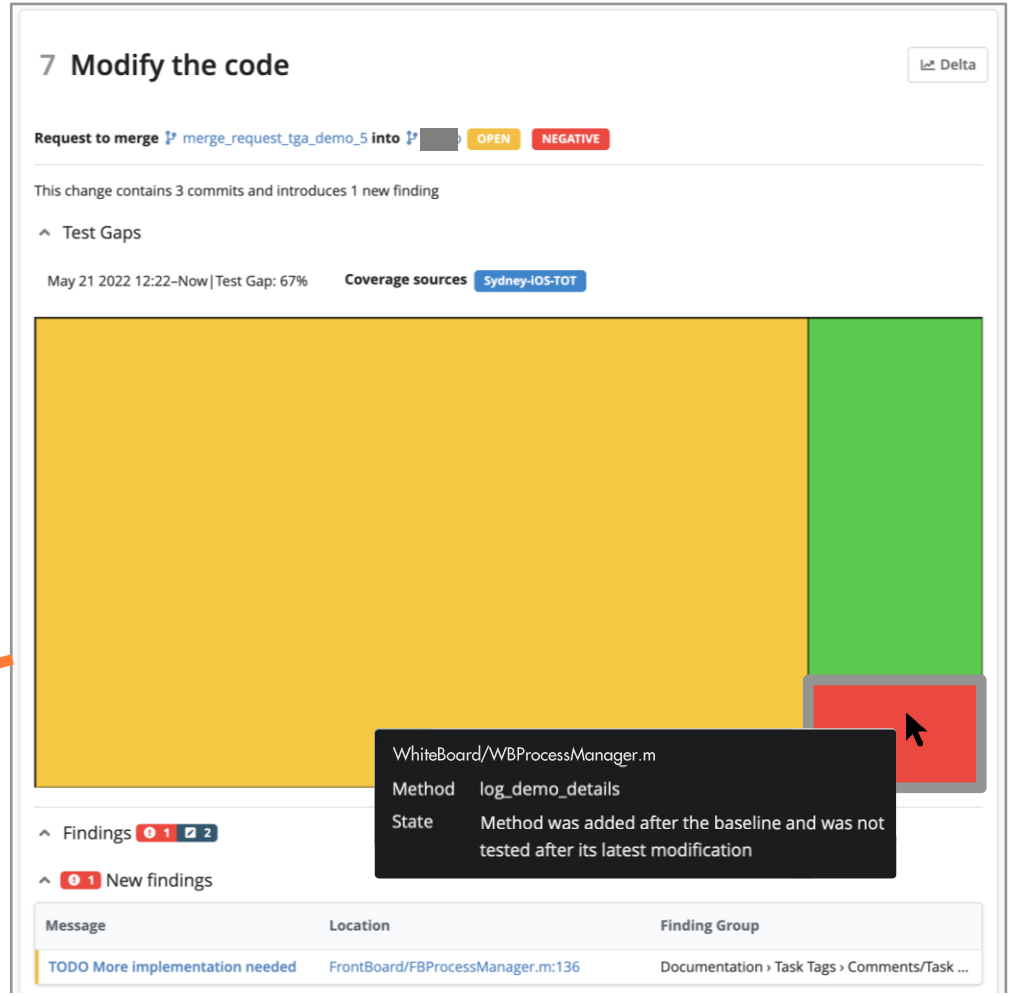
… im Test ausgeführt

Test Gaps (= geänderter, ungetesteter Code)

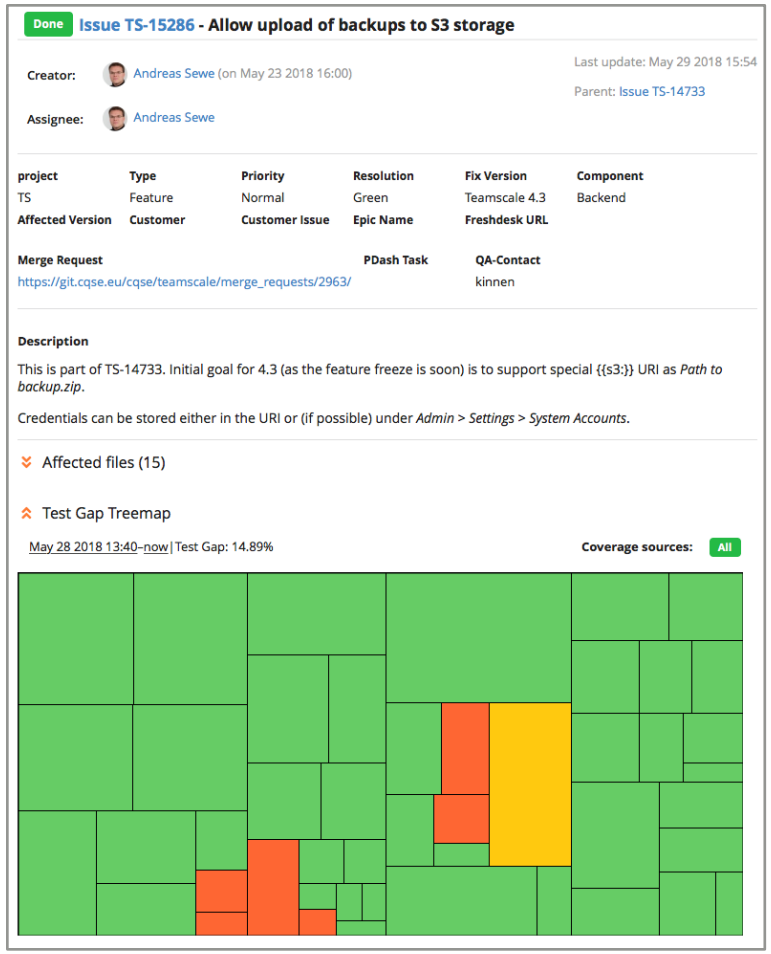


Annotation von Test Gaps an Merge/Pull Requests





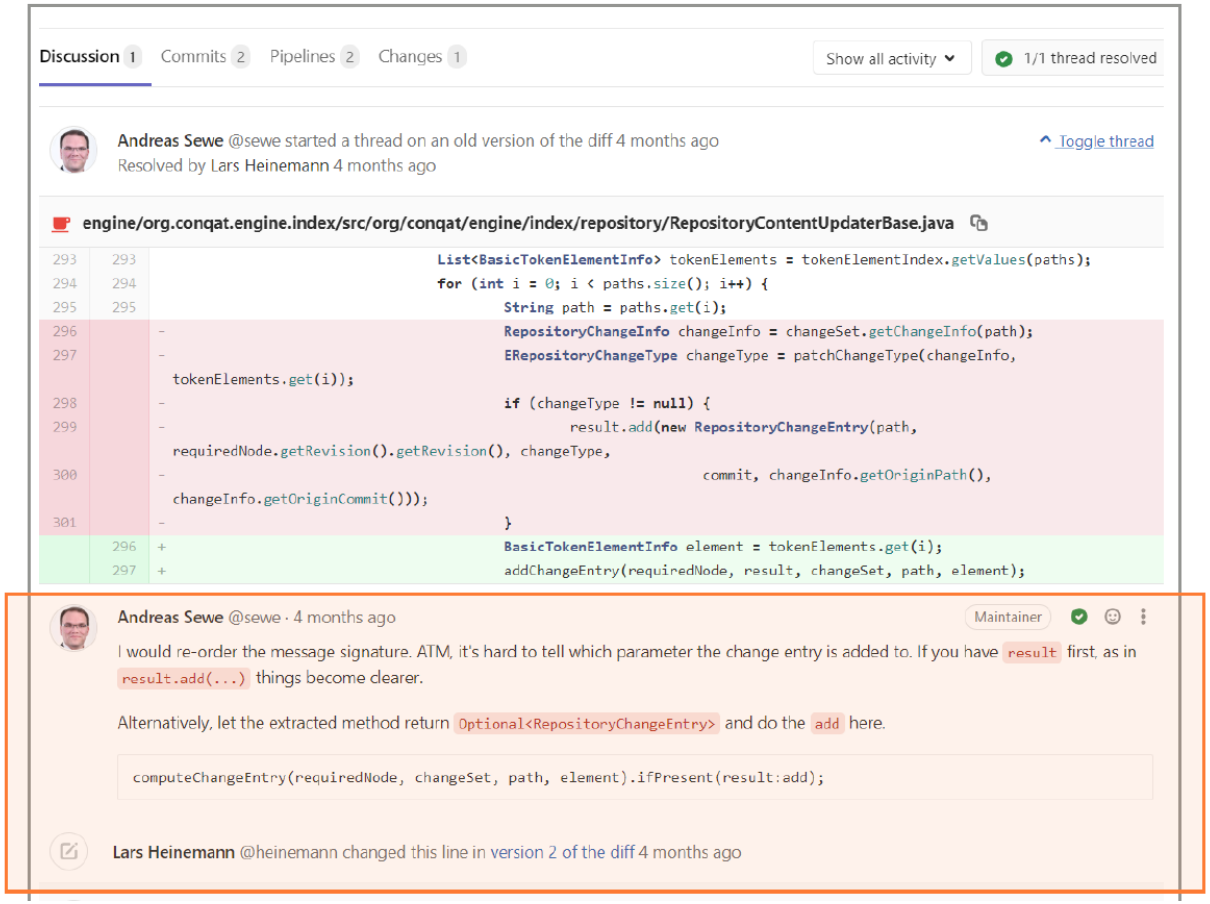
Test Gaps je Ticket



→ Test-Gap-Analyse schafft Transparenz über Test Gaps

→ Test-Gap-Analyse unterstützt beim effizienten Aufbau von Test-Sicherheitsnetz

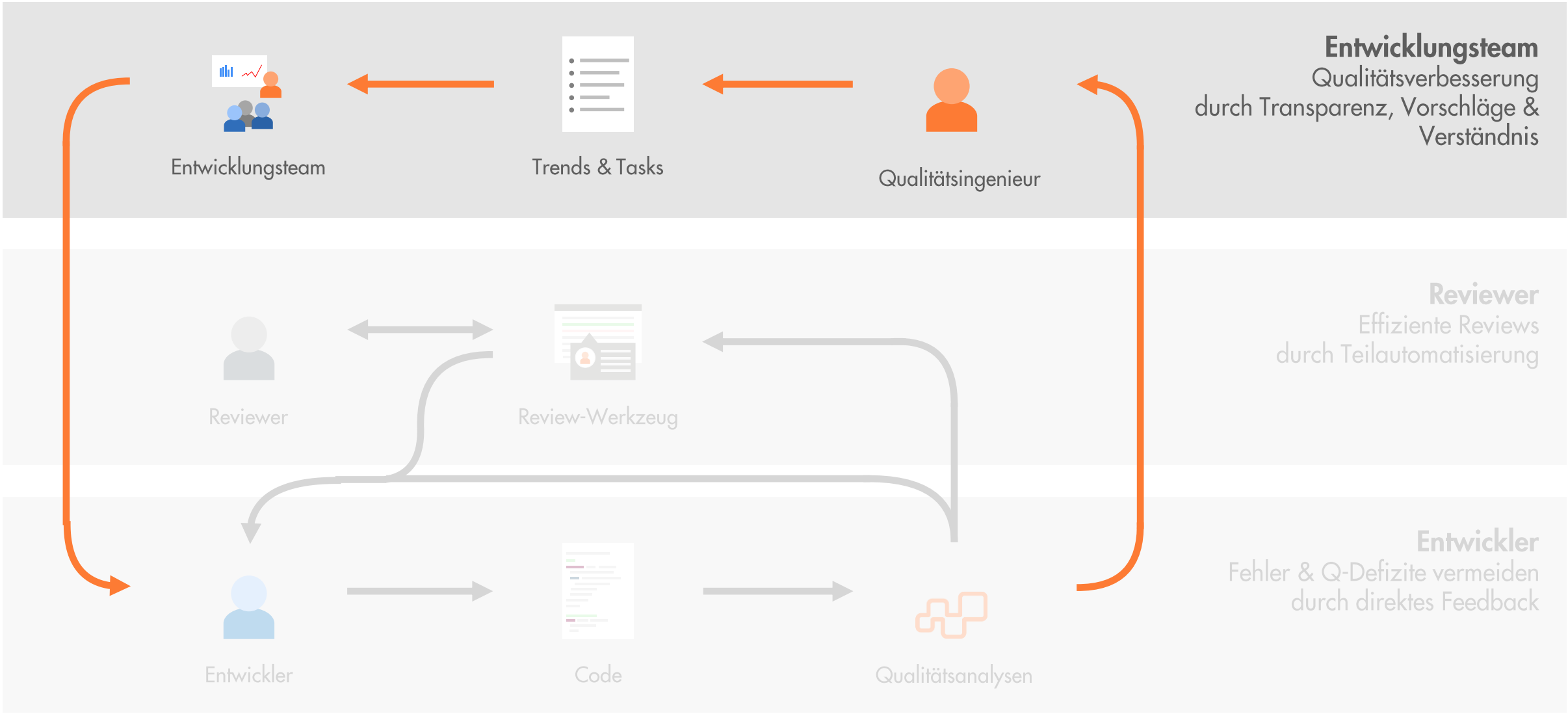
Code Review-Kommentar in GitLab



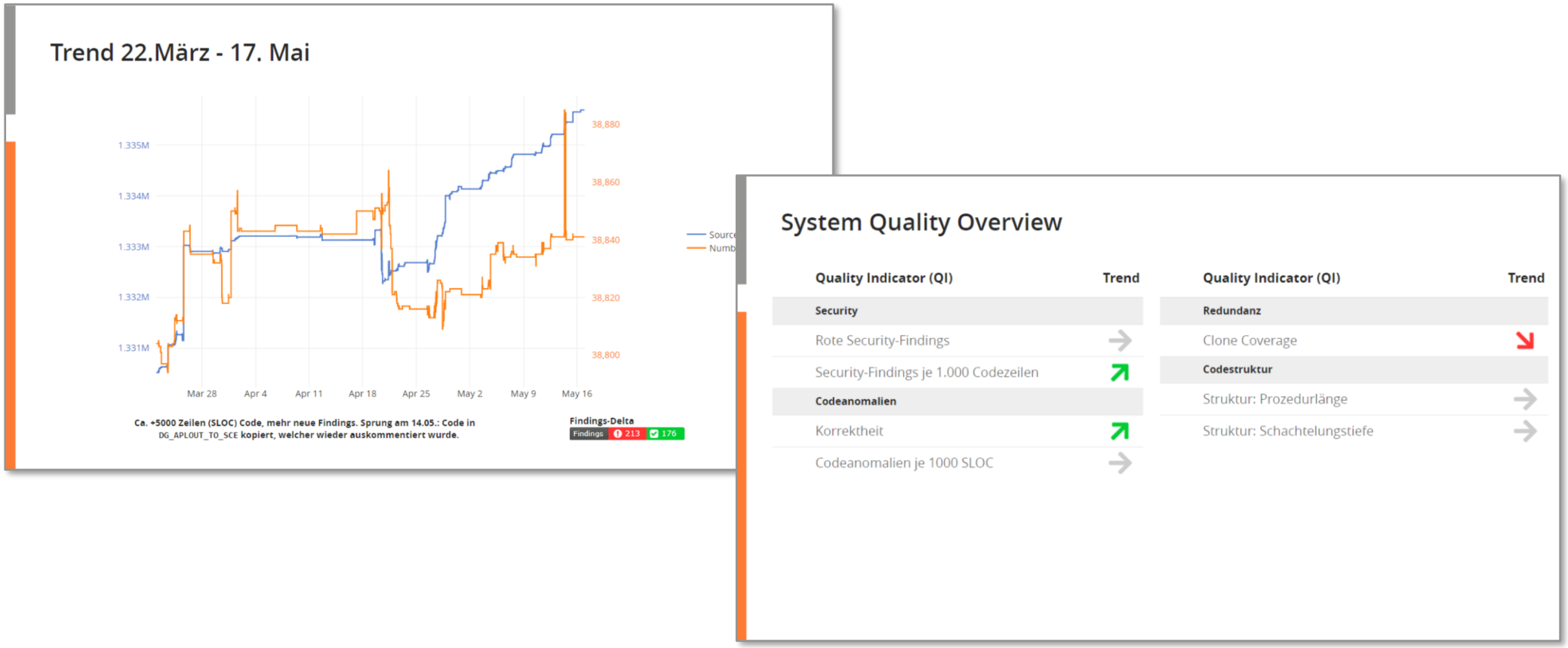
→ Hohe Eingangsqualität erlaubt effiziente, menschliche Code Reviews

Feedbackschleife 3:
Q-Retrospektiven unterstützen Entwicklungsteams bei Qualitätssicherung & -verbesserung

Feedbackschleife 3: Q-Retrospektiven unterstützen Entwicklungsteams bei Qualitätssicherung & -verbesserung

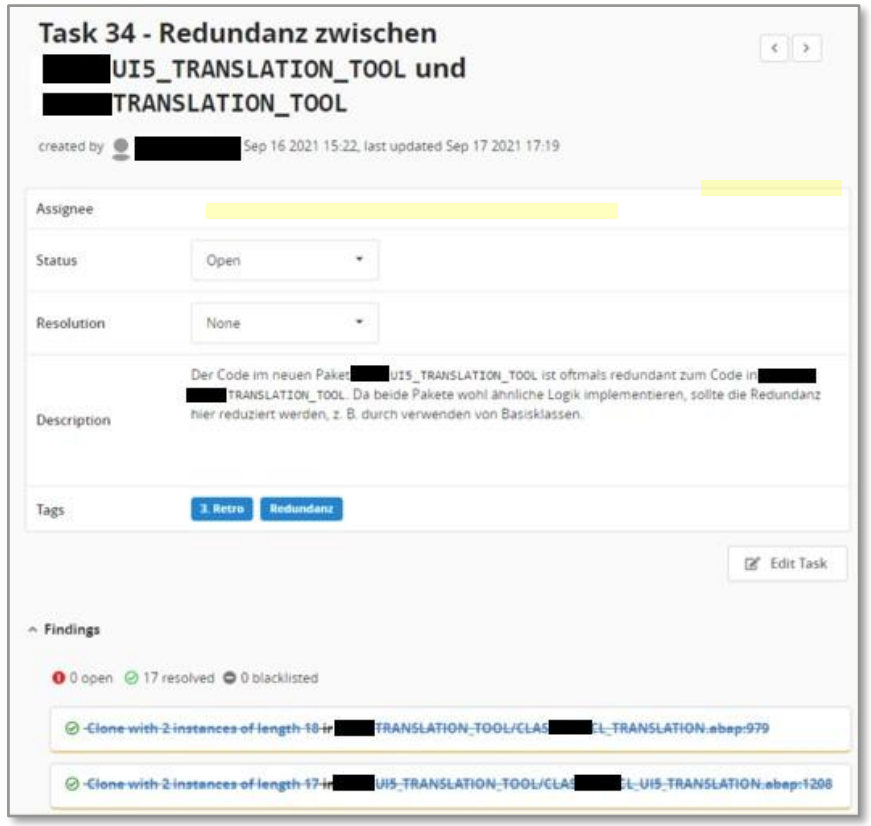


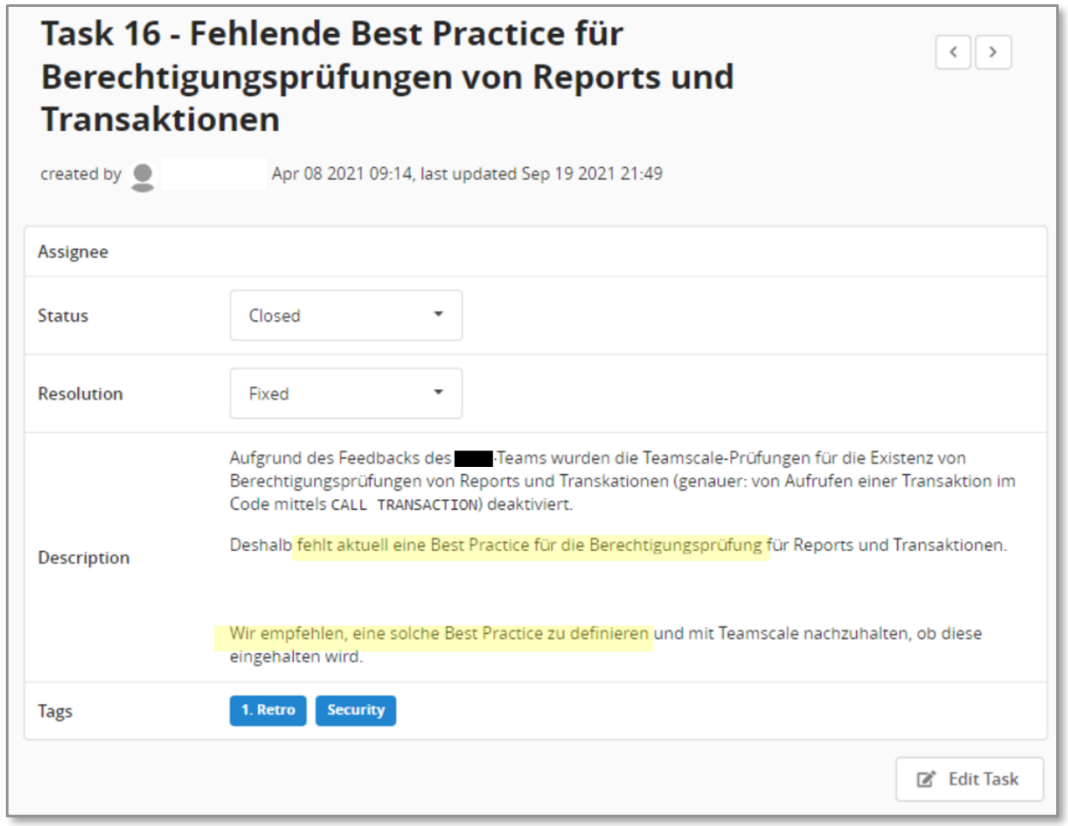
Analyse & Diskussion von Qualitätstrends



→ Trendanalyse schafft Transparenz für Team und Management

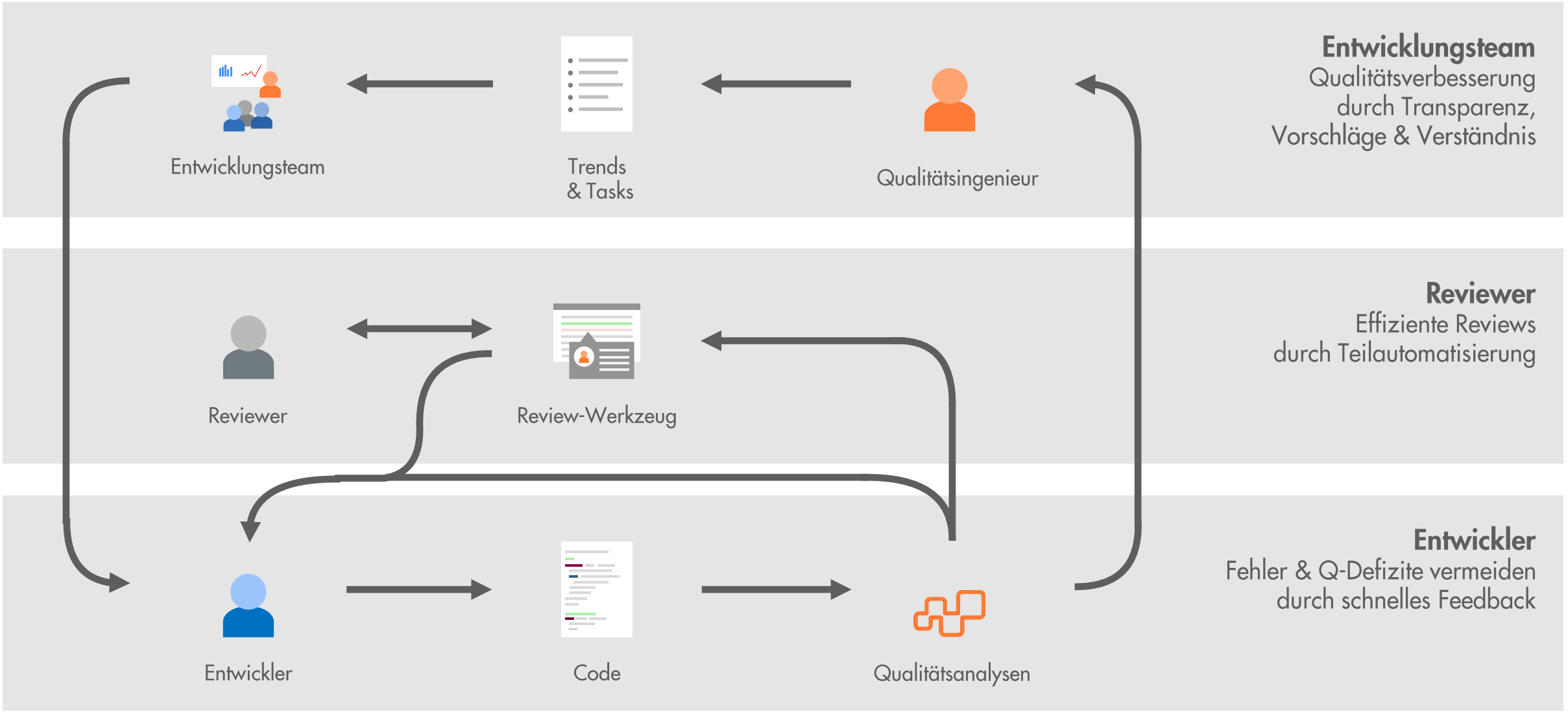
Konkrete Verbesserungsvorschläge



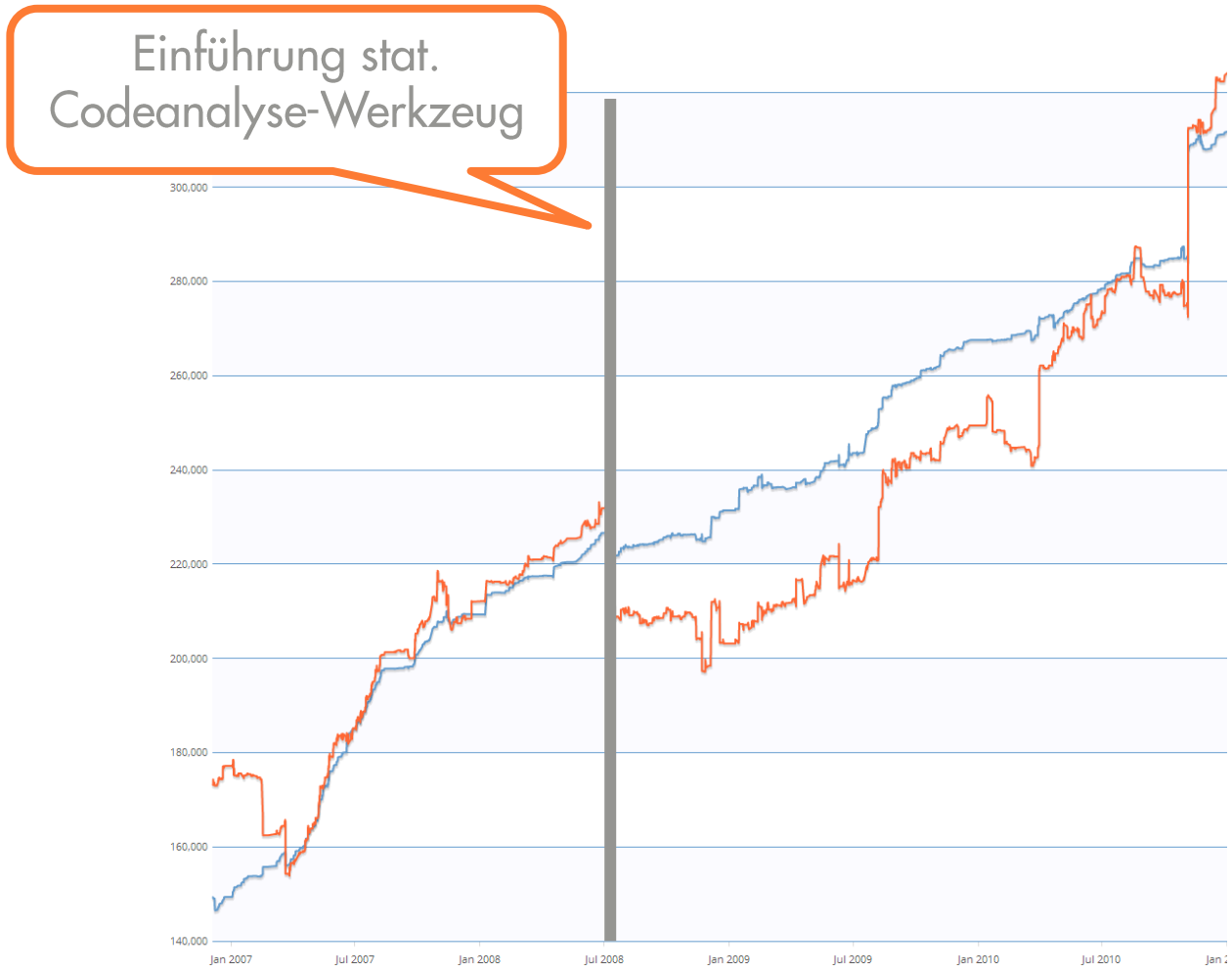


→ Verbesserungsvorschläge ermöglichen Einplanung und konkrete Verbesserung
→ Diskussionen schaffen gemeinsames Qualitätsverständnis und Best Practices

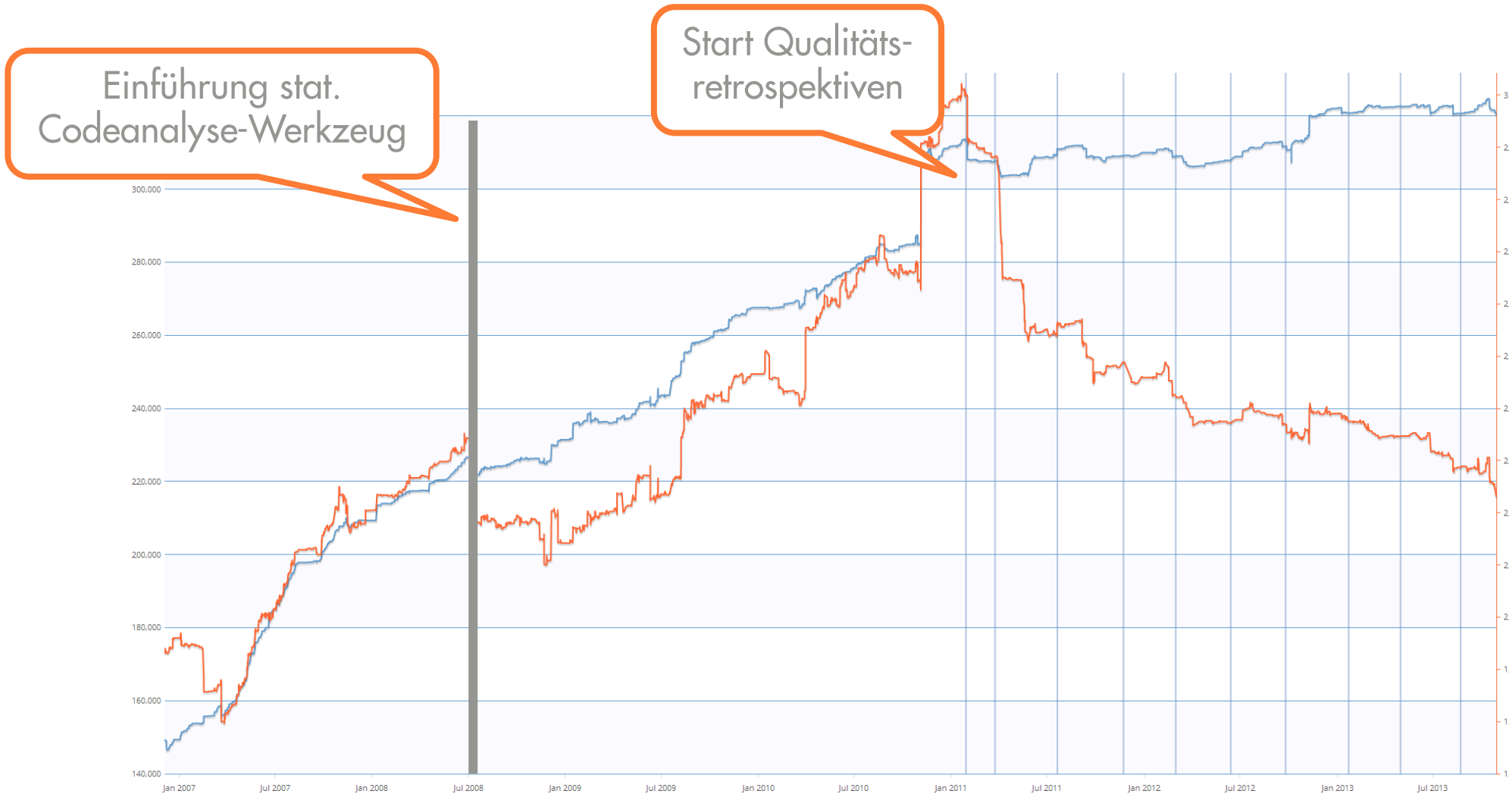
Continuous Quality Control-Feedbackschleifen



Was bringt Continuous Quality Control?

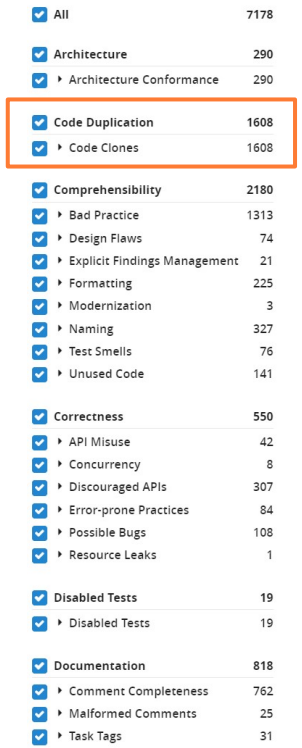


→ Werkzeug hat i.d.R. nur einen kurzfristigen Effekt



→ Kombination aus Werkzeug und Retrospektiven hat i.d.R. einen nachhaltigen Effekt

Nutzen der Klonanalyse für Munich Re



$$500 \frac{PT}{Jahr}^{*}$$

Munich Re spart durch Einsatz von Clone Detection jährlich ca. 500 PT Aufwand für Fehlerbehebung

$$\text{Ersparnis Aufwand} = \mathbf{6}\%^{*}$$

Munich Re spart durch Einsatz von Clone Detection jährlich 6% Aufwand durch vermiedene Redundanz ein.

→ Continuous Quality Control hat großen Nutzen

* Übertragbarkeit muss geprüft werden; für ausführliche, konservative Herleitung siehe CQSE-Workshop „Copy, Paste und dann …?“

Wie führt man Continuous Quality Control ein?

→ Technische, organisatorische und soziale Faktoren berücksichtigen

Onboarding-Prozess

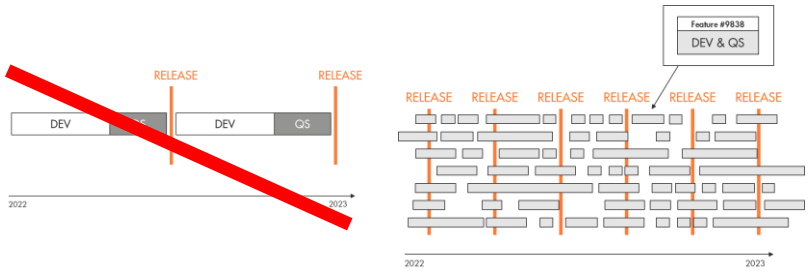
Kickoff & Abholung
(2 Wochen)

Konfiguration & Anpassungen
(2 Wochen)

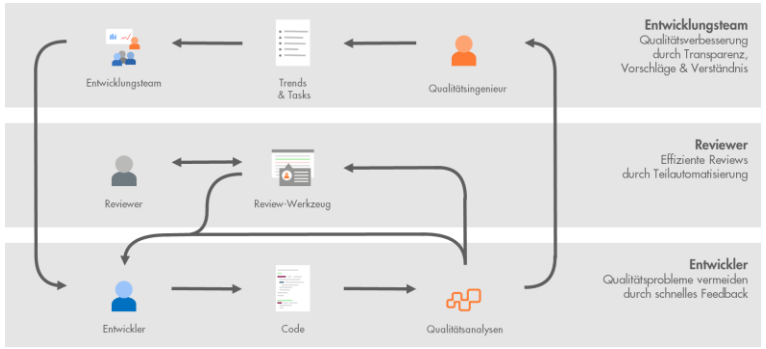
Schulung & Integration
(2 Wochen)

Continous Quality Control
(Kontinuierlich)

Zusammenfassung

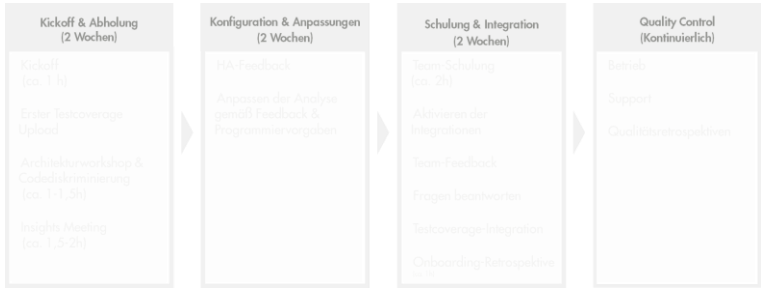


→ Releasefähigkeit erfordert jederzeitige Releasequalität und kontinuierliche QS



→ CQC ermöglicht dies durch Feedbackschleifen für Entwickler, Reviewer und Entwicklungsteam

→ CQC hat großen Nutzen (auch bei gewachsenen Anwendungen)



→ Einführung von CQC erfordert planvolles Vorgehen

Continuous Quality Control

Qualität trotz immer kürzerer Releasezyklen

Dienstag, 11. November 2025
10:30 bis 12:00 Uhr
online und kostenlos

Dr. Tobias Röhm

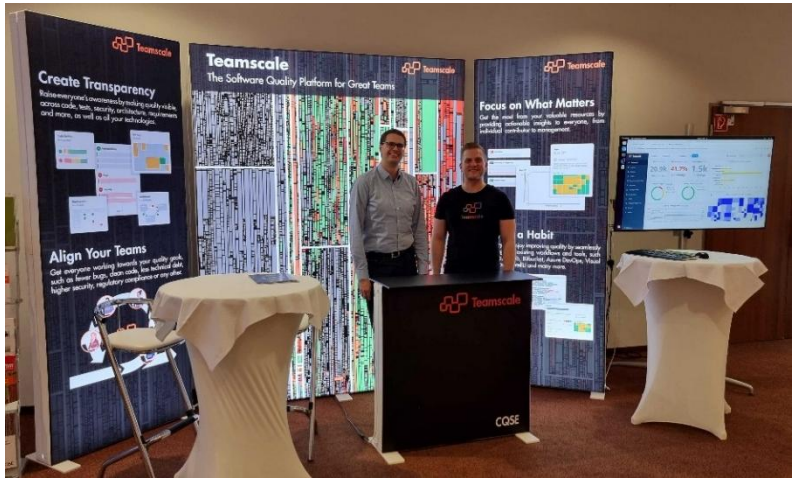
Tobias Wiese

Jetzt anmelden: tmscl.me/cqc2511-tjfs

Folien & Kontakt - Ich freue mich auf den Austausch ☺

Vortragsfolien:
tmscl.me/jfs_2025_releasezyklen

Dr. Tobias Röhm
roehm@cqse.eu
tmscl.me/coffee-tobias-roehm



CQSE-Stand