

Qualität im Vorbeigehen

Nachhaltige Softwarequalität im Takt deiner Feature-Deadlines



Dr. Philipp Straubinger

Dr. Philipp Straubinger



•msg



CQSE



2017 - 2020

2020 - 2025

2025 - heute



VER**SICHER**UNGS
KAMMER
BAYERN

LV 1871

Allianz

ERSTE
Digital

BayBG

...



SIEMENS

NORDEX

ASMPT

ROSEN

...



Vibracoustic



SEW
EURODRIVE

...



TTTech

**SALZGITTER
DIGITAL SOLUTIONS**
Ein Unternehmen der Salzgitter Gruppe

tebis
CAD/CAM MES
Software & Services

**dm
TECH**

Dolby

...

...

SW/M



HIMA **SMART
SAFETY.**

EnBW

DATEV

...

Qualität als Aufwandstreiber?

“(„Uns fehlt aktuell leider aufgrund anderer Themen die Zeit, uns mit Qualität zu beschäftigen.“

— Mindestens jeder 2. Entwickler im Projektalltag

VORURTEIL

„Qualität bremst uns aus!“

Dieses Standardargument entsteht, wenn Qualitätssicherung rein als **isolierter, manueller Prüfschritt** am Ende des Prozesses verstanden wird.

Der harte Projektalltag

Deadline- und Release-Druck: Qualität wird im Sprint oft weggespart.

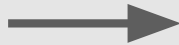
Die typische Folge: Technische Schulden steigen rasant, die Wartbarkeit sinkt, und spätere Änderungen werden extrem teuer.

Ziel des Vortrags

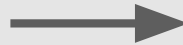
Wir zeigen, wie ein **nahtloser Entwicklungs-Workflow** funktioniert, der nachhaltige Softwarequalität ganz ohne Zeitverlust direkt in den Entwicklungsprozess integriert.



Entwickler



Code



Qualitätsanalysen

Entwickler
Fehler & Qualitäts-Defizite vermeiden
durch schnelles Feedback

Statische Code-Analyse

Ergebnisse direkt dort, wo sie benötigt werden – mit Fokus auf die wirklich relevanten Probleme.



Strukturmetriken

Komplexität im Blick behalten

Verschachtelungen

```
61         } else {
62             Class<?>[] testMethodParameters = testMethod.getParameterTypes();
63             Class<?>[] adapterMethodParameters = method.getParameterTypes();
64             if (testMethodParameters.length == adapterMethodParameters.length) {
65                 tested = true;
66                 for (int i = 0; i < adapterMethodParameters.length; i++) {
67                     if (!testMethodParameters[i].equals(adapterMethodParameters[i])) {
68                         tested = false;
69                     }
70                 }
71             }
72         }
73     }
74 }
```

Statische Code-Analyse in Echtzeit

Ergebnisse direkt dort, wo sie benötigt werden – mit Fokus auf die wirklich relevanten Probleme.



Strukturmetriken

Komplexität im Blick behalten



Klonerkennung

Redundanzen und
Code-Duplikate identifizieren

Klonerkennung

Findings **27645**

Tolerated **0**

False Positives **0**

Choose folder...

Treemap

Export

Configure Table

☐	Message	Category	Instances	Length
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 13	Redundancy › Code Clones › Intra Component Clones	50	13
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 13	Redundancy › Code Clones › Intra Component Clones	50	13
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 13	Redundancy › Code Clones › Intra Component Clones	50	13
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11
☐	⚠ Intra Component Clone with 50 instances of length 13	Redundancy › Code Clones › Intra Component Clones	50	13
☐	⚠ Intra Component Clone with 50 instances of length 11	Redundancy › Code Clones › Intra Component Clones	50	11

Klonerkennung

Findings **27645**

Tolerated **0**

False Positives **0**

Choose folder...

Treemap

Export

Configure Table

☐	Message	Category	Instances	Length
☐	 Intra Component Clone with 2 instances of length 563	Redundancy › Code Clones › Intra Component Clones	2	563
☐	 Intra Component Clone with 2 instances of length 563	Redundancy › Code Clones › Intra Component Clones	2	563
☐	 Intra Component Clone with 2 instances of length 496	Redundancy › Code Clones › Intra Component Clones	2	496
☐	 Intra Component Clone with 2 instances of length 496	Redundancy › Code Clones › Intra Component Clones	2	496
☐	 Intra Component Clone with 3 instances of length 412	Redundancy › Code Clones › Intra Component Clones	3	412
☐	 Intra Component Clone with 3 instances of length 412	Redundancy › Code Clones › Intra Component Clones	3	412
☐	 Intra Component Clone with 3 instances of length 412	Redundancy › Code Clones › Intra Component Clones	3	412
☐	 Intra Component Clone with 2 instances of length 396	Redundancy › Code Clones › Intra Component Clones	2	396
☐	 Intra Component Clone with 2 instances of length 396	Redundancy › Code Clones › Intra Component Clones	2	396
☐	 Intra Component Clone with 3 instances of length 392	Redundancy › Code Clones › Intra Component Clones	3	392

Statische Code-Analyse in Echtzeit

Ergebnisse direkt dort, wo sie benötigt werden – mit Fokus auf die wirklich relevanten Probleme.



Strukturmetriken

Komplexität im Blick behalten



Klonerkennung

Redundanzen und
Code-Duplikate identifizieren



Architekturanalyse

Einhaltung von Soll-Vorgaben
überwachen



Namensregeln & Kommentare

Konsistente Benennung und
Dokumentation

Benennung

```
470 Boolean result = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo, basicCaseConfigDataBE);
471 Boolean result_1 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_1, basicCaseConfigDataBE);
472 Boolean result_2 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_2, basicCaseConfigDataBE);
473 Boolean result_3 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_3, basicCaseConfigDataBE);
474 Boolean result_4 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_4, basicCaseConfigDataBE);
475 Boolean result_5 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_5, basicCaseConfigDataBE);
476 Boolean result_6 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_6, basicCaseConfigDataBE);
477 Boolean result_7 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_7, basicCaseConfigDataBE);
478 Boolean result_8 = this.service.isNotEqualsContinuing(basicCaseConfigDataRawTo_8, basicCaseConfigDataBE);
```

Statische Code-Analyse in Echtzeit

Ergebnisse direkt dort, wo sie benötigt werden – mit Fokus auf die wirklich relevanten Probleme.



Strukturmetriken

Komplexität im Blick behalten



Klonerkennung

Redundanzen und
Code-Duplikate identifizieren



Architekturanalyse

Einhaltung von Soll-Vorgaben
überwachen



Namensregeln & Kommentare

Konsistente Benennung und
Dokumentation



Bug Pattern

Auffällige Stellen und
Antipatterns aufspüren



SAST

Security-Schwachstellen und
Fehler aufspüren

Security

✓	Security	887
✓	▼ Directory Traversal	5
✓	Improper limitation of a pathname to a restricted directory ('Path Traversal') (Java) (SpotbugsPathTraversalAbsolute)	5
✓	▼ Hard-Coded Credentials	3
✓	Hard-coded password	3
✓	▼ SAST	877
✓	Quality	27
✓	Security	850
✓	▼ Potential Injection	2
✓	Improper neutralization of special elements used in an LDAP query ('LDAP Injection') (Java)	2

Statische Code-Analyse in Echtzeit

Ergebnisse direkt dort, wo sie benötigt werden – mit Fokus auf die wirklich relevanten Probleme.



Strukturmetriken

Komplexität im Blick behalten



Klonerkennung

Redundanzen und
Code-Duplikate identifizieren



Architekturanalyse

Einhaltung von Soll-Vorgaben
überwachen



Namensregeln & Kommentare

Konsistente Benennung und
Dokumentation



Bug Pattern

Auffällige Stellen und
Antipatterns aufspüren



SAST

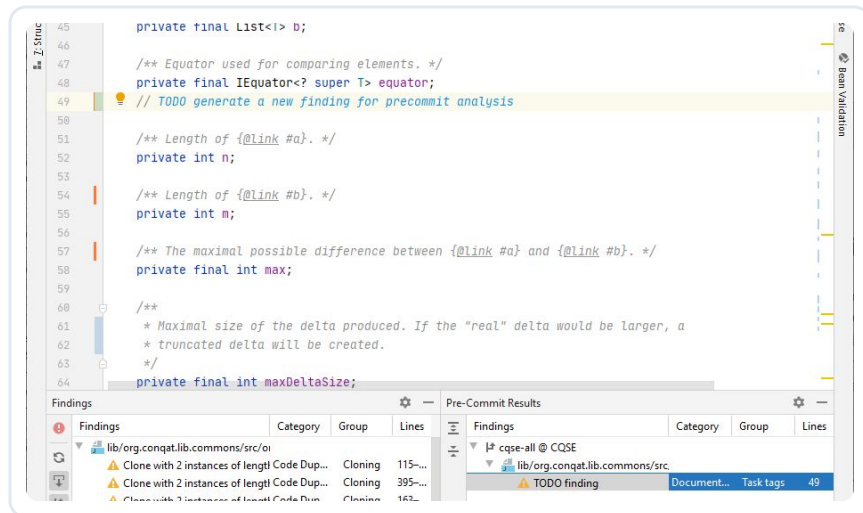
Security-Schwachstellen und
Fehler aufspüren



Sofortiges Feedback während der Entwicklung

Fehlervermeidung direkt im Arbeitsfluss, statt erst in späten Testphasen.

IDE-Integration & Precommit-Analyse



Precommit-Analyse

Ermöglicht wertvolles Qualitätsfeedback bereits vor dem Commit direkt in der Entwicklungsumgebung.



Inkrementelle Analyse

Liefert extrem schnelle Analyseergebnisse auch bei hochgradig komplexen Systemen.



Effektiver Qualitäts-Workflow

Feedback aus statischer Codeanalyse ist schnell, fokussiert und nahtlos integriert nutzbar.



Reviewer



Review-Werkzeug

Reviewer
Effiziente Reviews
durch Teilautomatisierung



Entwickler



Code



Qualitätsanalysen

Entwickler
Fehler & Qualitäts-Defizite vermeiden
durch schnelles Feedback

Annotationen an Merge Requests

CR#133750 Remove NPE Edit Code

Open Andrew Smith requested to merge `cr/133750_fix_nullpointer_` into `master` 38 minutes ago

Overview 3 Commits 1 Changes 1 3 unresolved threads

Teamscale

Findings	1	4	1 Test Gap (33%)	Impacted spec Items	0
----------	---	---	------------------	---------------------	---

Edited just now by Teamscale

Approve Approval is optional

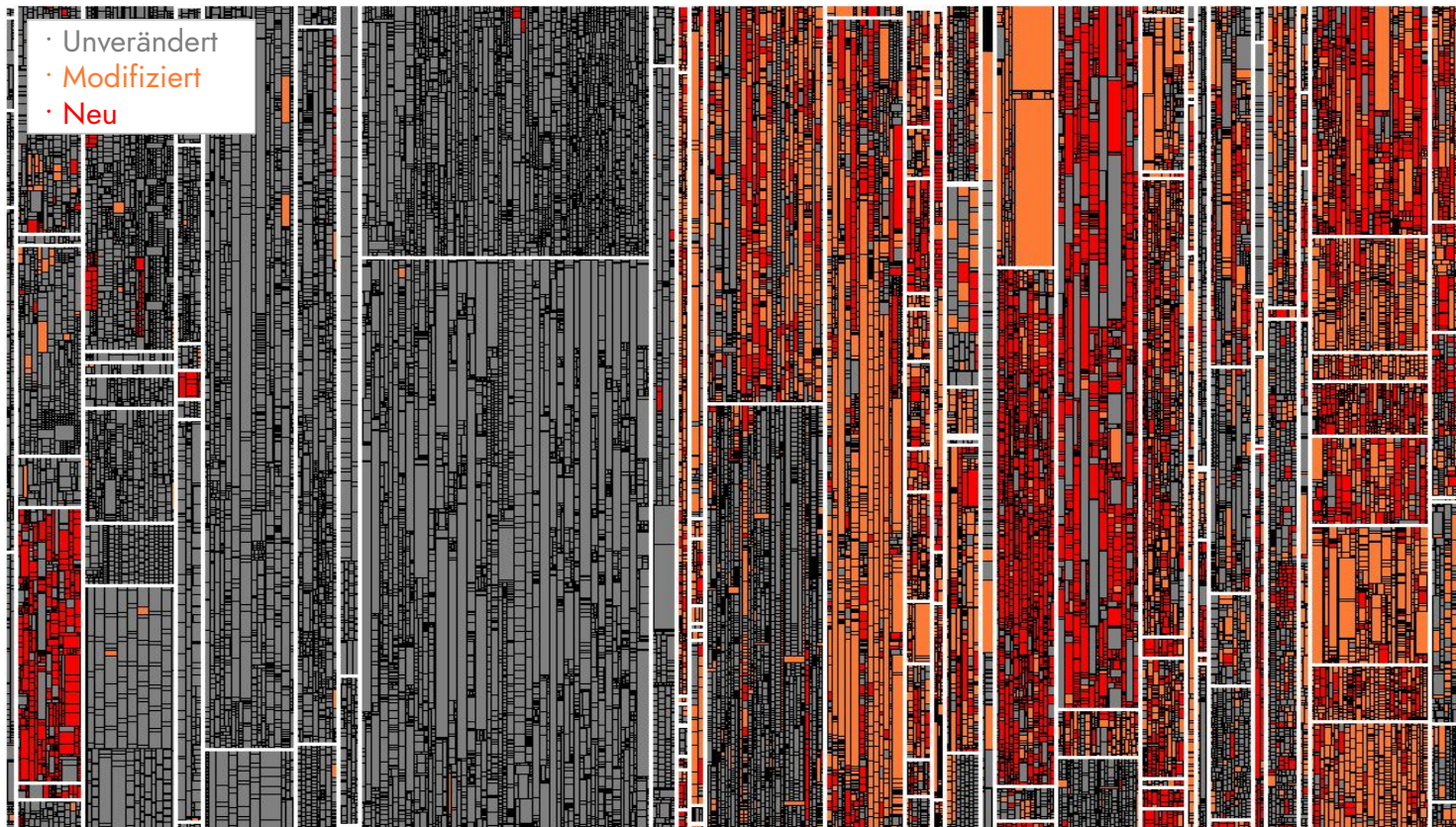
Merge blocked: all threads must be resolved. Jump to first unresolved thread Create issue to resolve all threads

Merge details

- 1 commit and 1 merge commit will be added to master.
- Source branch will be deleted.

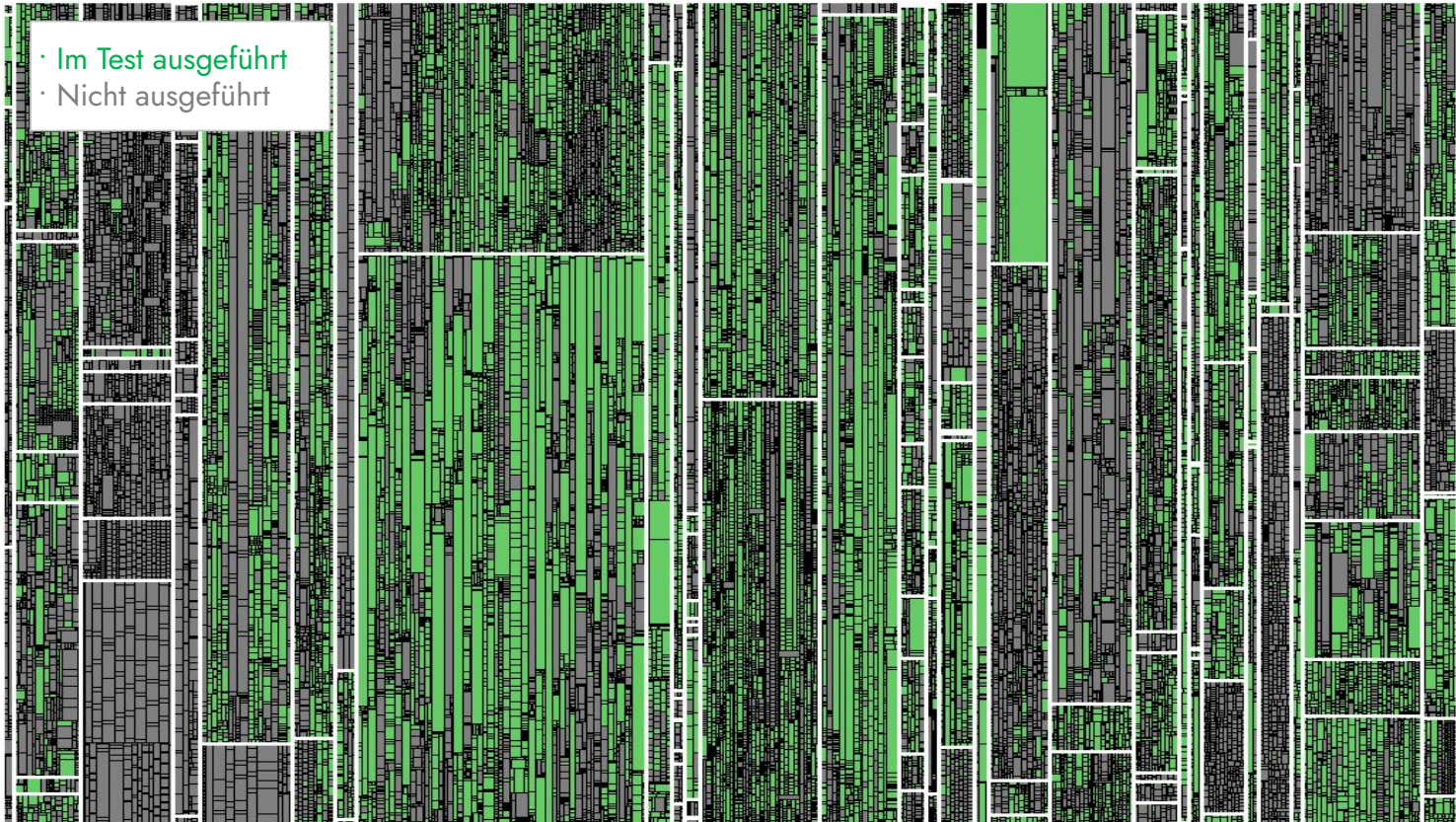
0 0 Oldest first Show all activity

Code-Änderungen

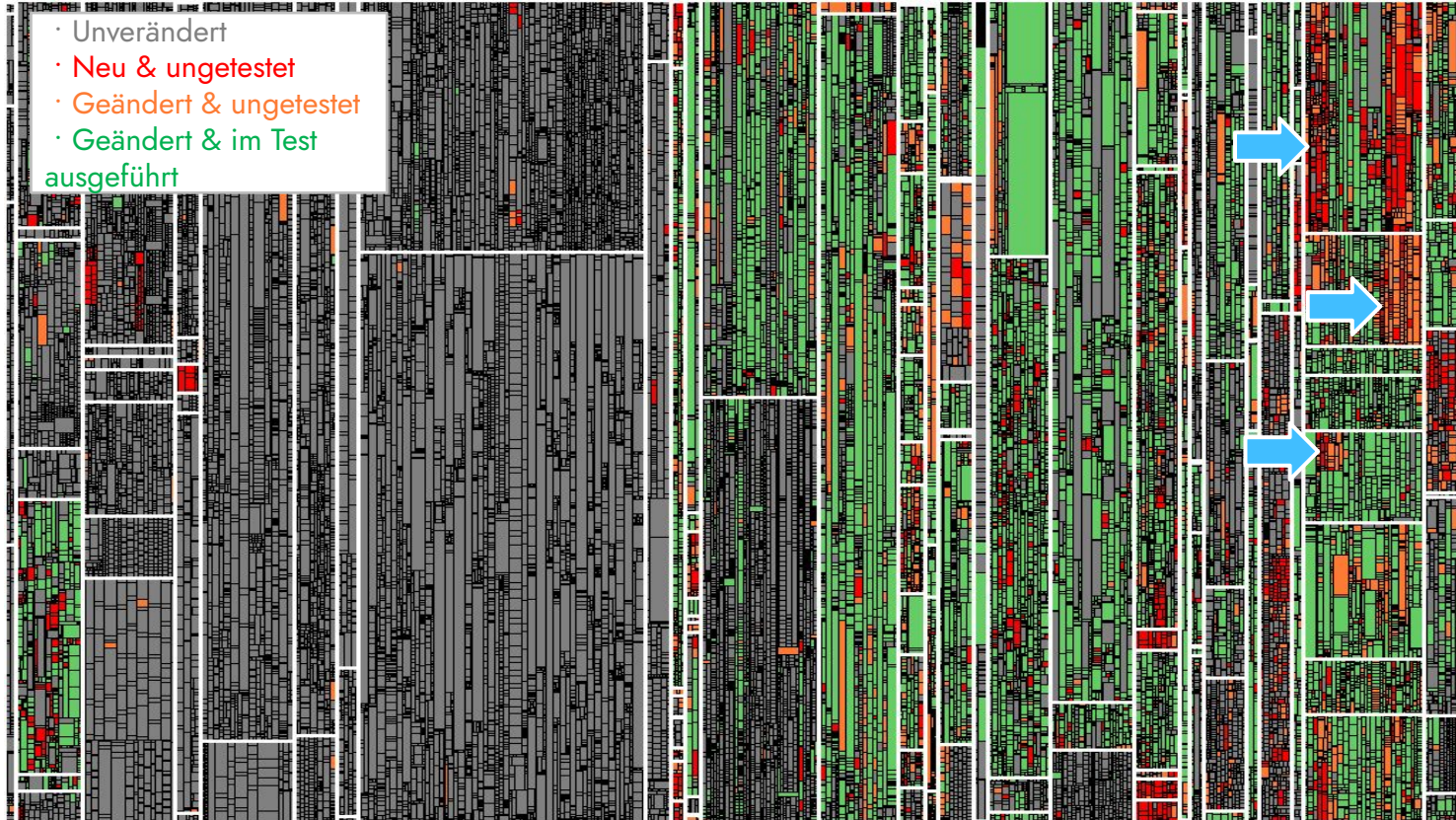


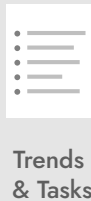
Coverage

- Im Test ausgeführt
- Nicht ausgeführt

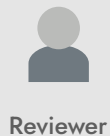


Test-Gap-Analyse

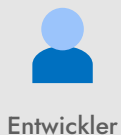




Entwicklungsteam
Qualitätsverbesserung
durch Transparenz,
Vorschläge & Verständnis

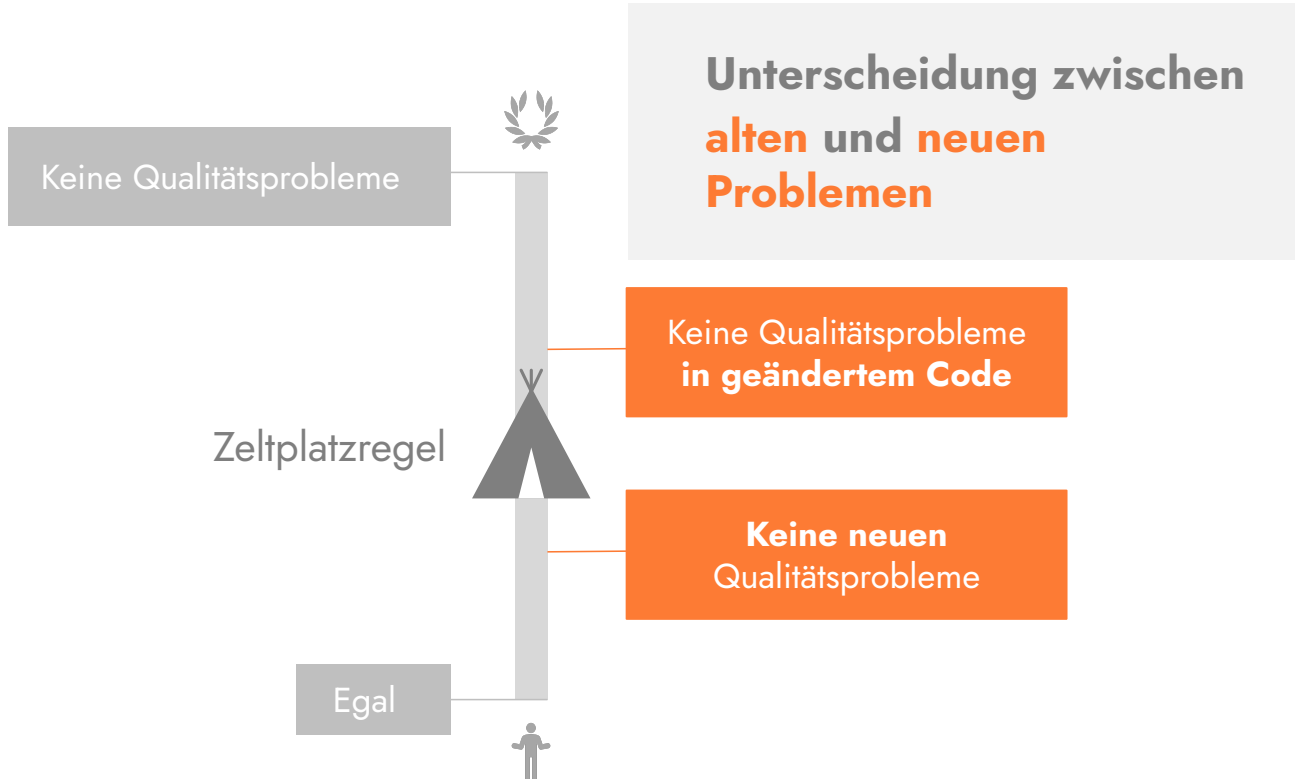


Reviewer
Effiziente Reviews
durch Teilautomatisierung



Entwickler
Fehler & Qualitäts-Defizite vermeiden
durch schnelles Feedback

Qualitätsziel



Codequalitäts-Retrospektiven

Strukturierter Prozess zur kontinuierlichen Verbesserung der Codequalität im Team



Findings & Defizite

Identifikation relevanter Findings basierend auf dem Qualitätsziel.



Vorstellung & Bericht

Präsentation neuer Tasks in einem kurzen Bericht (Foliensatz).

Codequalitäts-Retrospektiven



BASTA! Mainz - Codequal

Von 28.09.2026 bis 02.10.

Dr. Philipp Straubinger

Task 12 - Keine Konsolenausgaben

created by [Philipp Straubinger](#) at Sep 23 2026 09:42, last updated at Today 12:11



Assignee	PS Philipp Straubinger
Status	Open
Resolution	None
Description	Entferne alle Ausgaben auf die Konsole durch einen Logger.
Tags	2026.09

Edit Task

Findings 0 4 0 0 Code snippets 0

Replace this use of System.out by a logger in
TAPropertiesHandler.java:144

Replace this use of System.out by a logger in
DatenbankService.java:369

Replace this use of System.out by a logger in
MessageRefldGenerator.java:72

Replace this use of System.out by a logger in
XmlUtils.java:147

Codequalitäts-Retrospektiven

Strukturierter Prozess zur kontinuierlichen Verbesserung der Codequalität im Team



Findings & Defizite

Identifikation relevanter Findings basierend auf dem Qualitätsziel.



Vorstellung & Bericht

Präsentation neuer Tasks in einem kurzen Bericht (Foliensatz).



Diskussion & Feedback

Austausch über Findings, Vorschläge und Analyse-Anpassungen.



Verbesserungen

Erarbeiten konkreter Maßnahmen (z.B. Basis-Klasse einführen).



Team-Entscheidung

Gemeinsamer Beschluss über die Umsetzung von Qualitätstasks.



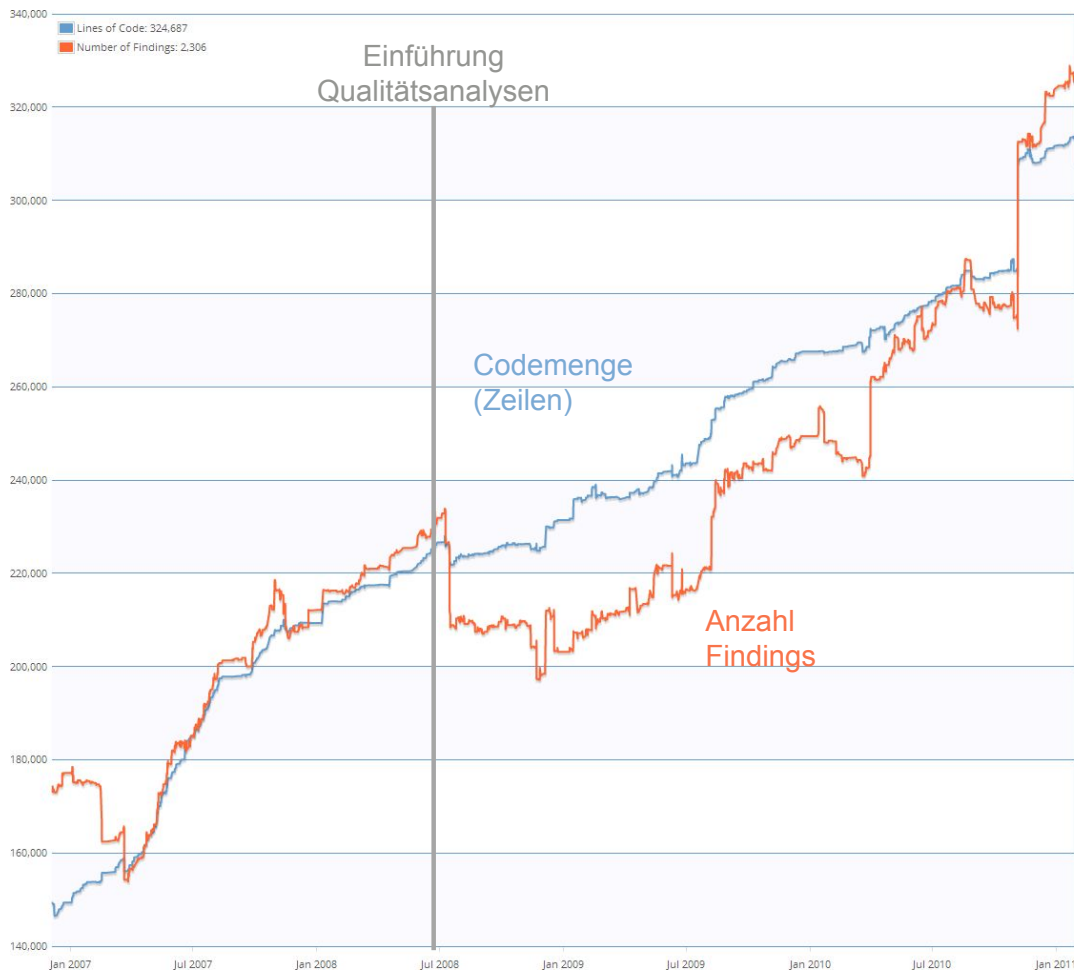
Review alter Tasks

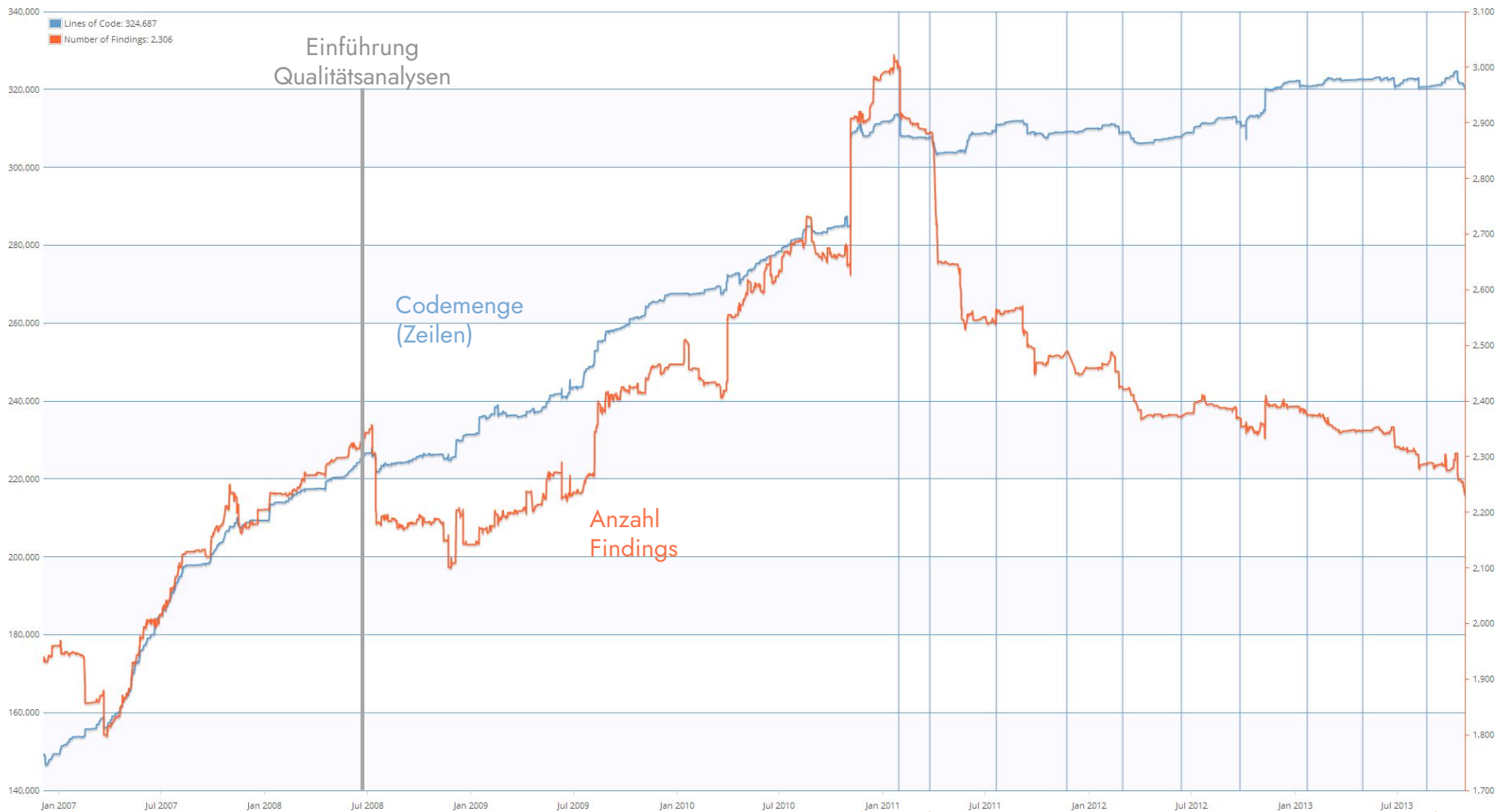
Rückblick und Nachverfolgung bereits beschlossener Maßnahmen.



Kollaborative Qualitätssicherung im Sprint-Rhythmus

Die Retrospektive fördert das gemeinsame Qualitätsverständnis und stellt sicher, dass Maßnahmen direkt im Alltag verankert werden.





Ohne Codequalitäts-Retrospektiven

Mit Codequalitäts-Retrospektiven

KUNDENSTIMME



»Der Einsatz von Teamscale und regelmäßige Qualitätsretrospektiven durch CQSE-Qualitätsingenieur:innen helfen uns, eine hohe Softwarequalität sicherzustellen, Wissen im Team zu verteilen und einen systematischen Umgang bezüglich Softwaresicherheit gegenüber Aufsichtsbehörden nachzuweisen.«

Rainer Midderhoff

Leiter Stab Cyber Incidents und Regulierung, LV 1871

Zusammenfassung: Qualität im Vorbeigehen



Sofortiges Feedback während der Entwicklung

Fehlervermeidung direkt im Arbeitsfluss, statt erst in späten Testphasen.



Effektiver Qualitäts-Workflow

Feedback aus statischer Codeanalyse ist schnell, fokussiert und nahtlos integriert nutzbar.



Kollaborative Qualitätssicherung im Sprint-Rhythmus

Die Retrospektive fördert das gemeinsame Qualitätsverständnis und stellt sicher, dass Maßnahmen direkt im Alltag verankert werden.

Contact



CQSE GmbH
Centa-Hafenbrädl-Straße 59
81249 München



Dr. Philipp Straubinger

straubinger@cqse.eu

+49 173 3568646