

Wie hole ich die anderen ins »Qualitätsboot«?

Erfahrungen aus 10 Jahren Qualitätsretrospektiven, OOP 2025



Dr. Tobias Röhm (@langelot)

Dr. Tobias Röhm



 Entwicklerin

 Tester

 Architekt

 Product Owner

Softwarequalität

 Anwendungs-
verantwortliche

 Projektleiterin

 QS-Spezialist

 Managerin

Welche Bezeichnung trifft Ihre aktuelle Rolle am besten (Mehrfachnennungen möglich)?

0

(Senior-)
Entwickler

0

Architekt

0

Anwendungsver-
antwortlicher

0

QS-Spezialist

0

(Senior-) Tester

0

Manager

0

Projektleite



 Entwicklerin

 Tester

 Architekt

 Product Owner

Softwarequalität

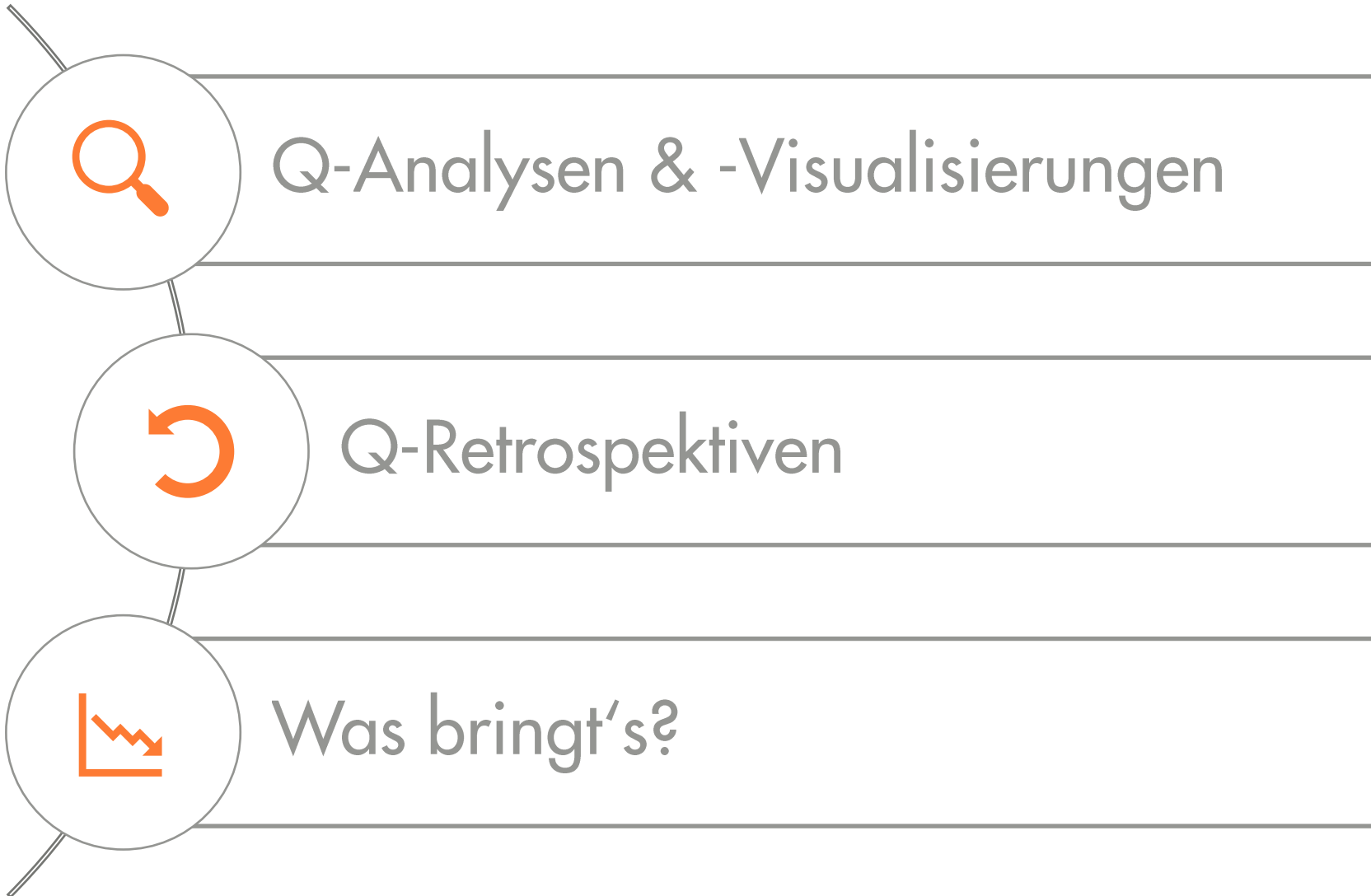
 Anwendungs-
verantwortliche

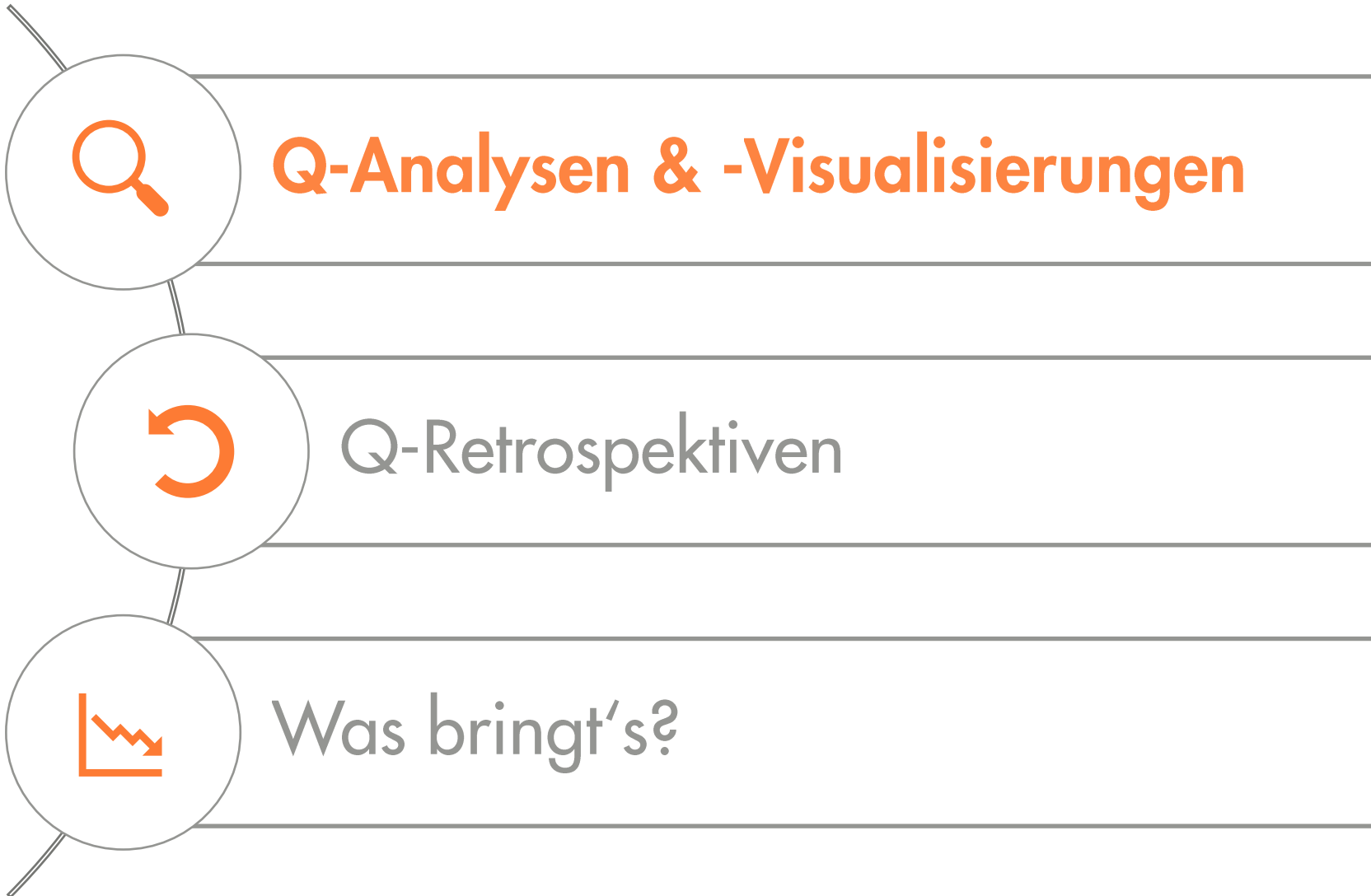
 Projektleiterin

 QS-Spezialist

 Managerin

→ **Wirksame Qualitätssicherung erfordert Einbeziehung aller Beteiligten**





Q-Analysen & -Visualisierungen



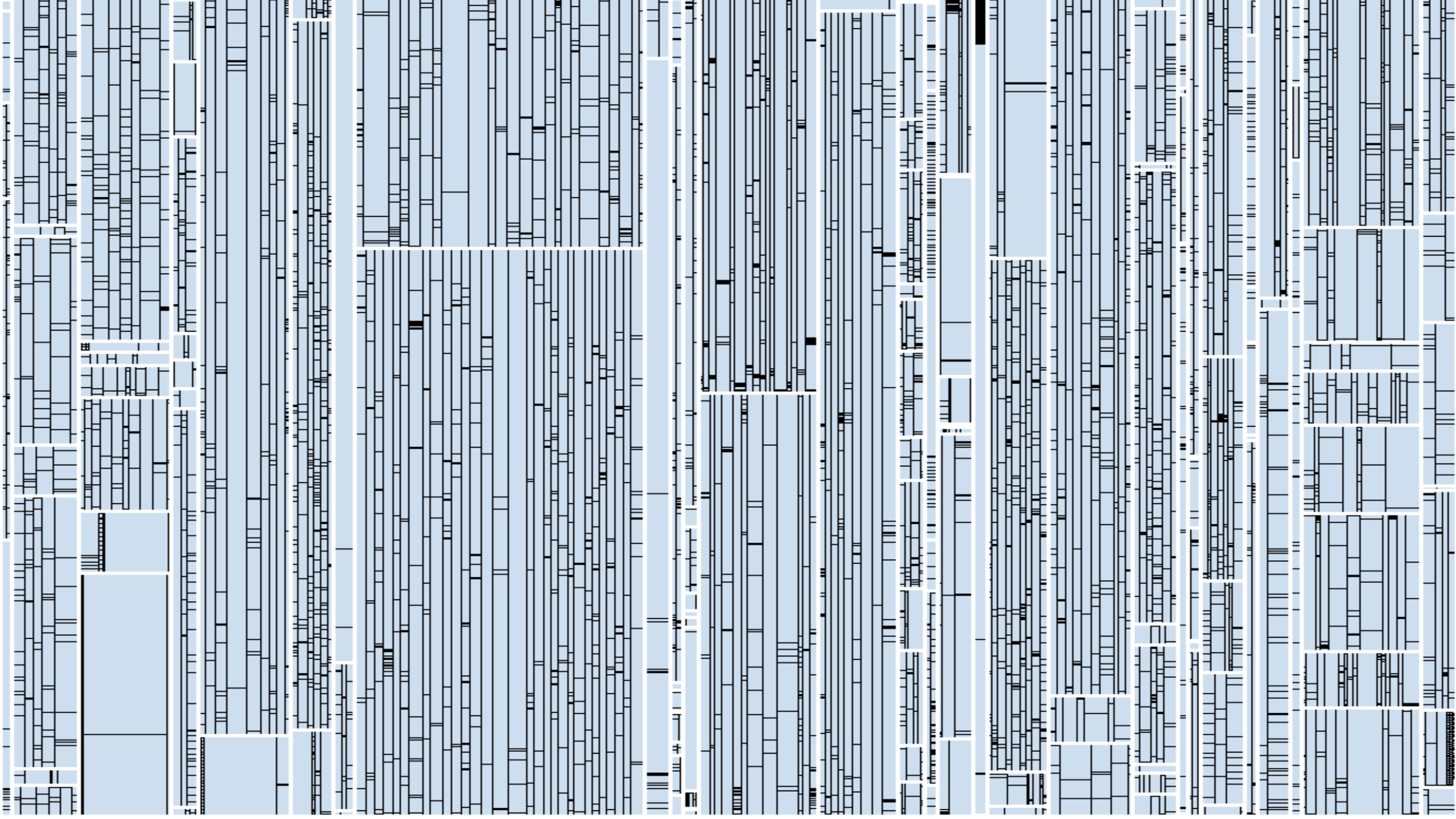
Q-Retrospektiven

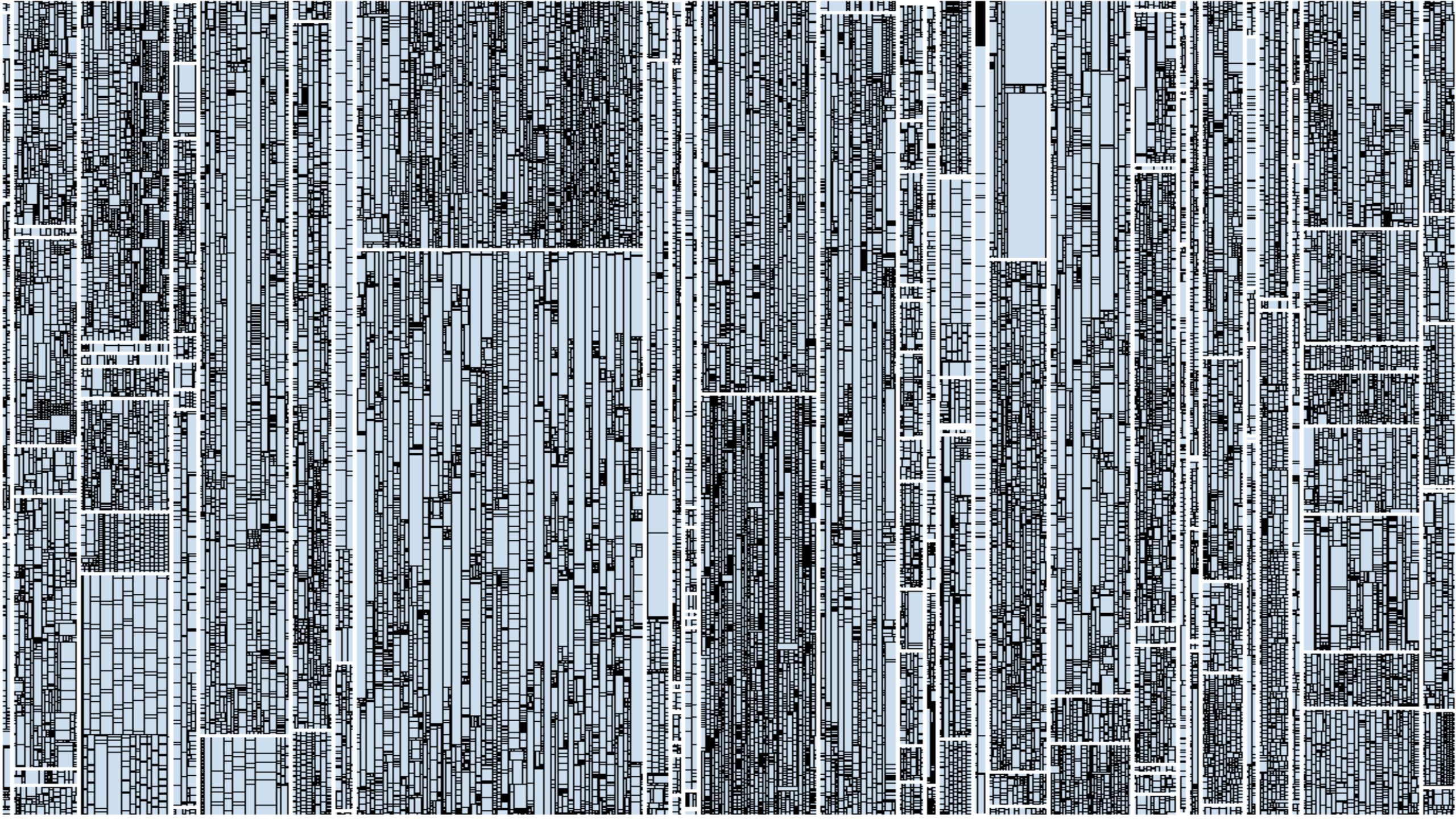


Was bringt's?

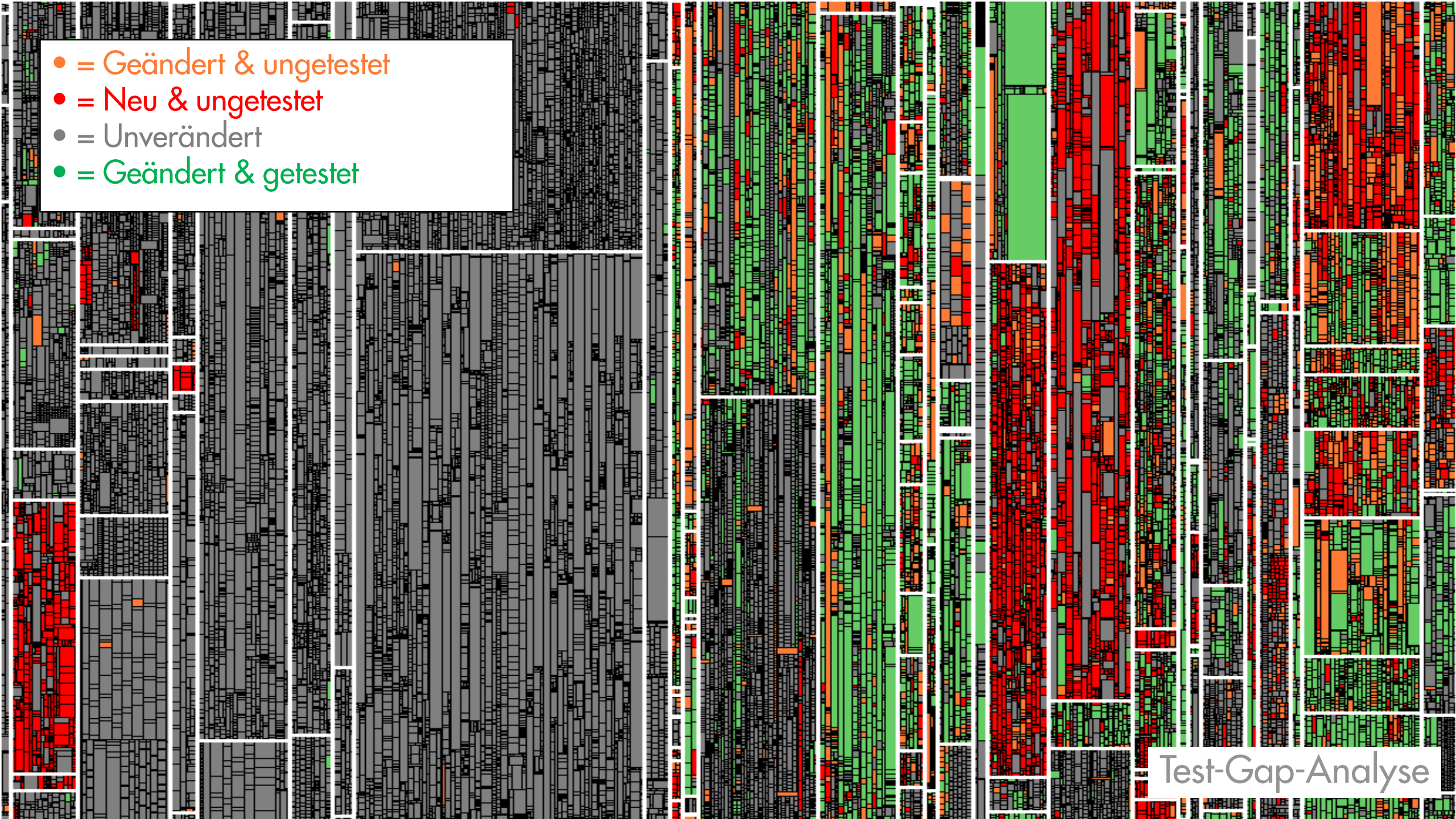


Teamscale

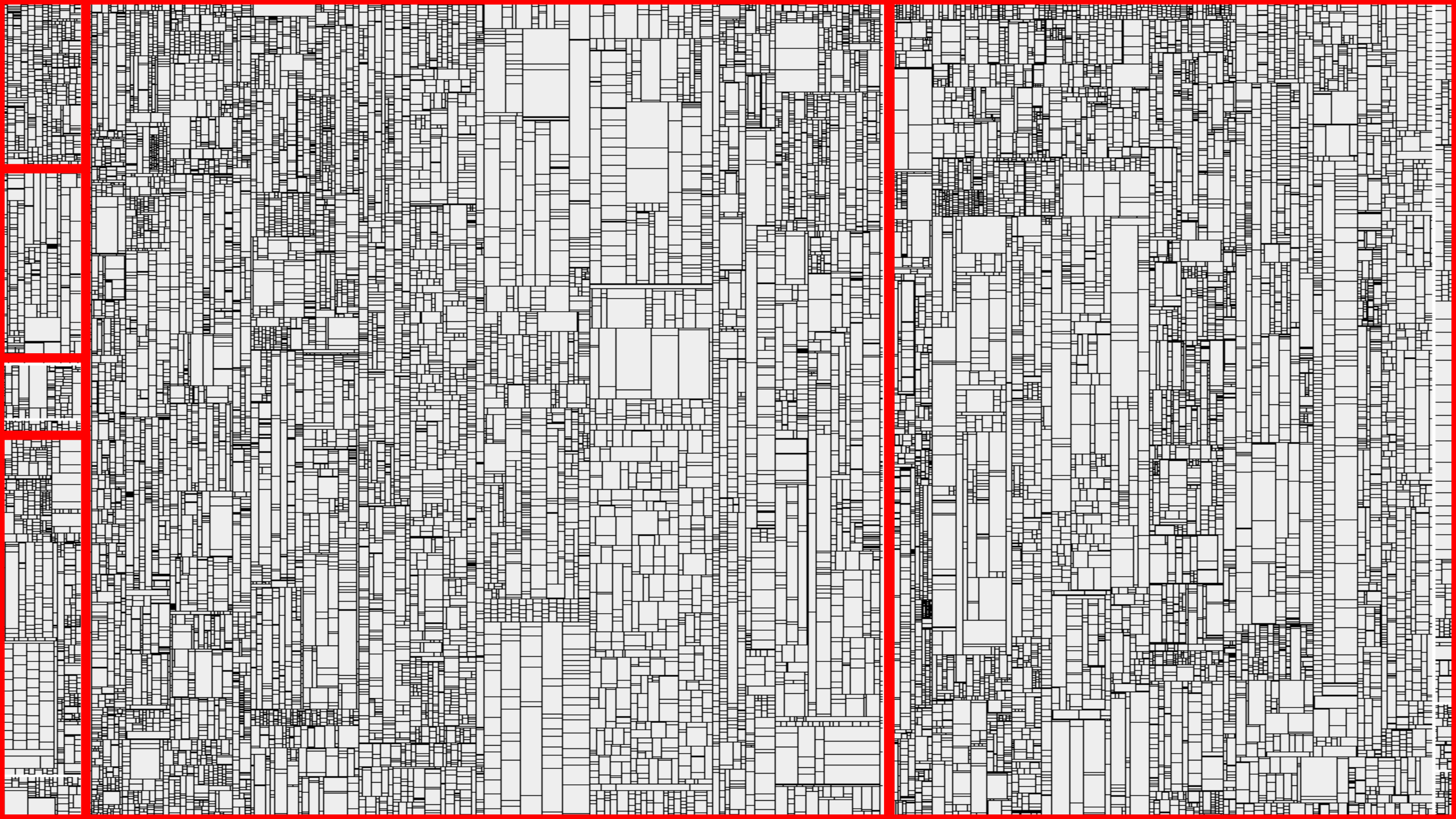


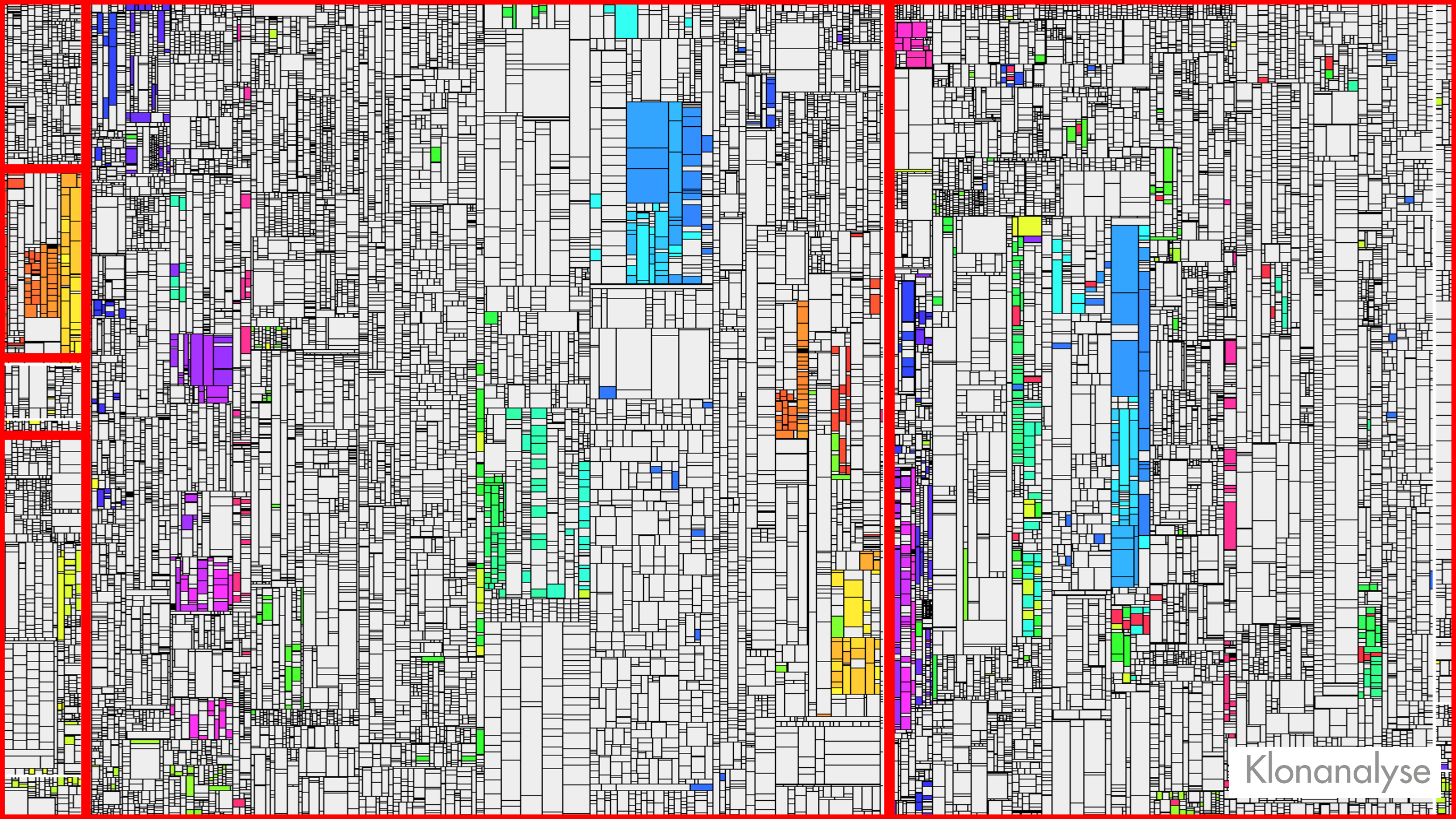


- = Geändert & ungetestet
- = Neu & ungetestet
- = Unverändert
- = Geändert & getestet



Test-Gap-Analyse





Klonanalyse

```

if (lhs==null && rhs==null) return;
if (lhs==null) fail("lhs is null while rhs="+rhs);
if (rhs==null) fail("rhs is null while lhs="+lhs);

Constructor<?> lc = findDataBoundConstructor(lhs.getClass());
Constructor<?> rc = findDataBoundConstructor(rhs.getClass());
assertEquals("Data bound constructor mismatch. Different type?", lc,rc);

List<String> primitiveProperties = new ArrayList<String>();

String[] names = ClassDescriptor.loadParameterNames(lc);
Class<?>[] types = lc.getParameterTypes();
assertEquals(names.length, types.length);
for (int i=0; i<types.length; i++) {
    Object lv = ReflectionUtils.getPublicProperty(lhs, names[i]);
    Object rv = ReflectionUtils.getPublicProperty(rhs, names[i]);

    if (Iterable.class.isAssignableFrom(types[i])) {
        Iterable lcol = (Iterable) lv;
        Iterable rcol = (Iterable) rv;
        Iterator ltr,rtr;
        for (ltr=lcol.iterator(), rtr=rcol.iterator(); ltr.hasNext() && rtr.hasNext();){
            Object litem = ltr.next();
            Object ritem = rtr.next();

            if (findDataBoundConstructor(litem.getClass())!=null) {
                assertEqualsDataBoundBeans(litem,ritem);
            } else {
                assertEquals(litem,ritem);
            }
        }
        assertFalse("collection size mismatch between "+lhs+" and "+rhs, ltr.hasNext() ^
    } else
    if (findDataBoundConstructor(types[i])!=null || (lv!=null && findDataBoundConstructo
        // recurse into nested databound objects
        assertEqualsDataBoundBeans(lv,rv);
    } else {
        primitiveProperties.add(names[i]);
    }
}

// compare shallow primitive properties
if (!primitiveProperties.isEmpty())
    assertEqualsBeans(lhs,rhs,Util.join(primitiveProperties,","));

```

```

if (lhs==null && rhs==null) return;
if (lhs==null) fail("lhs is null while rhs="+rhs);
if (rhs==null) fail("rhs is null while lhs="+lhs);

Constructor<?> lc = findDataBoundConstructor(lhs.getClass());
Constructor<?> rc = findDataBoundConstructor(rhs.getClass());
assertThat("Data bound constructor mismatch. Different type?", (Constructor)rc, is((Cons

List<String> primitiveProperties = new ArrayList<String>();

String[] names = ClassDescriptor.loadParameterNames(lc);
Class<?>[] types = lc.getParameterTypes();
assertThat(types.length, is(names.length));
for (int i=0; i<types.length; i++) {
    Object lv = ReflectionUtils.getPublicProperty(lhs, names[i]);
    Object rv = ReflectionUtils.getPublicProperty(rhs, names[i]);

    if (lv != null && rv != null && Iterable.class.isAssignableFrom(types[i])) {
        Iterable lcol = (Iterable) lv;
        Iterable rcol = (Iterable) rv;
        Iterator ltr,rtr;
        for (ltr=lcol.iterator(), rtr=rcol.iterator(); ltr.hasNext() && rtr.hasNext();){
            Object litem = ltr.next();
            Object ritem = rtr.next();

            if (findDataBoundConstructor(litem.getClass())!=null) {
                assertEqualsDataBoundBeans(litem,ritem);
            } else {
                assertThat(ritem, is(litem));
            }
        }
        assertThat("collection size mismatch between " + lhs + " and " + rhs, ltr.hasNext()
            is(false));
    } else
    if (findDataBoundConstructor(types[i])!=null || (lv!=null && findDataBoundConstructo
        // recurse into nested databound objects
        assertEqualsDataBoundBeans(lv,rv);
    } else {
        primitiveProperties.add(names[i]);
    }
}

// compare shallow primitive properties
if (!primitiveProperties.isEmpty())
    assertEqualsBeans(lhs,rhs,Util.join(primitiveProperties,","));

```

The variable token may contain a null value at this point and is dereferenced

Flag as False Positive

Flag as Tolerated

AI Actions

Correctness > Possible Bugs > Null pointer dereference

Dereferencing a potentially null pointer or null reference can cause runtime exceptions and unexpected behavior. Consider performing null checks before dereferencing pointers or using safer dereferencing methods.

What Does This Check Look For?

This check looks for null dereferencing problems and identifies potential instances where a program may attempt to access or manipulate an object or variable that has not been properly initialized or has been set to `null` (in Java and C#) or `NULL`, `0`, or `nullptr` (in C and C++).

Some notes related to C/C++ code: first, note that it is sometimes common to dereference a pointer after calling `malloc`. However, since `malloc` can fail, TeamScale will create a finding if a pointer is dereferenced after `malloc` without a prior null check. Second, TeamScale will identify the termination of a possible control flow when control flow termination keywords are used (such as `return` or `break`) or when the following functions

Show more

ScannerUtils.java:54

Code

Introduction Tasks Properties

```
44     * @throws IOException
45     *         thrown if scanner throws an IO exception
46     */
47     public static void readTokens(IScanner scanner, List<IToken> tokens, List<ScannerException> exceptions)
48         throws IOException {
49         IToken token = null;
50
51         do {
52             try {
53                 token = scanner.getNextToken();
54                 if (token.getType() != ETokenType.EOF) {
55                     tokens.add(token);
56                 }
57             } catch (ScannerException e) {
58                 exceptions.add(e);
59                 continue;
60             }
61         } while (token != null && token.getType() != ETokenType.EOF);
62     }
63
64     /**
```

NULL-Check fehlt
→ NPE-Exception

Show all lines (99 more) and all findings (4 more)

Korrektheitsanalyse

Missing authority check before CALL TRANSACTION

Code Anomalies / Security

CALL TRANSACTION executes a transaction with the given transaction code. Until SAP_BASIS release 7.40 no authority check was performed for CALL TRANSACTION, from SAP_BASIS release 7.40 on, a authority check is only performed, if the addition WITH AUTHORITY CHECK is used.

To solve this issue, the addition WITH AUTHORITY CHECK should be added (if SAP_BASIS version is 7.40 or higher) or the function module AUTHORITY_CHECK_TCODE should be called before.

in `PROGRAM` `...` `...` `...` `...` `...` `...`

```
536 CALL TRANSACTION ... USING ... MODE ...
537 MESSAGES INTO ...
```

Berechtigungsprüfung fehlt

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



IDE-Integration

The screenshot displays an IDE interface with a code editor at the top and two panels below it. The code editor shows a Java class with several private fields and a TODO comment. The Findings panel on the left lists various issues, including 'Clone with 2 instances of length!' and 'TODO finding'. The Pre-Commit Results panel on the right shows a single finding: 'TODO finding' at line 49. The status bar at the bottom indicates 'Pre-Commit Results for cqse-all @ CQSE: 1 finding added. No findings removed (3 minutes ago) 49:58 CRLF UTF-8 Tab* | cr/20332_test_precommit'.

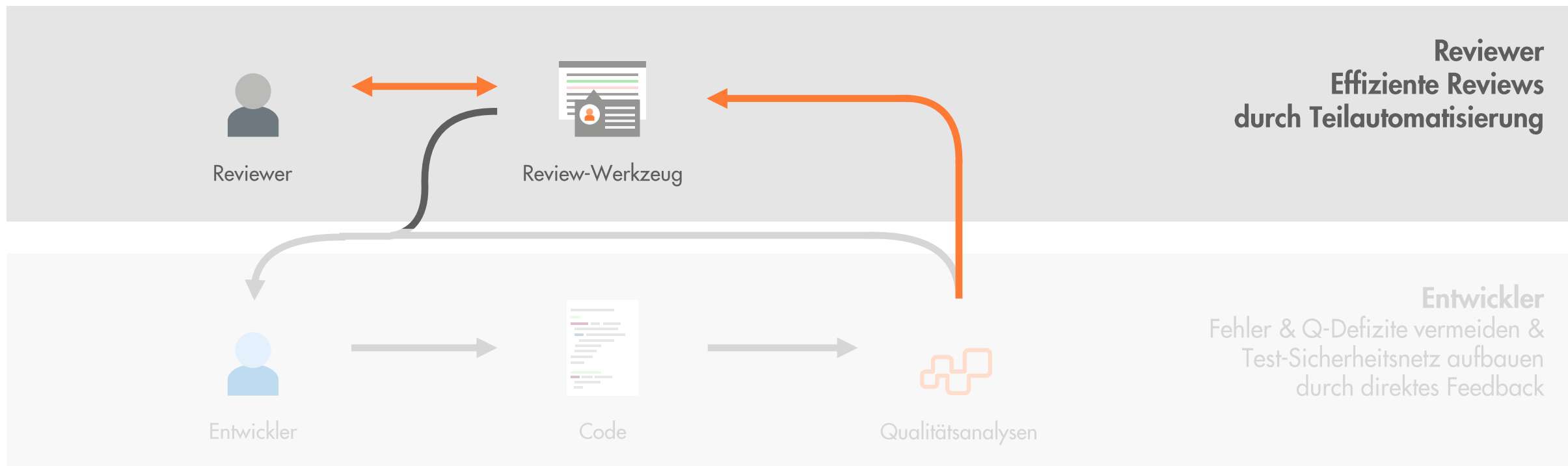
```
47  /** Equator used for comparing elements. */
48  private final IEquator<? super T> equator;
49  // TODO generate a new finding for precommit analysis
50
51  /** Length of {@link #a}. */
52  private int n;
53
54  /** Length of {@link #b}. */
55  private int m;
56
57  /** The maximal possible difference between {@link #a} and {@link #b}. */
58  private final int max;
59
60  /**
61   * Maximal size of the delta produced. If the "real" delta would be larger, a
62   * truncated delta will be created.
63   */
64  private final int maxDeltaSize;
```

Findings	Category	Group	Lines
lib/org.conqat.lib.commons/src/oi			
Clone with 2 instances of length!	Code Dup...	Cloning	115-...
Clone with 2 instances of length!	Code Dup...	Cloning	395-...
Clone with 2 instances of length!	Code Dup...	Cloning	163-...
Clone with 2 instances of length!	Code Dup...	Cloning	85-92
TODO finding	Documen...	Task tags	49
Unrelated Member Comment	Documen...	Comment...	57
Unrelated Member Comment	Documen...	Comment...	311
Unrelated Member Comment	Documen...	Comment...	355
Unrelated Member Comment	Documen...	Comment...	350

Pre-Commit Results	Category	Group	Lines
lib/org.conqat.lib.commons/src			
TODO finding	Document...	Task tags	49

Pre-Commit Results for cqse-all @ CQSE: 1 finding added. No findings removed (3 minutes ago) 49:58 CRLF UTF-8 Tab* | cr/20332_test_precommit

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



CCP-Integration durch Pull Request-Annotation

The screenshot displays a GitHub pull request page for a repository named 'Modify the code #7'. The pull request is open, showing 2 commits and 1 check. The check is titled 'Fix method not being called' and is associated with the Teamscale (SWE) service. The Teamscale (SWE) section shows a finding titled 'Added Findings' with the message 'This pull request would introduce 1 new finding'. The finding details include a Teamscale logo, a progress bar showing 1 finding and 2 test gaps (67%), and the specific finding: 'WhiteBoard/WBProcessManager.m#L115: TODO (AST) We should switch this to always log. (view in Teamscale)'. The annotations section shows a warning icon and the text 'Check warning on line 115 in WhiteBoard/WBProcessManager.m' with a link to the Teamscale finding details.

<> Code **Pull requests** 2 Projects Security Insights

Modify the code #7

Open asteckermeier wants to merge 2 commits into Sydro from merge_request_tga_demo_5 2

Conversation 0 Commits 2 Checks 1 Files changed 1

Fix method not being called ac2cbc2

Teamscale (SWE)

teamscale-findings Resolve

Teamscale (SWE) / teamscale-findings
Started 4m 8s ago

Added Findings

This pull request would introduce 1 new finding

DETAILS

Teamscale Findings 1 2 2 Test Gaps (67%)

- WhiteBoard/WBProcessManager.m#L115: TODO (AST) We should switch this to always log. (view in Teamscale)

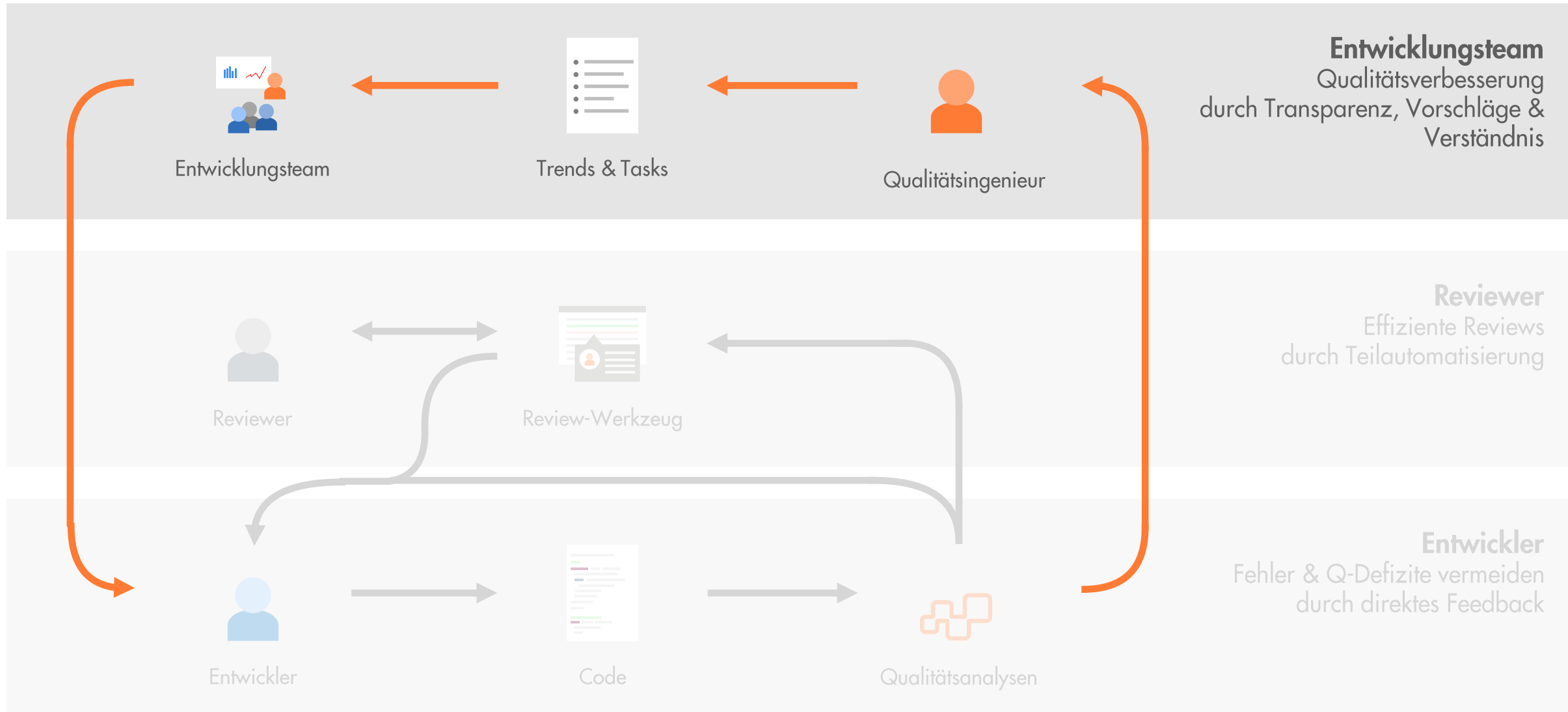
ANNOTATIONS

⚠ Check warning on line 115 in WhiteBoard/WBProcessManager.m

teamscale-swe / teamscale-findings
WhiteBoard/WBProcessManager.m#L115
TODO (AST) We should switch this to always log.
https://teamscale.my-company.com/findings.html#details/sam-faq-plms-project-whiteboard?=merge_request_tga_d

View more details on Teamscale (SWE)









Feedbackschleife 3: Q-Retrospektiven unterstützen Entwicklungsteams bei QS





Was ist eine Q-Retrospektive?



-  Regelmäßiger Workshop zur Rückschau auf Qualität
-  Team abholen, anhaltendes Q-Bewusstsein, gemeinsames Q-Verständnis
-  Entwicklungs-/Testteam & Qualitätsingenieur & ggf. weitere Beteiligte
-  Vorbereitet, moderiert und nachbereitet von Qualitätsingenieur
-  Wartbarkeit, Security, Korrektheit, Testen, ...
-  Monatlich/ quartärlich bzw. je Sprint/ Release
-  60-120 min
-  Basiert auf Q-Analysen

Diskussion von Q-Verbesserungsvorschlägen

Task 34 - Redundanz zwischen [REDACTED] UI5_TRANSLATION_TOOL und [REDACTED] TRANSLATION_TOOL



created by [REDACTED] Sep 16 2021 15:22, last updated Sep 17 2021 17:19

Assignee

Status

Open

Resolution

None

Description

Der Code im neuen Paket [REDACTED] UI5_TRANSLATION_TOOL ist oftmals redundant zum Code in [REDACTED] TRANSLATION_TOOL. Da beide Pakete wohl ähnliche Logik implementieren, sollte die Redundanz hier reduziert werden, z. B. durch verwenden von Basisklassen.

Tags

3 Retro

Redundanz

Edit Task

^ Findings

0 open 17 resolved 0 blacklisted

-Clone with 2 instances of length 18 in [REDACTED] TRANSLATION_TOOL/CLASS [REDACTED] EL_TRANSLATION.abap:979

-Clone with 2 instances of length 17 in [REDACTED] UI5_TRANSLATION_TOOL/CLASS [REDACTED] EL_UI5_TRANSLATION.abap:1208

Task 16 - Fehlende Best Practice für Berechtigungsprüfungen von Reports und Transaktionen



created by  Apr 08 2021 09:14, last updated Sep 19 2021 21:49

Assignee


Status

Closed

Resolution

Fixed

Description

Aufgrund des Feedbacks des -Teams wurden die Teamscale-Prüfungen für die Existenz von Berechtigungsprüfungen von Reports und Transaktionen (genauer: von Aufrufen einer Transaktion im Code mittels CALL TRANSACTION) deaktiviert.

Deshalb **fehlt aktuell eine Best Practice für die Berechtigungsprüfung** für Reports und Transaktionen.

Wir empfehlen, eine solche Best Practice zu definieren und mit Teamscale nachzuhalten, ob diese eingehalten wird.

Tags

1. Retro

Security

 Edit Task

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

Diskussion von Q-Indikatoren

Übersicht Produktqualität (Allgemein)

Quality Indicator (QI)	Trend	Quality Indicator (QI)	Trend
Architekturkonformität		Dokumentation	
Verletzungen		Kommentarvollständigkeit	
Nicht abgedeckte Typen		Struktur	
OWASP Dependency Findings		Methodenlänge	
Paketabhängigkeiten (ROT)		Schachtelungstiefe	
Paketabhängigkeiten (Gelb)		Dateigröße	
Redundanz		Testabdeckung	
Duplizierter Code		Testabdeckung	

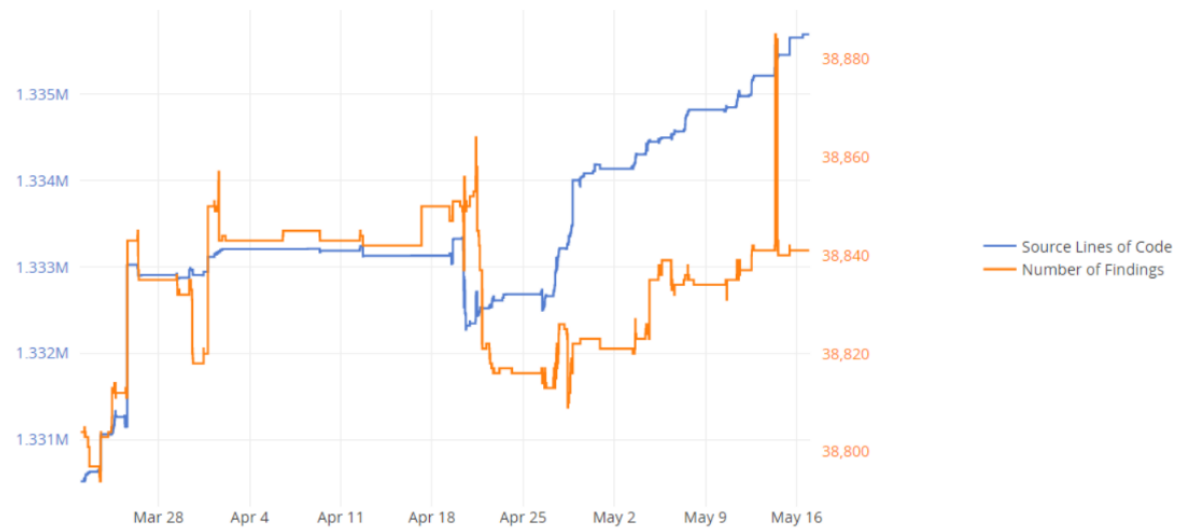
Übersicht Produktqualität (Codefindings)

Quality Indicator (QI)	Trend	Quality Indicator (QI)	Trend
Security		Verständlichkeit	
Kritische Security Findings		Kritische Verständlichkeitfindings	
Security Findings (Dichte)		Verständlichkeitfindings (Dichte)	
Korrektheit		Fehlerbehandlung	
Kritische Korrektheitsfindings		Kritische Fehlerbehandlungen	
Korrektheitsfindings (Dichte)		Fehlerbehandlungsfindings (Dichte)	
Effizienz			
Kritische Effizienzfindings			
Effizienzfindings (Dichte)			

- Q-Transparenz für Entwicklungsteam und Auftraggeber
- Einhaltung von Q-Indikatoren kann überprüft und ggfs. gegengesteuert werden

Diskussion von Q-Trends

Trend 22.März - 17. Mai

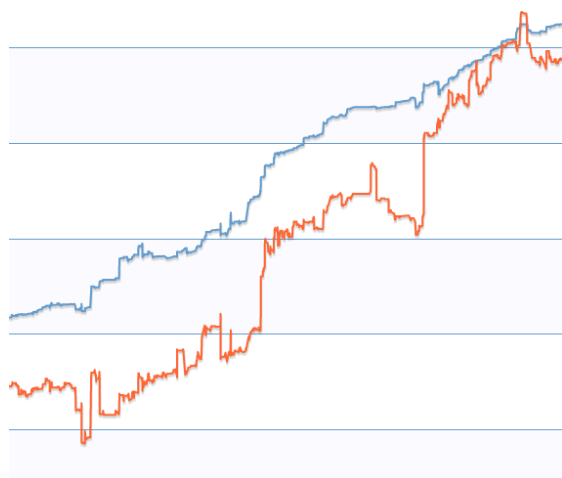


Ca. +5000 Zeilen (SLOC) Code, mehr neue Findings. Sprung am 14.05.: Code in DG_APL0UT_TO_SCE kopiert, welcher wieder auskommentiert wurde.

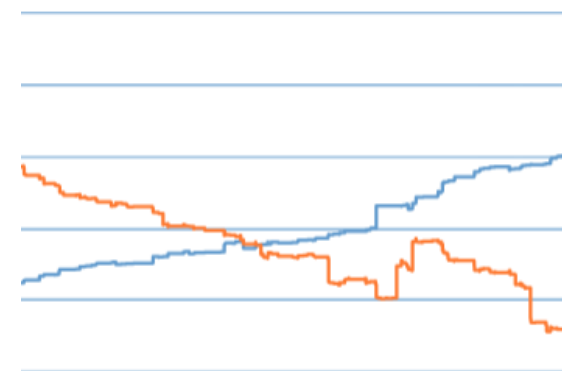
Findings-Delta

Findings -1 213 +1 176

Trend-Analyse: QS-Auswirkung



»Proportionales Wachstum«
→ Keine oder wirkungslose QS
=> Analyse & Gegensteuerung notwendig

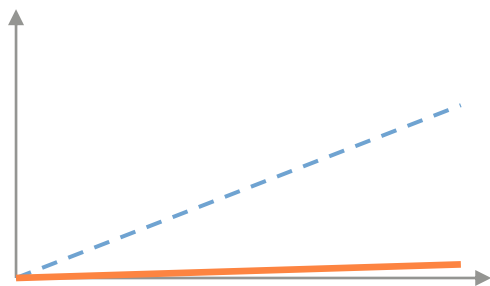


»X-Form«
→ Q-Verbesserung bei Codewachstum
=> Lob notwendig

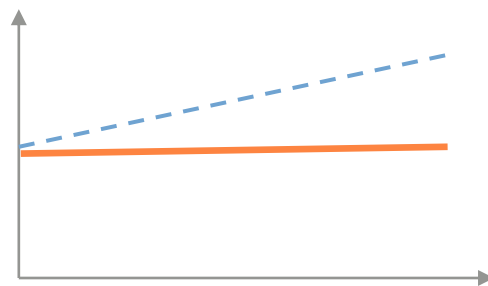
→ **Auswirkung der QS kann überprüft und ggfs. gegengesteuert werden**

Trend-Analyse: Erreichung von Q-Ziel

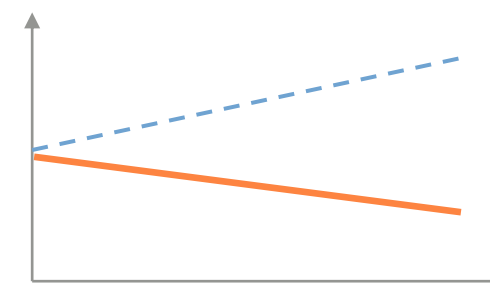
Neuentwicklung,
Q-Ziel »Perfekte Qualität«



Gewachsene Anwendung,
Q-Ziel »Qualität halten«

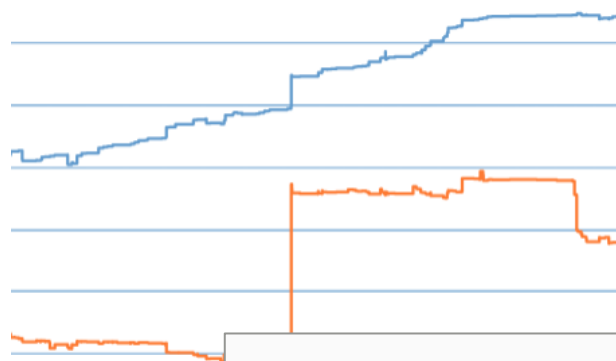


Gewachsene Anwendung,
Q-Ziel »Qualität mit Pfad-
finderregel verbessern«



→ Erreichung von Q-Zielen kann überprüft und ggfs. gegengesteuert werden

Trend-Analyse: Analyse von Q-Sprüngen



Projekte / Teamscale / Übergeordnete Breadc... / TS-38697

React warnings in Dev mode

+ Hinzufügen

Main HubSpot

Beschreibung

Since the update to React 18.3 we get even mode warnings in the browser console when running in Dev mode. The warnings are aimed to make the upgrade to React 19 easier as those features will be removed in React 19 (currently in beta).

See also [React 19 Upgrade Guide – React](#)



Merge branch 'ts/38697_semantic_warnings' into 'master'

by Florian Dreier as revision 9a6b3483 in master

Files: 110 added, 168 changed, 2 deleted

Issue: TS-38697: React warnings in Dev mode

Findings 757 48 27

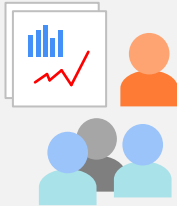
→ Ursachen für Q-Sprünge können verstanden und ggf. adressiert werden



⚠ WARNING
BRIGHT LIGHT
- Do not stare into light beam
- Do not view at close range
Failure to do so will cause
permanent eye damage

Trend 22.März - 17. Mai

38.880



- Qualität regelmäßig auf der Agenda
→ **Anhaltendes Q-Bewusstsein**
- Qualitätsdiskussionen und Erarbeitung von Best Practices
→ **Gemeinsames Q-Verständnis**
- Transparenz bezüglich Qualitätstrends & ggf. Identifizierung von Handlungsbedarf
→ **Mögliche Q-Steuerung**

→ **Abholung von & Unterstützung für Entwicklungsteam**
→ **Abholung von Management bezüglich Q-Steuerung**

System C

Quality In

Security

Rote Secu

Security-F

Codeanoma

Korrektthe

Codeanor

und

Berechtigungsprüfungen von Reports und Transaktionen (genauer: von Aufrufen einer Transaktion im Code mittels CALL TRANSACTION) deaktiviert.

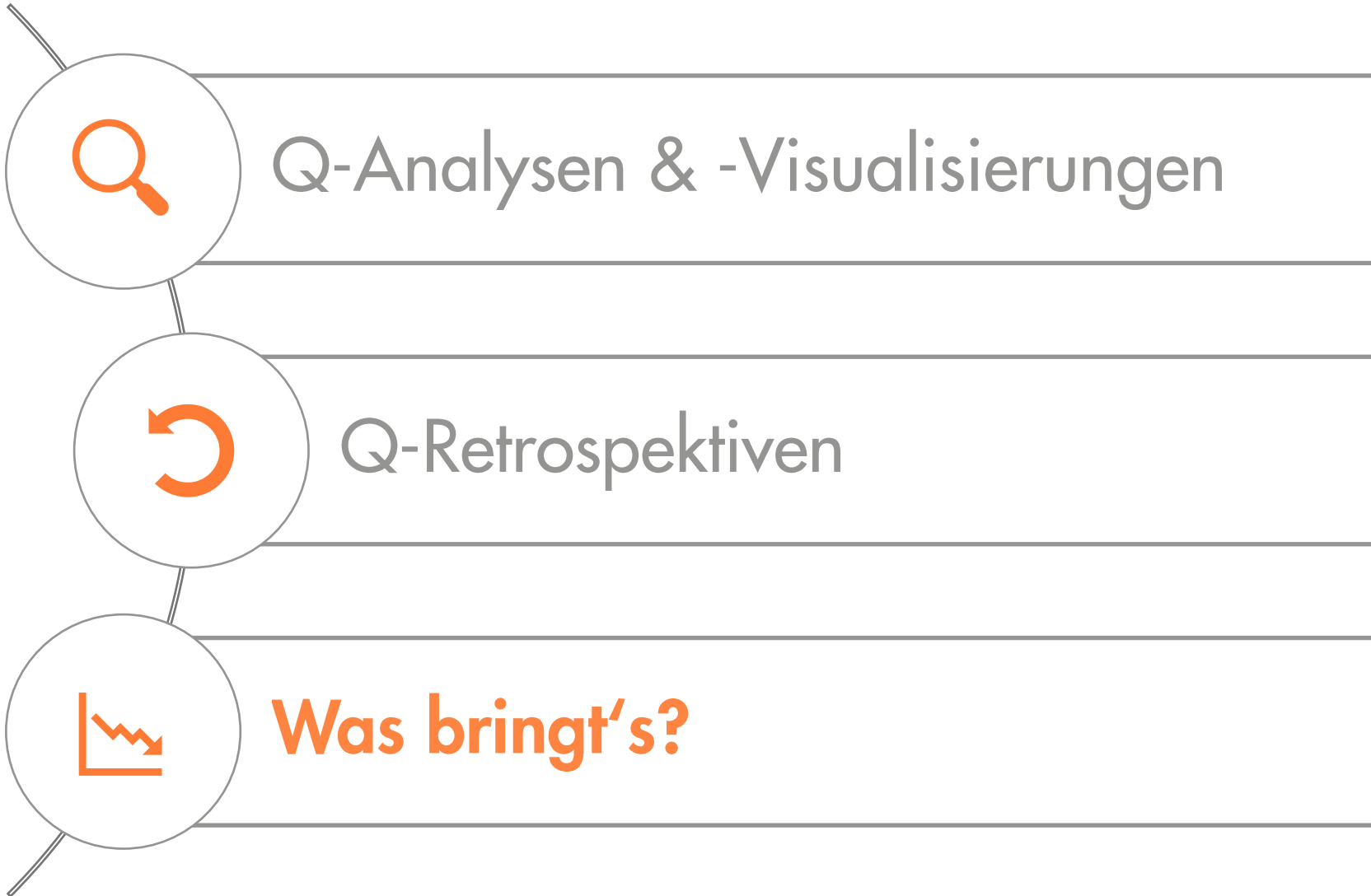
Description

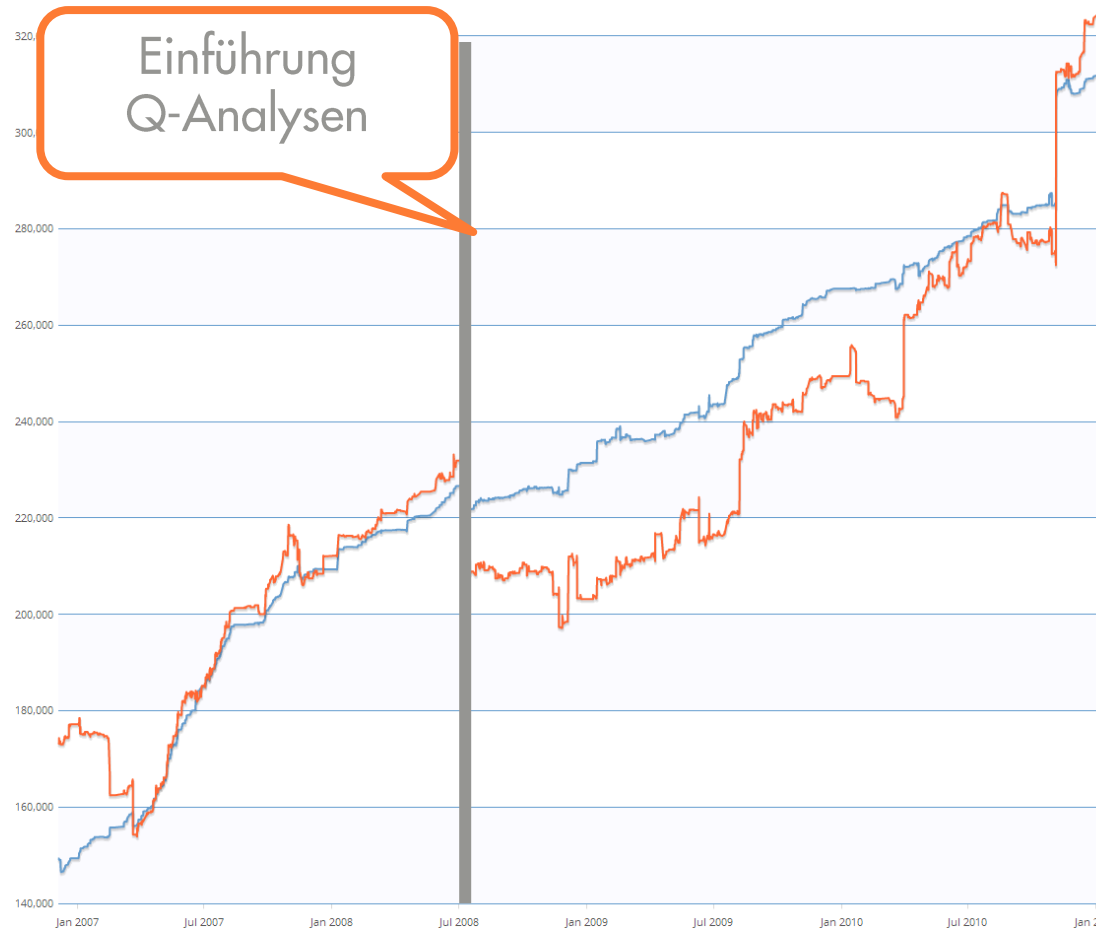
Deshalb fehlt aktuell eine Best Practice für die Berechtigungsprüfung für Reports und Transaktionen.

Wir empfehlen, eine solche Best Practice zu definieren und mit Teamscale nachzuhalten, ob diese eingehalten wird.

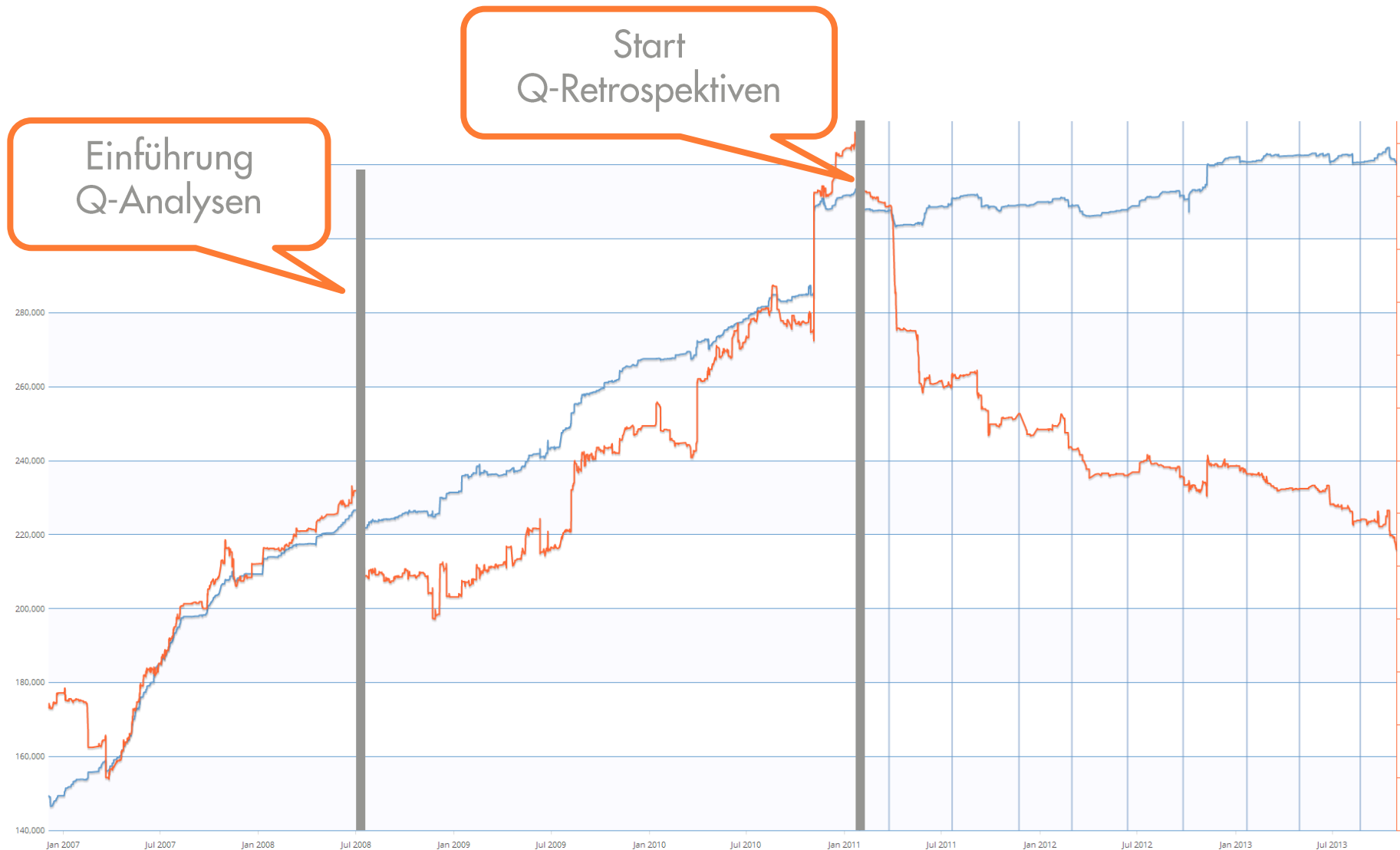
Tags

1. Retro Security



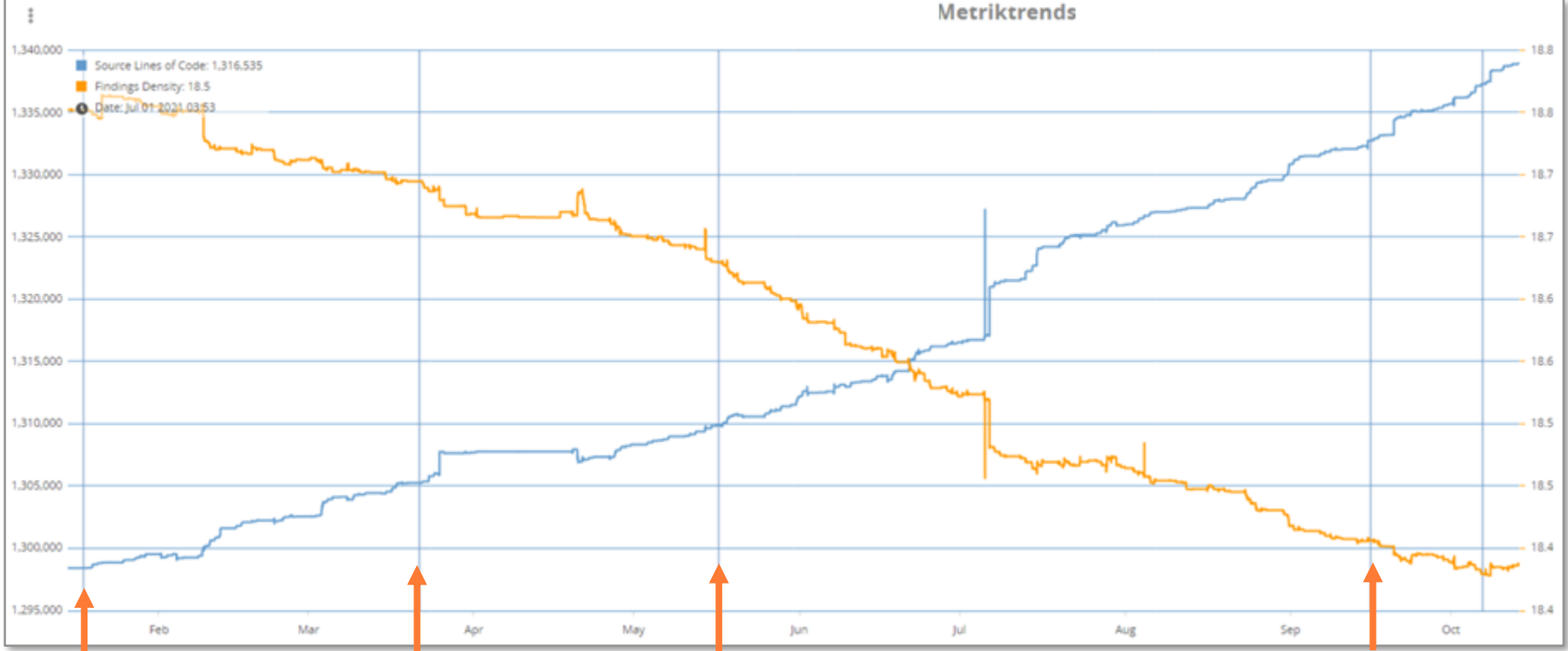


→ Q-Analysen haben i.d.R. nur einen kurzfristigen Effekt



→ Kombination aus Q-Analysen und Q-Retrospektiven hat i.d.R. einen anhaltenden Effekt

Metriktrends



Onboarding

1. Q-Retro

2. Q-Retro

3. Q-Retro

Metriktrends



Onboard



Zusammenfassung

Quality Indicator (QI)	Trend	Quality Indicator (QI)	Trend
Security		Redundanz	
Rote Security-Findings	→	Clone Coverage	↘
Security-Findings je 1.000 Codezeilen	→	Codestruktur	
Codeanomalien		Struktur: Prozedurlänge	↘
Korrektheit	→	Struktur: Schachtelungstiefe	↘
Codeanomalien je 1000 SLOC	↘		

1. Q-Retrospektive

System Quality Overview

Quality Indicator (QI)	Trend	Quality Indicator (QI)	Trend
Security		Redundanz	
Rote Security-Findings	→	Clone Coverage	↘
Security-Findings je 1.000 Codezeilen	↗	Codestruktur	
Codeanomalien		Struktur: Prozedurlänge	→
Korrektheit	↗	Struktur: Schachtelungstiefe	→
Codeanomalien je 1000 SLOC	→		

3. Q-Retrospektive

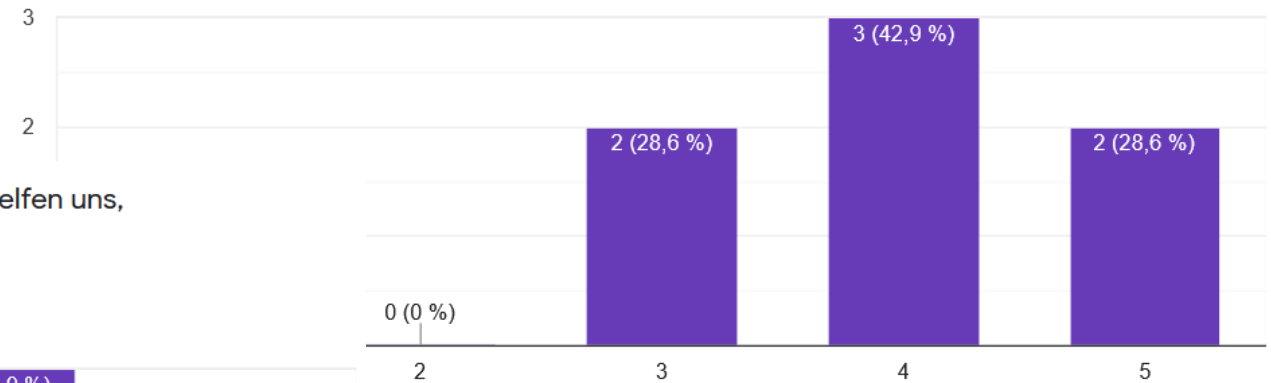
2. Q-Retrospektive

Quality Indicator (QI)	Trend	Quality Indicator (QI)	Trend
Security		Redundanz	
Rote Security-Findings	→	Clone Coverage	→
Security-Findings je 1.000 Codezeilen	→	Codestruktur	
Codeanomalien		Struktur: Prozedurlänge	→
Korrektheit	↗	Struktur: Schachtelungstiefe	→
Codeanomalien je 1000 SLOC	↗		

Entwicklerumfrage* zu Q-Retrospektiven

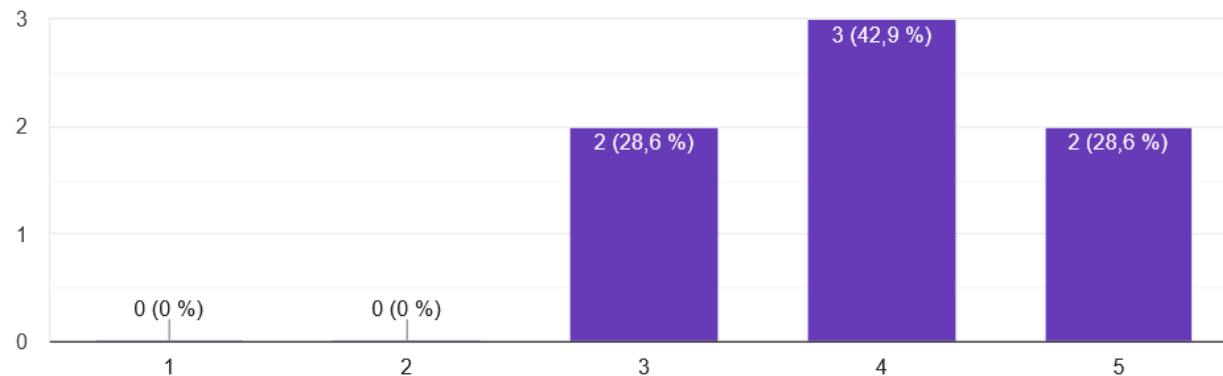
Wie ist Deine Einschätzung zu der Aussage "Durch die Qualitätsretrospektiven können wir unseren Code verbessern"?

7 Antworten



Wie ist Deine Einschätzung zu der Aussage "Die Qualitätsretrospektiven helfen uns, gemeinsame Best Practices und Standards zu etablieren"?

7 Antworten



* 7 Antworten im Mai 2021, 1: „Stimme überhaupt nicht zu“ – 5 „Stimme voll und ganz zu“



Christian Finkbeiner,
Softwarearchitekt,
SEW-EURODRIVE

»Der Weg zu einer **nachhaltigen
Qualitätsverbesserung** geht
nicht über ein Werkzeug,
sondern über den Prozess«

Vom Wiegen allein wird die Sau nicht fett

Wie ProSiebenSat.1, Munich Re, Allianz und Co.
Softwarequalität verbessern

Dienstag, 11. März

11:00 bis 12:00 Uhr

Online und kostenlos

Jetzt anmelden: www.tmscl.me/sit-253-oop



Dr. Sven Amann
(CQSE)

Zusammenfassung



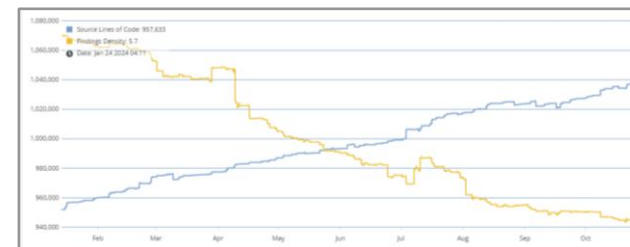
→ Abholung aller Beteiligten ist essenziell für wirksame QS



→ Q-Retrospektiven schaffen gemeinsames Q-Verständnis im Entwicklungsteam und Q-Steuerbarkeit für Management



→ Q-Analysen & -Visualisierungen machen Qualität sichtbar und diskutierbar



→ Kombination von Q-Analysen und Q-Retrospektiven führt zu wirksamer, nachweisbarer Qualitätsverbesserung

Folien & Kontakt - Ich freue mich auf Fragen & Austausch 😊



Vortragsfolien:
tmscl.me/3008JnF



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