

Wie hole ich die anderen ins »Qualitätsboot«?

Erfahrungen aus 10 Jahren Qualitätsretrospektiven
MedConf 2025

Dr. Tobias Röhm



Freelancer



2004 - 2010

2010 - 2015

2015 - Heute

 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

Projektleiter



 Entwicklerin

 Tester

 Architekt

Softwarequalität

 Product Owner

 Anwendungs-
verantwortliche

 Projektleiterin

 QS-Spezialist

 Managerin

→ **Wirksame Qualitätssicherung erfordert Einbeziehung aller Beteiligten**



Q-Analysen & -Visualisierungen



Q-Retrospektiven



Was bringt's?



Q-Analysen & -Visualisierungen



Q-Retrospektiven



Was bringt's?



Teamscale

(Kritische) Anforderungen

Testfälle

Show test results from: **All**

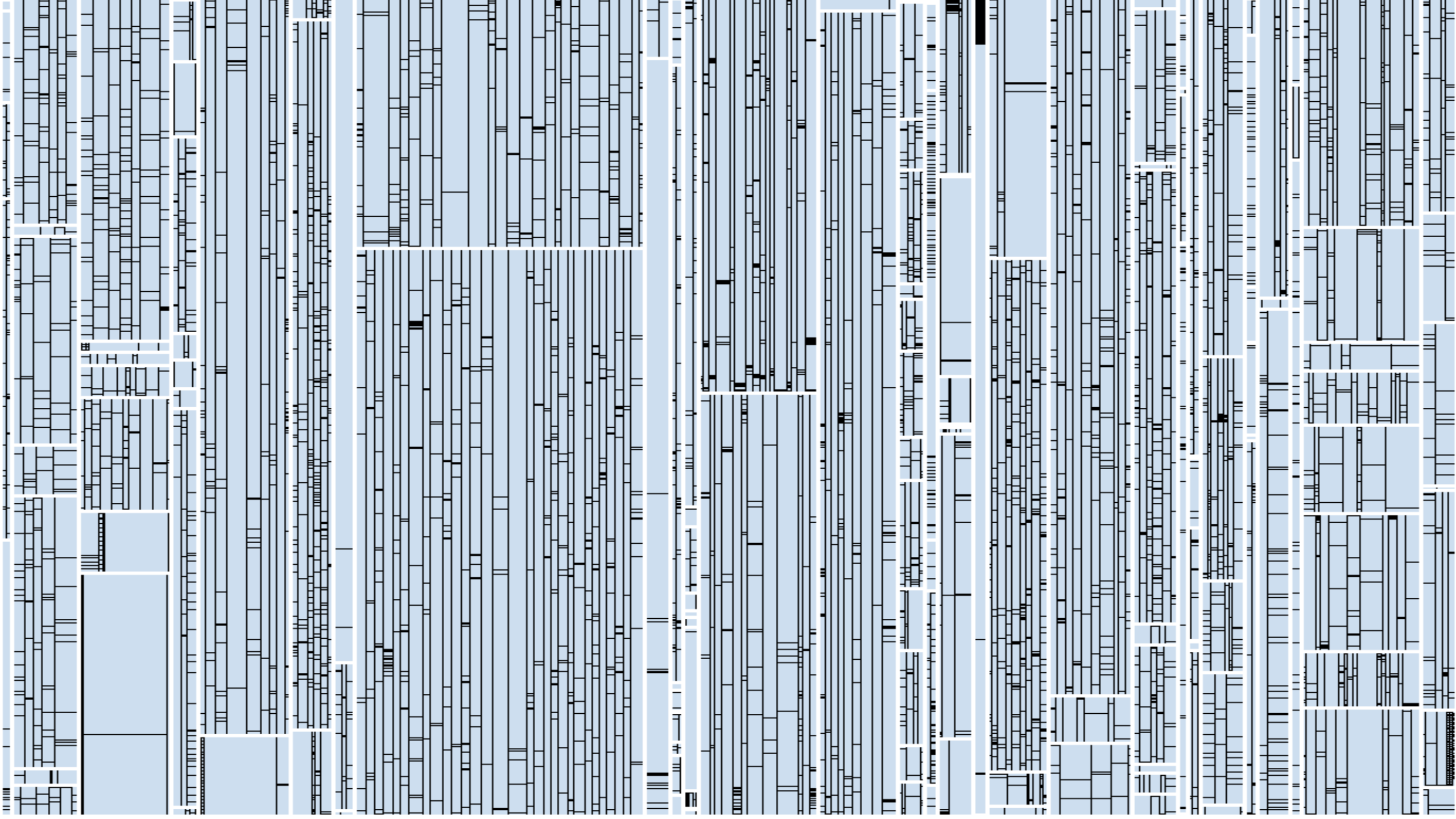
Tests	Specification Items		REQ DP-539	REQ DP-540	TST DP-547	TST DP-548
	Σ	1		0		
data_structures/tests/stack_tests.c/StackTests/Push	1					
data_structures/tests/stack_tests.c/StackTests/Pop	1					
data_structures/tests/stack_tests.c/StackTests/Peek	1					

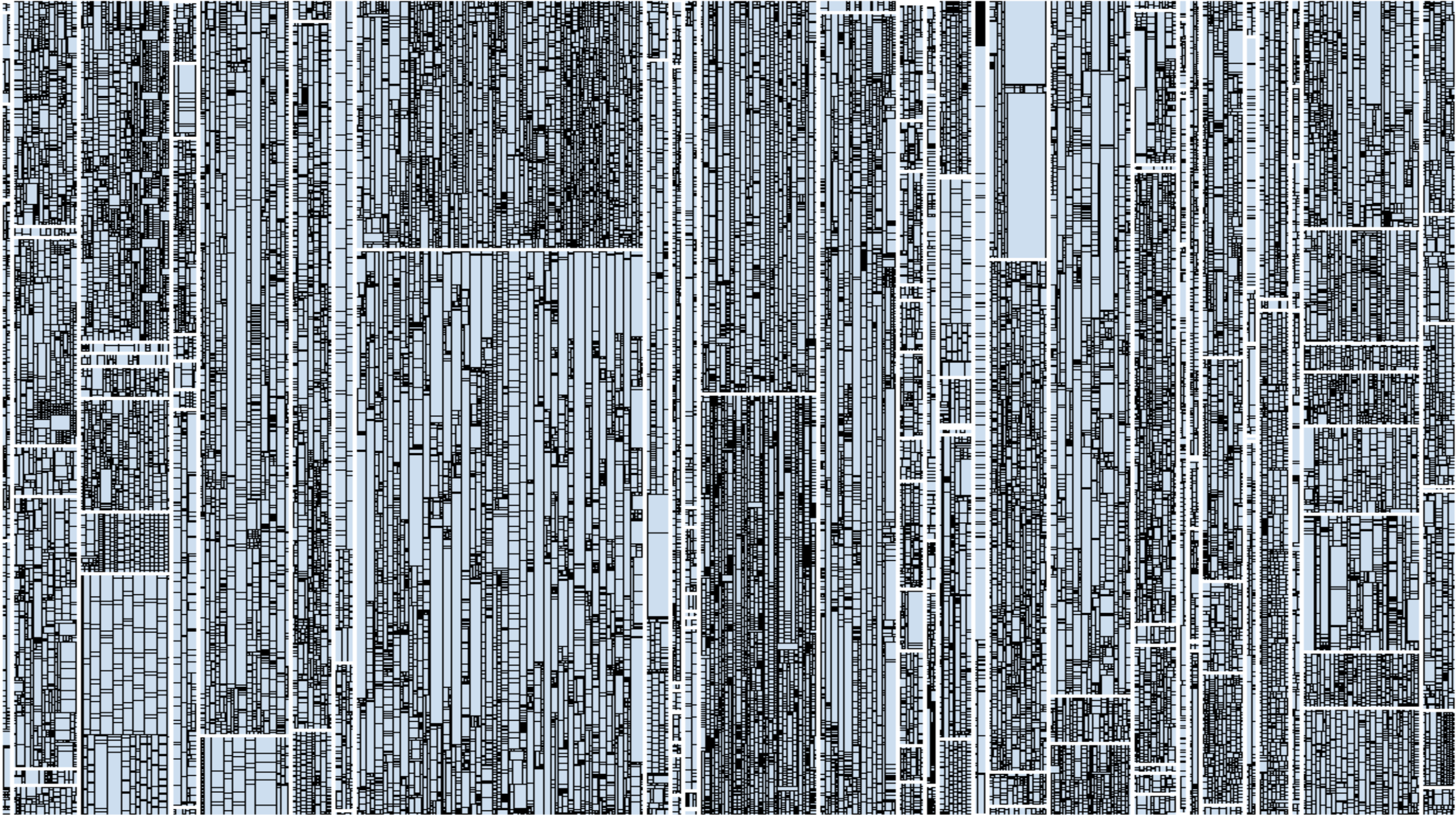
Linux

Windows

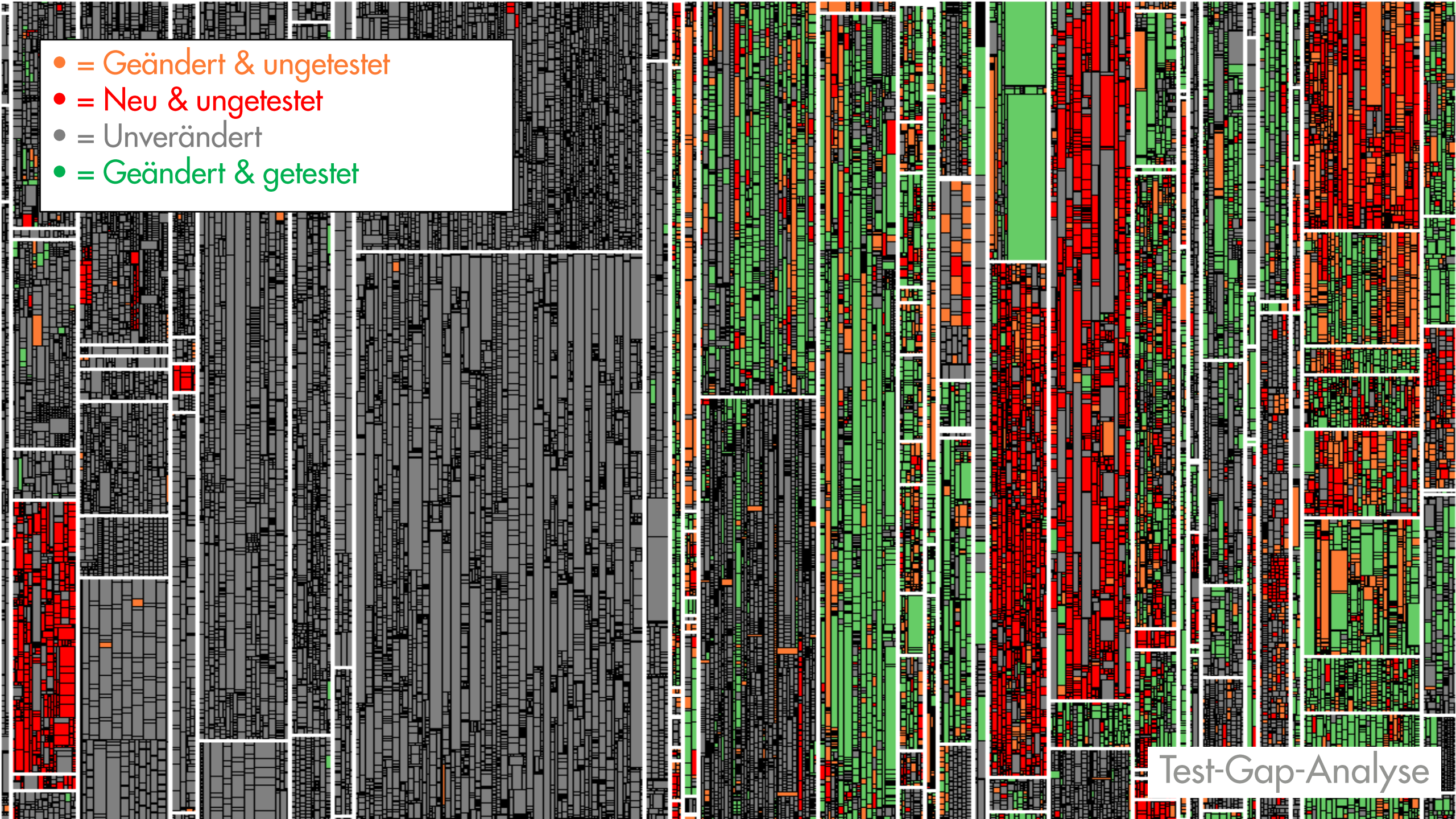
SKIPPED

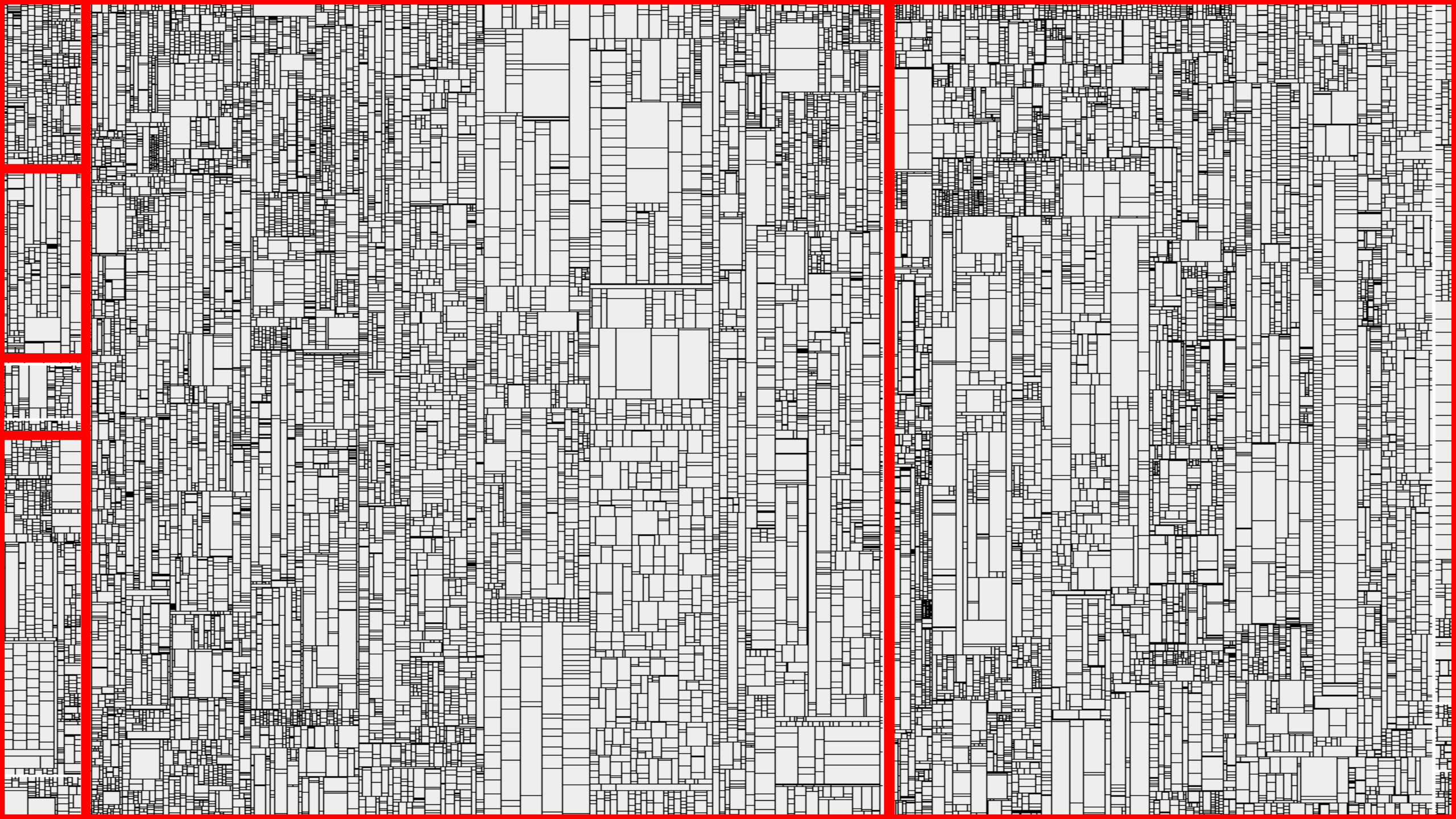
FAILURE

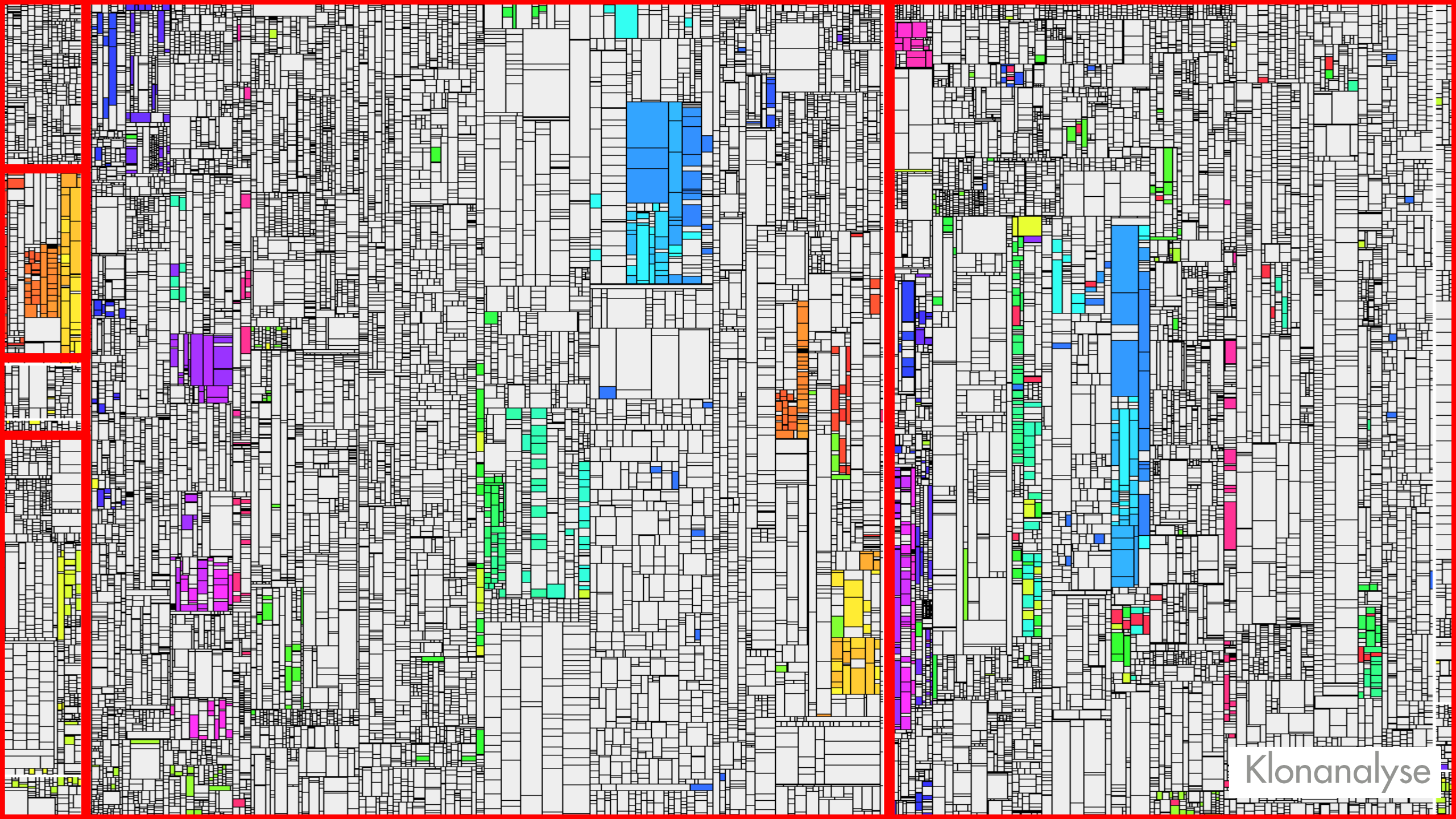




- = Geändert & ungetestet
- = Neu & ungetestet
- = Unverändert
- = Geändert & getestet







```

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`, `break`, or `throw`) 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 an 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 [Pittman, 1988](#), [1991](#), [1993](#), [1994](#), [1995](#), [1996](#), [1997](#), [1998](#), [1999](#), [2000](#), [2001](#), [2002](#), [2003](#), [2004](#), [2005](#), [2006](#), [2007](#), [2008](#), [2009](#), [2010](#), [2011](#), [2012](#), [2013](#), [2014](#), [2015](#), [2016](#), [2017](#), [2018](#), [2019](#), [2020](#), [2021](#), [2022](#), [2023](#), [2024](#), [2025](#), [2026](#), [2027](#), [2028](#), [2029](#), [2030](#), [2031](#), [2032](#), [2033](#), [2034](#), [2035](#), [2036](#), [2037](#), [2038](#), [2039](#), [2040](#), [2041](#), [2042](#), [2043](#), [2044](#), [2045](#), [2046](#), [2047](#), [2048](#), [2049](#), [2050](#), [2051](#), [2052](#), [2053](#), [2054](#), [2055](#), [2056](#), [2057](#), [2058](#), [2059](#), [2060](#), [2061](#), [2062](#), [2063](#), [2064](#), [2065](#), [2066](#), [2067](#), [2068](#), [2069](#), [2070](#), [2071](#), [2072](#), [2073](#), [2074](#), [2075](#), [2076](#), [2077](#), [2078](#), [2079](#), [2080](#), [2081](#), [2082](#), [2083](#), [2084](#), [2085](#), [2086](#), [2087](#), [2088](#), [2089](#), [2090](#), [2091](#), [2092](#), [2093](#), [2094](#), [2095](#), [2096](#), [2097](#), [2098](#), [2099](#), [2100](#), [2101](#), [2102](#), [2103](#), [2104](#), [2105](#), [2106](#), [2107](#), [2108](#), [2109](#), [2110](#), [2111](#), [2112](#), [2113](#), [2114](#), [2115](#), [2116](#), [2117](#), [2118](#), [2119](#), [2120](#), [2121](#), [2122](#), [2123](#), [2124](#), [2125](#), [2126](#), [2127](#), [2128](#), [2129](#), [2130](#), [2131](#), [2132](#), [2133](#), [2134](#), [2135](#), [2136](#), [2137](#), [2138](#), [2139](#), [2140](#), [2141](#), [2142](#), [2143](#), [2144](#), [2145](#), [2146](#), [2147](#), [2148](#), [2149](#), [2150](#), [2151](#), [2152](#), [2153](#), [2154](#), [2155](#), [2156](#), [2157](#), [2158](#), [2159](#), [2160](#), [2161](#), [2162](#), [2163](#), [2164](#), [2165](#), [2166](#), [2167](#), [2168](#), [2169](#), [2170](#), [2171](#), [2172](#), [2173](#), [2174](#), [2175](#), [2176](#), [2177](#), [2178](#), [2179](#), [2180](#), [2181](#), [2182](#), [2183](#), [2184](#), [2185](#), [2186](#), [2187](#), [2188](#), [2189](#), [2190](#), [2191](#), [2192](#), [2193](#), [2194](#), [2195](#), [2196](#), [2197](#), [2198](#), [2199](#), [2200](#), [2201](#), [2202](#), [2203](#), [2204](#), [2205](#), [2206](#), [2207](#), [2208](#), [2209](#), [2210](#), [2211](#), [2212](#), [2213](#), [2214](#), [2215](#), [2216](#), [2217](#), [2218](#), [2219](#), [2220](#), [2221](#), [2222](#), [2223](#), [2224](#), [2225](#), [2226](#), [2227](#), [2228](#), [2229](#), [2230](#), [2231](#), [2232](#), [2233](#), [2234](#), [2235](#), [2236](#), [2237](#), [2238](#), [2239](#), [2240](#), [2241](#), [2242](#), [2243](#), [2244](#), [2245](#), [2246](#), [2247](#), [2248](#), [2249](#), [2250](#), [2251](#), [2252](#), [2253](#), [2254](#), [2255](#), [2256](#), [2257](#), [2258](#), [2259](#), [2260](#), [2261](#), [2262](#), [2263](#), [2264](#), [2265](#), [2266](#), [2267](#), [2268](#), [2269](#), [2270](#), [2271](#), [2272](#), [2273](#), [2274](#), [2275](#), [2276](#), [2277](#), [2278](#), [2279](#), [2280](#), [2281](#), [2282](#), [2283](#), [2284](#), [2285](#), [2286](#), [2287](#), [2288](#), [2289](#), [2290](#), [2291](#), [2292](#), [2293](#), [2294](#), [2295](#), [2296](#), [2297](#), [2298](#), [2299](#), [2300](#), [2301](#), [2302](#), [2303](#), [2304](#), [2305](#), [2306](#), [2307](#), [2308](#), [2309](#), [2310](#), [2311](#), [2312](#), [2313](#), [2314](#), [2315](#), [2316](#), [2317](#), [2318](#), [2319](#), [2320](#), [2321](#), [2322](#), [2323](#), [2324](#), [2325](#), [2326](#), [2327](#), [2328](#), [2329](#), [2330](#), [2331](#), [2332](#), [2333](#), [2334](#), [2335](#), [2336](#), [2337](#), [2338](#), [2339](#), [2340](#), [2341](#), [2342](#), [2343](#), [2344](#), [2345](#), [2346](#), [2347](#), [2348](#), [2349](#), [2350](#), [2351](#), [2352](#), [2353](#), [2354](#), [2355](#), [2356](#), [2357](#), [2358](#), [2359](#), [2360](#), [2361](#), [236](#)

The screenshot shows a SQL query in a code editor. A callout box with an orange border points to the line `CALL TRANSACTION 'LERN' USING 'LERN' MODE 'NORM'`. The text inside the box says "Berechtigungsprüfung fehlt" (Permission check is missing). The query is as follows:

```
536 CALL TRANSACTION 'LERN' USING 'LERN' MODE 'NORM'
537 MESSAGES INTO 'LERN'.
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```

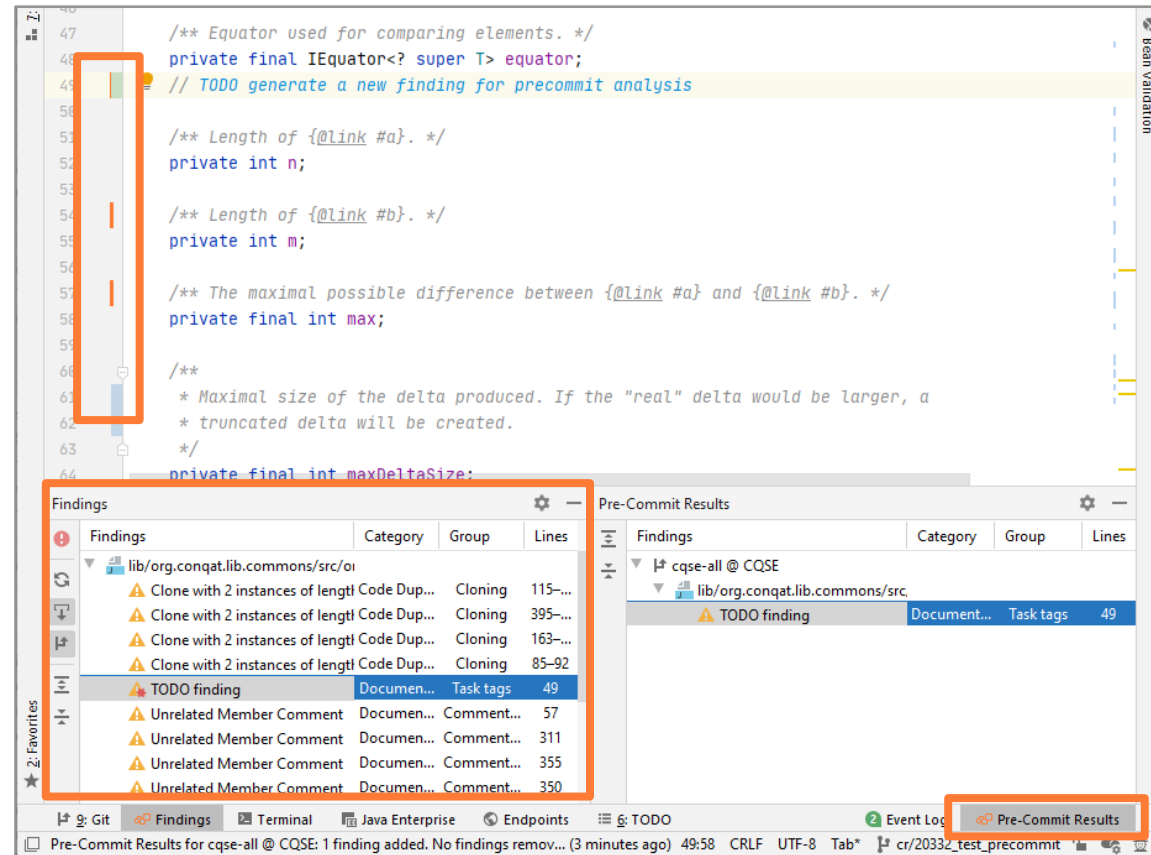
Berechtigungsprüfung fehlt

Static Application Security Testing (SAST)

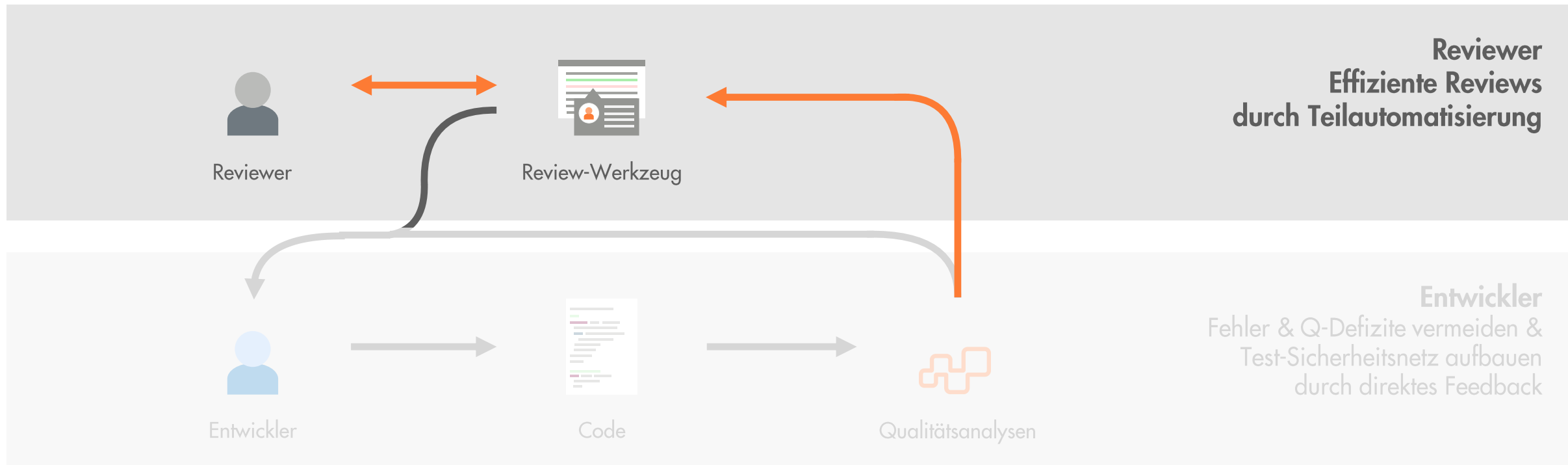
Feedbackschleife 1: Statische Codeanalyse hilft Entwicklern, Fehler & Q-Defizite zu vermeiden



IDE-Integration



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 titled "Modify the code #7". The interface includes tabs for Code, Pull requests (2), Projects, Security, and Insights. The pull request is open, showing 2 commits and 1 check. A Teamscale (SWE) finding is highlighted, indicating a new finding introduced by the pull request. The finding details show a warning icon, the Teamscale logo, and a progress bar with 1 finding and 2 test gaps (67%). The finding text is: "WhiteBoard/WBProcessManager.m#L115: TODO (AST) We should switch this to always log." The annotations section provides further details, including the file path and a link to view more details on Teamscale (SWE).

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

Modify the code #7

Open asteckermeier wants to merge 2 commits into `Sydra` 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

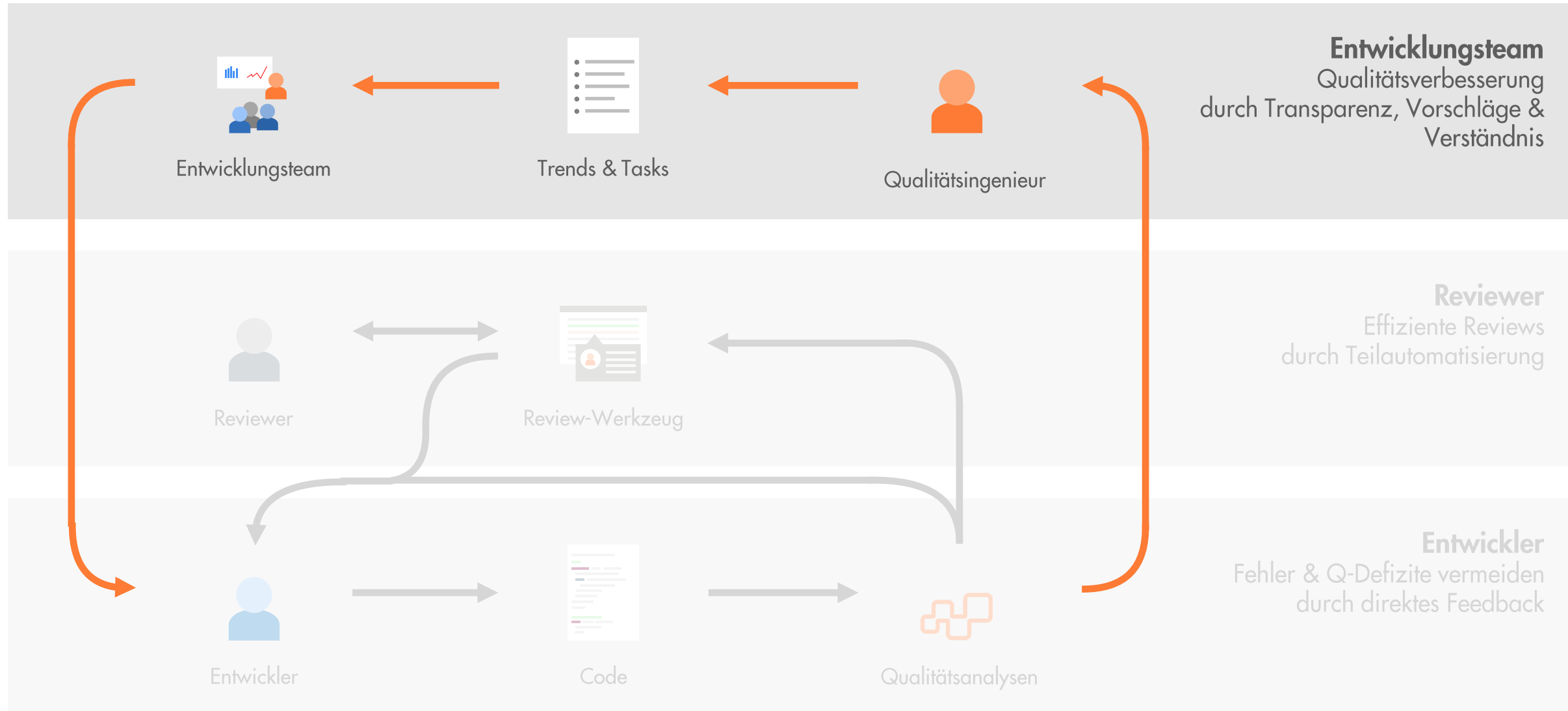
WhiteBoard/WBProcessManager.m#L115

TODO (AST) We should switch this to always log.

https://teamscale.my-company.com/findings.html#details/sam-faqplms-project-whiteboard?merge_request_tga_d

View more details on Teamscale (SWE)

Feedbackschleife 3: Q-Retrospektiven unterstützen Entwicklungsteams bei QS





Q-Analysen & -Visualisierungen



Q-Retrospektiven



Was bringt's?

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

(Kritische) Anforderungen

Testfälle

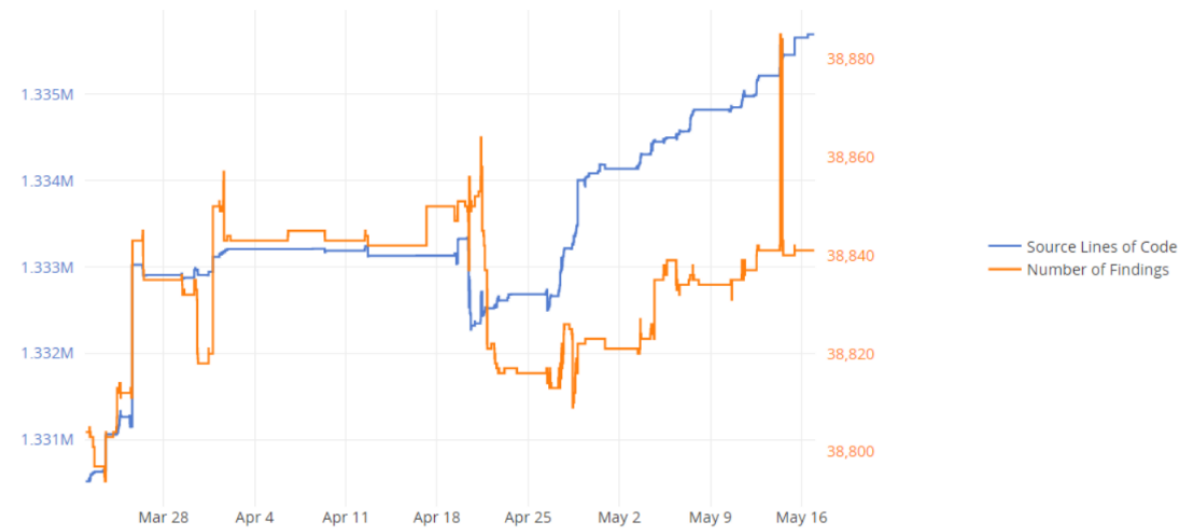
Show test results from: **All**

Tests	Specification Items	REQ DP-539	REQ DP-540	TST DP-547	TST DP-548
	Σ	1 	0 	1 	1 
data_structures/tests/stack_tests.c/StackTests/Push	1 				
data_structures/tests/stack_tests.c/StackTests/Pop	1 				
data_structures/tests/stack_tests.c/StackTests/Peek	1 				

Linux SKIPPED
Windows FAILURE

Diskussion von Q-Trends

Trend 22.März - 17. Mai

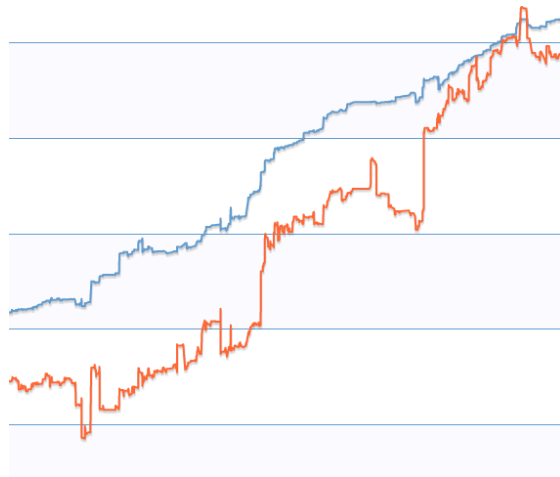


Ca. +5000 Zeilen (SLOC) Code, mehr neue Findings. Sprung am 14.05.: Code in DG_APL0UT_TO_SCE kopiert, welcher wieder auskommentiert wurde.

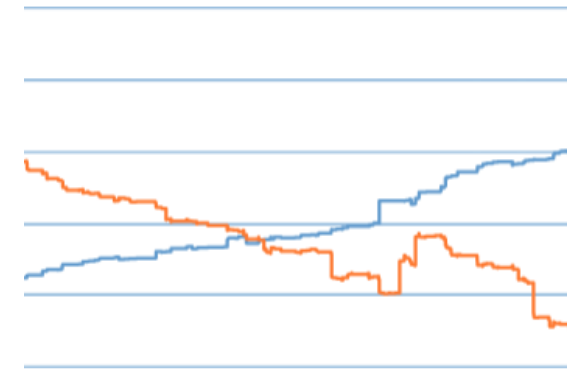
Findings-Delta

Findings 🔴 213 🟢 176

Trend-Analyse: QS-Auswirkung



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

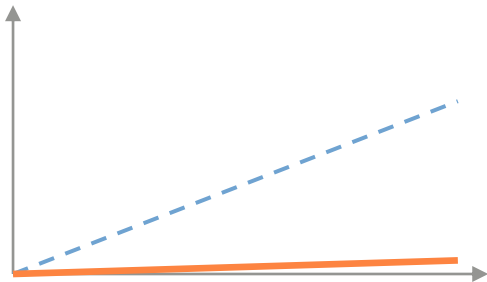


»X-Form«
→ Q-Verbesserung bei Codewachstum
=> Lob notwendig

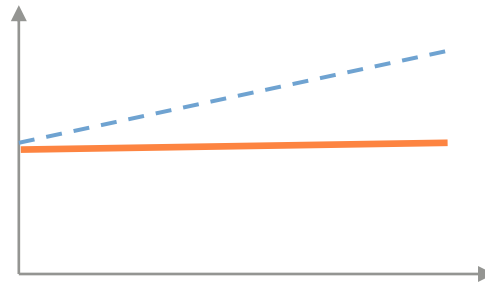
→ **Auswirkung der QS kann überprüft und ggfs. gegengesteuert werden**

Trend-Analyse: Erreichung von Q-Ziel

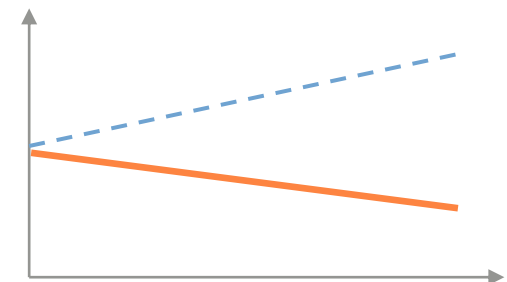
Neuentwicklung oder
Standard-Verpflichtung,
Q-Ziel »Perfekte Qualität«



Gewachsene Anwendung,
Q-Ziel »Qualität halten«



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



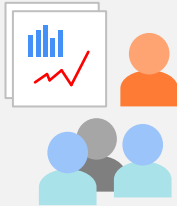
→ Erreichung von Q-Zielen kann überprüft und ggfs. gegengesteuert werden



⚠ 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

38.800



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

System C

Quality In

Security

Rote Secu

Security-F

Codeanoma

Korrektur

Codeanor

und

Description

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



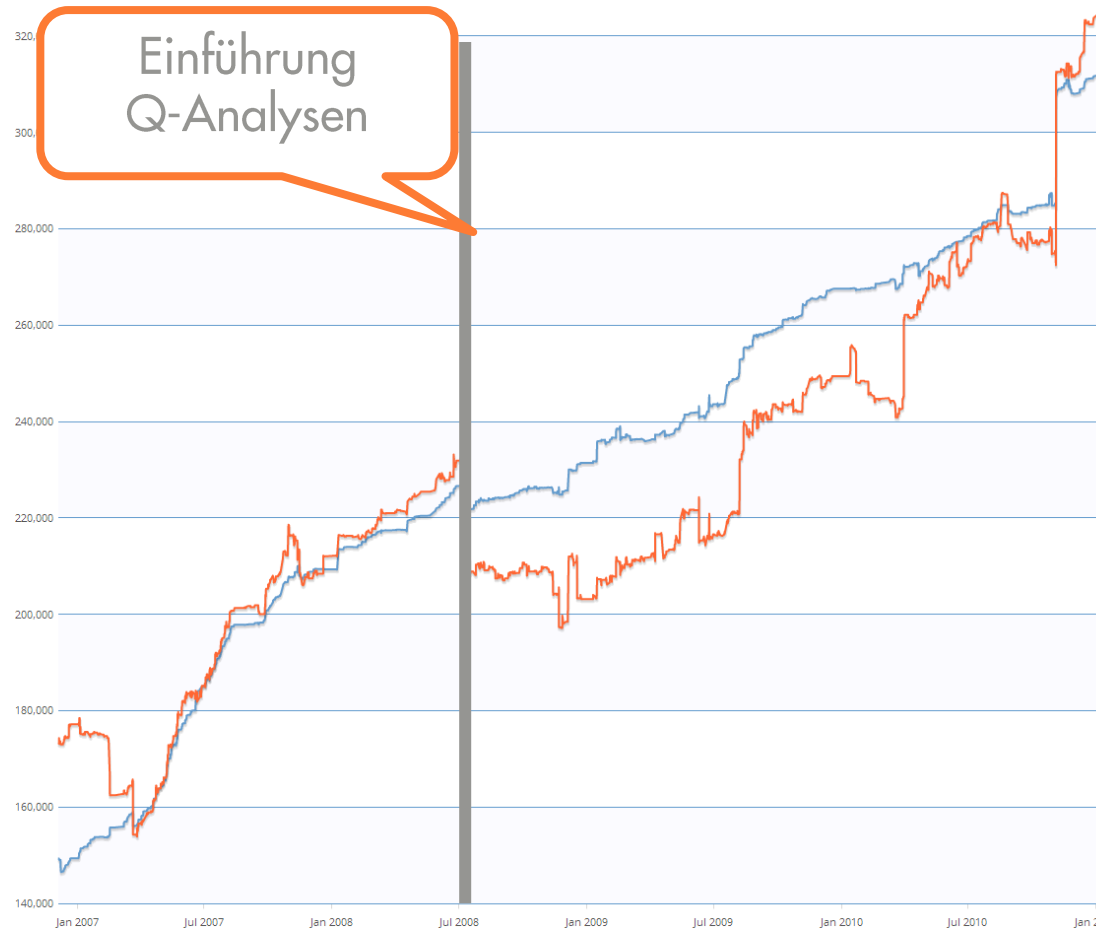
Q-Analysen & -Visualisierungen



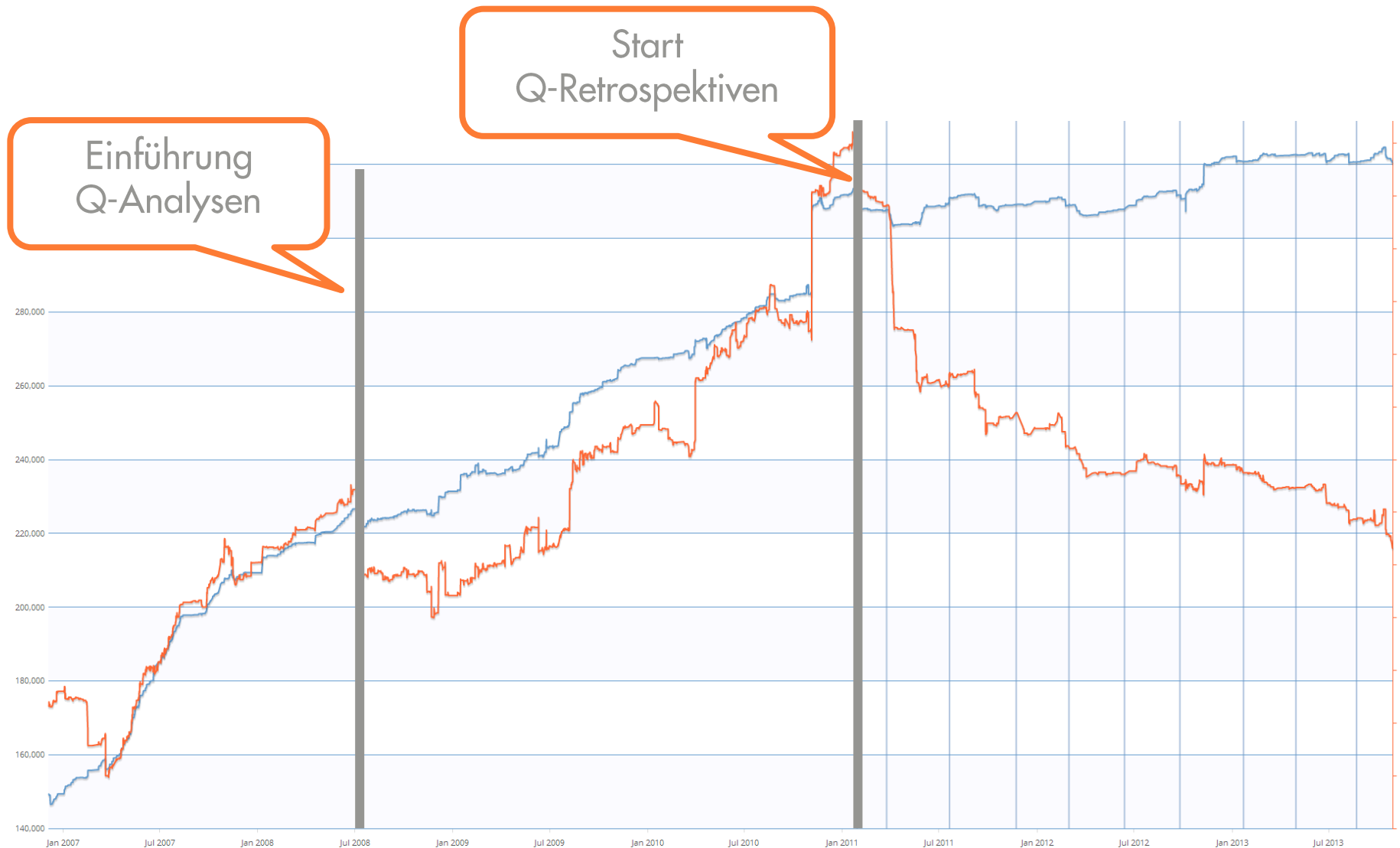
Q-Retrospektiven



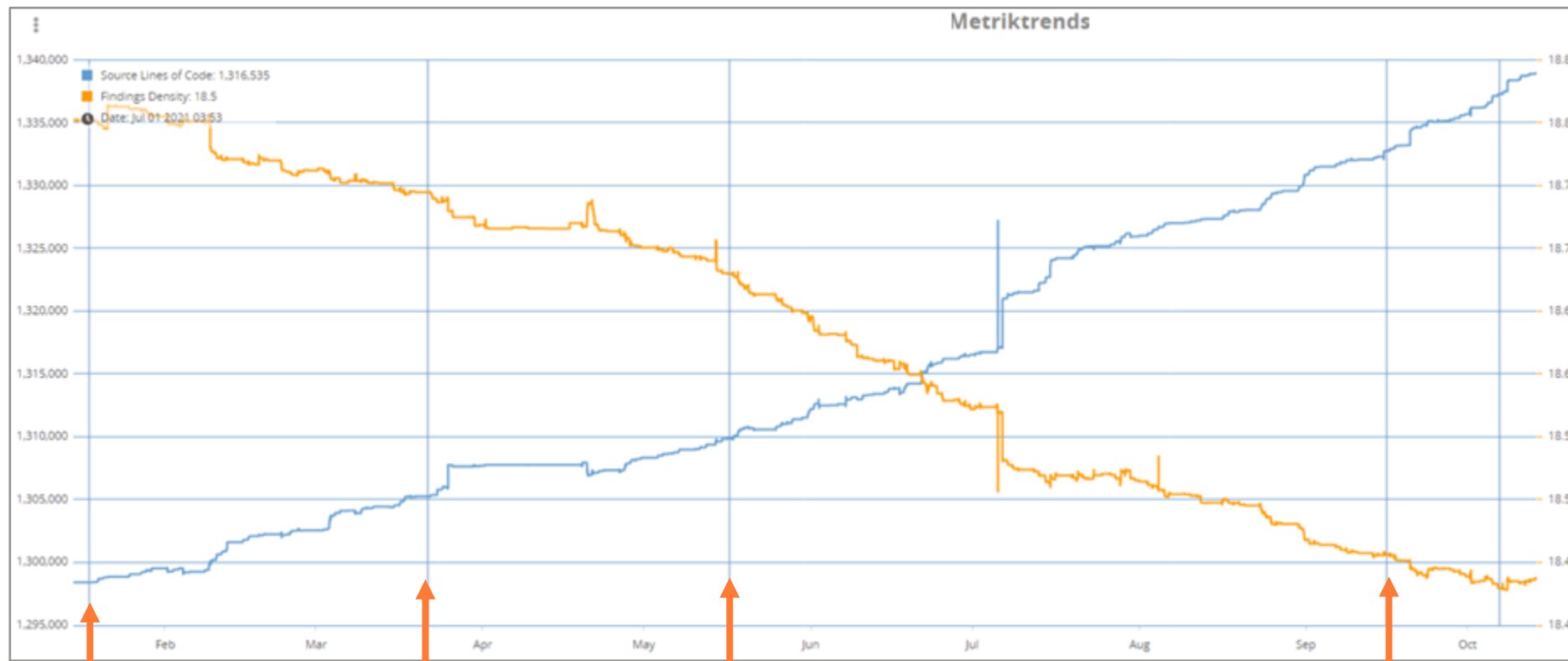
Was bringt's?



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



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



Onboarding

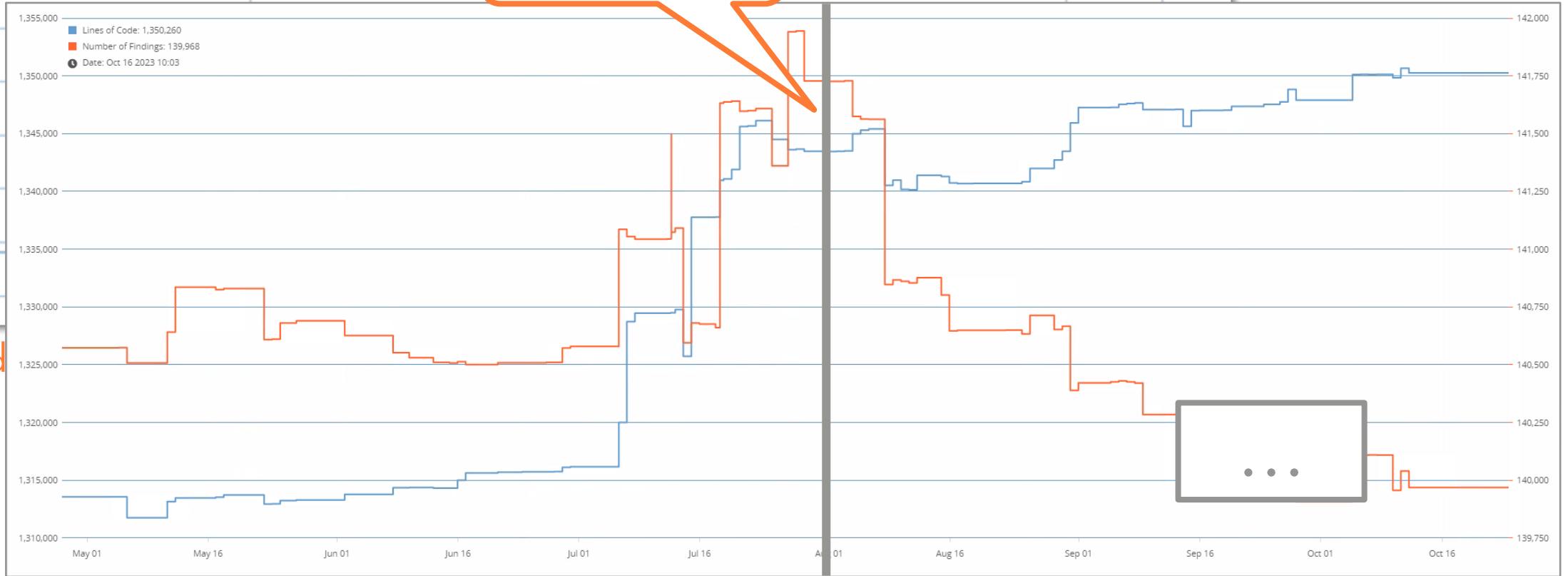
1. Q-Retro

2. Q-Retro

3. Q-Retro



Onboard



Zusammenfassung

Quality Indicator (QI)	Trend
Security	
Rote Security-Findings	→
Security-Findings je 1.000 Codezeilen	→
Codeanomalien	
Korrektheit	→
Codeanomalien je 1000 SLOC	↘

1. Q-Retrospektive

Quality Indicator (QI)	Trend
Redundanz	
Clone Coverage	↘
Codestruktur	
Struktur: Prozedurlänge	↘
Struktur: Schachtelungstiefe	↘

System Quality Overview

Quality Indicator (QI)	Trend
Security	
Rote Security-Findings	→
Security-Findings je 1.000 Codezeilen	↗
Codeanomalien	
Korrektheit	↗
Codeanomalien je 1000 SLOC	→

3. Q-Retrospektive

Quality Indicator (QI)	Trend
Redundanz	
Clone Coverage	↘
Codestruktur	
Struktur: Prozedurlänge	→
Struktur: Schachtelungstiefe	→

2. Q-Retrospektive

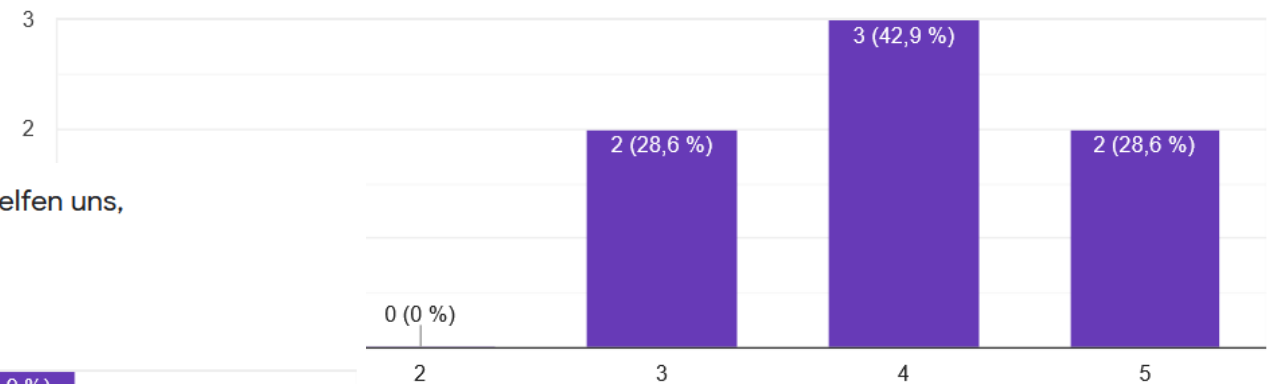
Quality Indicator (QI)	Trend
Security	
Rote Security-Findings	→
Security-Findings je 1.000 Codezeilen	→
Codeanomalien	
Korrektheit	↗
Codeanomalien je 1000 SLOC	↗

Quality Indicator (QI)	Trend
Redundanz	
Clone Coverage	→
Codestruktur	
Struktur: Prozedurlänge	→
Struktur: Schachtelungstiefe	→

Entwicklerumfrage* zu Q-Retrospektiven

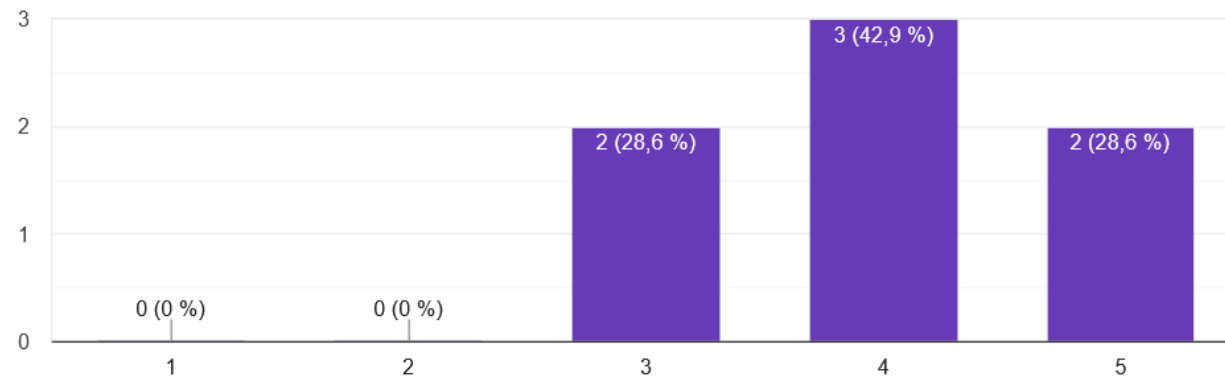
Wie ist Deine Einschätzung zu der Aussage "Durch die Qualitätsretrospektiven können wir unseren Code verbessern"?

7 Antworten



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

7 Antworten



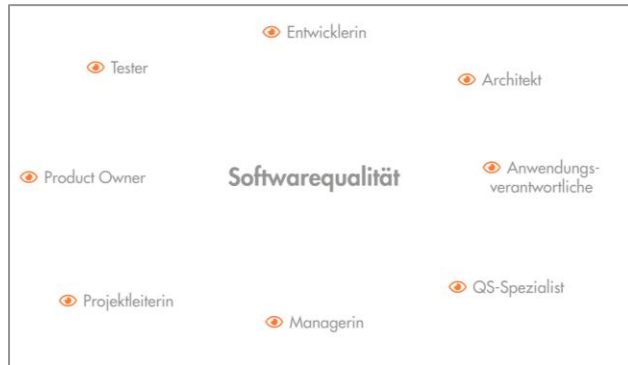
* 7 Antworten im Mai 2021, 1: „Stimme überhaupt nicht zu“ – 5 „Stimme voll und ganz zu“



Christian Finkbeiner,
Softwarearchitekt,
SEW-EURODRIVE

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

Zusammenfassung



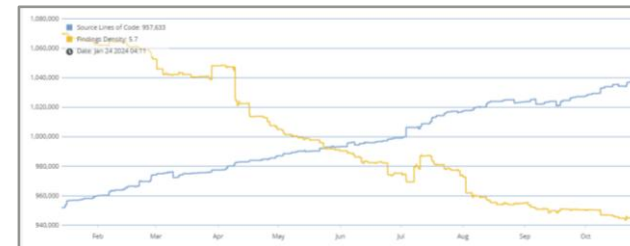
→ Abholung aller Beteiligten ist essenziell für wirksame QS



→ 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

Qualität für alle und alle für Qualität

Wie Softwarequalität zum Teamspirit wird

Mittwoch, 28. Mai, 2025

10:30 bis 12:00 Uhr

online und kostenlos

Jetzt anmelden: www.tmscl.me/qr-255-mc



Dr. Tobias Röhm



Marcel Bruckner

(Un)reif für die Cloud

Wie viel Cloud ist machbar und sinnvoll für mein System?

Mittwoch, 21. Mai 2025

10:30 bis 12:00 Uhr

online & kostenlos

Jetzt anmelden: www.tmscl.me/cloud-255-mc



Dr. Florian Deißeböck



Dr. Nils Göde

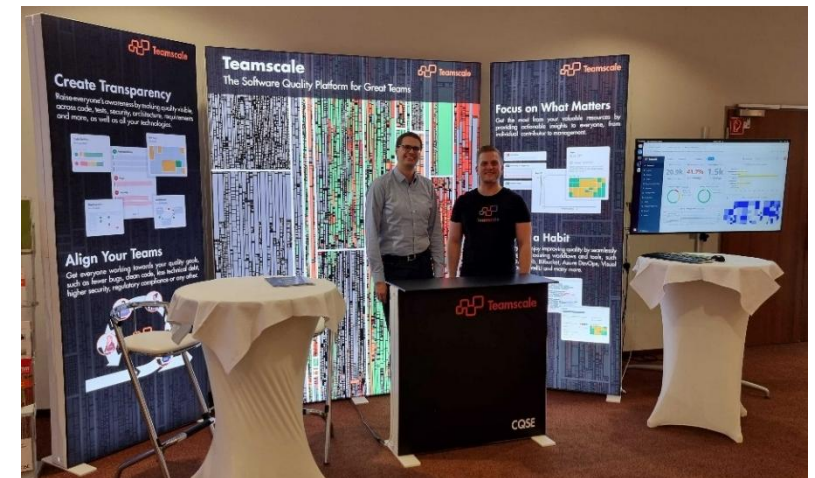
Folien & Kontakt - Ich freue mich auf Fragen & Austausch 😊



Vortragsfolien:
[tmscl.me/
qualitaetsboot_
medconf2025](https://tmscl.me/qualitaetsboot_medconf2025)



Dr. Tobias Röhm
roehm@cqse.eu
[tmscl.me/
coffee-tobias-roehm](https://tmscl.me/coffee-tobias-roehm)



CQSE-Stand