Wie gut kann man Code-Qualität wirklich messen?

Dr. Elmar Juergens

CQSE
Continuous Quality in Software Engineering

Nutzbarkeit

Effizienz

Korrektheit

Zuverlässigkeit

Funktionale Sicherheit

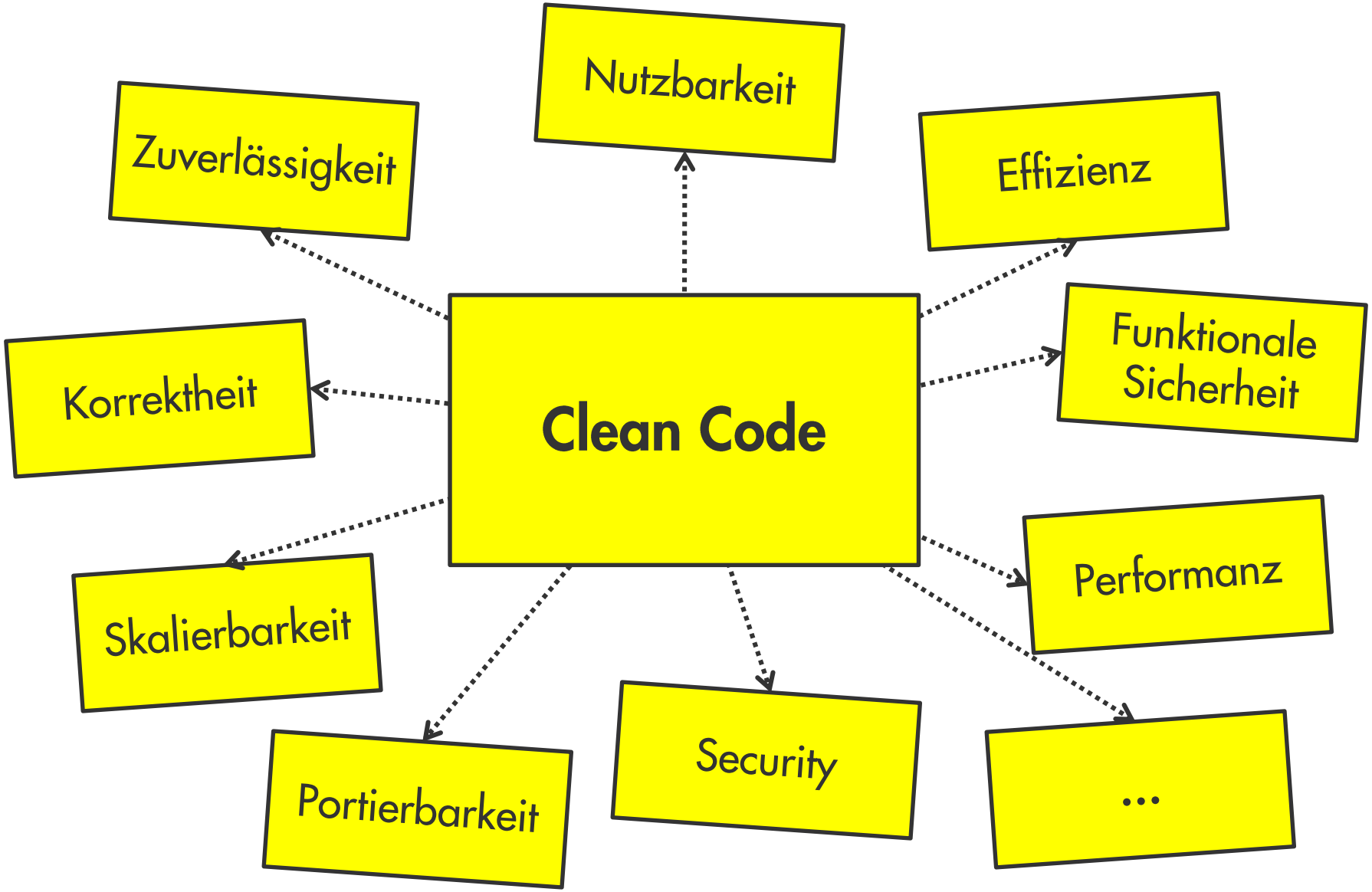
Performanz

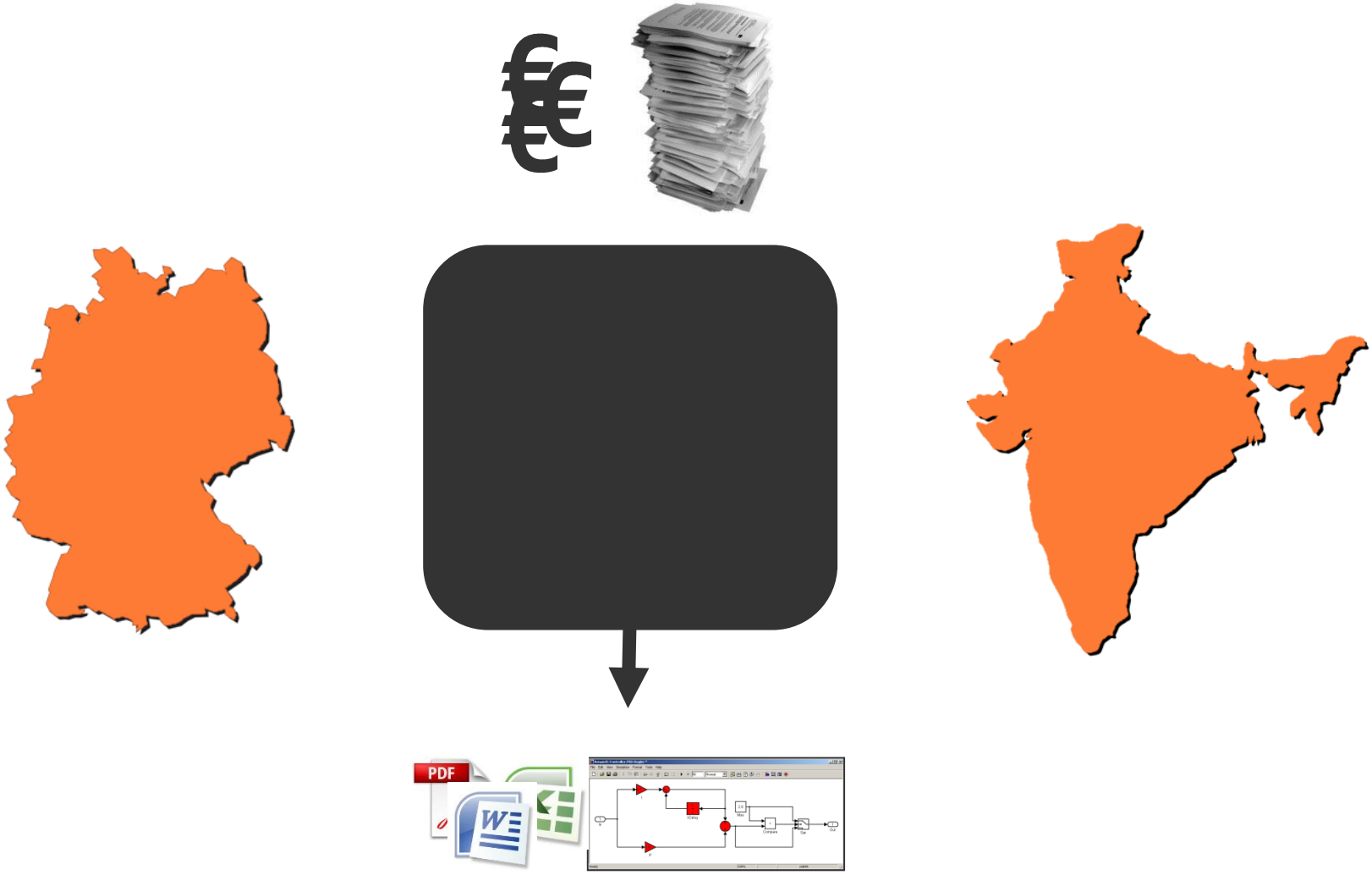
Skalierbarkeit

Security

...

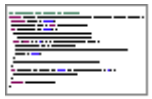
Portierbarkeit


Clean Code
Nutzbarkeit
Effizienz
Funktionale Sicherheit
Performanz
...
Security
Portierbarkeit
Skalierbarkeit
Korrektheit
Zuverlässigkeit


PDF

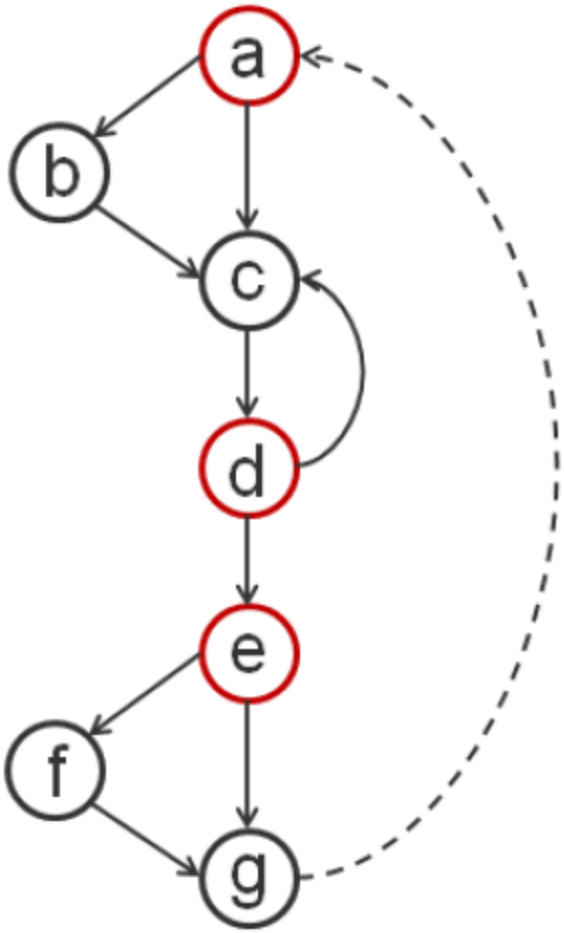
```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
   boolean found = false;
   StringBuffer res = new StringBuffer();
   if (elements != null) {
      Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
      for (int i = 0; i < elements.length; i++) {
         ModelElement el = elements[i];
         res.append(showElementLink(el)).append(HTML.LINE_BREAK);
         found = true;
      }
      Index.getInstance().resetCurrentRenderer();
   }
   if (!found && nomsg != null && nomsg.length() > 0) {
      res.append(HTML.italics(nomsg));
   }
   return res.toString();
}
```

```java
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
                FlatReferenceRenderer.getInstance());
        for (int i = 0; i < elements.length; i++) {
            ModelElement el = elements[i];
            res.append(showElementLink(el)).append(HTML.LINE_BREAK);
            found = true;
        }
        Index.getInstance().resetCurrentRenderer();
    }
    if (!found && nomsg != null && nomsg.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}
```



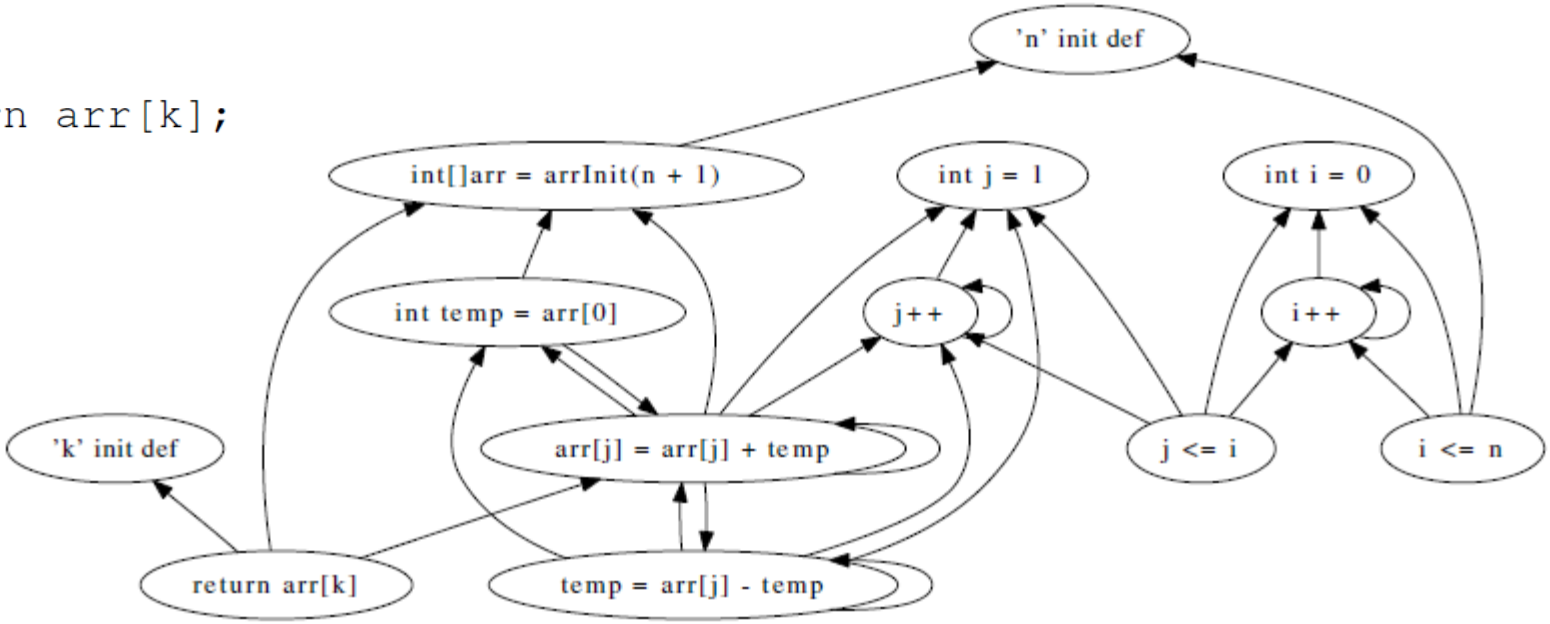
$\ll 1\%$

```
1  public void exampleCode(int a, int b, int c) {
2      if(a < 0) {
3          System.out.println("A ist negativ");
4      }
5
6      do {
7          b += 1;
8      } while(b < 0);
9
10     if(c > 0) {
11         System.out.println("C ist positiv");
12     }
13
14     int z = a * b - c;
15 }
```



Engel 2014: *Sinn und Unsinn von Software-Metriken (Seminararbeit)*

```
// Require: n >= k >= 0
int bico(int n, int k) {
  int[] arr = arrInit(n+1);
  for (int i = 0; i <= n; i++) {
    int temp = arr[0];
    for (int j = 1; j < i; j++) {
      arr[j] = arr[j] + temp;
      temp = arr[j] - temp;
    }
  }
  return arr[k];
}
```

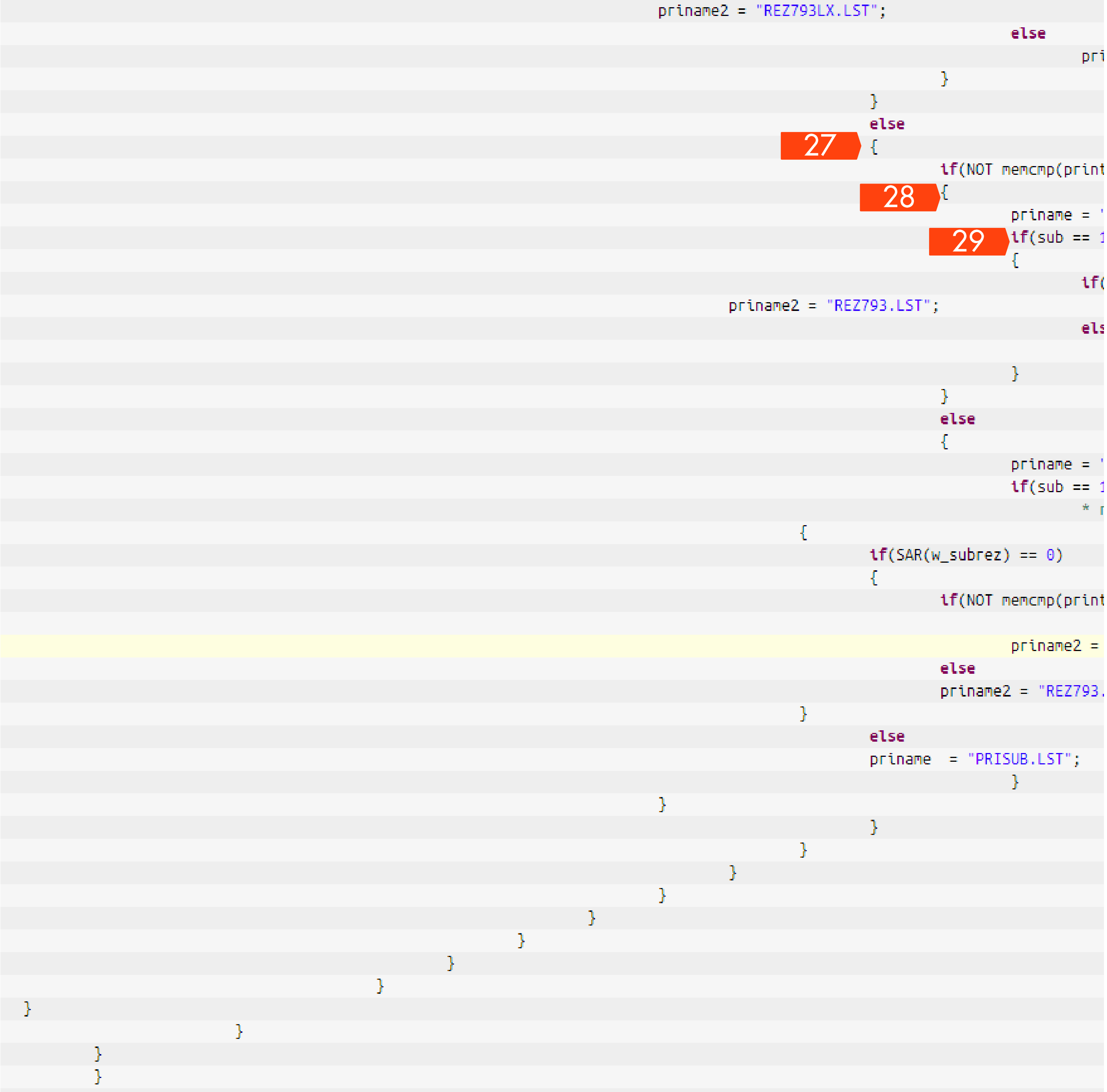


Beyer, Fararooy, ICPC 2010: *A simple and effective measure for complex low-level dependencies*


1659 LOC / 1301 SLOC

```
1418:      IF NOT lt_zeile_num IS INITIAL.
1419:    * Anzahl = 1
1420:        IF l_anz = 1.
1421:          LOOP AT lt_zeile_num INTO ls_zeile_num.
1422:            READ TABLE gt_gui_liste_angebot_0500
1423:                 INTO ls_gui_liste_angebot INDEX ls_zeile_
1424:
1425:            IF ls_gui_liste_angebot-auftyp  EQ con_angeb
1426:              OR p_vbsto EQ con_ein.
1427:    * geändertes Verschoben bis Datum in Datenbanktabell
1428:              CALL FUNCTION 'CONVERSION_EXIT_ALPHA_INPUT
1429:                EXPORTING
1430:                  input  = ls_gui_liste_angebot-vbeln
1431:                IMPORTING
1432:                  output = l_vbeln.
1433:
1434:              SELECT * FROM          /guigra INTO TABLE
1435:                     WHERE vbeln  EQ l_vbeln
1436:                       AND aktiv  EQ con_ein.
1437:
1438:              IF sy-subrc IS INITIAL.
1439:                LOOP AT lt_guigra ASSIGNING <ls_guigra>
1440:                     WHERE NOT verdat IS INITIAL.
1441:    * Es ist ein "Verschoben bis Datum" vorhanden
1442:                  EXIT.
1443:                ENDLOOP.
1444:
1445:                IF sy-subrc IS INITIAL.
1446:    * Es ist ein "Verschoben bis Datum" vorhanden
1447:                  ls_fields-tabname = '          /GUIGRA'
1448:                  ls_fields-fieldname = 'VERDAT'.
1449:                  ls_fields-value = sy-datum.
1450:
1451:                  APPEND ls_fields TO lt_fields.
1452:    * Bei Änderung Verschoben bis Datum Eingabepopup auf
1453:                  CALL FUNCTION 'POPUP_GET_VALUES_USER_C
1454:                    EXPORTING
1455:                      formname        = 'CHECK_DATE_VERD
1456:                      popup_title     = text-ver
1457:                      programname     = '/          'GUI_LISTE_ANGEBOT'
1458:                    IMPORTING
1459:                      returncode      = l_returncode
1460:                    TABLES
1461:                      fields          = lt_fields
1462:                    EXCEPTIONS
1463:                      error_in_fields = 1
1464:                      OTHERS          = 2.
1465:
```

```
1465:
1466:                IF sy-subrc IS INITIAL AND l_returncode NE con_a.
1467:                  READ TABLE lt_fields INTO ls_fields  INDEX 1.
1468:                  IF sy-subrc IS INITIAL.
1469:                    LOOP AT lt_guigra ASSIGNING <ls_guigra>
1470:                         WHERE NOT verdat IS INITIAL.
1471:                      IF ls_fields-value EQ space.
1472:                        CLEAR <ls_guigra>-verdat.
1473:                      ELSE.
1474:                        <ls_guigra>-verdat = ls_fields-value.
1475:                      ENDIF.
1476:                    ENDLOOP.
1477:                    IF sy-subrc IS INITIAL.
1478:                      UPDATE /          /guigra FROM TABLE lt_guigra.
1479:                    ENDIF.
1480:                    IF sy-subrc IS INITIAL.
1481:                      COMMIT WORK.
1482:                    ENDIF.
1483:
1484:                  ENDIF.
1485:                ELSE.
1486:    * keine Änderungen vorgenommen
1487:                  MESSAGE i255(/          40_reports)  DISPLAY LIKE 'W'.
1488:                ENDIF.
1489:              ELSE.
1490:                MESSAGE e258(          /40_reports)  DISPLAY LIKE 'E'.
1491:              ENDIF.
1492:            ENDIF.
1493:          ELSE.
1494:            MESSAGE e256(          /40_reports)  DISPLAY LIKE 'E'.
1495:          ENDIF.
1496:        ENDLOOP.
1497:      ELSE.
1498:        MESSAGE e253(          /40_reports)  DISPLAY LIKE 'E'.
1499:      ENDIF.
1500:    ELSE.
1501:      MESSAGE e028(          /40_reports)  DISPLAY LIKE 'E'.
1502:    ENDIF.
1503:
```

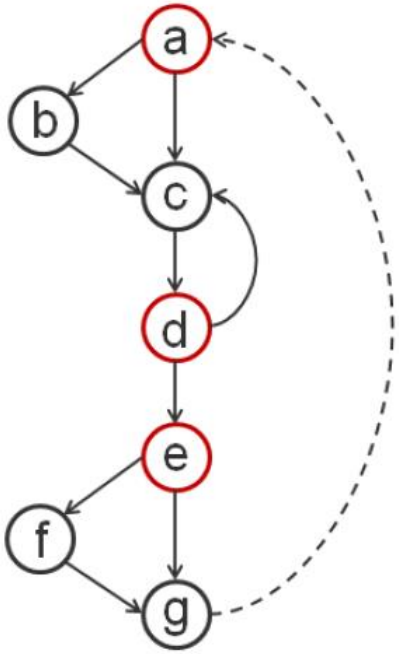


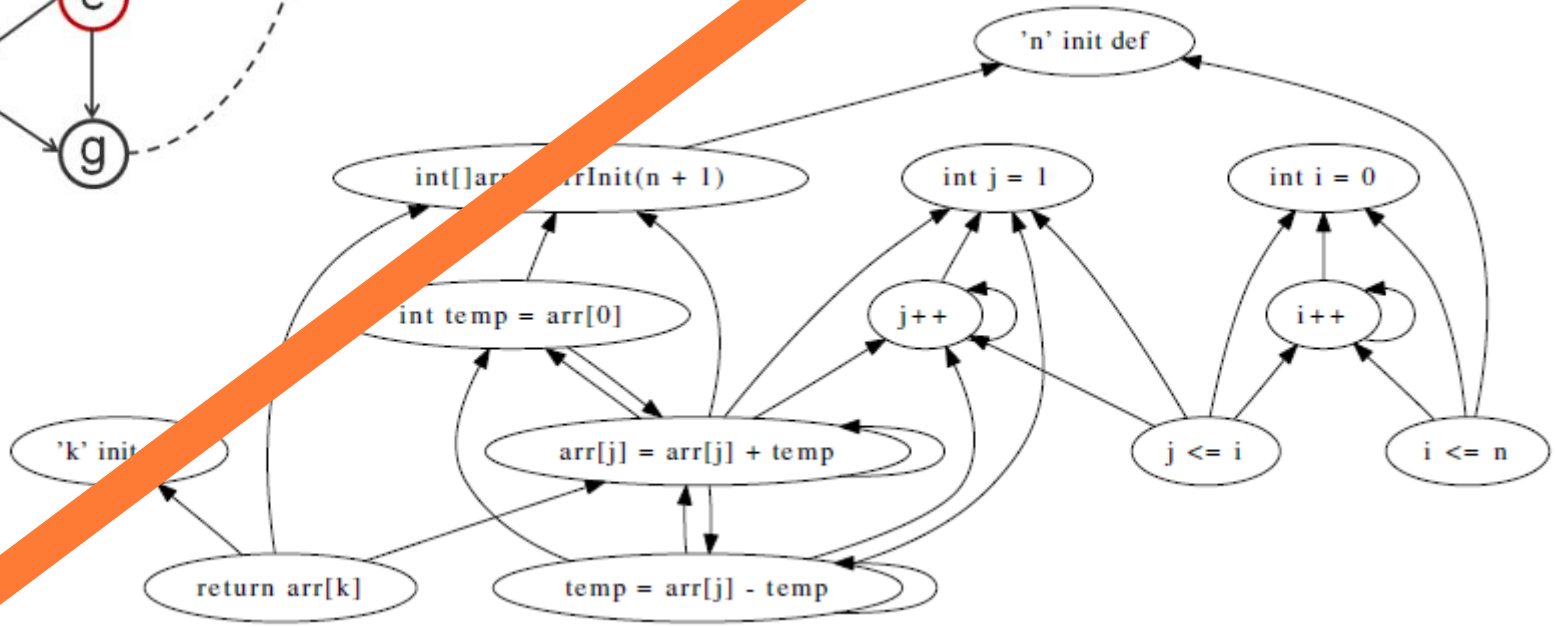
```
									priname2 = "REZ793LX.LST";
														else
															pri
													}
												}
												else
												{
													if(NOT memcmp(print
													{
														priname = '
														if(sub == 1
														{
															if(
										priname2 = "REZ793.LST";
															els

														}
													}
													else
													{
														priname = '
														if(sub == 1
															* r
											{
												if(SAR(w_subrez) == 0)
												{
													if(NOT memcmp(print

														priname2 =
													else
													priname2 = "REZ793.
											}
												else
												priname  = "PRISUB.LST";
														}
									}
												}
											}
										}
									}
								}
							}
						}
					}
}
			}
	}
	}
```

```java
String getMonthName (int month) {
        switch (month) {
                case 0: return "January";
                case 1: return "February";
                case 2: return "March";
                case 3: return "April";
                case 4: return "May";
                case 5: return "June";
                case 6: return "July";
                case 7: return "August";
                case 8: return "September";
                case 9: return "October";
                case 10: return "November";
                case 11: return "December";
                default: throw new IllegalArgumentException();
        }
}
```

```java
int sumOfNonPrimes(int limit) {
        int sum = 0;
        OUTER: for (int i = 0; i < limit; ++i) {
                if (i <= 2) {
                        continue;
                }
                for (int j = 2; j < i; ++j) {
                        if (i % j == 0) {
                                continue OUTER;
                        }
                }
                sum += i;
        }
        return sum;
}
```

Katzmarski, Koschke, ICPC 2012: *Program Complexity Metrics and Programmer Opinions*


a
b
c
d
e
f
g


'n' init def
int[]arr = arrInit(n + 1)
int j = 1
int i = 0
int temp = arr[0]
j++
i++
'k' init
arr[j] = arr[j] + temp
j <= i
i <= n
return arr[k]
temp = arr[j] - temp

$$171 - 5.2 \cdot \ln(avgHV) - 0.23 \cdot avgCC(g') - 16.2 \cdot \ln(avgLOC) + 50 \cdot \sin(\mathrm{sqrt}(2.4 \cdot perCM))$$

HV:	Halstead Volume	CC:	Cyclomatic Complexity
LOC:	lines of code	perCM:	% Comment Lines

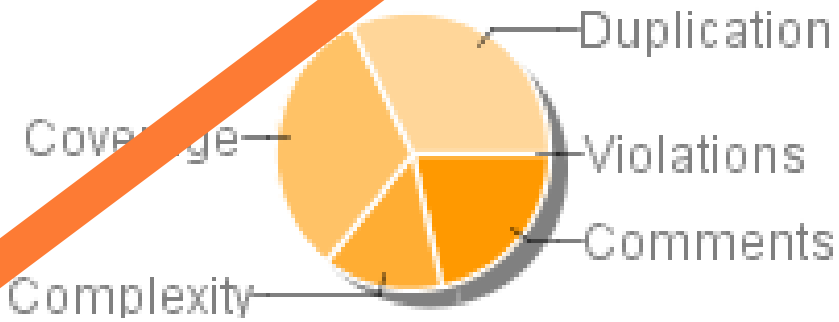
?

?

Technical Debt

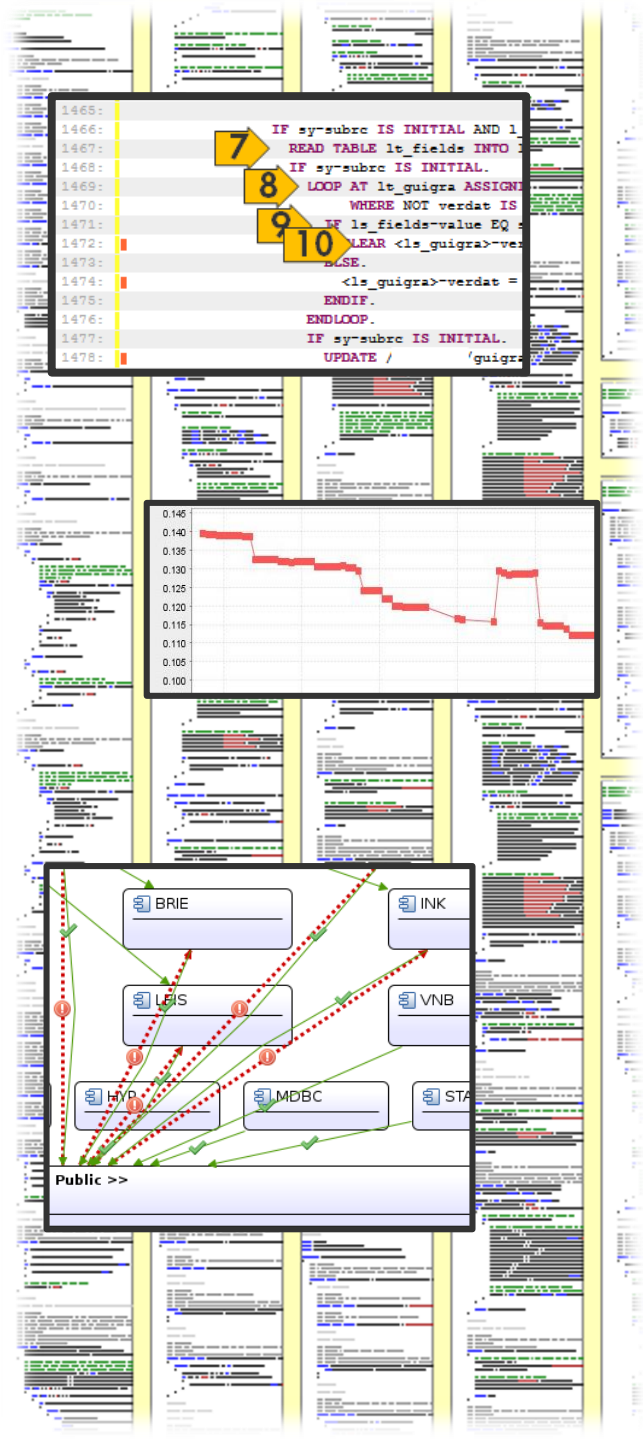
$ 14'605

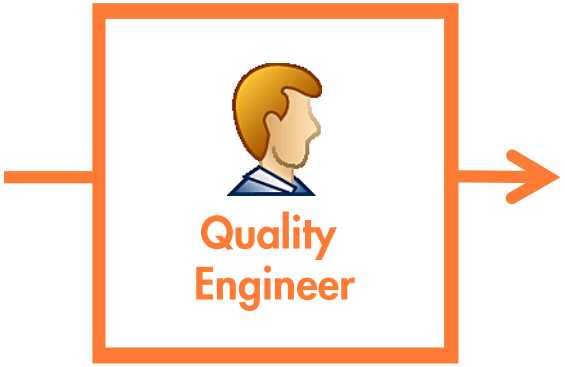
29 man days

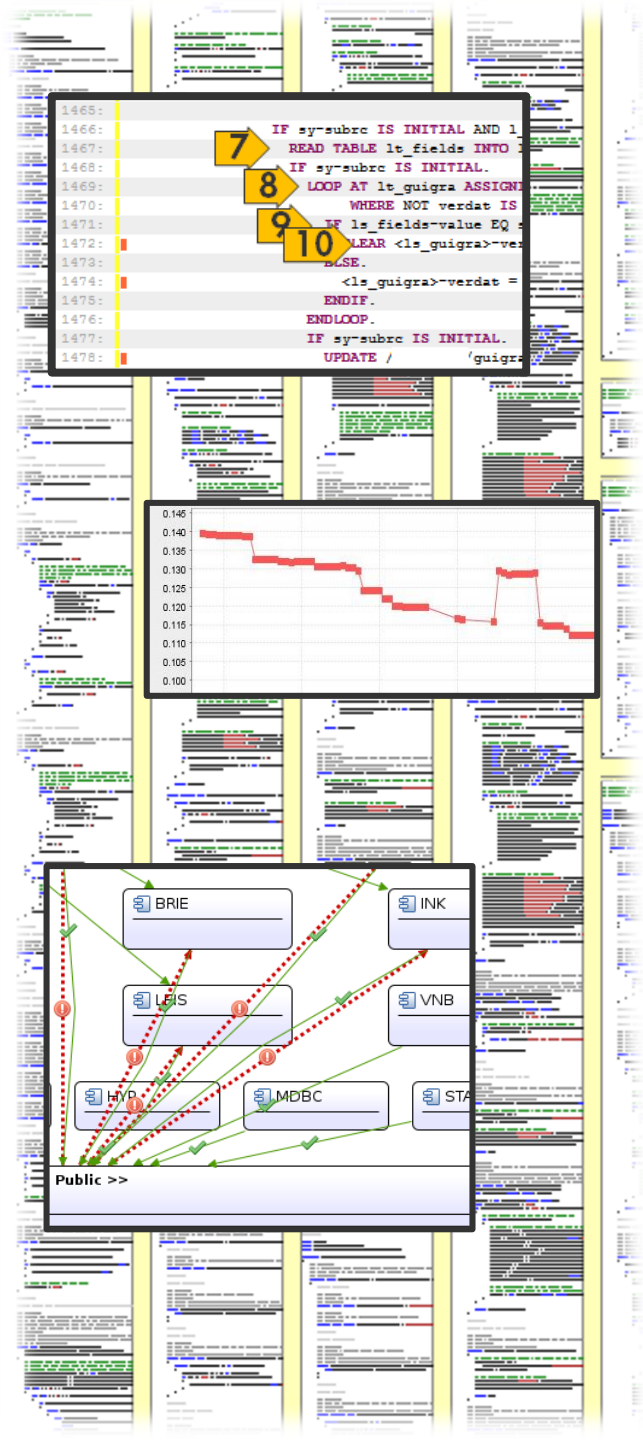


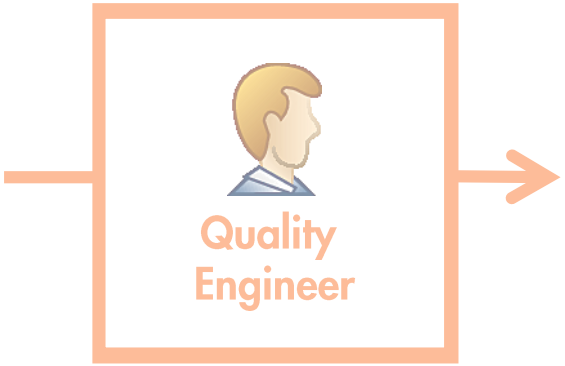
http://www.sonarqube.org/


5
6
7
8
9
4
10
2
11
1
12
14
13


7
8
9
10
0.145
0.140
0.135
0.130
0.125
0.120
0.115
0.110
0.105
0.100
BRIE
INK
LEIS
VNB
MDBC
Public >>

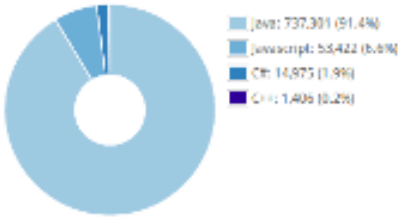

Quality
Engineer


7
8
9
10
0.145
0.140
0.135
0.130
0.125
0.120
0.115
0.110
0.105
0.100
BRIE
INK
LEIS
VNB
MDBC
Public >>


Quality
Engineer

Teamscale Dashboard Activity Findings Tasks Code Architecture Delta Projects System Admin

LOC Distribution for cqse-all



Clone Coverage for cqse-all

4.7%

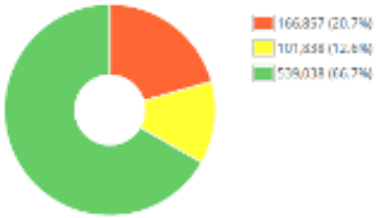
-0.05%

Findings Density for cqse-all

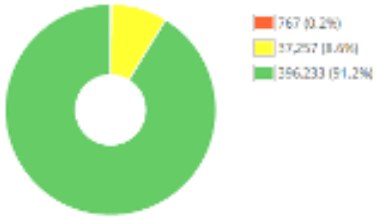
15

+1.1

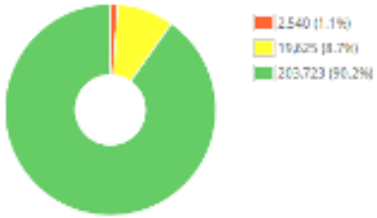
Review Rating for cqse-all



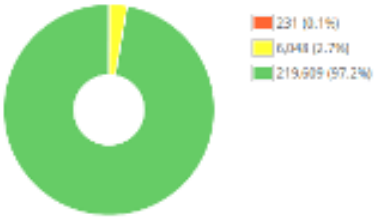
File Length for cqse-all



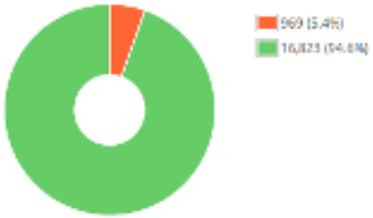
Method Length for cqse-all



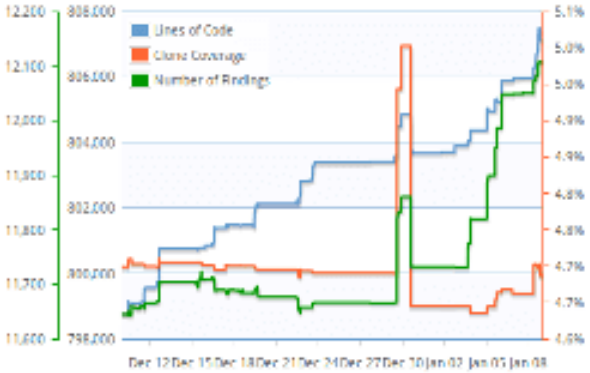
Nesting Depth for cqse-all



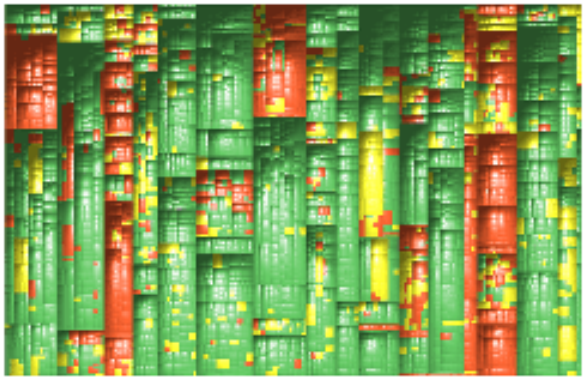
Comment Completeness for cqse-



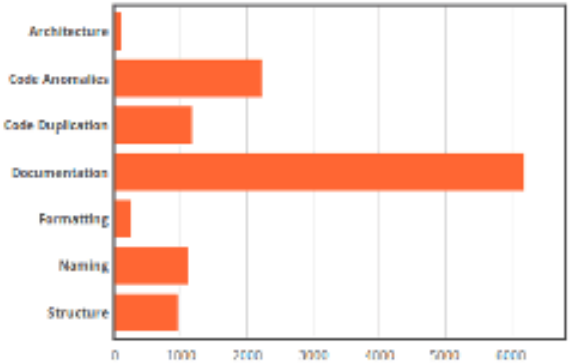
30 Day Trend for cqse-all

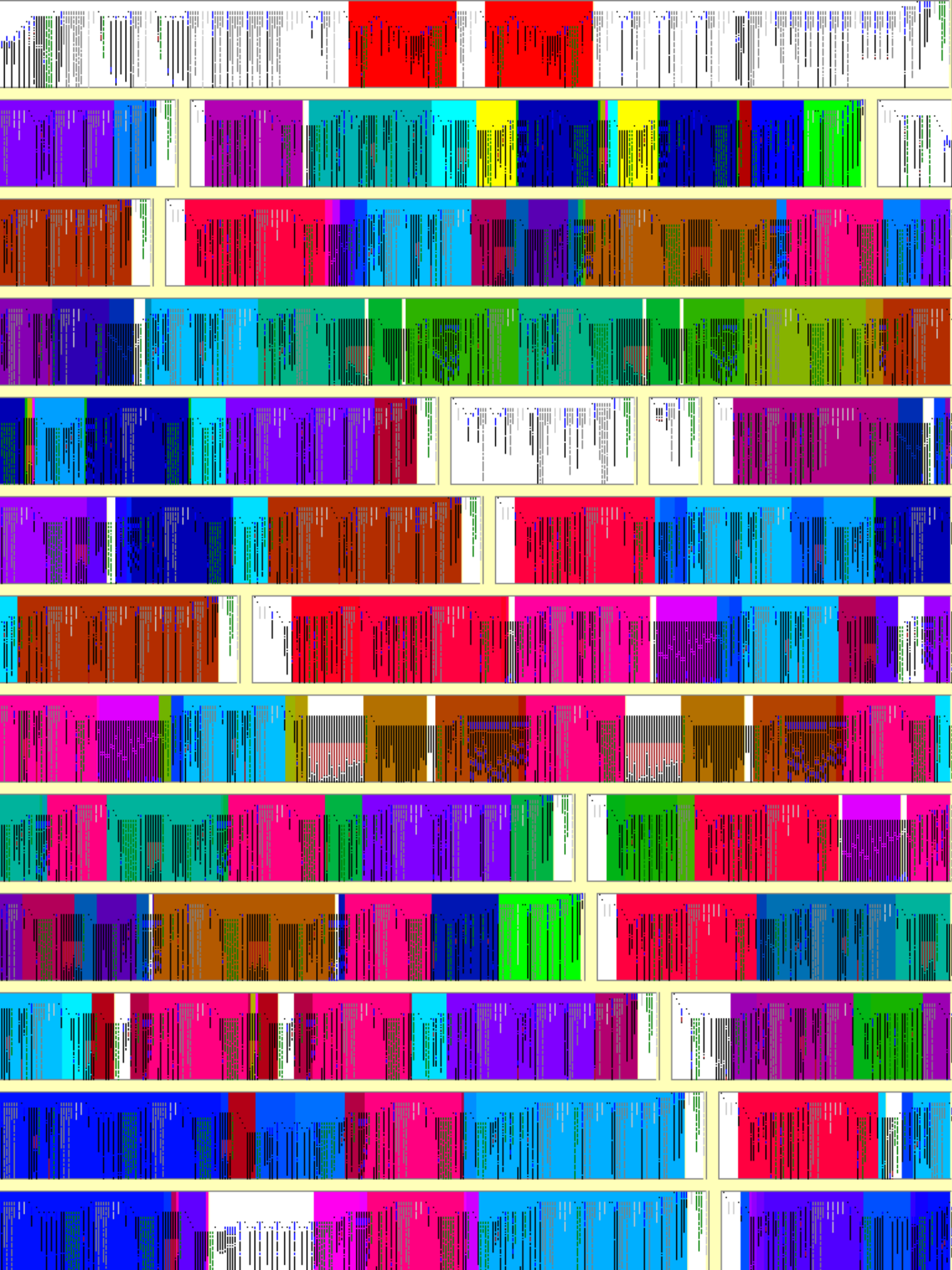


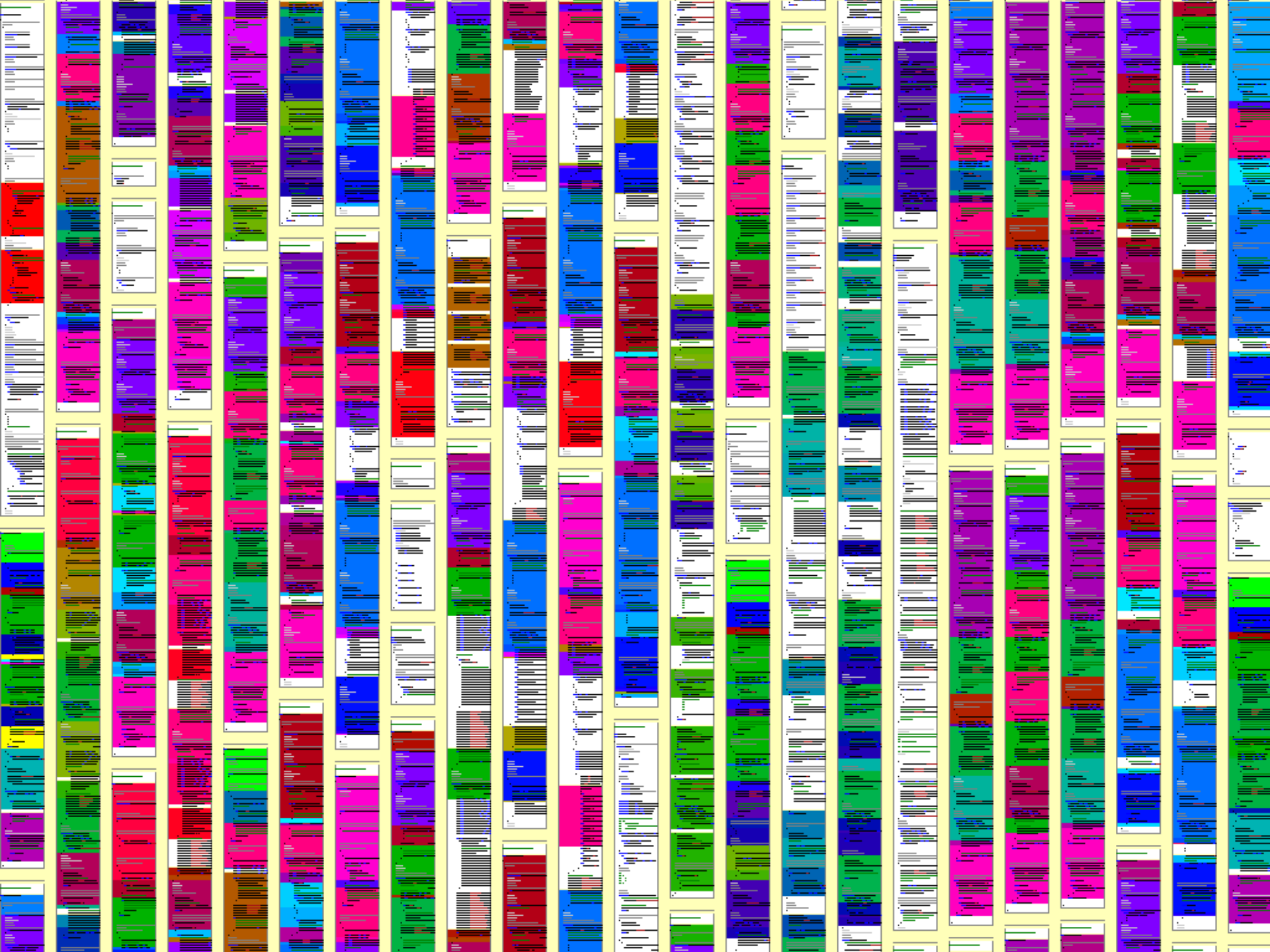
Rating Treemap for cqse-all

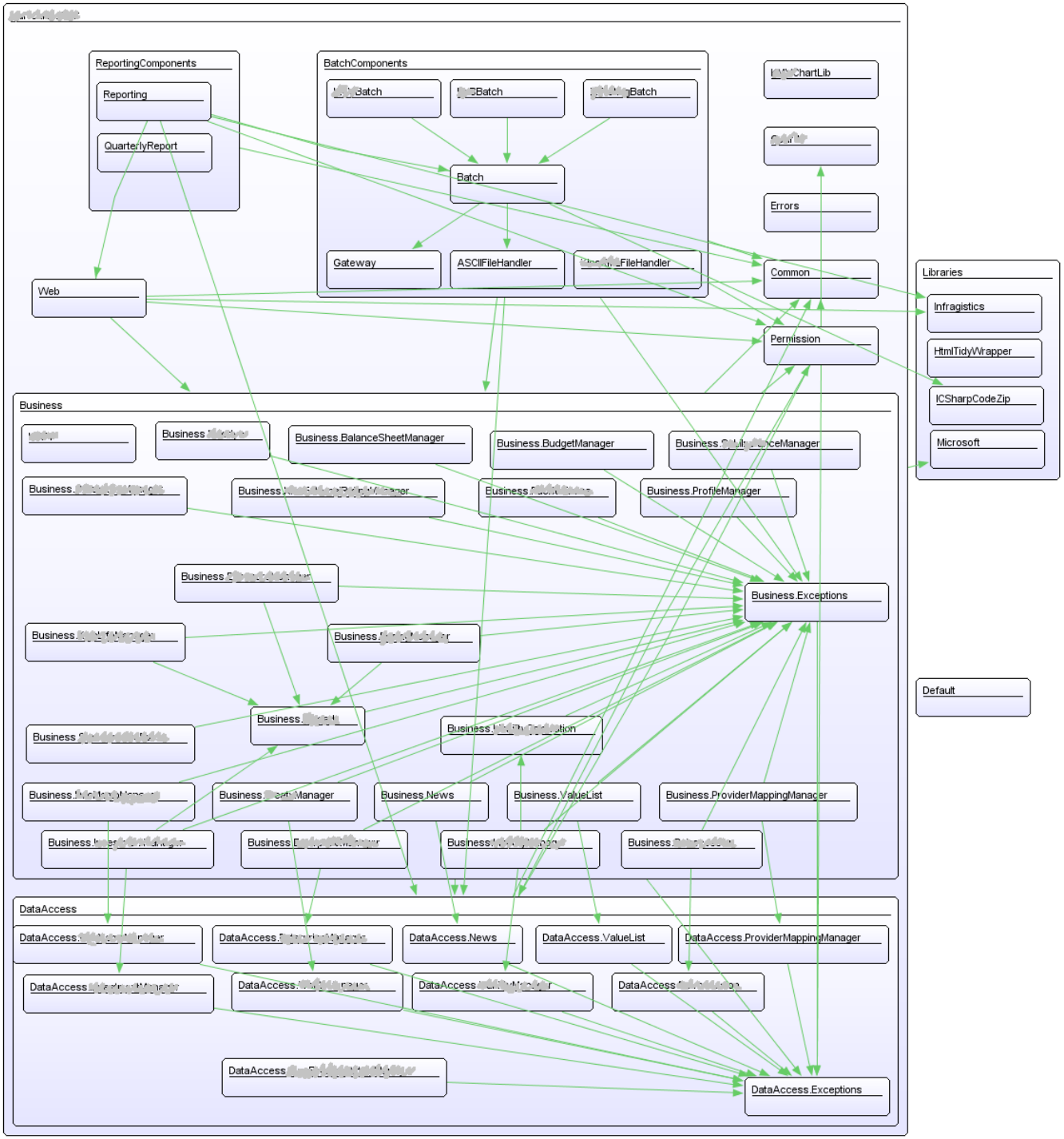


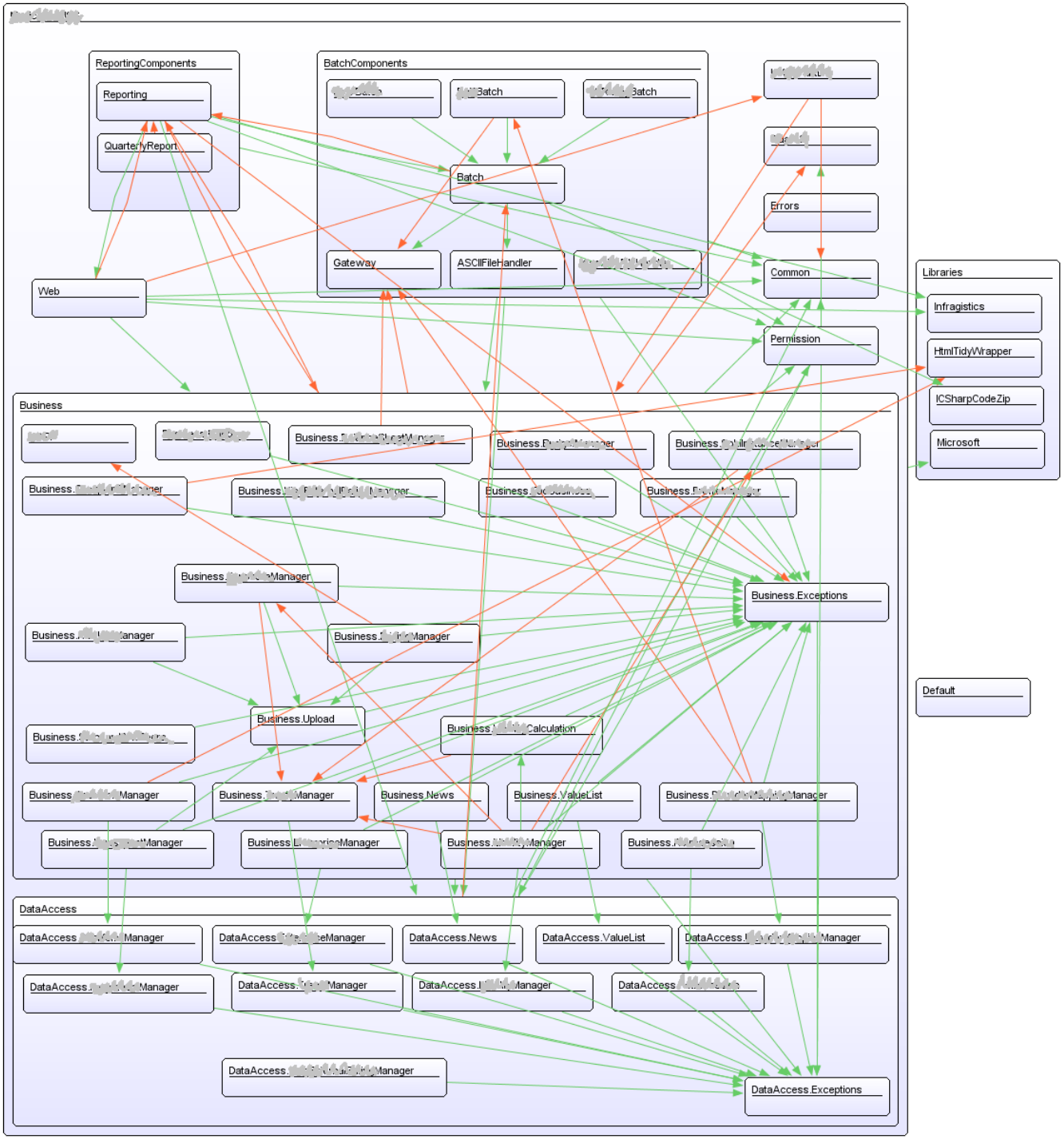
Findings Summary Bar Chart for cqse-all

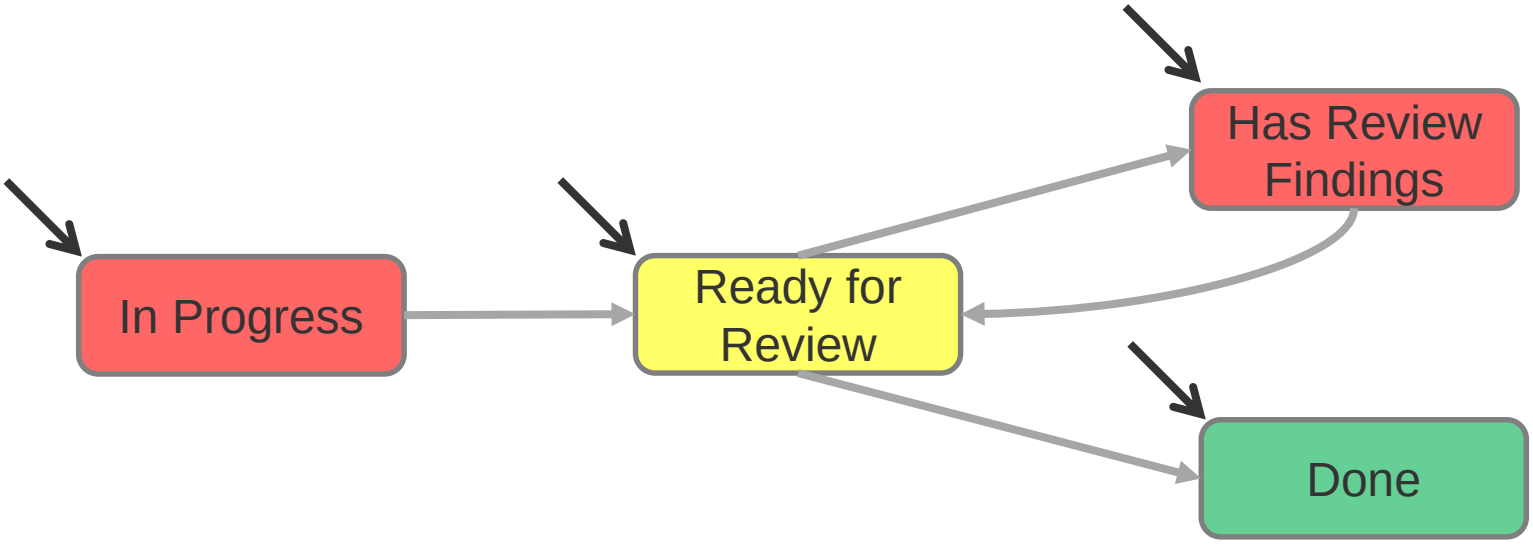


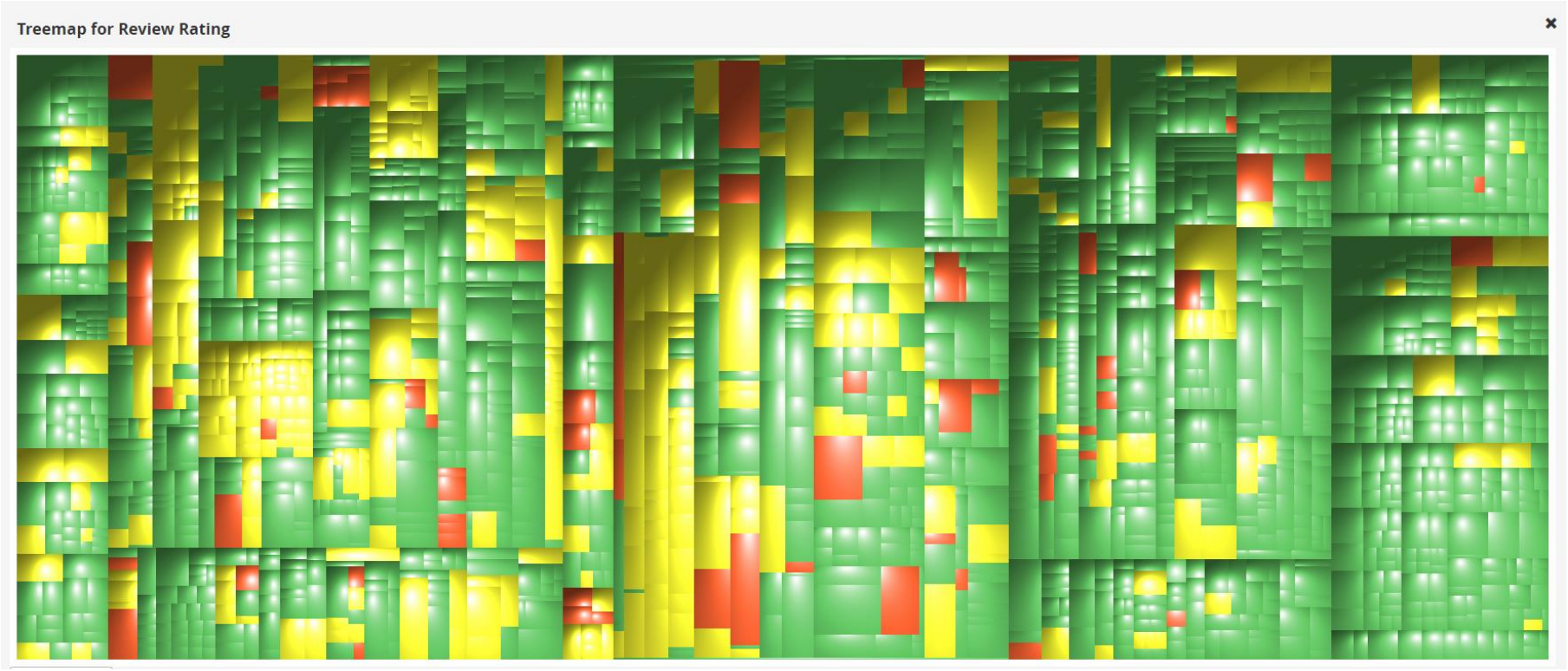


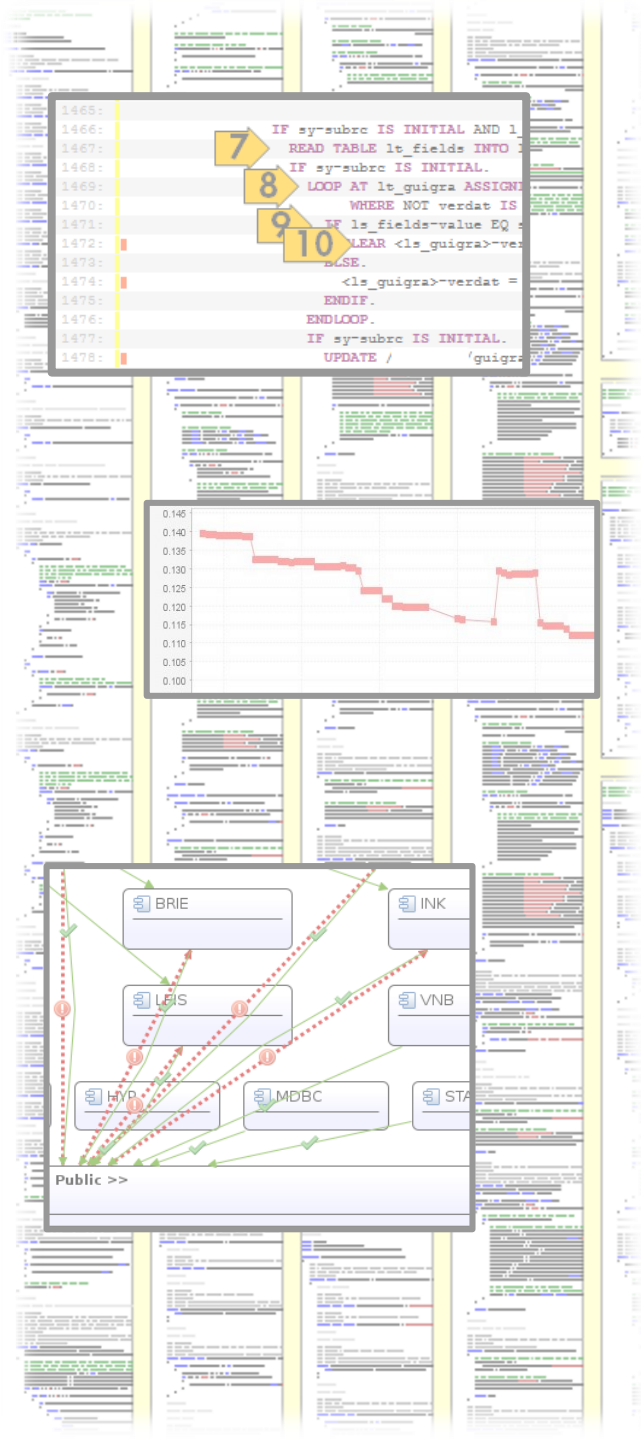


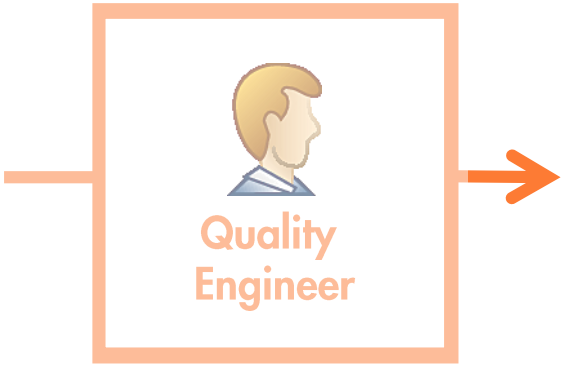

ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Gateway
ASCIIFileHandler
Web
Errors
Common
Permission
Libraries
Infragistics
HtmlTidyWrapper
ICSharpCodeZip
Microsoft
Business
Business.BalanceSheetManager
Business.BudgetManager
Business.ProfileManager
Business.Exceptions
Business.News
Business.ValueList
Business.ProviderMappingManager
Default
DataAccess
DataAccess.News
DataAccess.ValueList
DataAccess.ProviderMappingManager
DataAccess.Exceptions


ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Gateway
ASCIIFileHandler
Errors
Common
Permission
Web
Libraries
Infragistics
HtmlTidyWrapper
ICSharpCodeZip
Microsoft
Business
Business.Exceptions
Business.Upload
Business.News
Business.ValueList
Default
DataAccess
DataAccess.News
DataAccess.ValueList
DataAccess.Exceptions

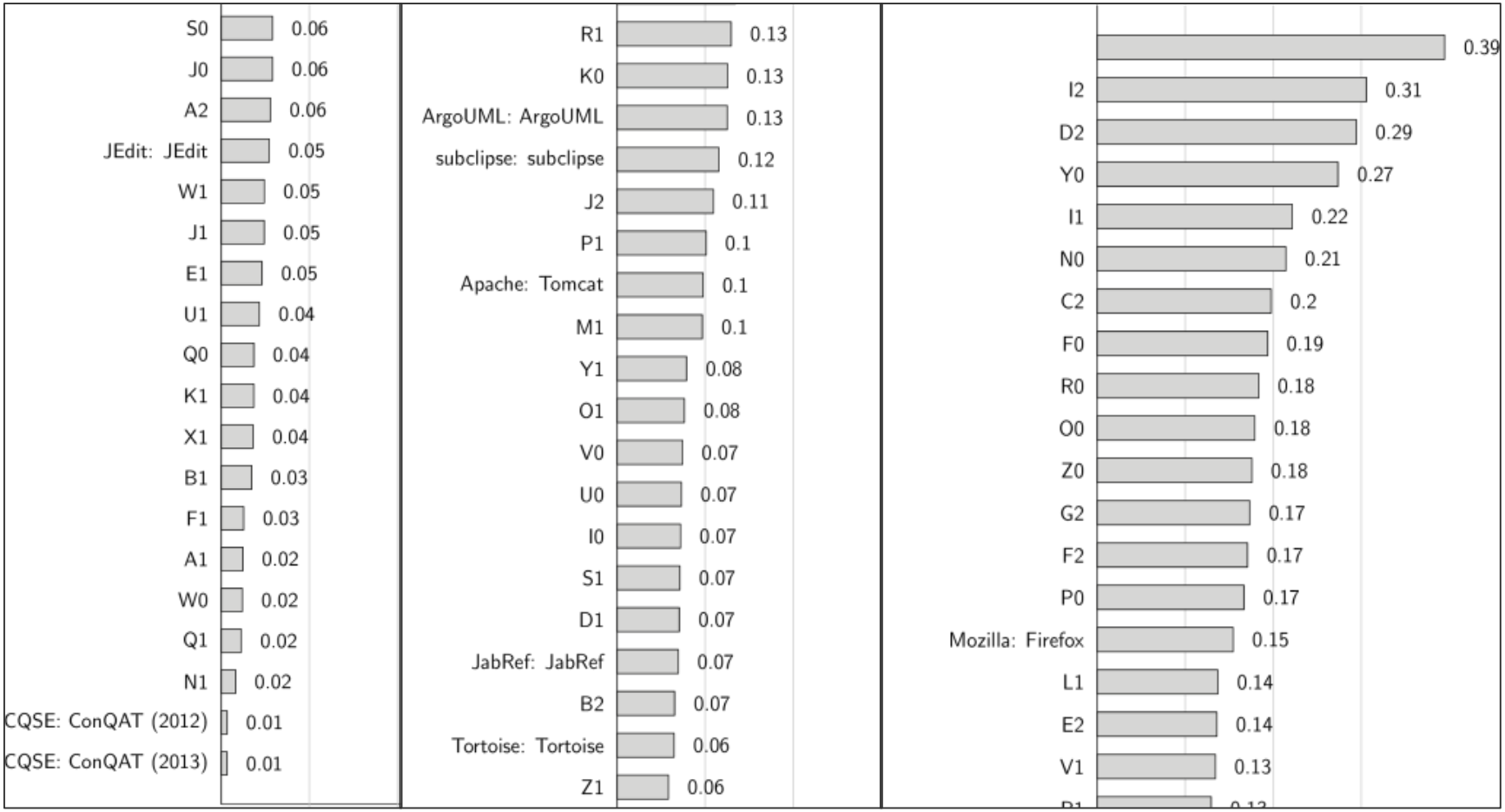

In Progress
Ready for Review
Has Review Findings
Done

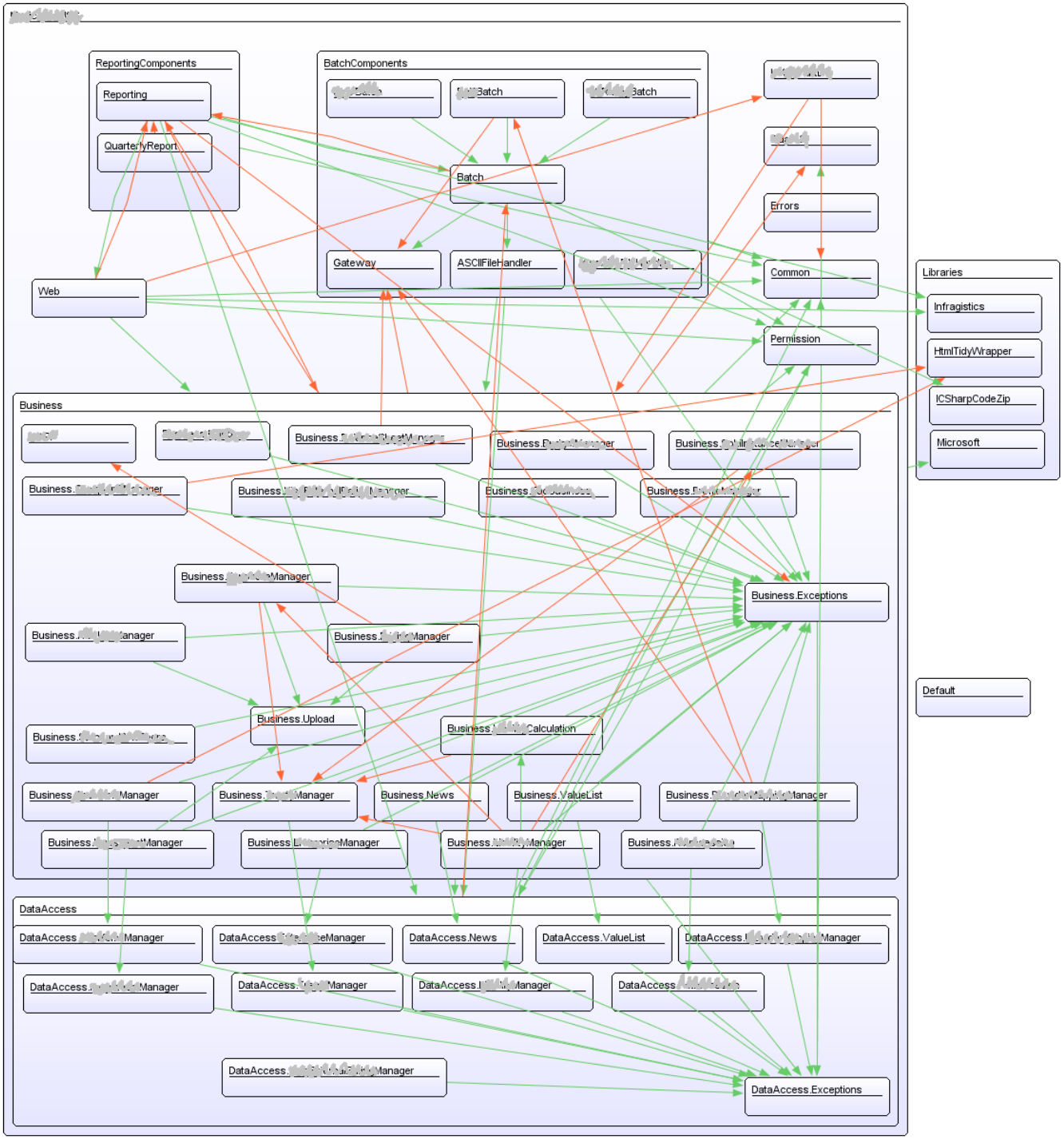

Treemap for Review Rating

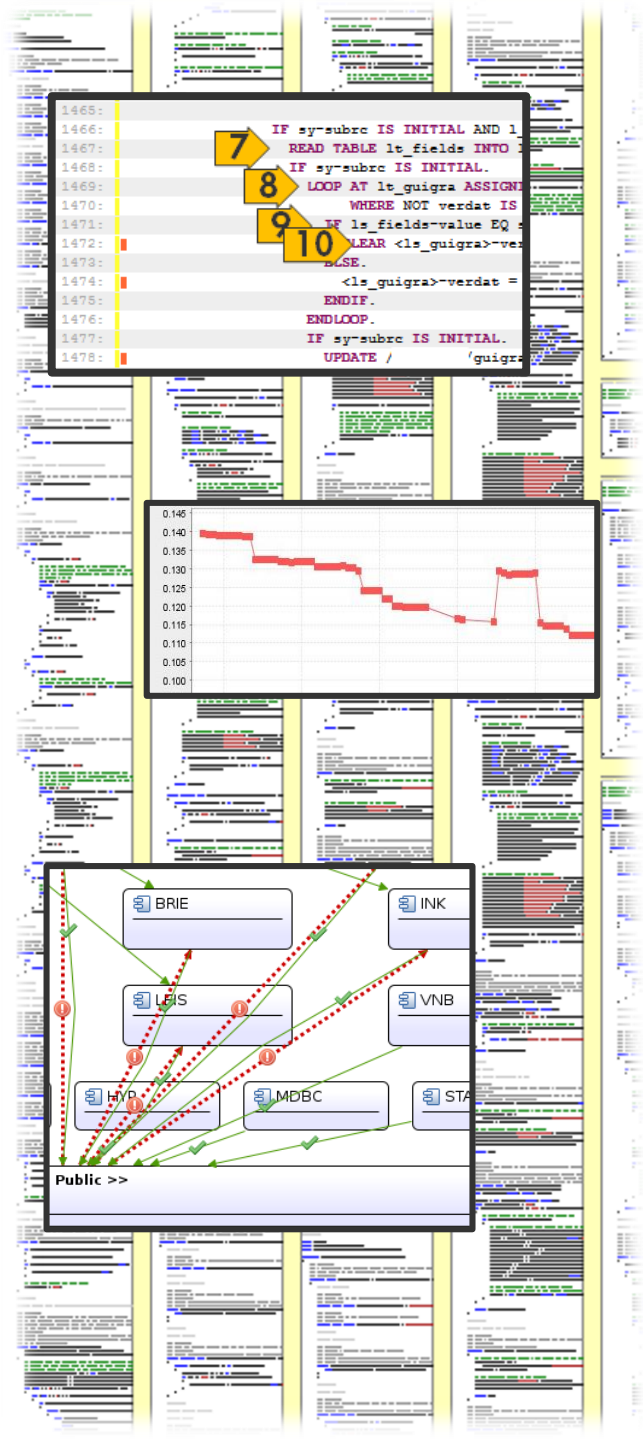

7
8
9
10
0.145
0.140
0.135
0.130
0.125
0.120
0.115
0.110
0.105
0.100
BRIE
INK
LEIS
VNB
HYP
MDBC
Public >>

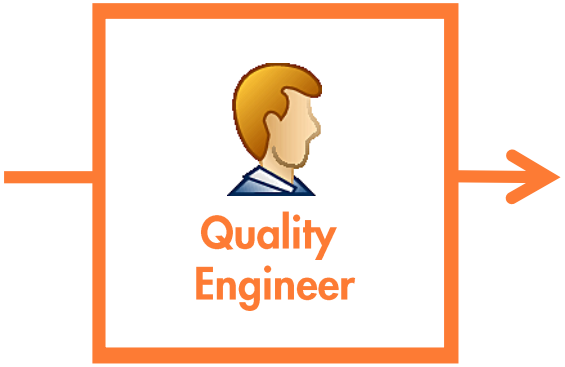

Quality
Engineer

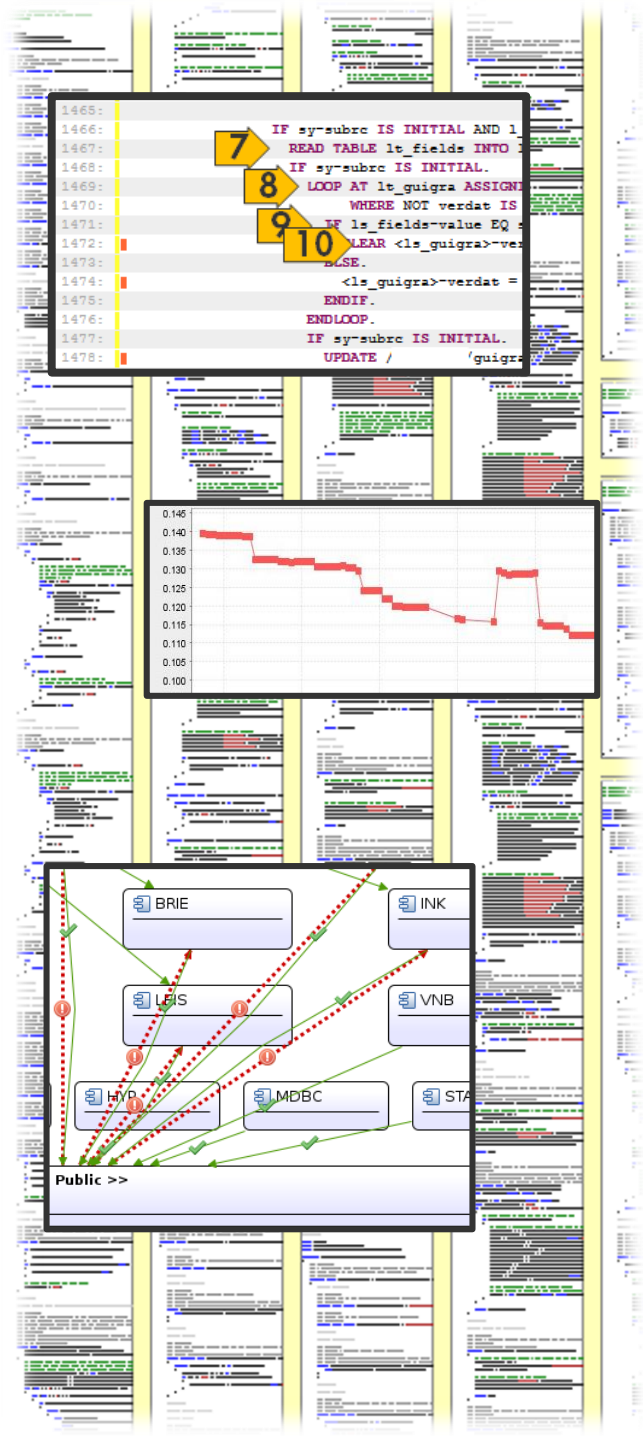
Benchmark (C#/Java/C++)

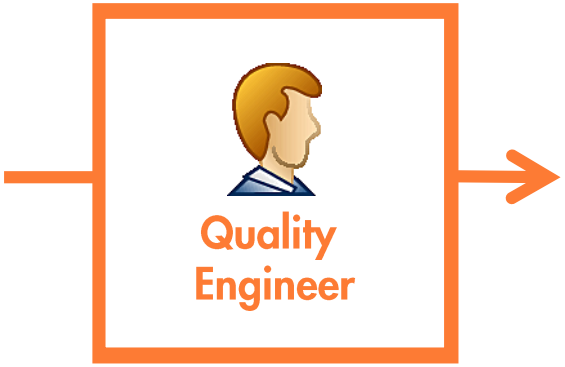



ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Gateway
ASCIIFileHandler
Errors
Common
Permission
Web
Libraries
Infragistics
HtmlTidyWrapper
ICSharpCodeZip
Microsoft
Business
Business.Exceptions
Business.Upload
Business.News
Business.ValueList
Default
DataAccess
DataAccess.News
DataAccess.ValueList
DataAccess.Exceptions


7
8
9
10
0.145
0.140
0.135
0.130
0.125
0.120
0.115
0.110
0.105
0.100
BRIE
INK
LEIS
VNB
MDBC
Public >>


Quality
Engineer

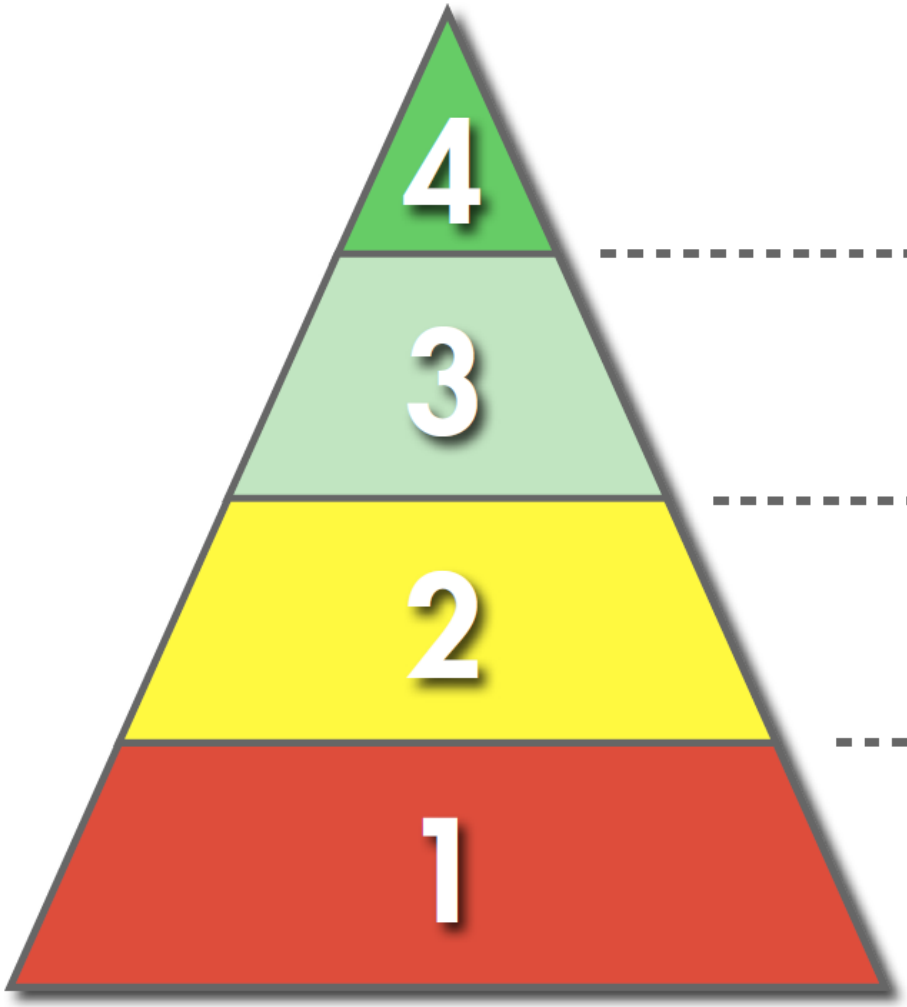

7
8
9
10
IF sy-subrc IS INITIAL AND l
READ TABLE lt_fields INTO
IF sy-subrc IS INITIAL.
LOOP AT lt_guigra ASSIGN
WHERE NOT verdat IS
IF ls_fields-value EQ
CLEAR <ls_guigra>-ver
ELSE.
<ls_guigra>-verdat =
ENDIF.
ENDLOOP.
IF sy-subrc IS INITIAL.
UPDATE /
'guigra
0.145
0.140
0.135
0.130
0.125
0.120
0.115
0.110
0.105
0.100
BRIE
INK
LEIS
VNB
HYP
MDBC
STA
Public >>


Quality
Engineer

Task List

```java
/** Pace statistics. */
public static Result pace(int year) throws SQLException,
        PermissionException {
    requirePermission(currentUser().isMa());

    PairList<Date, Double> pace = new PairList<>();

    AggregatedSalaryData aggregatedSalaryData = new AggregatedSalaryData(
            SalaryData.getSalaryDataForEmployees(year), year);

    List<Costs> costs = Costs.getCostsDataForYear(year,
            aggregatedSalaryData.employeeHeadCount,
            aggregatedSalaryData.studentsHeadCount);
    CostDataForYear aggregatedCostData = new CostDataForYear(costs);

    RevenueData revenueData = new RevenueData(year, aggregatedSalaryData,
            aggregatedCostData);

    for (int month = 0; month < 12; ++month) {
        Date start = DateUtils.getMonthStart(year, month);
        Date end = DateUtils.getMonthEnd(year, month);

        PaceInfo paceInfo = new PaceInfo(start, end);
        int available = paceInfo.getTotalAvailableAll(false);
        if (available == 0) {
            pace.add(start, 0.);
        } else {
            pace.add(start, paceInfo.getTotalBilledAll(false)
                    / (double) available);
        }
    }

    Date start = DateUtils.getStartOfYear(year).getTime();
    Date end = DateUtils.getEndOfYear(year).getTime();

    PaceInfo paceInfo = new PaceInfo(start, end);
    double dailyRate = paceInfo.getAverageRate();
    if (dailyRate == 0) {
        dailyRate = 1000;
    }

    List<LabeledValue> infos = new ArrayList<>();

    VacationPredictor vacationPredictor = new VacationPredictor(year);
    int available = paceInfo.getTotalAvailableAll(false)
            - (int) (vacationPredictor.getOverallExpectedVacation() * 60 * 8);

    LicenseDevChartInfo licenseInfo = Licenses
            .obtainLicenseDevChartInfo(year);

    PairList<Date, Double> minPace = new PairList<>();
    PairList<Date, Double> bonusPace = new PairList<>();
    PairList<Date, Double> targetPace = new PairList<>();

    Date plotEnd = DateUtils.getMonthStart(year, 11);
    if (available > 0) {
        infos.add(new LabeledValue("Current pace: ", paceInfo
                .getTotalBilledAll(false) / (double) available));

        int partialAvailable = paceInfo.getTotalAvailableAll(true);
        if (partialAvailable > 0) {
            infos.add(new LabeledValue("Current partial pace: ", paceInfo
                    .getTotalBilledAll(true) / (double) partialAvailable));
        }

        double cost = aggregatedSalaryData.sumTotalCosts
                + aggregatedCostData.sumExpenses;
        double survivalPace = ((cost - licenseInfo.getLicenseGain()) / dailyRate)
                * 8 * 60 / available;
        infos.add(new LabeledValue("Survival pace: ", survivalPace));
        minPace.add(start, survivalPace);
        minPace.add(plotEnd, survivalPace);

        double bonusPaceValue = ((revenueData.targetRevenueBoni - licenseInfo
                .getLicenseGain()) / dailyRate) * 8 * 60 / available;
        infos.add(new LabeledValue("Bonus pace: ", bonusPaceValue));
        bonusPace.add(start, bonusPaceValue);
        bonusPace.add(plotEnd, bonusPaceValue);

        double targetPaceValue = ((revenueData.targetRevenueEbit - licenseInfo
                .getLicenseGain()) / dailyRate) * 8 * 60 / available;
        infos.add(new LabeledValue("Target pace: ", targetPaceValue));
        targetPace.add(start, targetPaceValue);
        targetPace.add(plotEnd, targetPaceValue);
    }

    return Json(new PaceStatistics(infos, pace, minPace, bonusPace,
            targetPace));
}
```


```java
/** Pace statistics. */
public static Result pace(int year) throws SQLException,
        PermissionException {
    requirePermission(currentUser().isMa());

    PairList<Date, Double> pace = new PairList<>();

    AggregatedSalaryData aggregatedSalaryData = new AggregatedSalaryData(
            SalaryData.getSalaryDataForEmployees(year), year);

    List<Costs> costs = Costs.getCostsDataForYear(year,
            aggregatedSalaryData.employeeHeadCount,
            aggregatedSalaryData.studentsHeadCount);
    CostDataForYear aggregatedCostData = new CostDataForYear(costs);

    RevenueData revenueData = new RevenueData(year, aggregatedSalaryData,
            aggregatedCostData);

    for (int month = 0; month < 12; ++month) {
        Date start = DateUtils.getMonthStart(year, month);
        Date end = DateUtils.getMonthEnd(year, month);

        PaceInfo paceInfo = new PaceInfo(start, end);
        int available = paceInfo.getTotalAvailableAll(false);
        if (available == 0) {
            pace.add(start, 0.);
        } else {
            pace.add(start, paceInfo.getTotalBilledAll(false)
                    / (double) available);
        }
    }
```

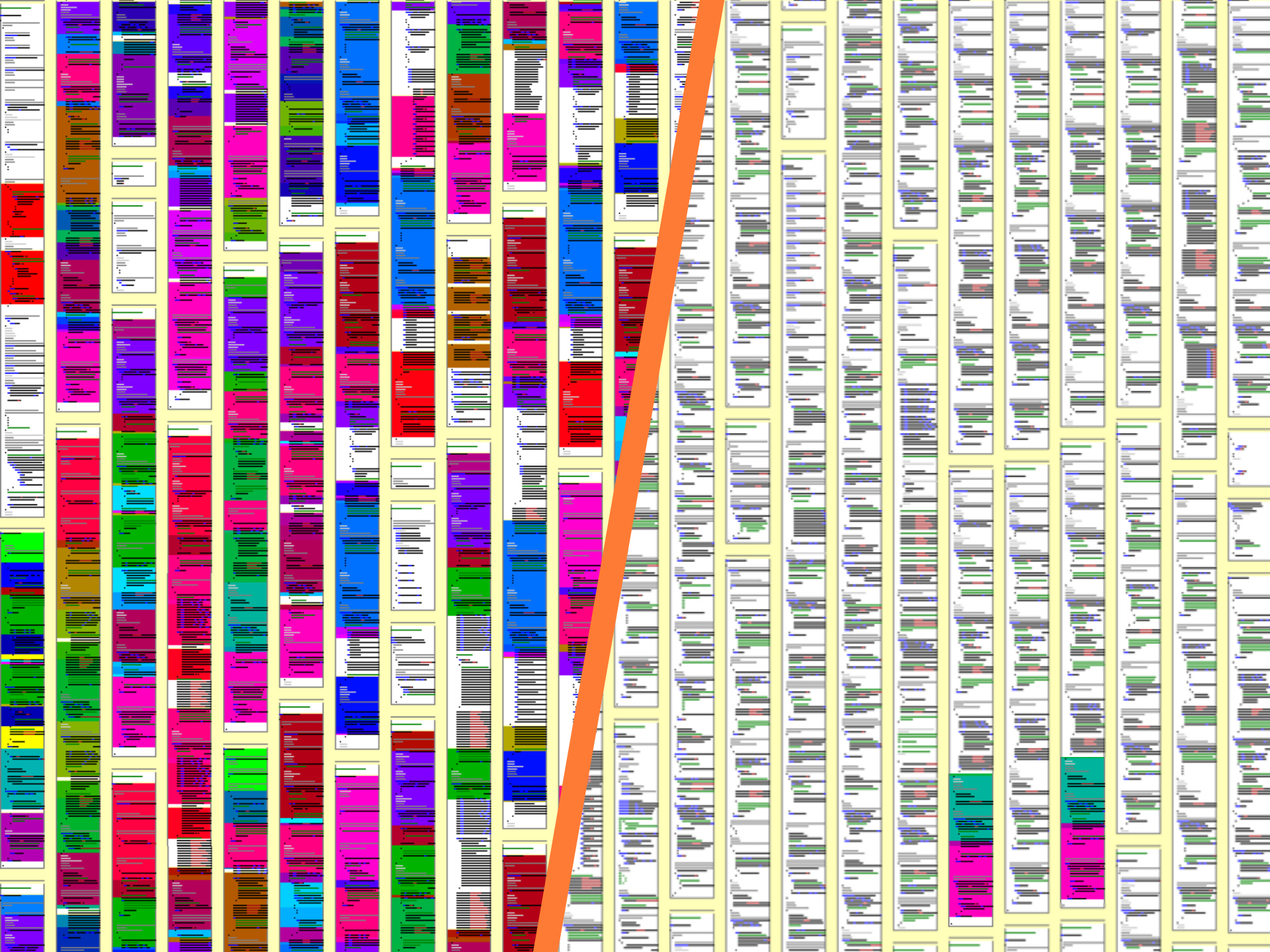


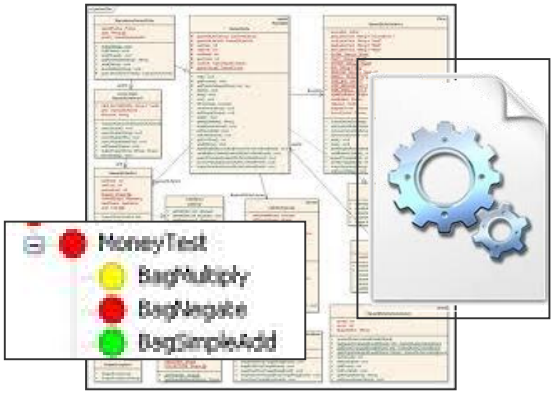
Keine Defizite

Keine Defizite in geändertem Code

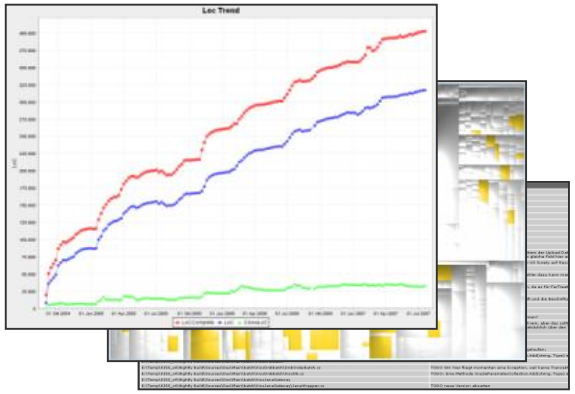
Keine neuen Defizite

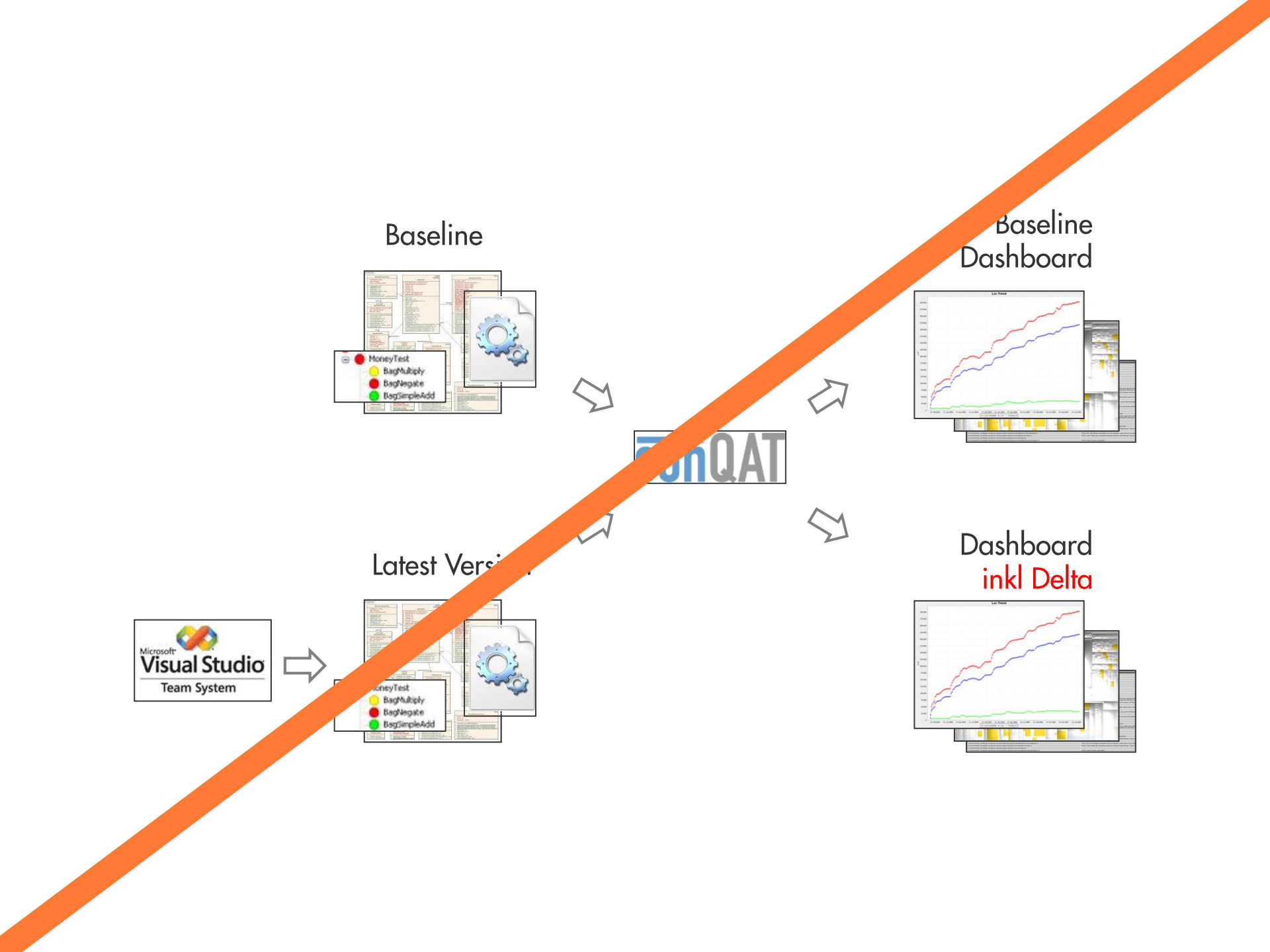
Egal

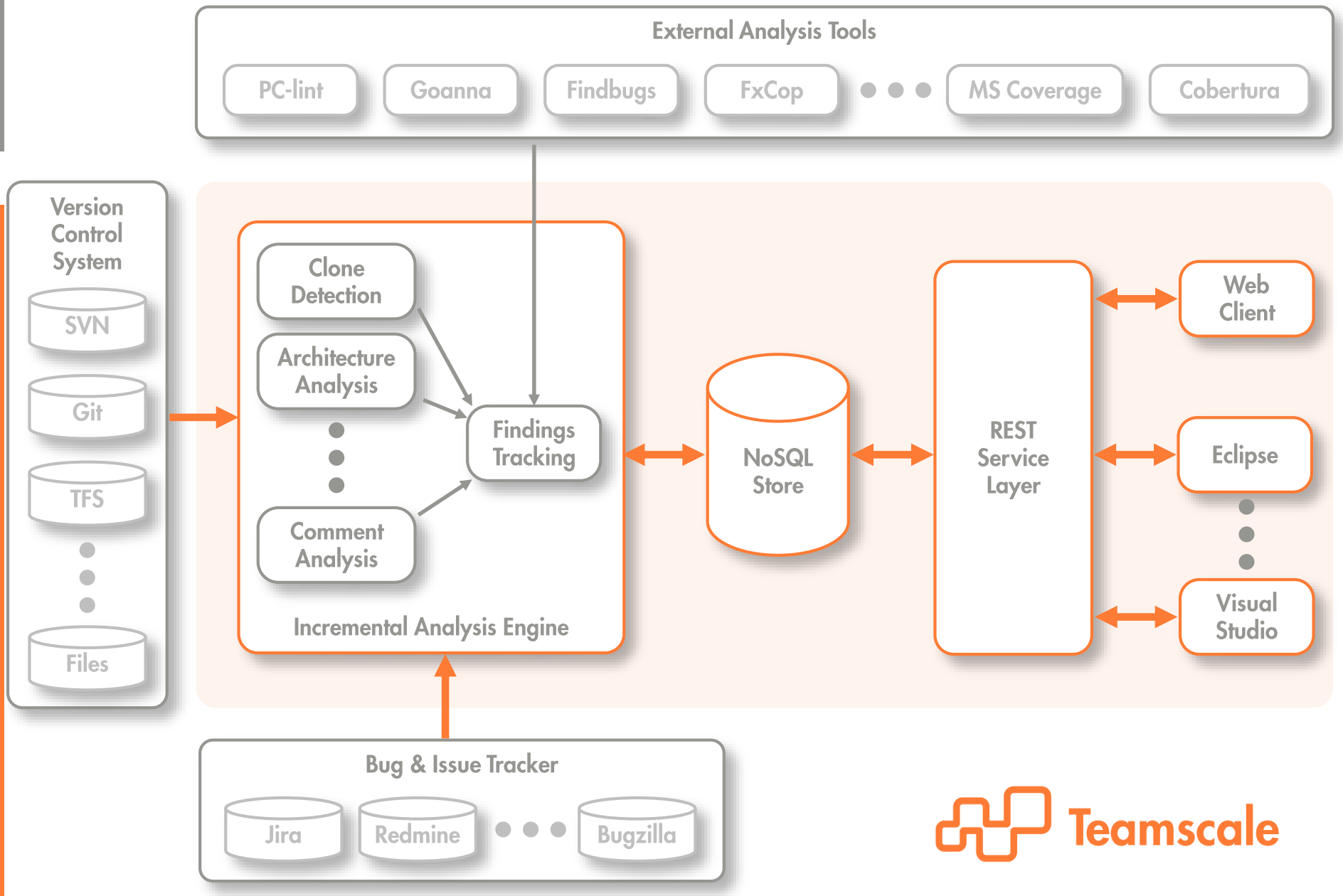



MoneyTest
BugMultiply
BugNegate
BugSimpleAdd


conQAT


Loc Trend


Baseline
MoneyTest
BagMultiply
BagNegate
BagSimpleAdd
Baseline
Dashboard
Latest Vers
Microsoft
Visual Studio
Team System
Dashboard
inkl Delta


External Analysis Tools
PC-lint
Goanna
Findbugs
FxCop
MS Coverage
Cobertura
Version Control System
SVN
Git
TFS
Files
Clone Detection
Architecture Analysis
Comment Analysis
Findings Tracking
Incremental Analysis Engine
NoSQL Store
REST Service Layer
Web Client
Eclipse
Visual Studio
Bug & Issue Tracker
Jira
Redmine
Bugzilla
Teamscale

ACT-1270 Fixing Inconsistent handling of serializable process variables
by Victoria King in revision e1aa41b4b133d269980fff3f81d008da8f21a109 (git)

Jun 29 2012
16:05

changed 2 files
-4 findings

ACT-1258 Merging Pablo's work into trunk
by Jacob Nelson in revision 9e664a1f0676cedcbe03415a253e8c3e4a58944c (git)

Jun 29 2012
14:41

added 3 files, changed 2 files
-1 findings

Fix for ACT-1059: Task#setDelegationState(DelegationState) was not saved in database
by Michael Harris in revision 1f48dcad04bc4a621e60af047fb121ae161bca30 (git)

Jun 28 2012
21:45

changed 3 files
+2 findings

ACT-991 Removed user id from exception message in order not to leak sensitive information
by Michael Harris in revision e9a09424e6309c854c44ac5d08740a8ffb082fc9 (git)

Jun 28 2012
15:26

changed 2 files
-2 findings

Teamscale

```
/// <param name="authority">Die Zuweisung die deaktiviert werden soll</param
void IAuthorityListManager.DeactivateAuthority(DecMemoIdentifier decMemoId,
{
   this.delegateManager.DeactivateAuthority(decMemoId, authorityList as Autho
}

/// <summary>
/// Führt die Laufliste fort (geht zur nächsten Zuweisung über wenn die aktuelle Z
/// </summary>
/// <param name="decMemoId">Identifikator der Entscheidungsvorlage</param
/// <param name="authorityList">Die Laufliste die fortgeführt werden soll</para
/// <returns>Die nach dem Fortführen aktive Zuweisung der Laufliste</returns>
IAuthorityAssignment IAuthorityListManager.Proceed(DecMemoIdentifier decMem
{
   if (!this.CheckCurrentUserMayProceed(authorityList as AuthorityList))
   {
      throw new AuthorityListException(Error_27.CurrentUserMayNotProceedAut
   }
   if (authorityList.State == AuthorityListState.InProgress)
   {
      DTOComplex decMemoData = this.GetDecMemo(decMemoId, Currency.Neu
      ((IDecMemoState)this).SubmitDecMemo(decMemoId, decMemoData);
      IAuthorityAssignment newActiveAssignment = this.delegateManager.Proce
      return newActiveAssignment;
   }
   else
   {
      return this.delegateManager.Proceed(decMemoId, authorityList as Authorit
   }
}
...
```

```
void IAuthorityListManager.DeactivateAuthority(DecMemoIdentifier decMe
{
   this.delegateManager.DeactivateAuthority(decMemoId, authorityList as
}

/// <summary>
/// Führt die Laufliste fort (geht zur nächsten Zuweisung über wenn die ak
/// </summary>
/// <param name="decMemoId">Identifikator der Entscheidungsvorlage<
/// <param name="authorityList">Die Laufliste die fortgeführt werden soll
/// <returns>Die nach dem Fortführen aktive Zuweisung der Laufliste</re
IAuthorityAssignment IAuthorityListManager.Proceed(DecMemoIdentifier
{
   if (!this.CheckCurrentUserMayProceed(authorityList as AuthorityList))
   {
      throw new AuthorityListException(Error_27.CurrentUserMayNotProce
   }
   if (authorityList.State == AuthorityListState.InProgress)
   {
      DTOComplex decMemoData = ((ICedentDecMemoStore)this).GetCed
      ((IDecMemoState)this).SubmitDecMemo(decMemoId, decMemoData);
      IAuthorityAssignment newActiveAssignment = this.delegateManager
      base.CommitTransaction();
      return newActiveAssignment;
   }
   else
   {
      return this.delegateManager.Proceed(decMemoId, authorityList as A
   }
}
...
```

fixed: latest change is no longer lost when assigning entry to a keyword group while it is being edited

by jzieren in revision e0ca9a51b50c8b01f579f4eef79028bff6c34028 (git)

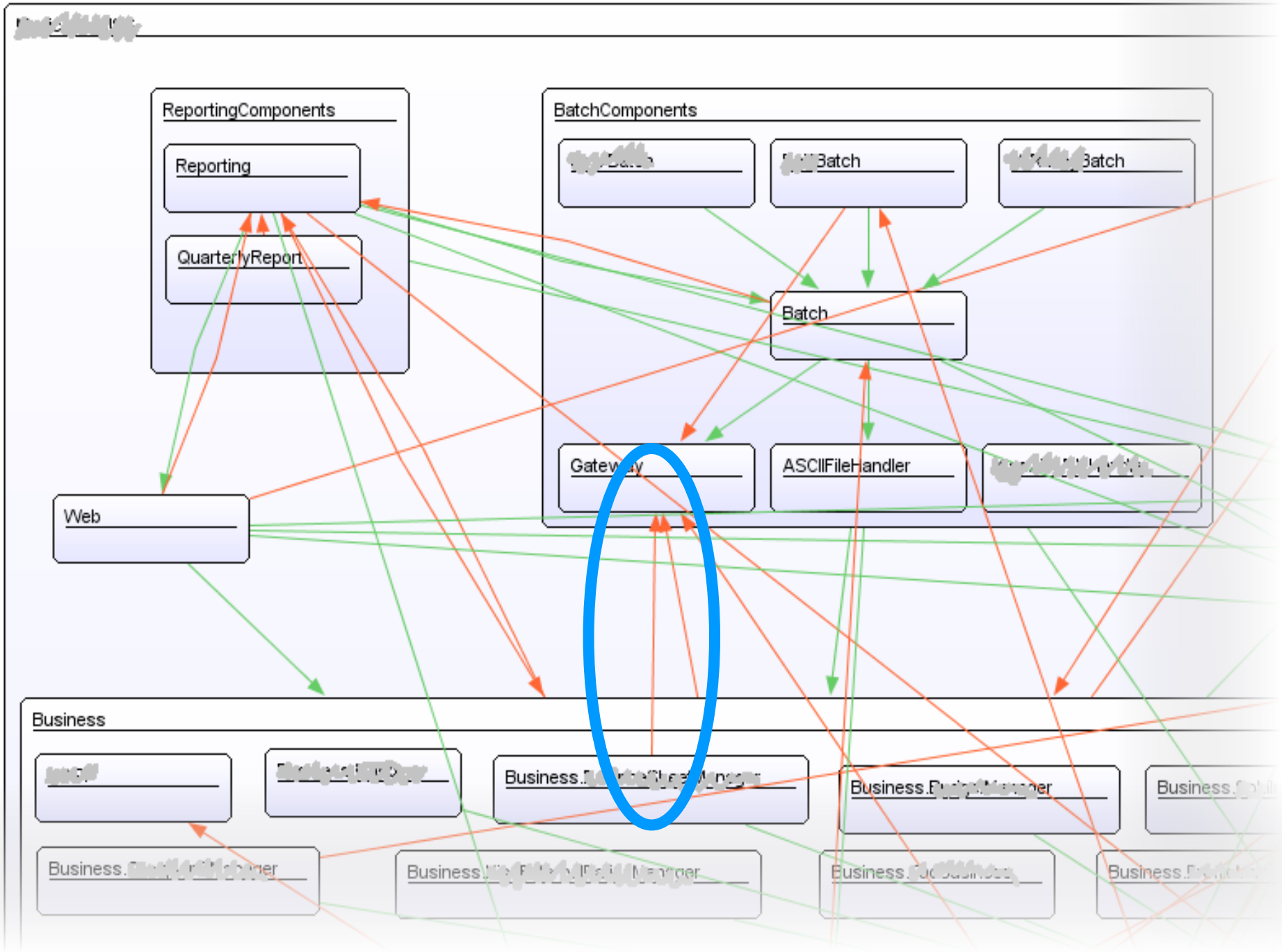
May 26 2005
15:58

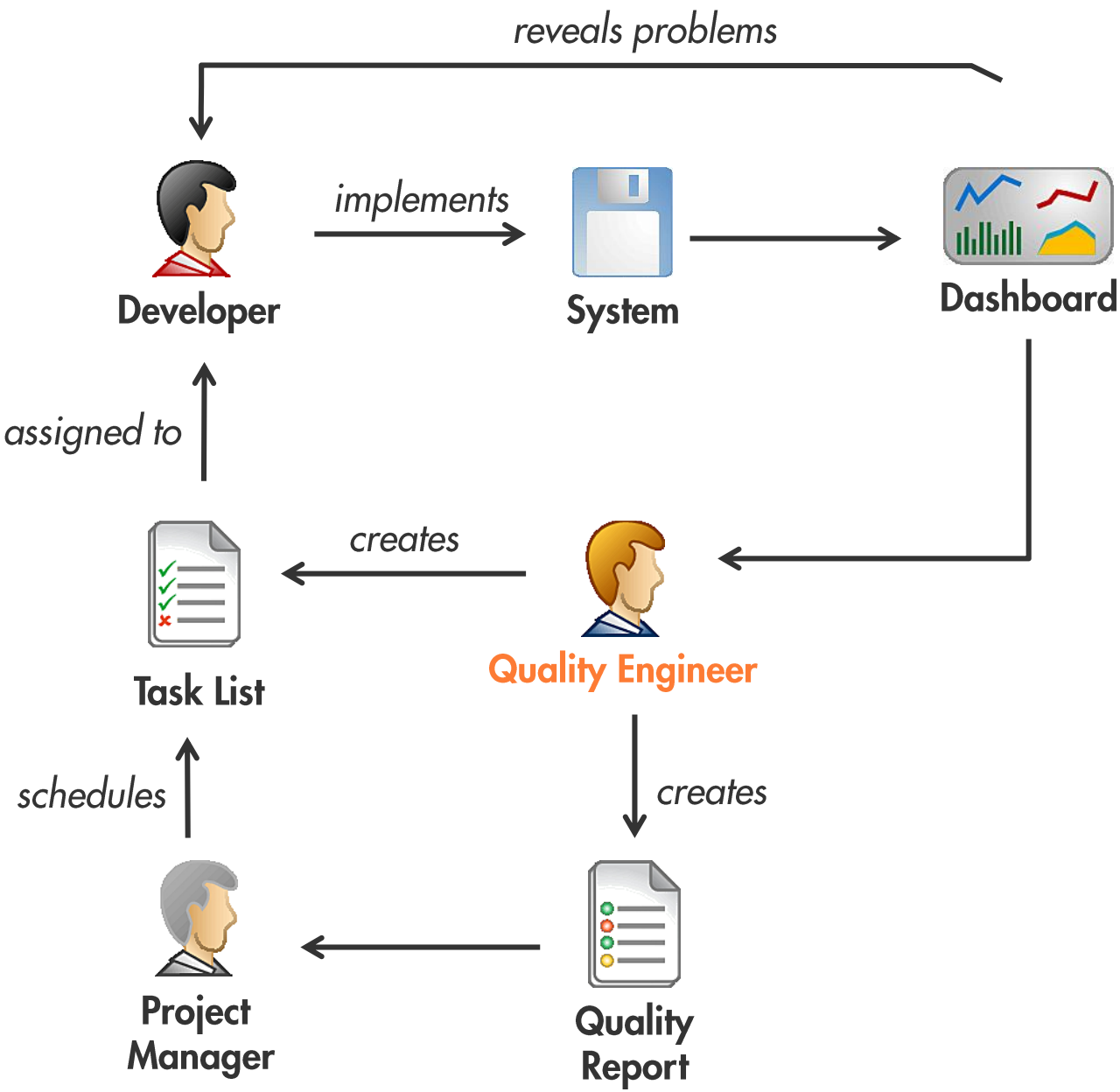
1 alerts:

Message	Context
Found potential inconsistent clone change in RightClickMenu.java	[Broken clone] [Old clone finding] [Code change]

2 removed findings:

Message	Location	Finding Group
Clone with 2 instances of length 10	src/java/net/sf/.../RightClickMenu.java:366-380	Code Duplication / Cloning
Clone with 2 instances of length 10	src/java/net/sf/.../RightClickMenu.java:340-354	Code Duplication / Cloning


ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Batch
Batch
Batch
Gateway
ASCIIFileHandler
Web
Business
Business.
Business.
Business.
Business.
Business.
Business.
Business.
Business.


reveals problems
implements
Developer
System
Dashboard
assigned to
creates
Task List
Quality Engineer
schedules
creates
Project Manager
Quality Report

Weitere Best Practices

- Mit Standortbestimmung starten
- Spezifische Sichten für Stakeholder (aber: Transparenz!)
- Code Peer-Reviews
- Projektspezifisches Tailoring der Analysen
- Manuelle Reviews von KPI Verbesserungen
- Vollautomatische *Messung*. Manuelle *Bewertung*.
- Schnellstmögliches Feedback (Integration in IDE, Daily Builds, …)
- …

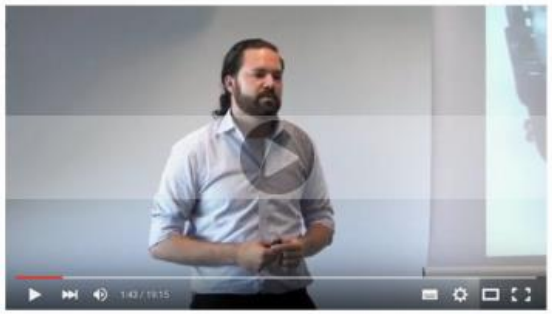
Fazit

Für belastbare Ergebnisse muss Messung und Bewertung voneinander getrennt werden. Werkzeuge messen, Menschen bewerten.

Wirksame Qualitätsanalysewerkzeuge unterstützen daher den Quality Engineer und versuchen nicht, ihn zu ersetzen.

Wie sag ichs meinem Chef?

Mit Vorträgen über Softwarequalität sind wir regelmäßig auf Industriekonferenzen oder Kunden-internen Workshops vertreten.



Impulsvorträge

Gerne kommen wir auch zu Ihnen ins Haus, beispielsweise für interne Konferenzen oder Workshops. Unsere Themen reichen von Qualitätsanalysen über Qualitätscontrolling bis hin zu Testcontrolling oder der Einführung von Reviews. Oder aber Sie schlagen uns ein Thema Ihrer Wahl vor.

DAS ANGEBOT

- 60-90 MIN VORTRAG
- SOFTWAREQUALITÄT ALS THEMA
- BEI IHNEN IM HAUS
- € NUR UNSERE ANREISEKOSTEN
- TERMIN NACH VEREINBARUNG

IMPULSVORTRAG ANFRAGEN

Do Code Clones Matter?

Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, Stefan Wagner
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{juergens,deissenb,hummelb,wagnerst}@in.tum.de

Abstract

Code cloning is not only assumed to inflate maintenance costs but also considered defect-prone as inconsistent changes to code duplicates can lead to unexpected behavior. Consequently, the identification of duplicated code, clone detection, has been a very active area of research in recent years. Up to now, however, no substantial investigation of the consequences of code cloning on program correctness has been carried out. To remedy this shortcoming, this paper presents the results of a large-scale case study that was undertaken to find out if inconsistent changes to cloned code can represent faults. For the analyzed commercial and open source systems we not only found that inconsistent changes to clones are very frequent but also identified a significant number of faults induced by such changes. The clone detection tool used in the case study implements a novel algorithm for the detection of inconsistent clones. It is available as open source to enable other researchers to use it as basis for further investigations.

1. Clones & correctness

Research in software maintenance has shown that many programs contain a significant amount of duplicated (cloned) code. Such cloned code is considered harmful for two reasons: (1) multiple, possibly unnecessary, duplicates of code increase maintenance costs and, (2) inconsistent changes to cloned code can create faults and, hence, lead to incorrect program behavior [19, 28]. While clone detection has been a very active area of research in recent years, up to now, there is no thorough understanding of the degree of harmfulness of code cloning. In fact, some researchers even started to doubt the harmfulness of cloning at all [16].

To shed light on the situation, we investigated the effects of code cloning on program correctness. It is important to understand, that clones do not directly cause faults but inconsistent changes to clones can lead to unexpected program behavior. A particularly dangerous type of change to cloned code is the *inconsistent bug fix*. If a fault was found in cloned code but not fixed in *all* clone ins
the system is likely to still exhibit the incorrect be
To illustrate this, Fig. 1 shows an example, where a
null-check was retrofitted in only one clone instance

This paper presents the results of a large-scale cas
that was undertaken to find out (1) if clones are chan
consistently, (2) if these inconsistencies are introdu
tentionally and, (3) if unintentional inconsistencies c
resent faults. In this case study we analyzed three c
cial systems written in C#, one written in Cobol a
open-source system written in Java. To conduct th
we developed a novel detection algorithm that ena
to detect inconsistent clones. We *manually* inspecte
900 clone groups to handle the inevitable false positi
discussed each of the over 700 inconsistent clone
with the *developers* of the respective systems to de
if the inconsistencies are intentional and if they re
faults. Altogether, around 1800 individual clone gr
sessments were manually performed in the course
case study. The study lead to the identification of 10
that have been confirmed by the systems' developer

Research Problem Although most previous work
that code cloning poses a problem for software
nance, "there is little information available concern
impacts of code clones on software quality" [28].
consequences of code cloning on program correctn
particular, are not fully understood today, it remains
how *harmful* code clones really are. We consider
sence of a thorough understanding of code cloning
ous for software engineering research, education an
tice.

Contribution The contribution of this paper is t
First, we extend the existing empirical knowledge by
study that demonstrates that clones get changed in
tently and that such changes can represent faults. S
we present a novel suffix-tree based algorithm for the
tion of inconsistent clones. In contrast to other alg
for the detection of inconsistent clones, our tool
made available for other researchers as open source.

The Loss of Architectural Knowledge during System Evolution: An Industrial Case Study

Martin Feilkas and Daniel Ratiu and Elmar Jürgens
Institut für Informatik
Technische Universität München
Boltzmannstr. 3, D-85748 Garching
feilkas|ratiu|juergens@in.tum.de

Abstract

Architecture defines the components of a system and their dependencies. The knowledge about how the architecture is intended to be implemented is essential to keep the system structure coherent and thereby comprehensible. In practice, this architectural knowledge is explicitly formulated only in the documentation (if at all), which usually gets outdated very soon. This leads to a growing amount of implicit knowledge during evolution that is especially volatile in projects with high developer fluctuation.

In this paper we present a case study about the loss of architectural knowledge in three industrial projects to answer the following research questions: 1) to what degree is the architectural documentation kept in conformance with the code? 2) how well does the documentation reflect the intended architecture?, 3) how big is the architectural decay?, and 4) what are the causes for nonconformances? We answer these questions by investigating the architecture documentation, the source code, and by performing interviews with developers.

The most important outcomes of our study are: the informal documentation and the source code are not kept in conformance with each other, none of them completely reflects the intended architecture, and even developers taken individually are not completely aware of the intended architecture. Quantitatively, between 70% and 90% of these nonconformances are caused by flaws in the documentation and between 10% and 30% represent architectural violations in the code.

1 Introduction

The architecture defines the structure of a software system in terms of components and (allowed) dependencies. A suitable architecture is a fundamental prerequisite for evolvable and understandable systems [5]. Developers need knowledge about the intended architecture of a system whenever they do any modification. Without this knowledge, programmers can break the architectural integrity of the system accidentally, even when making only small code changes.

Today's widely used programming languages offer only very primitive mechanisms for making the architecture in the code explicit. Therefore, in everyday industrial practice, the information about the architecture is contained in external documentation in form of diagrams and natural language texts that often originate from early phases of the system design. During system evolution, the architecture often needs to be adapted, extended and modified in response to changes to the requirements, additional features or simply new insights about shortcomings of the initial design. These changes are inevitable even if an 'optimal design strategy' is used [13]. Needless to say, this effect is amplified in an industrial environment. When these changes to the intended architecture happen, they are often (unintentionally) not introduced into the architecture documentation and not propagated to other team members [10]. This leads to a gap between the intended architecture of the system, how different developers perceive it, how it is made explicit in the documentation and how it is actually implemented in the code.

Figure 1-left illustrates the ideal situation: All developers possess accurate knowledge about the system's architecture, the architecture is accurately documented and accurately implemented in the code. The right side of the figure illustrates the situation typically encountered in industrial projects: Different developers understand the architecture of big systems in (slightly) different manners, with none of them having an accurate view of the intended architecture. Furthermore, only a part of the intended architecture is documented and only a part of the code complies with it. As depicted in Figure 1-right, the *loss of architectural knowledge* can be observed in different forms: missing architectural information in the documentation, violations of

https://www.cqse.eu/publications/

Software Quality Blog

Practical Guide to Code Clones (Part 1)

Posted on 07/16/2014 *by* Dr. Benjamin Hummel

One well known principle in software engineering states *don't repeat yourself*, also known as the DRY principle. A very obvious violation of DRY is the application of copy/paste to create duplicates of large portions of source code within the same code base. These duplicate pieces of code, also known as *code clones*, have been subject to lots of research in the last two decades. In this two-part post I want to summarize those parts of the current knowledge that I find most relevant to the practitioner, especially the impact of clones on software dev

Practical Guide to Code Clones (Part 2)

Posted on 07/30/2014 *by* Dr. Benjamin Hummel

In the previous part we introduced the notion of code clones and discussed, whether and under which circumstances cloning in your code base can be a problem for development and maintenance. In this post, I will introduce ways and tools to deal with code clones in your code base. After reading this, you should be able to select and apply a detection tool to inspect the clones in your own code base.

https://www.cqse.eu/en/blog/practical-guide-to-code-clones-part1/

Visit our Teamscale online demo

Get a quick impression of Teamscale and what it can do to help you create high quality code.

Read our brief tutorial to get you started.

VISIT TEAMSCALE ONLINE DEMO

https://www.cqse.eu/en/products/teamscale/try/

Teamscale Lizenz

Mail an juergens@cqse.eu

- Betreff: Teamscale Lizenz
- Bis Ende Juni

Ich schicke Euch eine komplett offene Lizenz für ½ Jahr

Kontakt

Ich bin am Nachmittag hier und freue mich auf Diskussionen ☺

Bei Interesse an einem Job in diesem Bereich bitte melden ☺

Dr. Elmar Jürgens · juergens@cqse.eu · +49 179 675 3863

@ElmarJuergens
@teamscale
www.cqse.eu/en/blog

CQSE GmbH, Lichtenbergstraße 8,
85748 Garching bei München

CQSE
Continuous Quality in Software Engineering