BASTA!

.NET, WINDOWS, JAVASCRIPT

Dr. Elmar Juergens | CQSE GmbH & TU München

Vom Wiegen wird die Sau nicht fett

Von Qualitätsanalyse zu wirksamer Qualitätsverbesserung

Über Mich

Forschung

- Clone Detection, Architekturanalyse
- Effektivität und Effizient von QS-Maßnahmen

Beratung

- Gründer
- Qualitäts-Bewertung & Qualitäts-Controlling

Gesellschaft für Informatik

- Zum Junior-Fellow ernannt
- Erfahruzngsaustausch Forschung <-> Praxis

XmlWellformedWriter.cs

```
        if (localName == null || localName.Length == 0)
        {
            throw new ArgumentException(SR.Xml_EmptyLocalName);
        }
        CheckNCName(localName);

        AdvanceState(Token.StartElement);

        // lookup prefix / namespace
        if (prefix == null)
        {
            if (ns != null)
            {
                prefix = LookupPrefix(ns);
            }
            if (prefix == null)
            {
                prefix = string.Empty;
            }
        }
        else if (prefix.Length > 0)
        {
            CheckNCName(prefix);
            if (ns == null)
            {
                ns = LookupNamespace(prefix);
            }
            if (ns == null || (ns != null && ns.Length == 0))
            {
                throw new ArgumentException(SR.Xml_PrefixForEmptyNs);
            }
        }
        if (ns == null)
        {
            ns = LookupNamespace(prefix);
            if (ns == null)
            {
                Debug.Assert(prefix.Length == 0);
                ns = string.Empty;
```

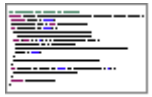
XmlWellformedWriter.cs

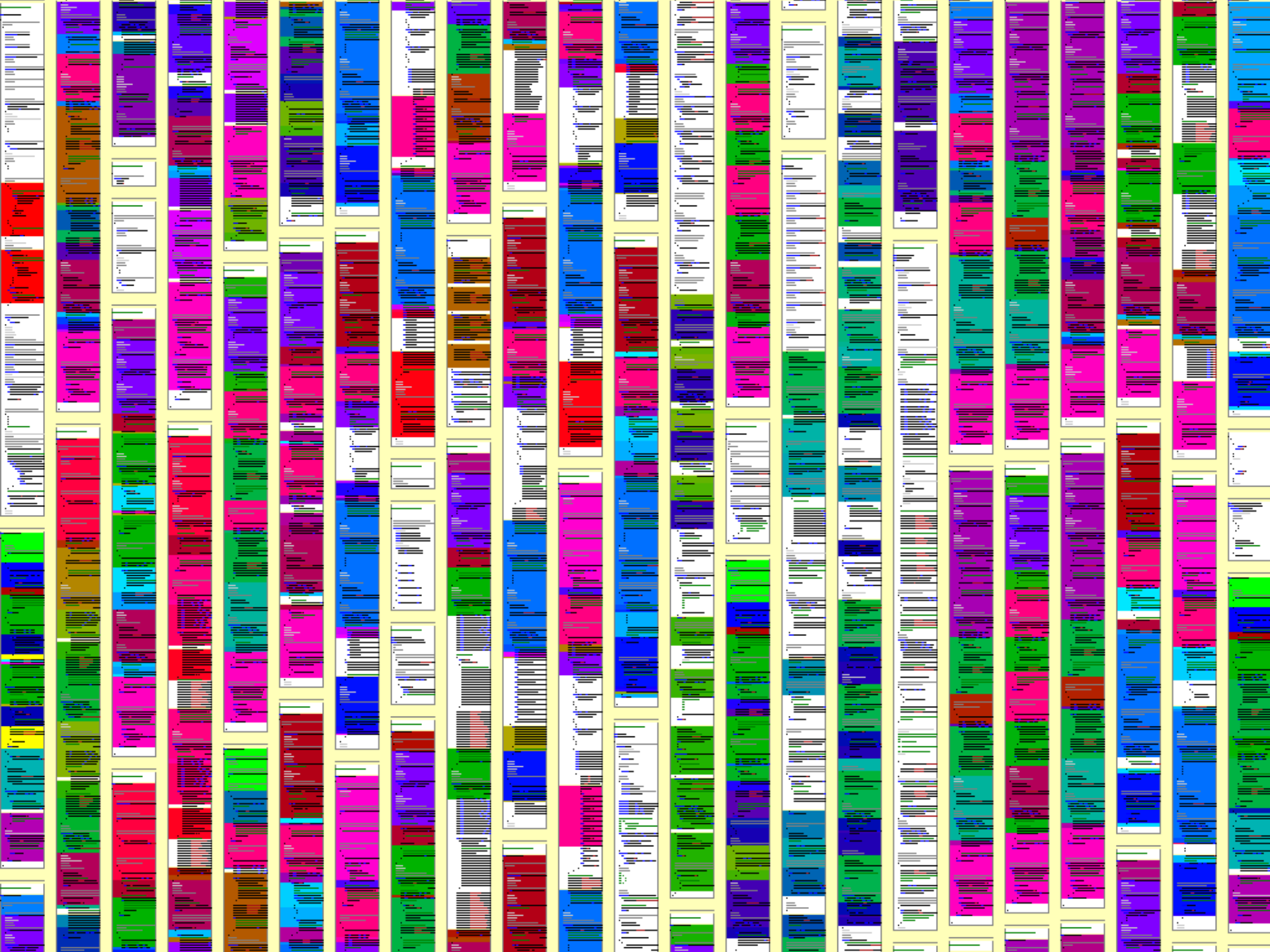
```
        if (string.IsNullOrEmpty(localName))
        {
            throw new ArgumentException(SR.Xml_EmptyLocalName);
        }
        CheckNCName(localName);

        AdvanceState(Token.StartElement);

        // lookup prefix / namespace
        if (prefix == null)
        {
            if (ns != null)
            {
                prefix = LookupPrefix(ns);
            }
            if (prefix == null)
            {
                prefix = string.Empty;
            }
        }
        else if (prefix.Length > 0)
        {
            CheckNCName(prefix);
            if (ns == null)
            {
                ns = LookupNamespace(prefix);
            }
            if (string.IsNullOrEmpty(ns))
            {
                throw new ArgumentException(SR.Xml_PrefixForEmptyNs);
            }
        }
        if (ns == null)
        {
            ns = LookupNamespace(prefix);
            if (ns == null)
            {
                Debug.Assert(prefix.Length == 0);
                ns = string.Empty;
```

```
// 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();
}
```

```
// 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.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();
}
```





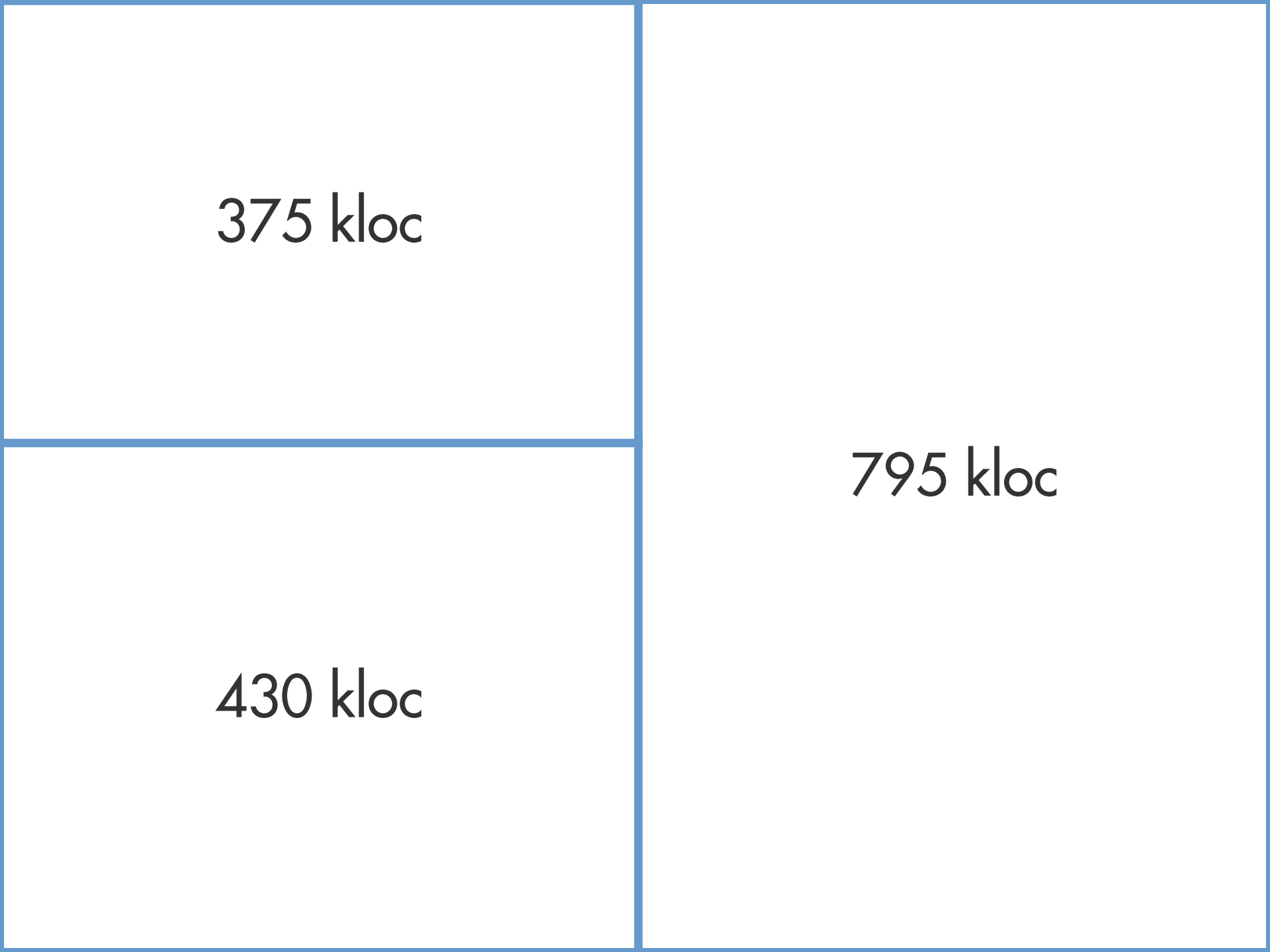
Studie

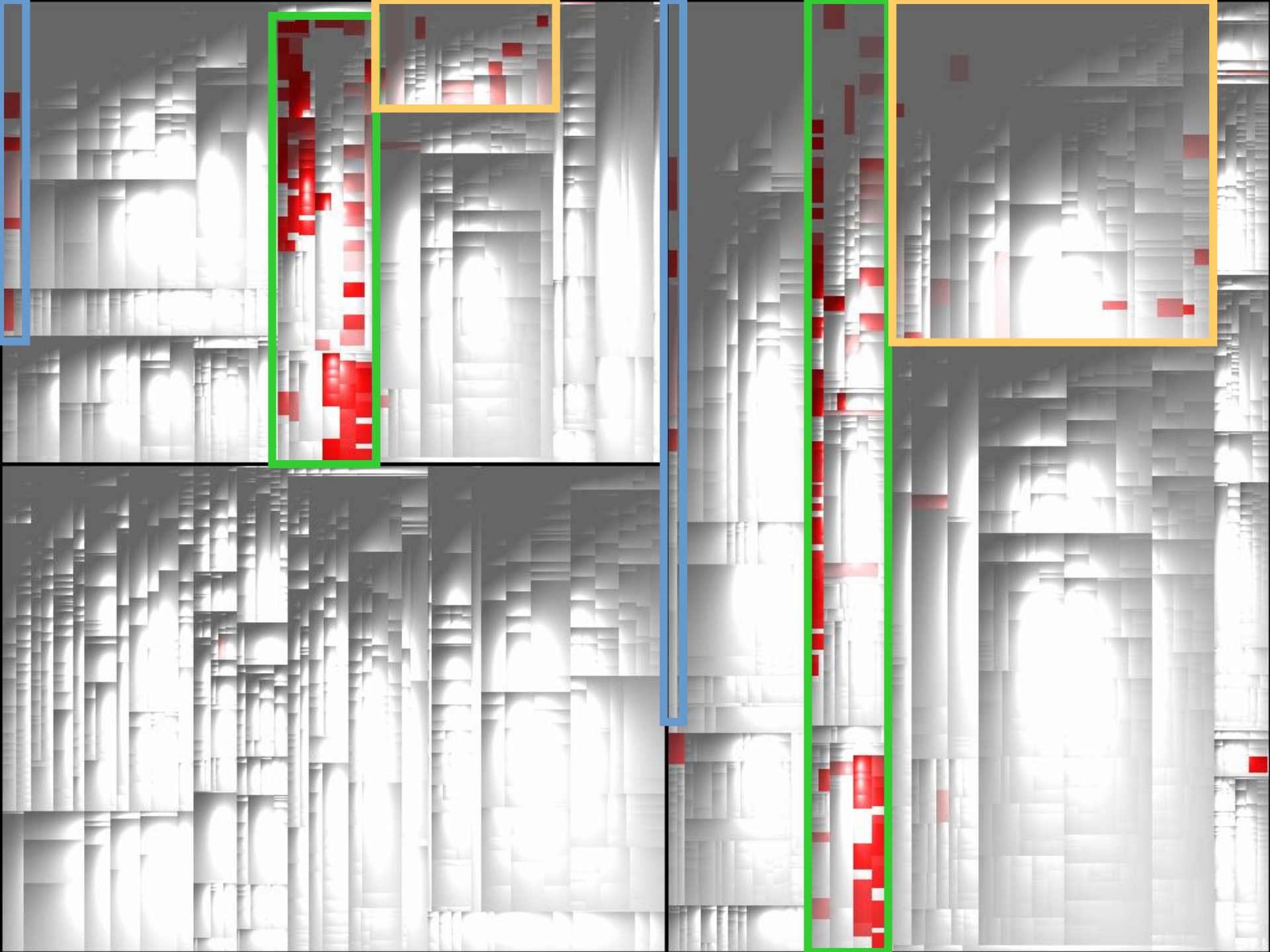
Munich Re LV 1871

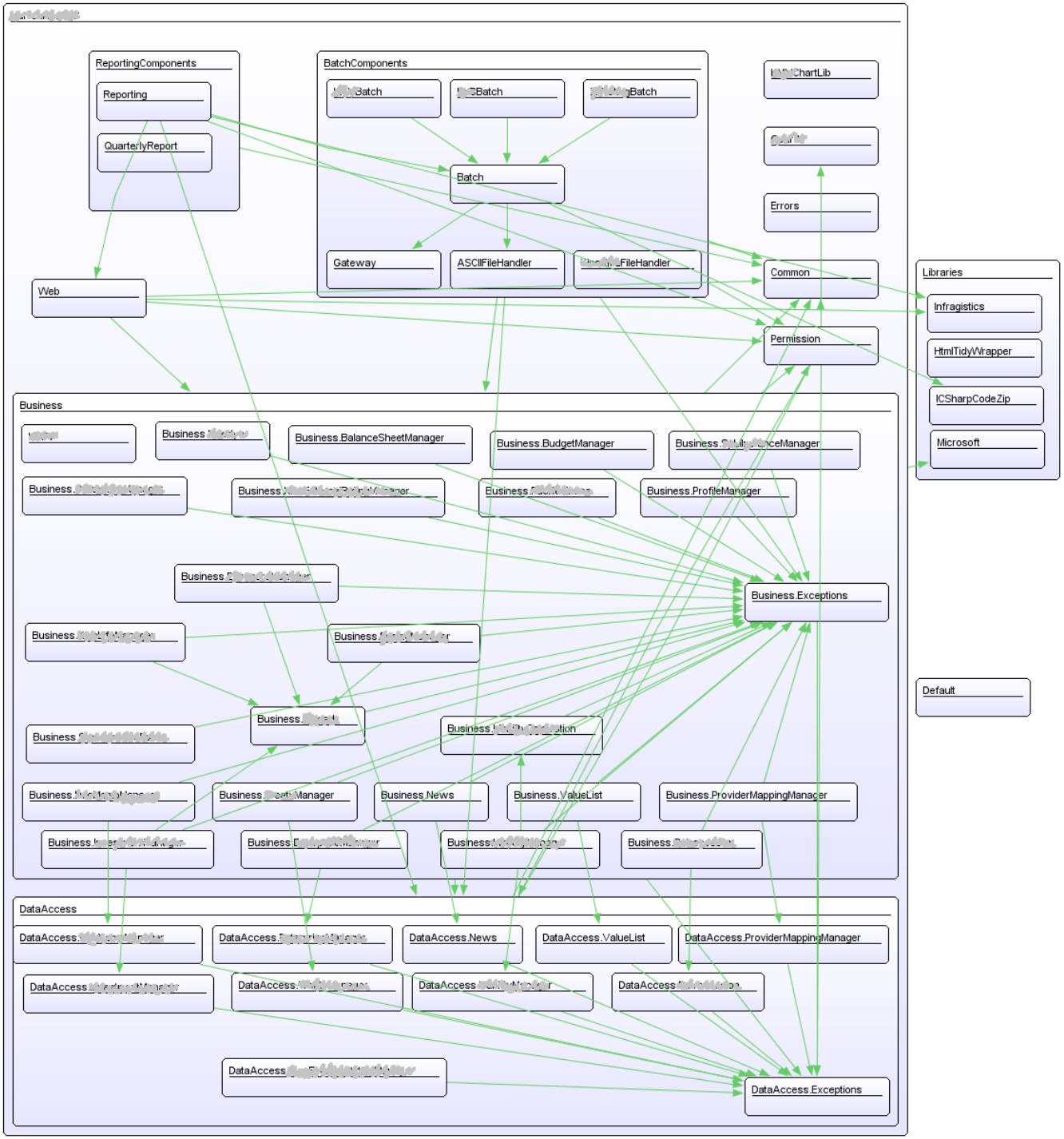
- Über 100 Fehler in produktiver Software

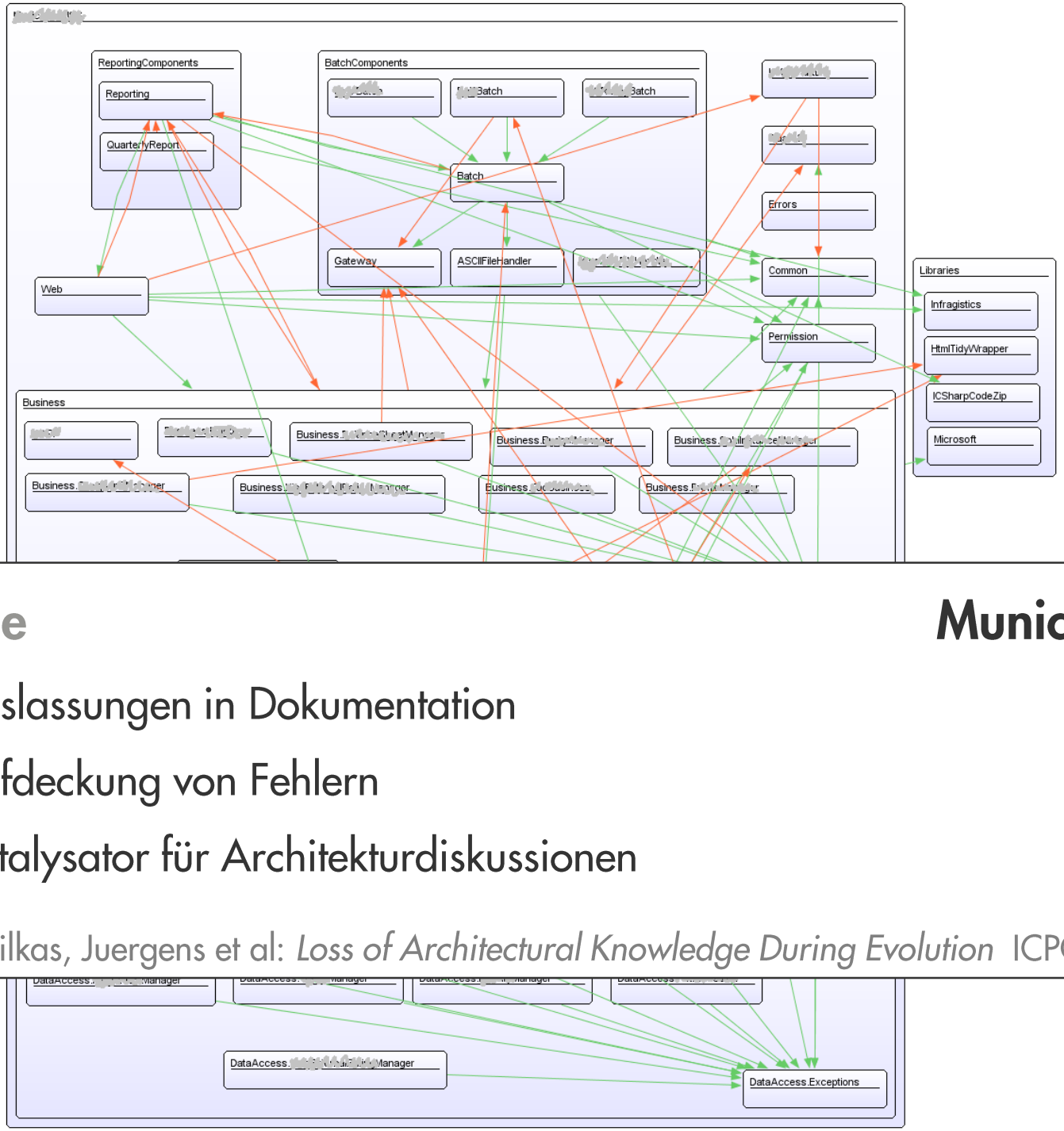
- 52% aller ungewollten Unterschiede fehlerhaft

Juergens, Deissenboeck et al: *Do Code Clones Matter?* ICSE 2009


375 kloc
795 kloc
430 kloc




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
Default
DataAccess
DataAccess.News
DataAccess.ValueList
DataAccess.ProviderMappingManager
DataAccess.Exceptions

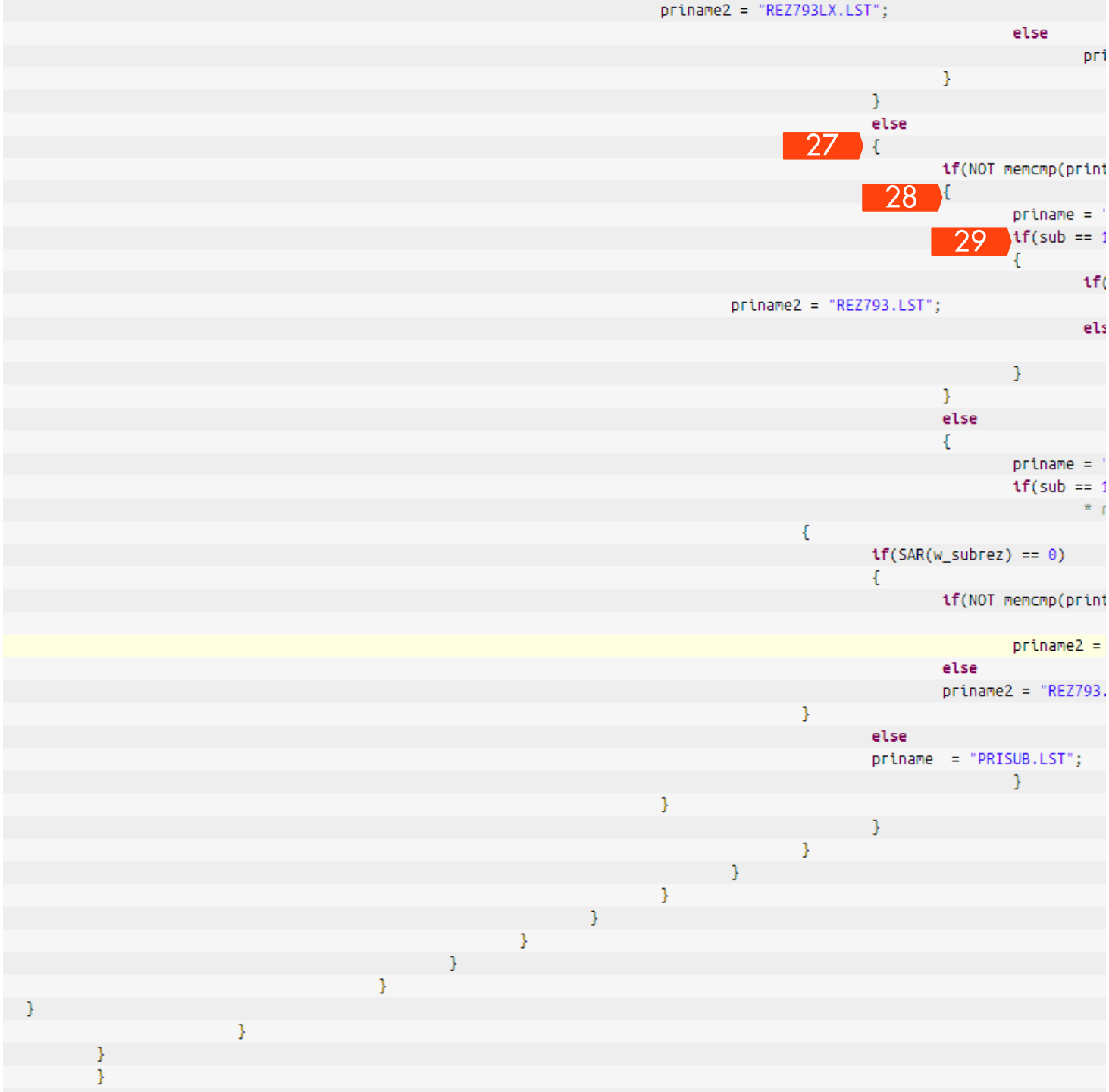


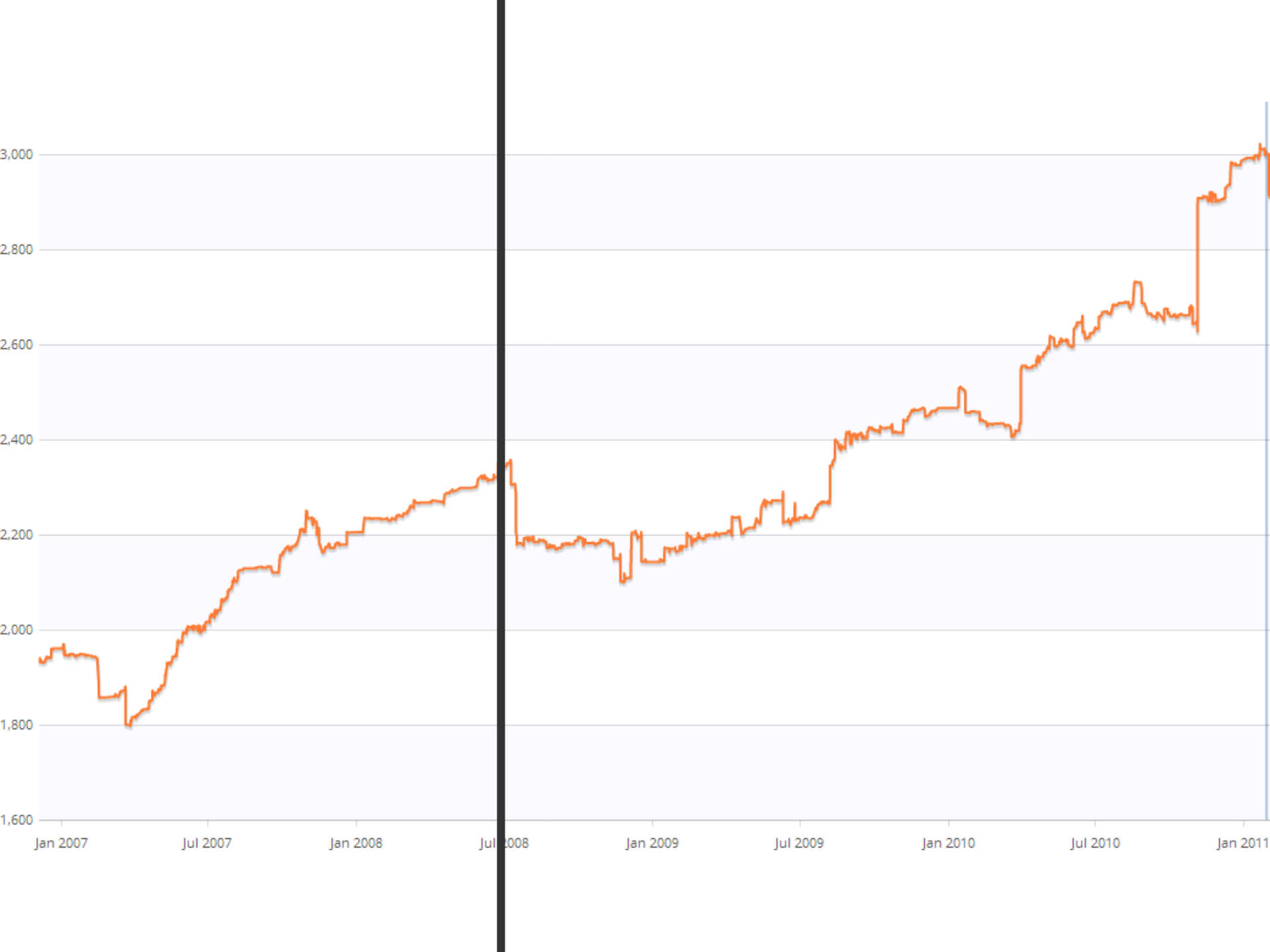
Studie

Munich Re

- Auslassungen in Dokumentation
- Aufdeckung von Fehlern
- Katalysator für Architekturdiskussionen

Feilkas, Juergens et al: *Loss of Architectural Knowledge During Evolution* ICPC 2009


priname2 = "REZ793LX.LST";
else
pri
}
}
else
27
{
if(NOT memcmp(print
28
{
priname = '
29
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";
}
}
}
}
}
}
}
}
}
}
}
}
}


3,000
2,800
2,600
2,400
2,200
2,000
1,800
1,600
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011

+

=


soft cloths
soft cloths
BANKERS BOX
33250
33250
SOLE F80


WARNING
BRIGHT LIGHT

$$171 - 5.2 \cdot \ln(avgHV) - 0.23 \cdot \text{?}gCC(g') - 16.2 \cdot \ln(avgLOC) + 50 \cdot \text{si}(\text{sqrt}(2.4 \cdot perCM))$$

?

HV: Halstead Volume

CC: Cyclomatic Complexity

LOC: lines of cod

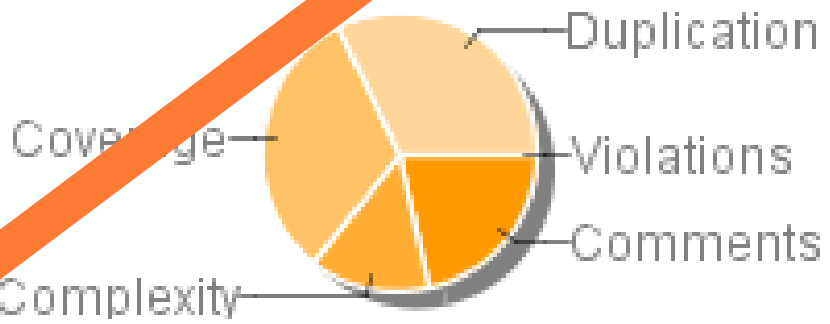
perCM: % Comment Lines

?

Technical Debt

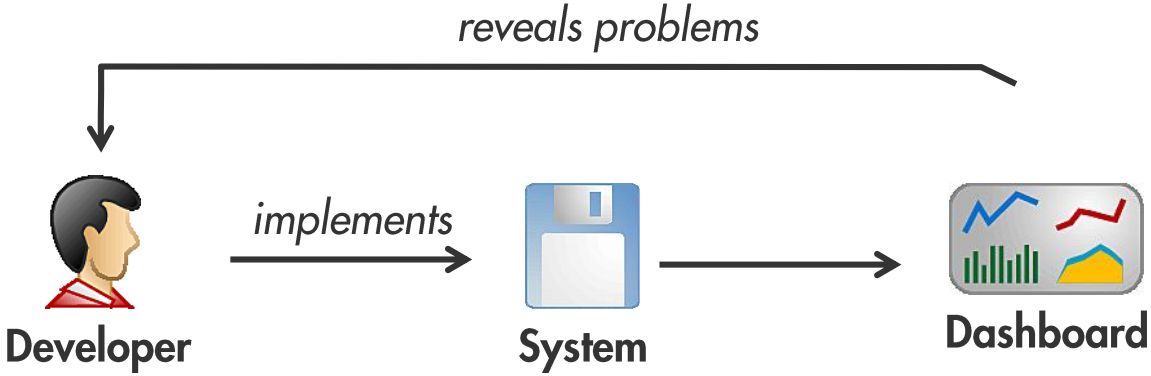
$ 14'605

29 man days



Best Practice: Zielorientierte Analysen

- Objektiv
- Auswirkungen von Code-Änderungen verständlich
- Actionable
- Nachvollziehbarer Zusammenhang zu Wartungstätigkeit


reveals problems
implements
Developer
System
Dashboard

Best Practice: Diskriminierung (von Code)

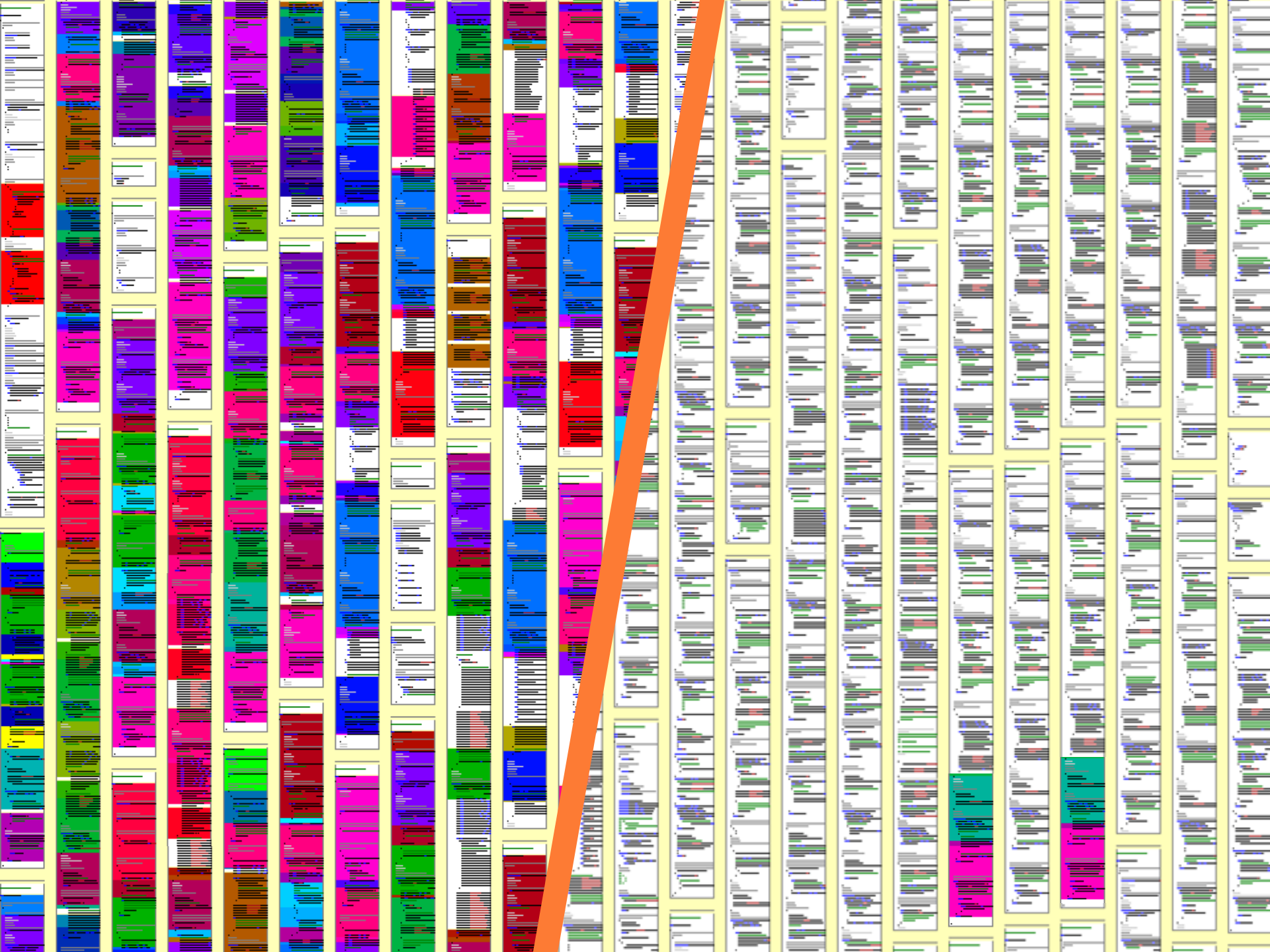
Art der Wartung

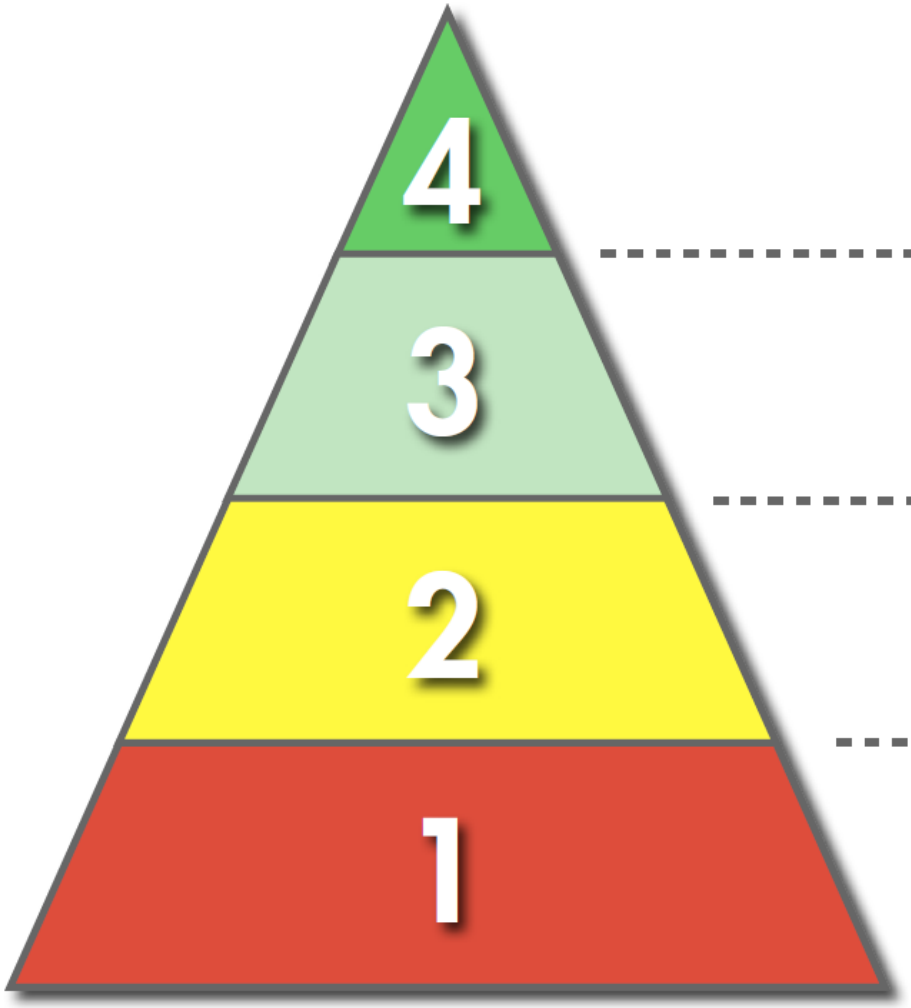
- Manuell
- Generator
- Überhaupt nicht (Wegwerf-Prototyp)

Aufgabe im Projekt

- Teil der Anwendung
- Test
- Hilfswerkzeug

Mit manuell gewartetem Applikationscode beginnen



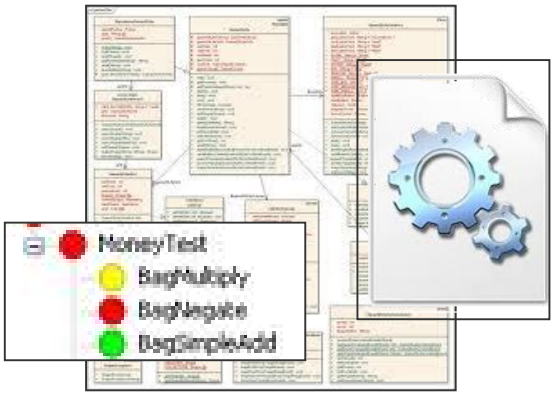


Keine Defizite

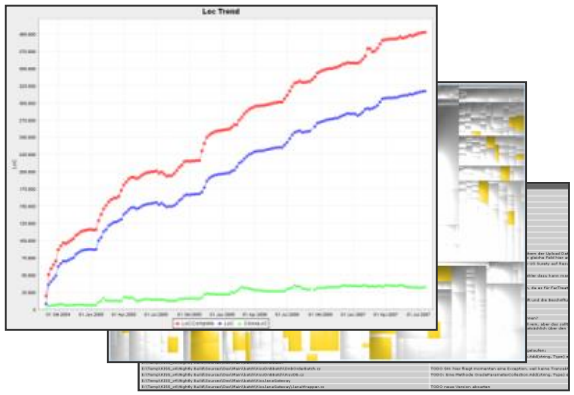
Keine Defizite in geändertem Code

Keine neuen Defizite

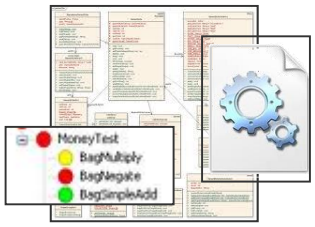
Egal


MoneyTest
BagMultiply
BagNegate
BagSimpleAdd

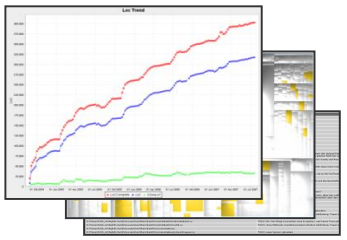

conQAT


Loc Trend

Baseline

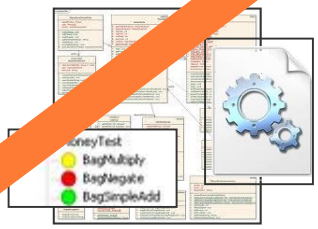


Baseline
Dashboard

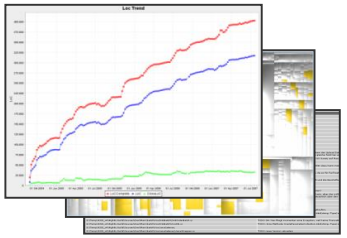


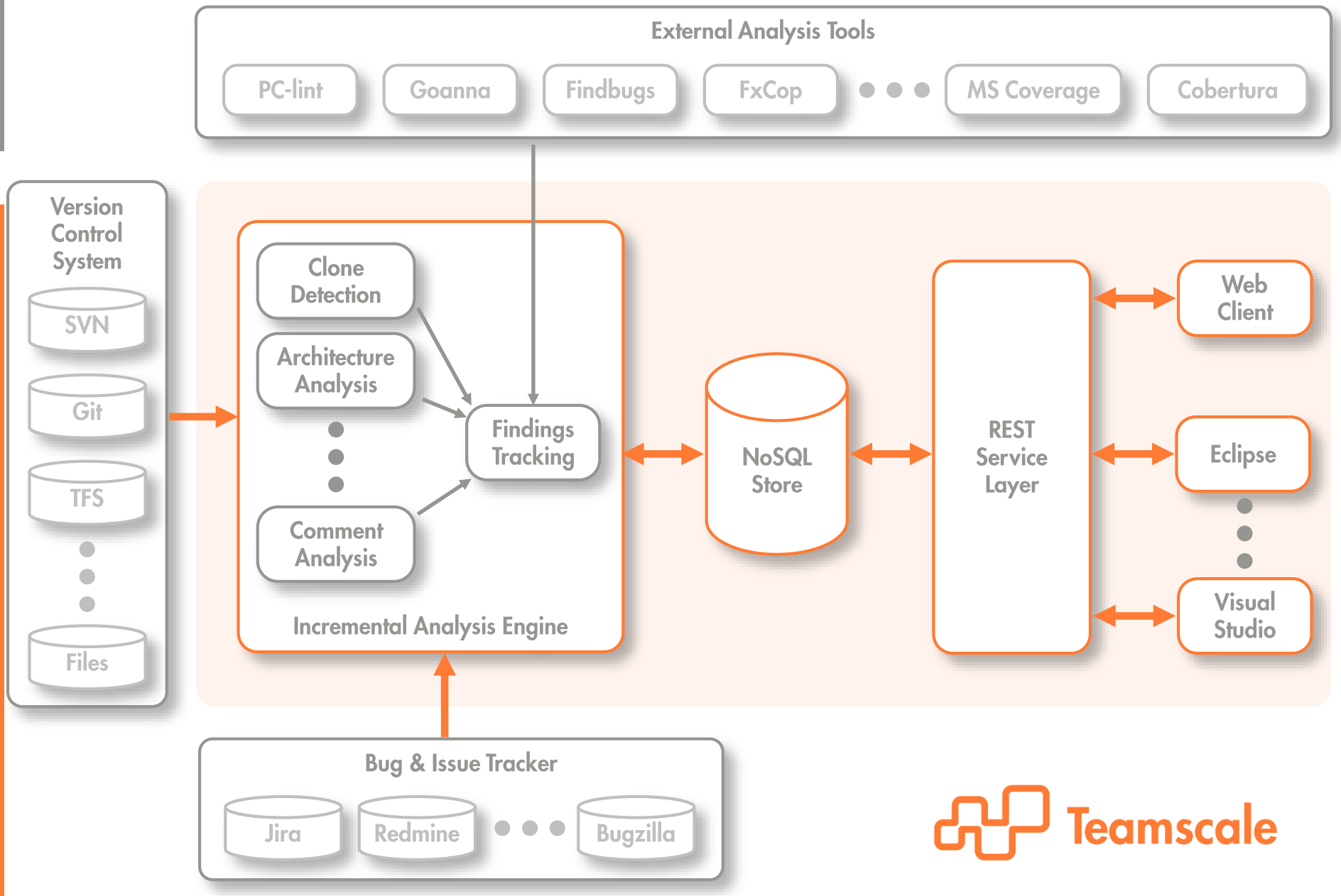
Microsoft Visual Studio Team System

Latest Versi



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)

changed 2 files
-4 findings

Jun 29 2012
16:05

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

added 3 files, changed 2 files
-1 findings

Jun 29 2012
14:41

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

changed 3 files
+2 findings

Jun 28 2012
21:45

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

changed 2 files
-2 findings

Jun 28 2012
15:26

Teamscale

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:

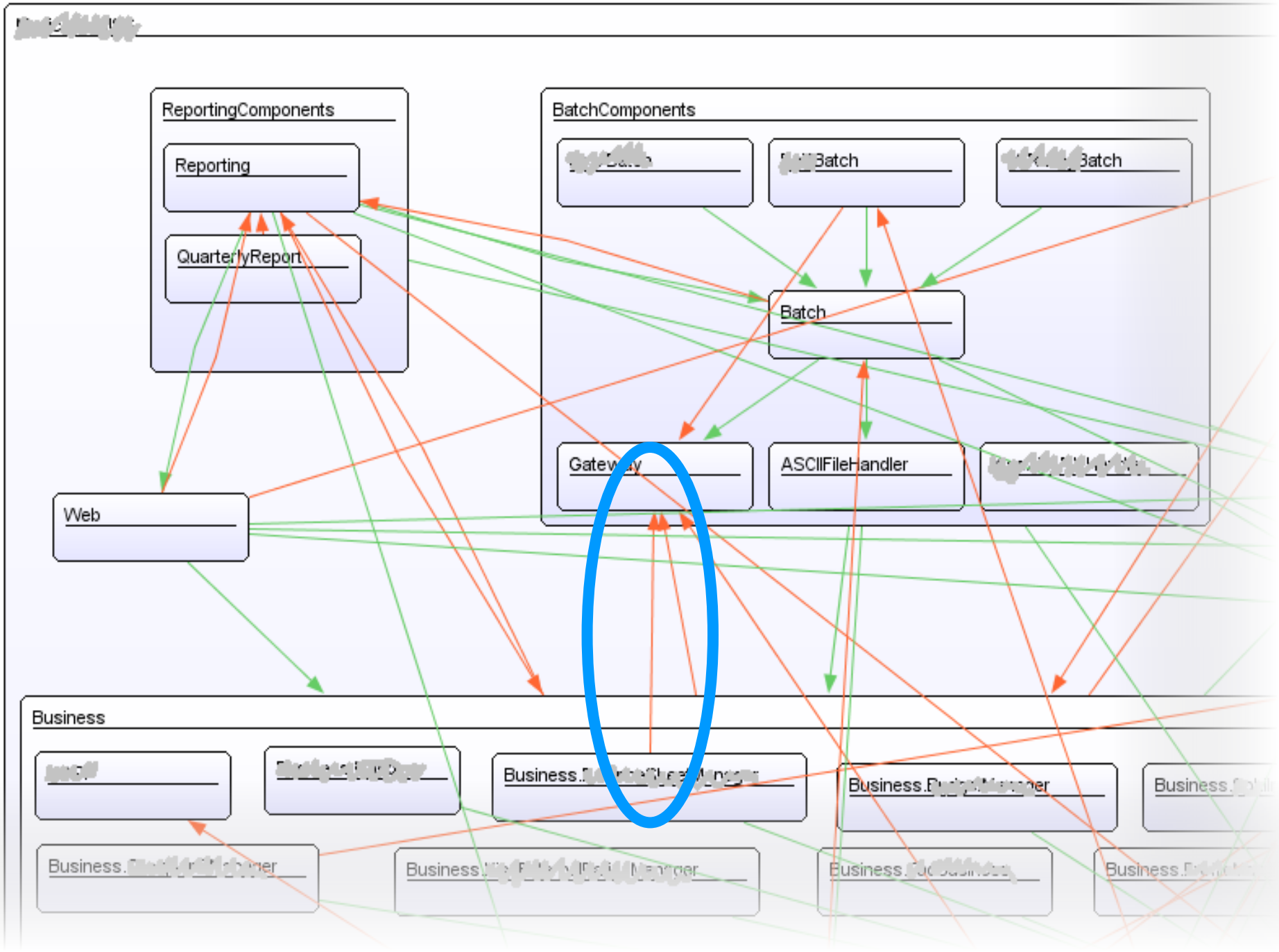
Message	Context
Found potential inconsistent clone change in RightClickMenu.java	[Broken clone] [Old clone finding] [Code change]

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

```
/// <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
      }
   }
   ...
```

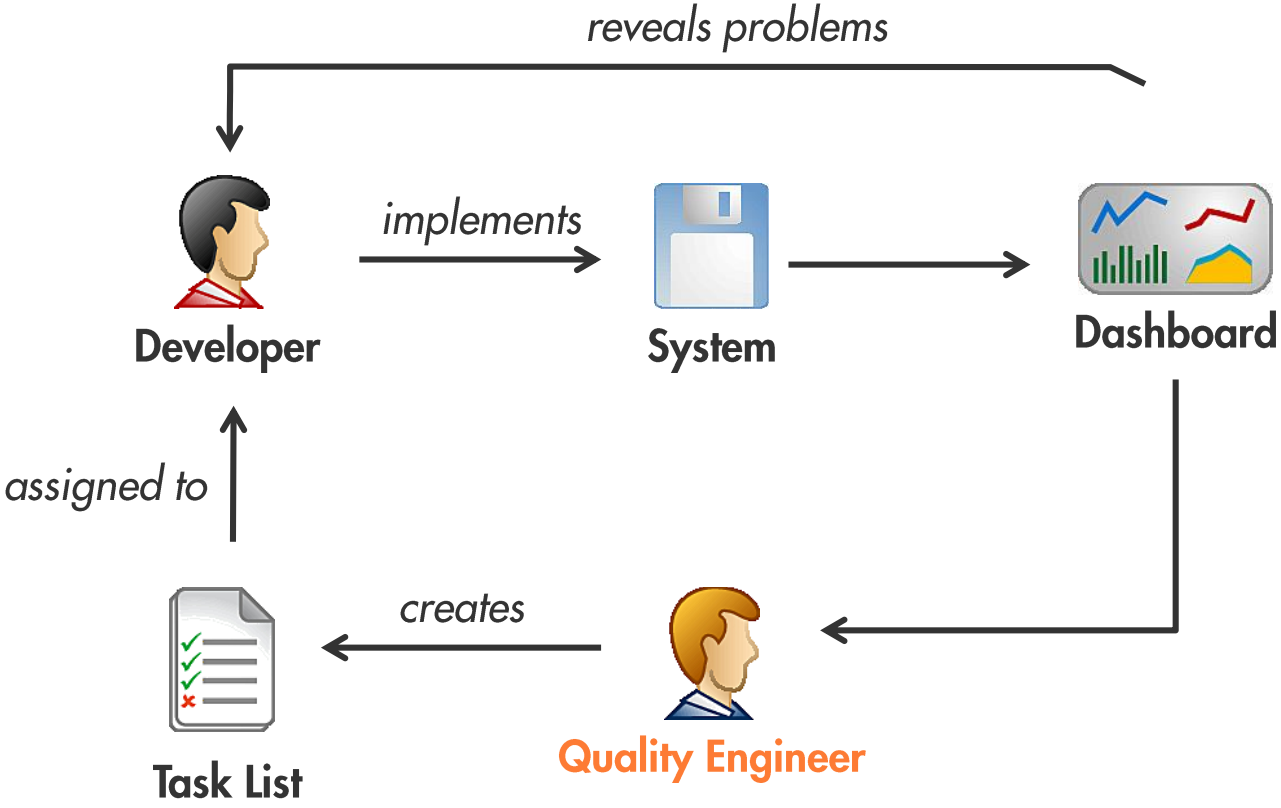

ReportingComponents
Reporting
QuarterlyReport
BatchComponents
Batch
Gateway
ASCIIFileHandler
Web
Business

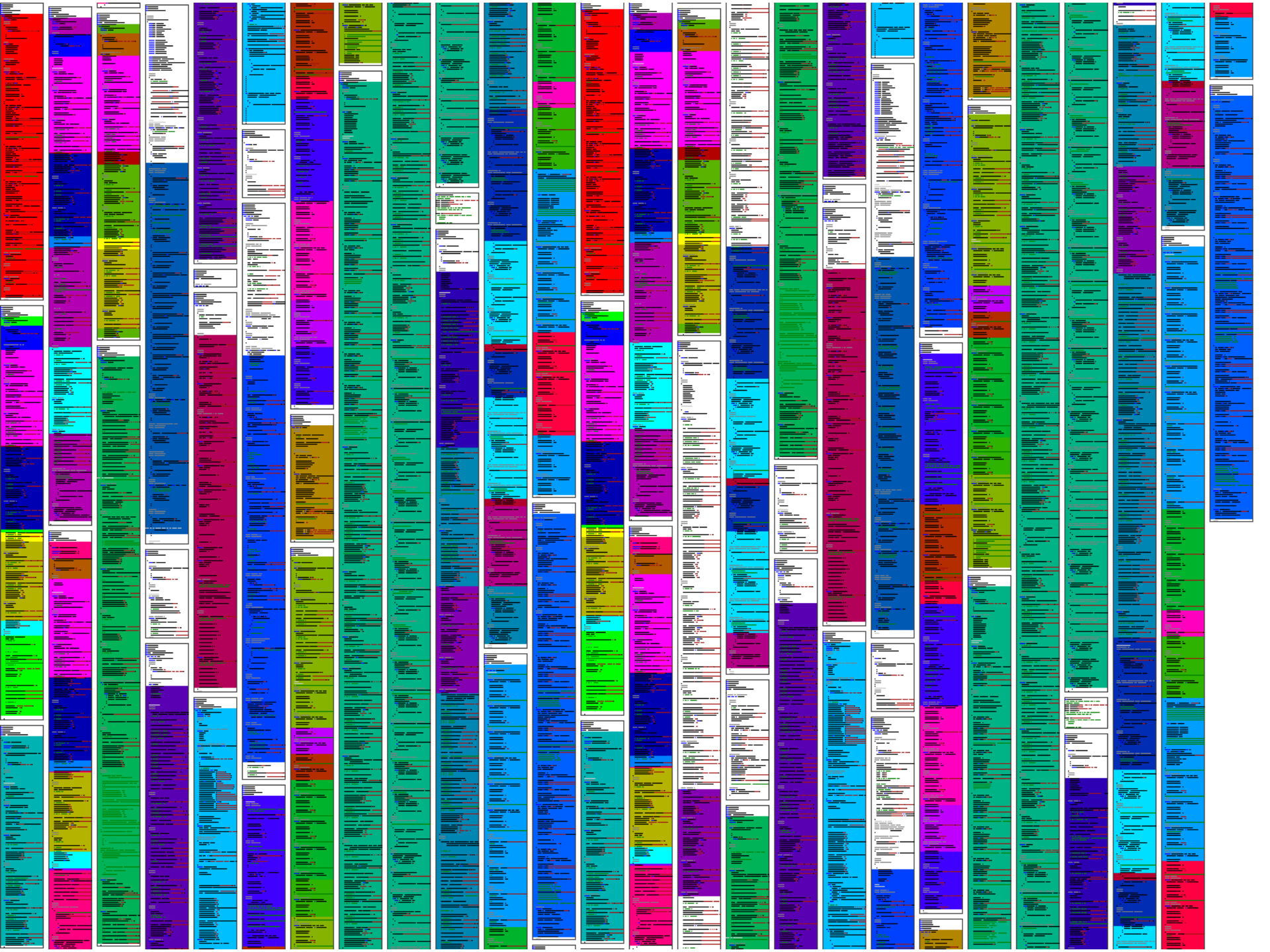
Teamscale Lizenz

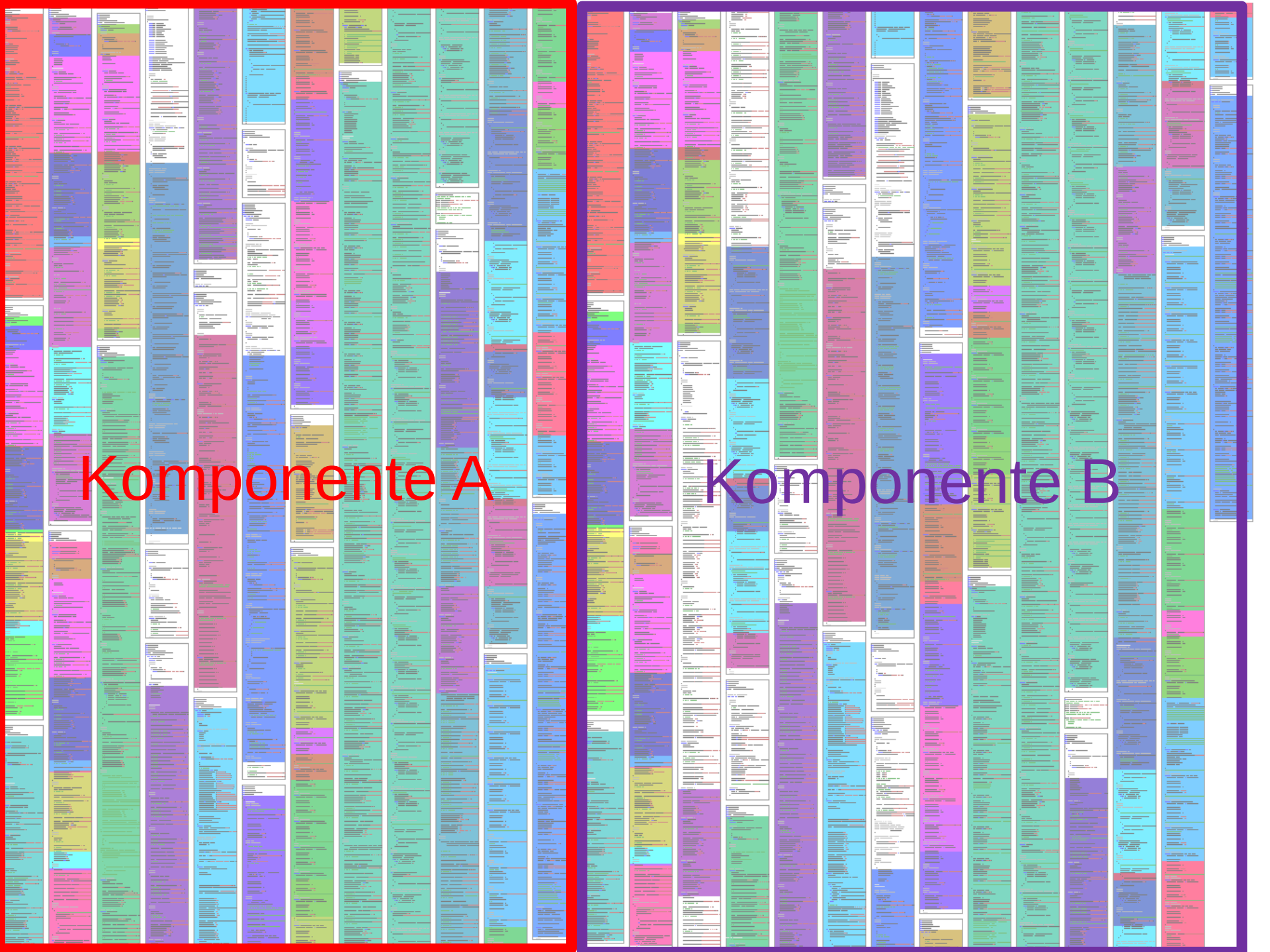
Mail an juergens@cqse.eu

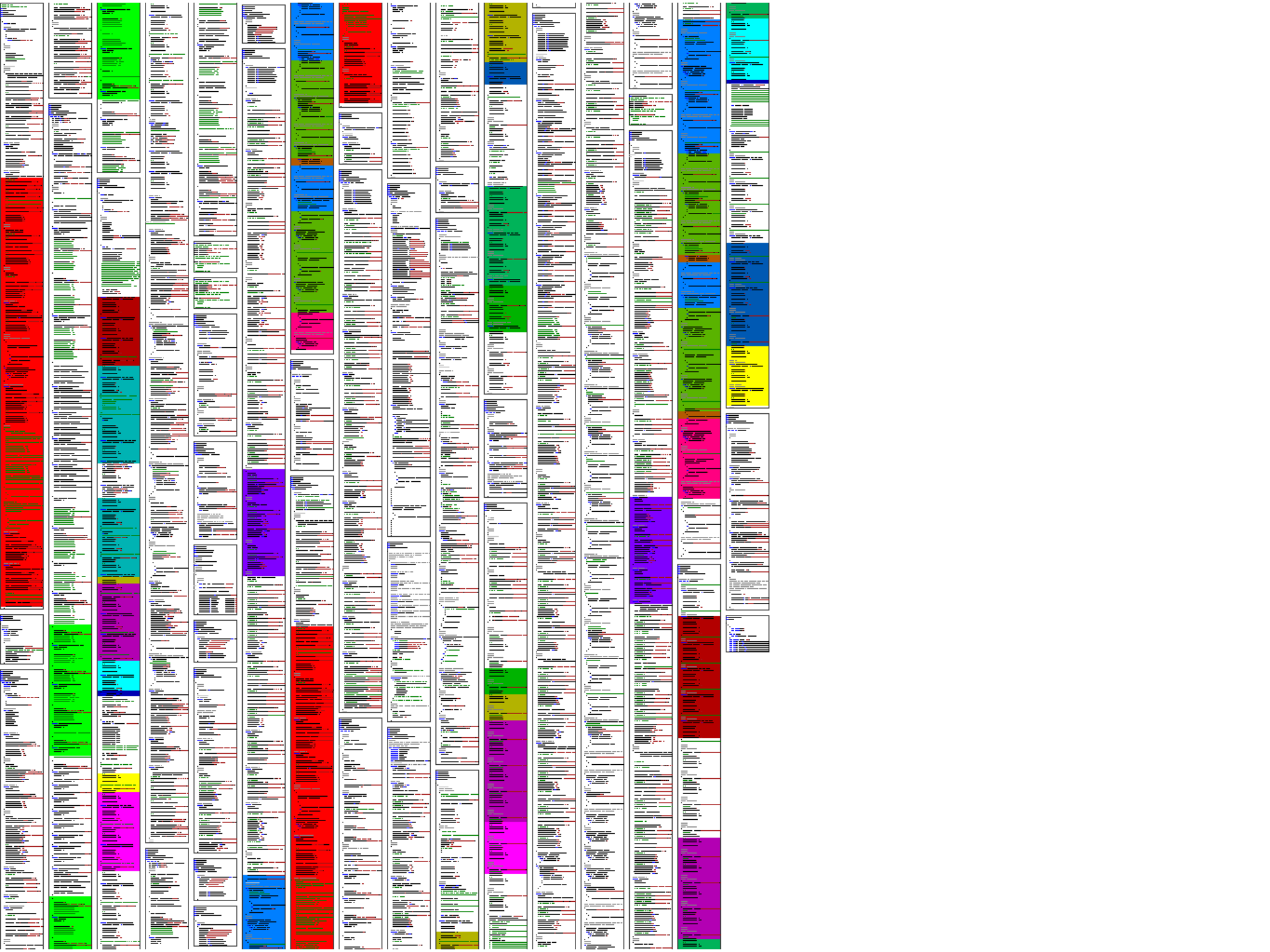
- Betreff: Teamscale Lizenz
- Bis Ende Juni

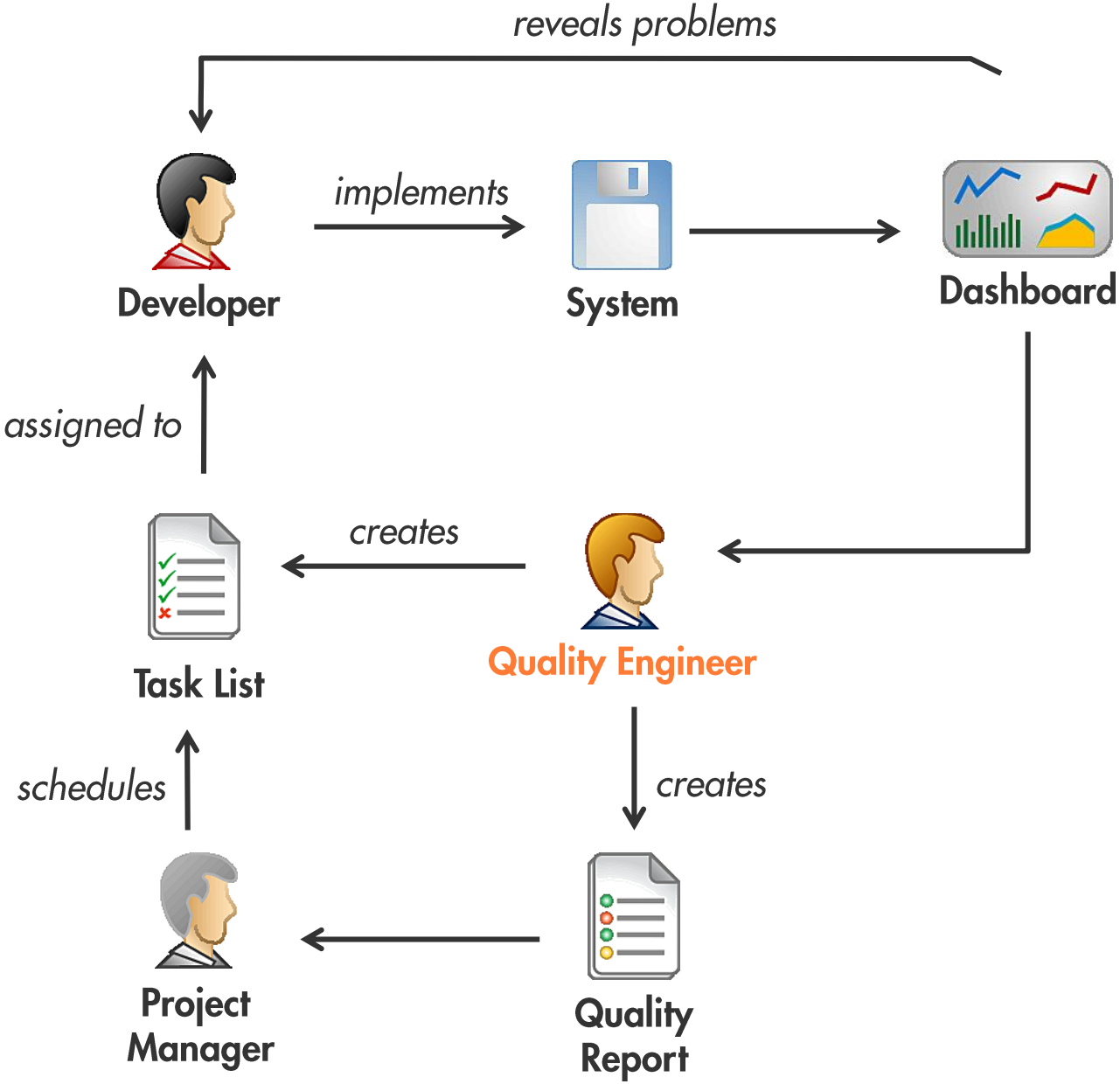
Ich schicke Euch eine komplett offene Lizenz für 2015


reveals problems
implements
Developer
System
Dashboard
assigned to
creates
Task List
Quality Engineer




Komponente A
Komponente B



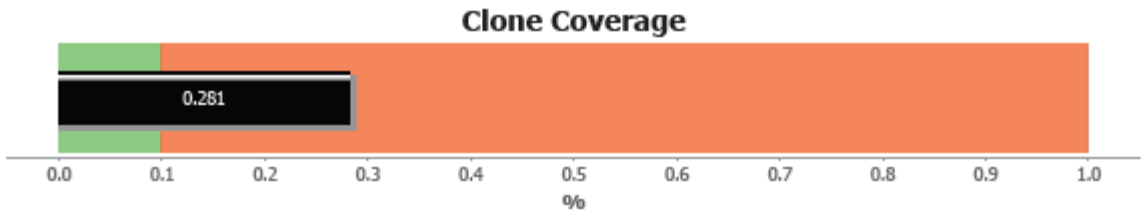

reveals problems
implements
Developer
System
Dashboard
assigned to
creates
Task List
Quality Engineer
schedules
creates
Project Manager
Quality Report

Best Practice: Quality Reports

Quality Indicator	Full Assessment		Delta Assessment	
Stability of Regular Team Builds	Build is stable.	✓	At the time of the last report, there was no automated build at all.	↗
Architecture Conformanc	There are zero violations.	✓	Architecture specification was completed and is now fully	↗
Stability of U Tests				
Code Cover				
Compiler Warnings				
Coding Guidelines Violations				
Duplicated Code	With a clone coverage of 12.2% the threshold of 10% is slightly violated.	✗	Clone coverage did not change significantly.	→
File Size	The threshold regarding files > 400 LOC is violated.	✗	Use of partial classes improves the metric values but does not improve code quality	→

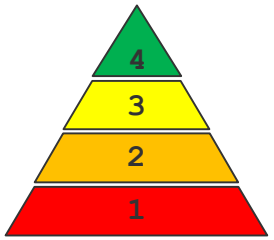
1.7 Duplicated Code ✗ ↗

TQE Target: Clone coverage of less than 10%.



Best Practice: TQE Tasks

- Bei jedem Report identifiziert
- Passend zu Quality Goal des Projekts
- Projekt entscheidet *ob* und *wann*



form routine.

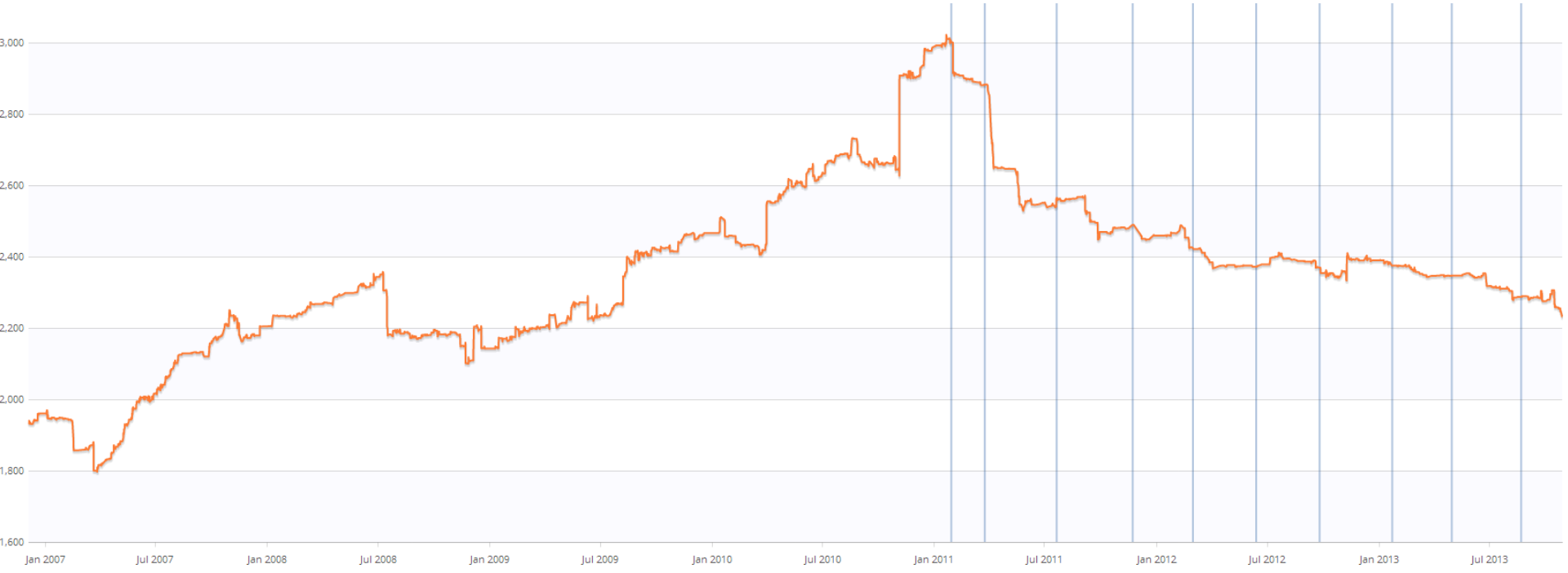
99902 *[Duplicated Code]* Remove duplicated code in function modules [redacted] (from l. 169), [redacted] (from l. 274) and [redacted] (from ll. 279, 577 and 1037) which recently were equally modified in 5 clone instances. Each duplicate is 91 lines long and equal unless two literals.

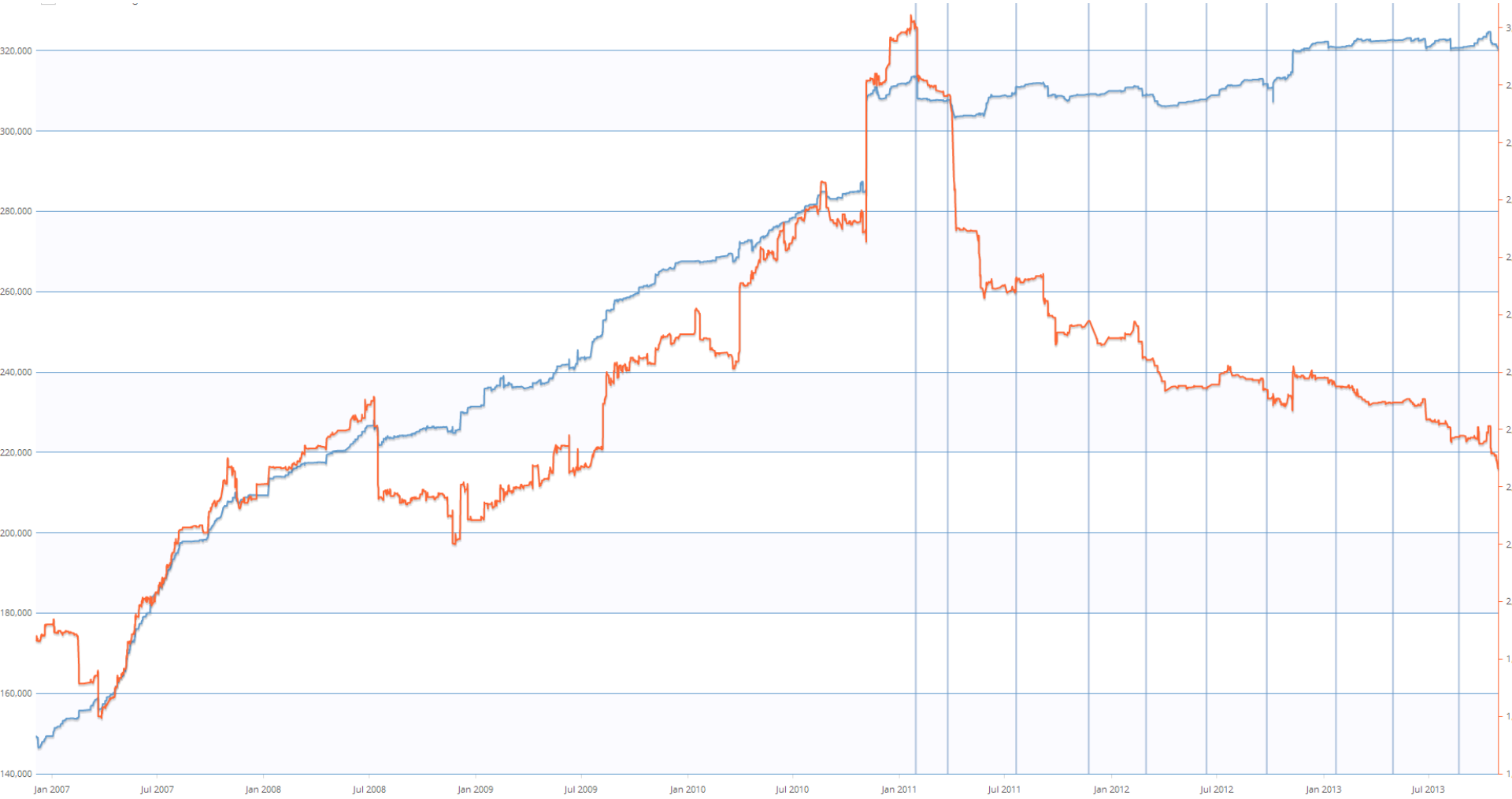
log[redacted] parts.

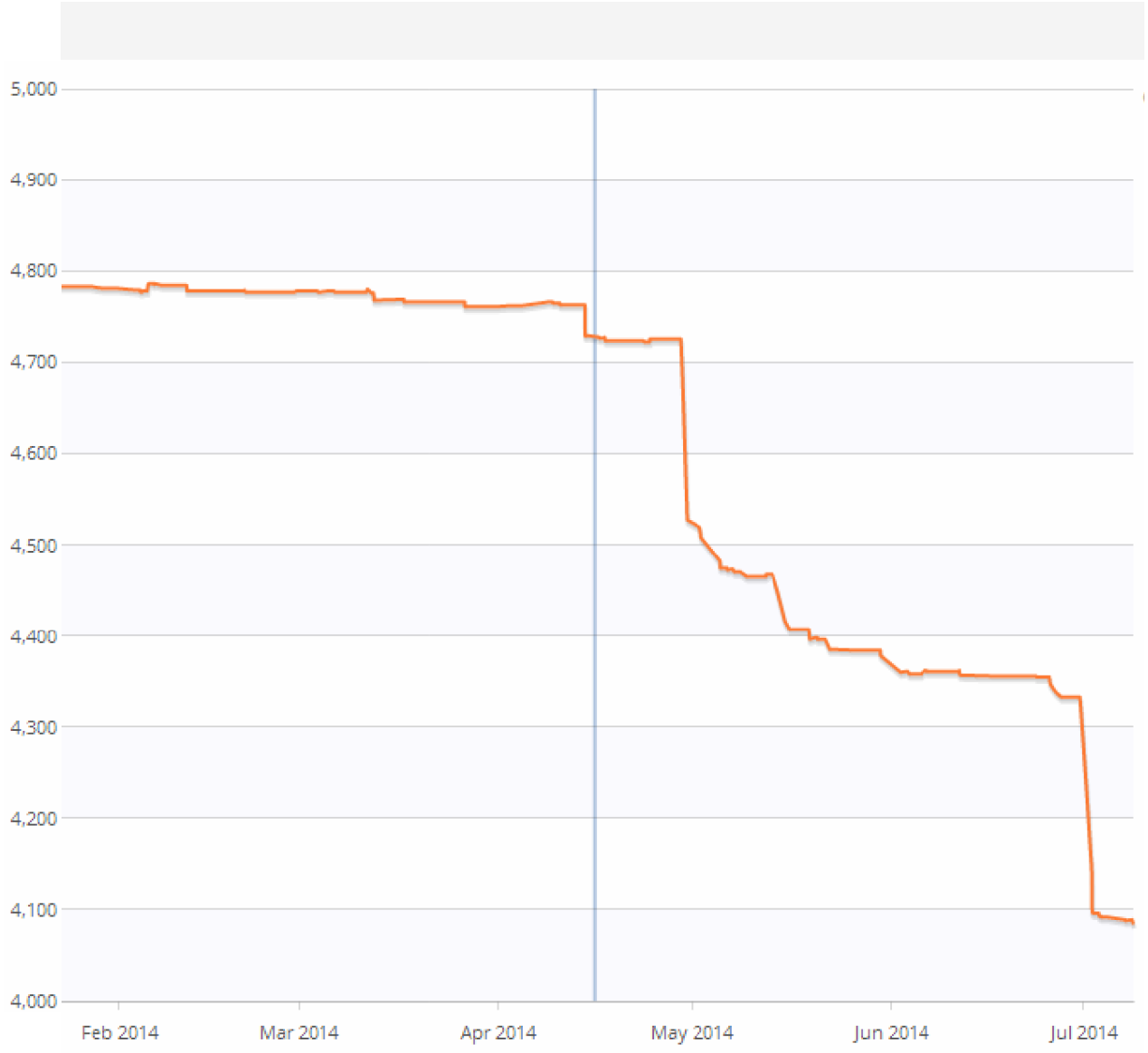
99914 *[Nesting Depth]* Restructure function module [redacted] to reduce the deep nesting which was added. E.g. by extracting code within loops to helper function modules.

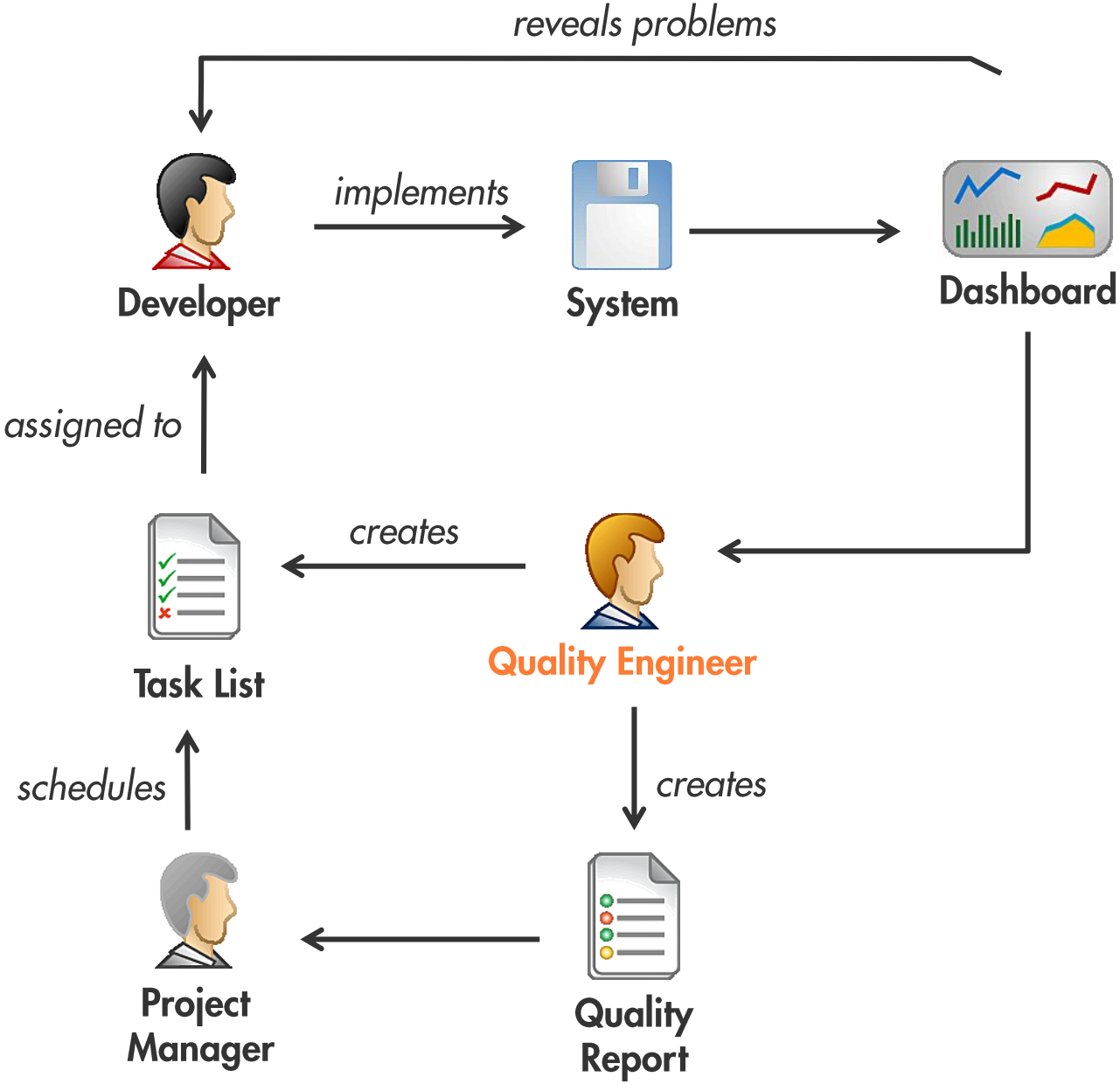
99915 *[Nesting Depth]* Restructure function module [redacted]


3,000
2,800
2,600
2,400
2,200
2,000
1,800
1,600
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011


3,000
2,800
2,600
2,400
2,200
2,000
1,800
1,600
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011
Jul 2011
Jan 2012
Jul 2012
Jan 2013
Jul 2013


320,000
300,000
280,000
260,000
240,000
220,000
200,000
180,000
160,000
140,000
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011
Jul 2011
Jan 2012
Jul 2012
Jan 2013
Jul 2013


5,000
4,900
4,800
4,700
4,600
4,500
4,400
4,300
4,200
4,100
4,000
Feb 2014
Mar 2014
Apr 2014
May 2014
Jun 2014
Jul 2014


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, …)
- …


Plakate
ankleben
verboten
Plakate
annageln
erlaubt?
Barbara.

Fazit

Qualitätsanalysewerkzeuge sind notwendig, aber nicht hinreichend für Verbesserung der Software-Qualität.

Nachhaltige Verbesserung erfordert die Unterstützung von Entwicklern und Management und die Integration in den Entwicklungsprozess.

Mindestens einer muss sich hierfür verantwortlich fühlen.

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
resent faults. In this case study we analyzed three co
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' developers

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
- Für ½ Jahr

Ich schicke Euch eine komplett offene Lizenz für ½ Jahr

Kontakt

Ich bin noch länger hier freue mich auf Diskussionen ☺

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