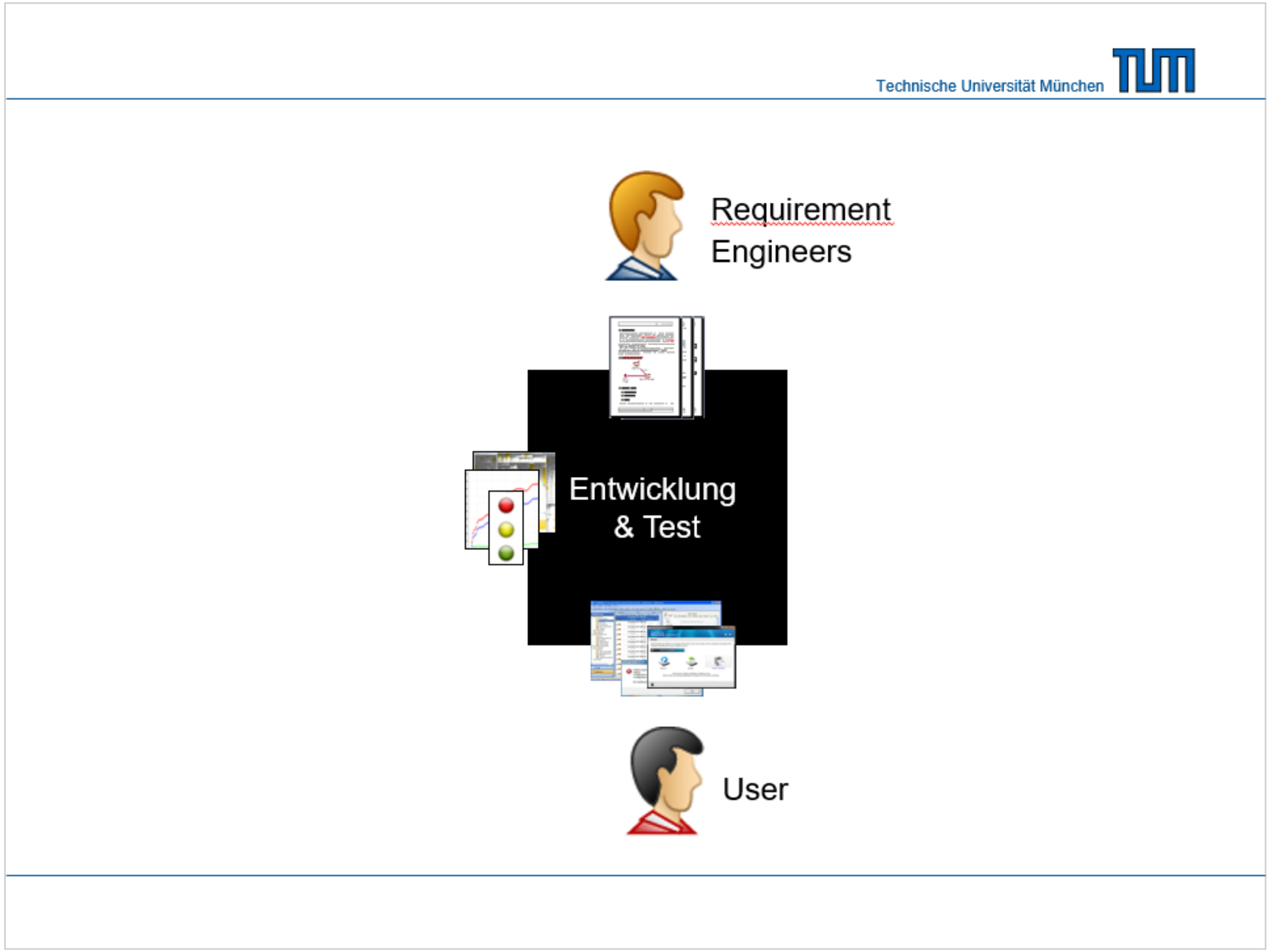
CQSE Software Intelligence Talk

Agenda

- Begrüßung
- Kosten-Nutzen-Berechnung von Qualitätsanalysen
  Erfahrungen bei der Munich Re
  Uwe Proft
  Münchener Rückversicherungs-Gesellschaft AG
  Dr. Elmar Juergens
  CQSE GmbH
- Q&A Session

Kosten-Nutzen-Berechnung von Qualitätsanalysen
Erfahrungen bei der Munich Re

Uwe Proft (Munich Re)
Elmar Jürgens (CQSE)



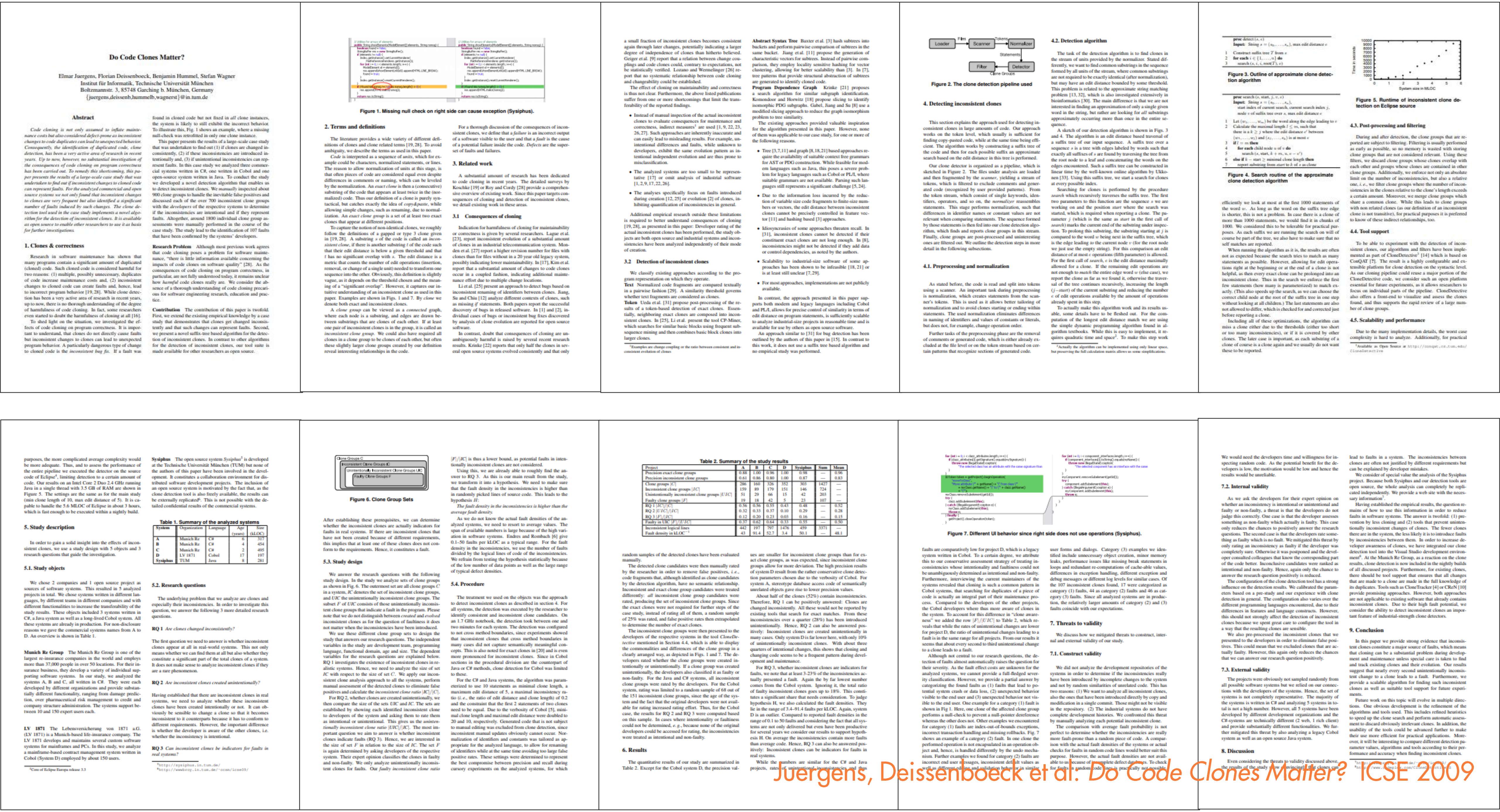
2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020

TreeAdministrationQuarterly.cs

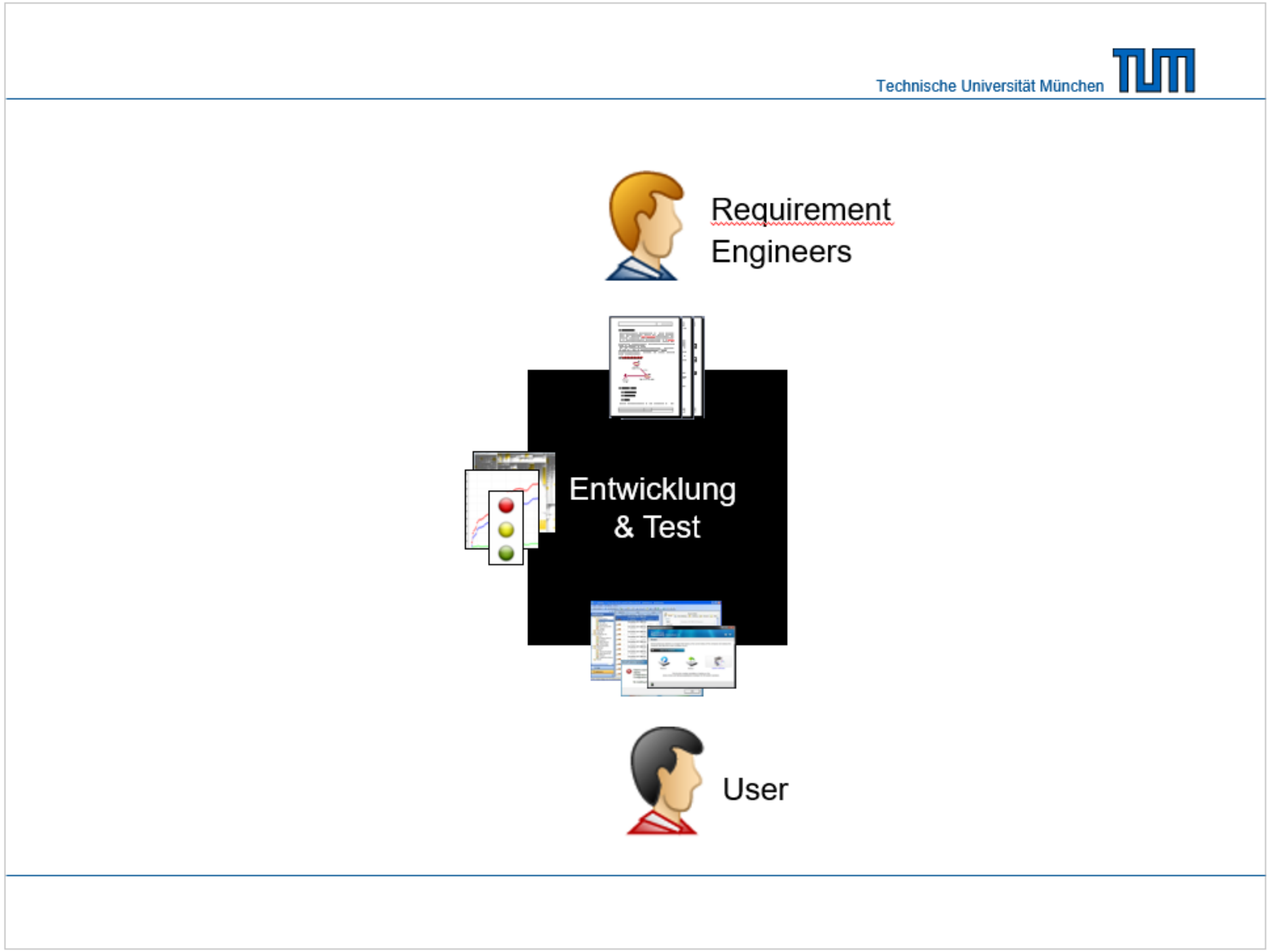
```
/// </summary>
protected override bool DoLazyLoading(
    UltraTreeNode node)
{
 // Some types of segments always have the
     expand icon
 if ( node.Tag is GaSegment ||
   node.Tag is MainSegment ||
   node.Tag is GeneralMainSegment ||
   node.Tag is Branch )
  return true;
 // All others only if there are child nodes
 return false;
}

/// <summary>
/// Structure segments return only loss or
    premium segments according to
/// the mode.
/// </summary>
protected override IList GetChildSegments(
    ISegment parent, ref bool alreadySorted )
{
 if ( parent is StructureSegment )
 {
  // Show either preminum or loss beyond
      structure segment
  StructureSegment strucSeg = (StructureSegment
      )parent;
  if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
      TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
      )
   return strucSeg.GetLossProcessingSegments();
  else
   return strucSeg.GetPremiumProcessingSegments
       ();
```

TreeAdministrationYearly.cs

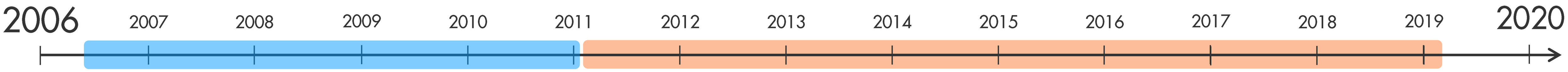
```
/// </summary>
protected override bool DoLazyLoading(
    UltraTreeNode node)
{
 // Some types of segments always have the
     expand icon
 if ( node.Tag is GaSegment ||
   node.Tag is MainSegment ||
   node.Tag is GeneralMainSegment ||
   node.Tag is Branch )
  return true;
 // All others only if there are child nodes
 return false;
}

/// <summary>
/// Structure segments return only loss or
    premium segments according to
/// the mode.
/// </summary>
protected override IList GetChildSegments(
    ISegment parent, ref bool alreadySorted )
{
 if ( parent is StructureSegment )
 {
  // Show either preminum or loss beyond
      structure segment
  StructureSegment strucSeg = (StructureSegment
      )parent;
  if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
      TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
      )
   return strucSeg.GetLossProcessingSegments();
  else
   return strucSeg.GetPremiumProcessingSegments
       ();
```



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



Technische Universität München
Requirement
Engineers
Entwicklung
& Test
User



2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020

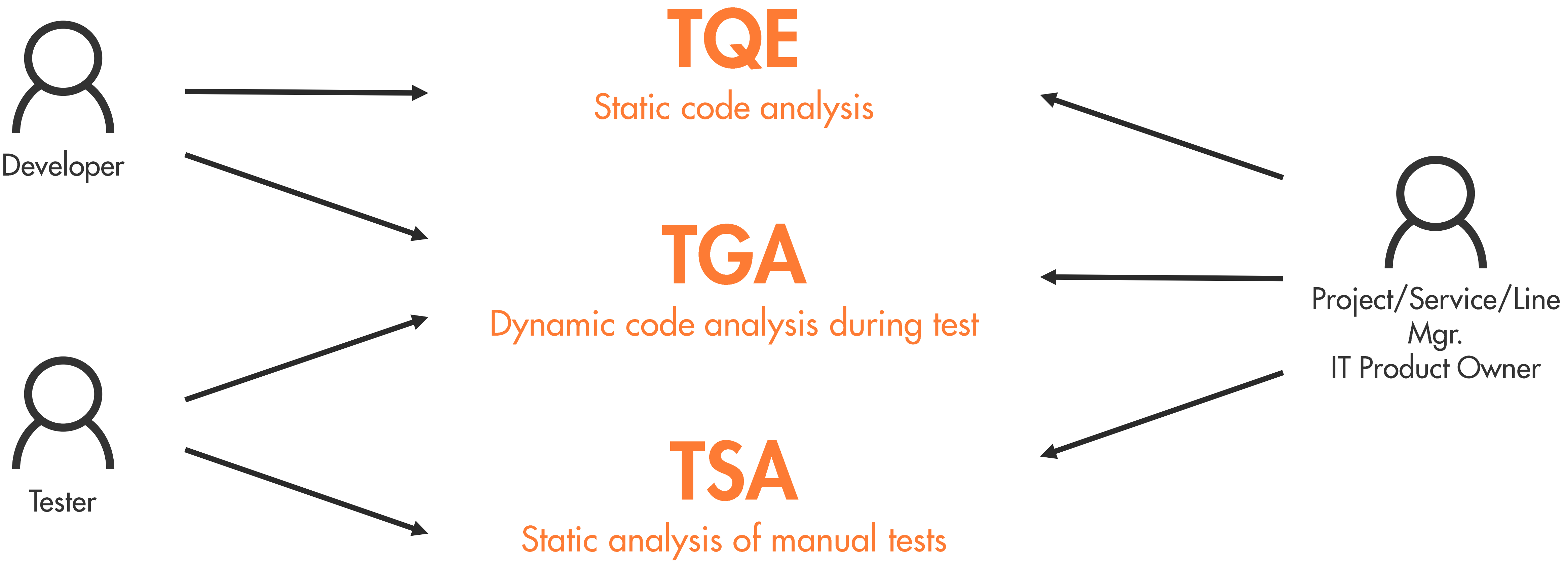
Wie sehen die Werkzeuge & Prozesse bei der Munich Re aus?

Uwe Proft

- Background im Software Engineering und Provider Management

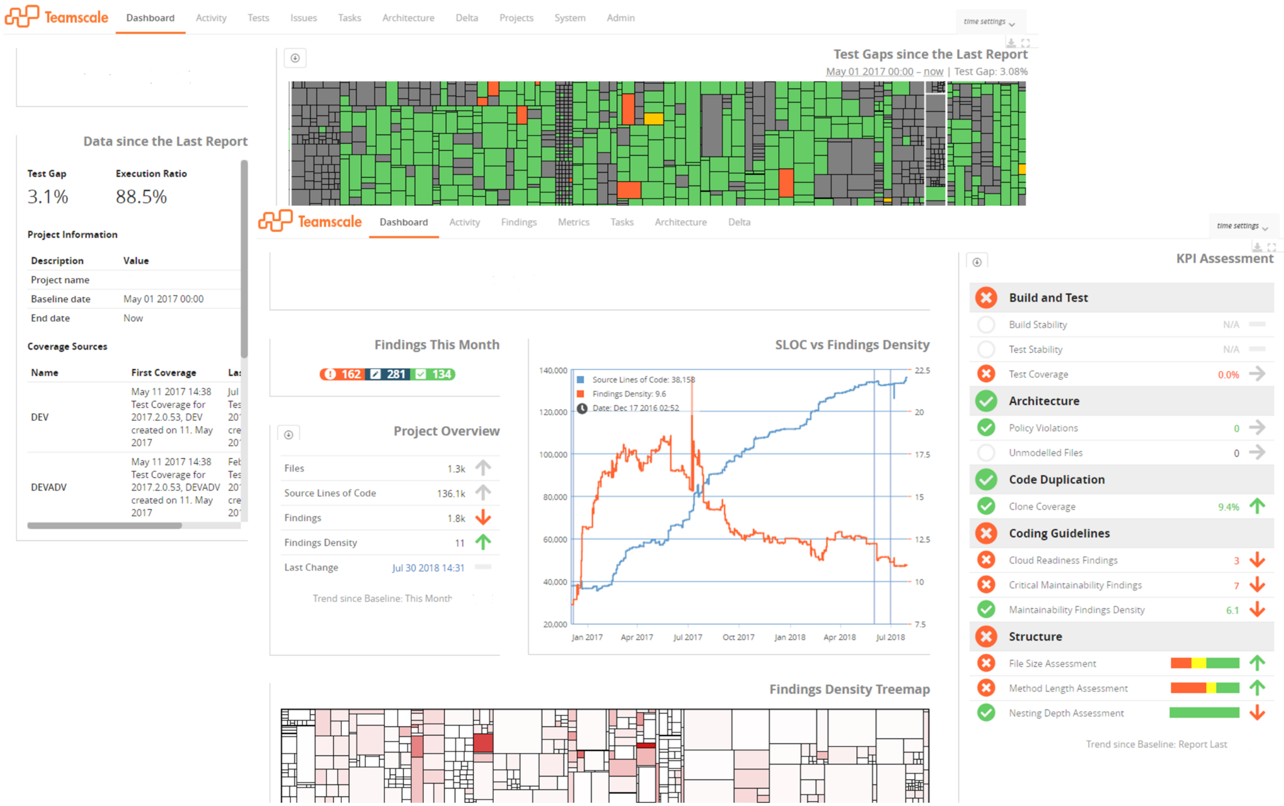
- Seit 7 Jahren bei der MR in Rollen zum Qualitätsmanagement
  - Erläutern Nutzen und Aufwand intern
  - Ausrollen, auch international an unterschiedlichen Standorten
  - Change-Management
  - Vermittlung der Messergebnisse für Beurteilung der Qualität von Zulieferern und Projekten
  - Steuerung des Teams der Quality Engineers (CQSE) bei der Munich Re

Quality Tools

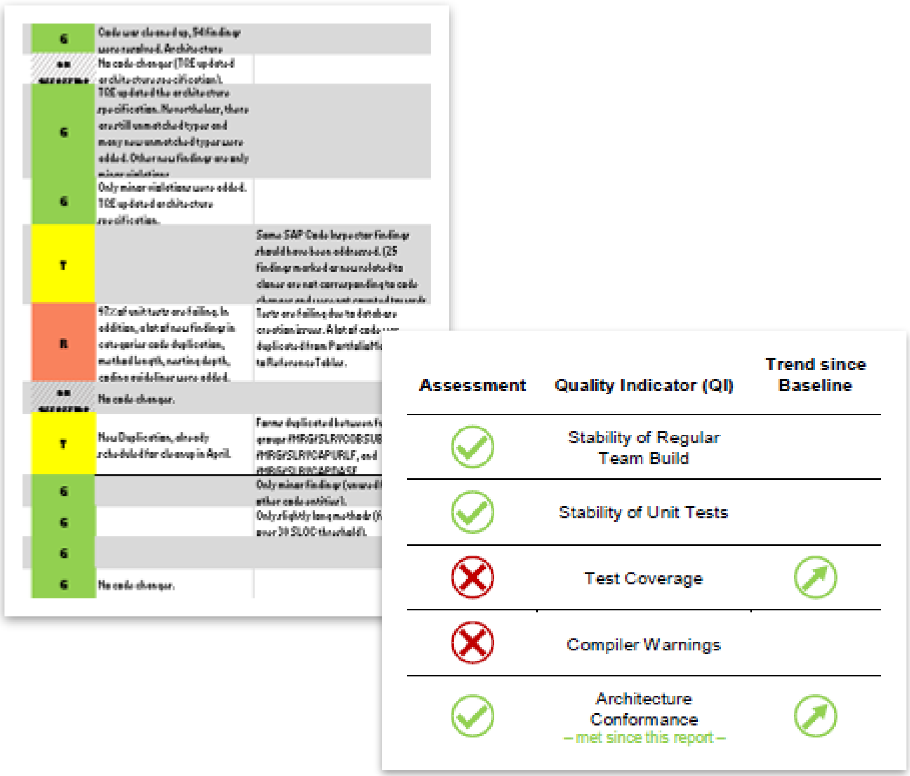


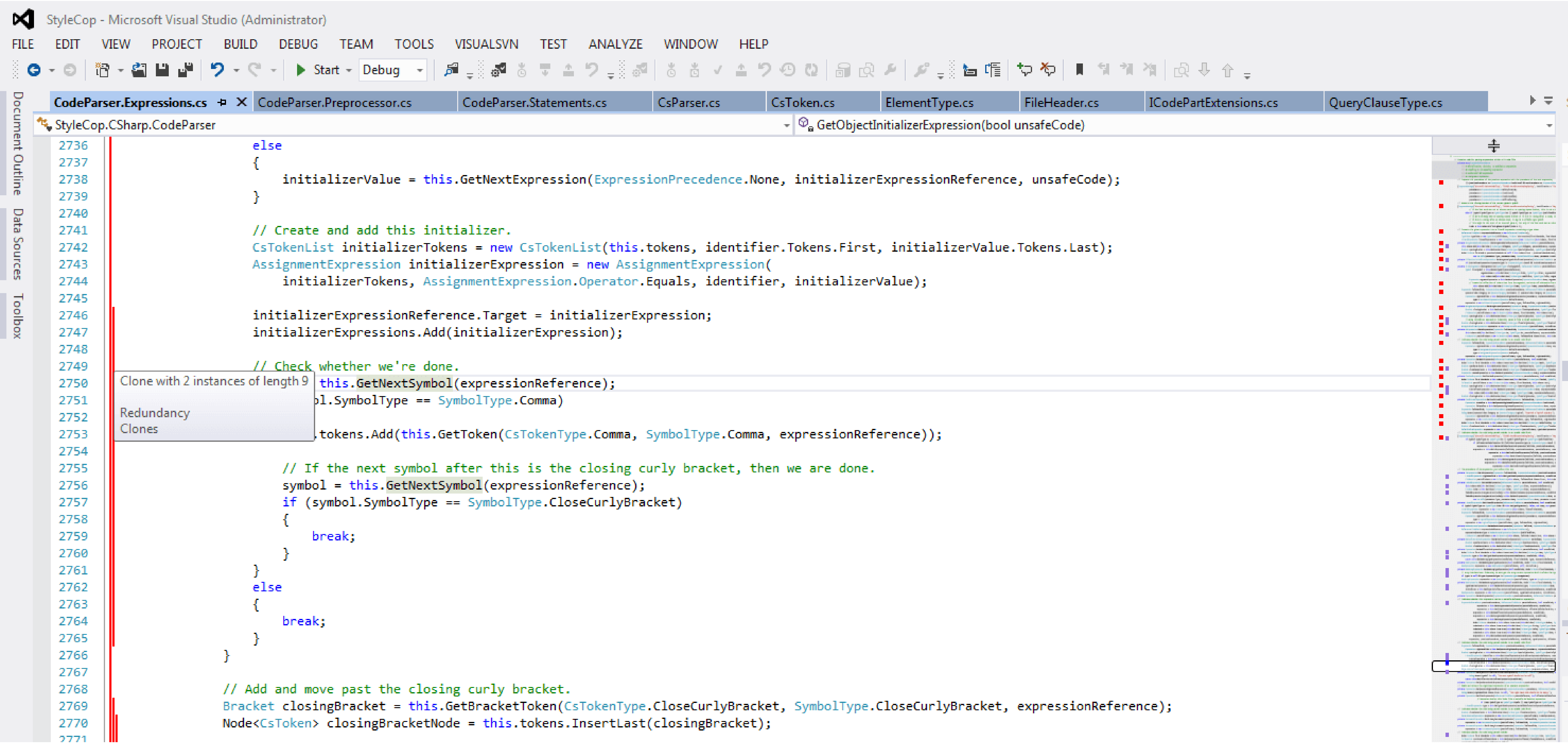
Munich Re Internal Services

Dashboards, IDE Plugin, Azure DevOps

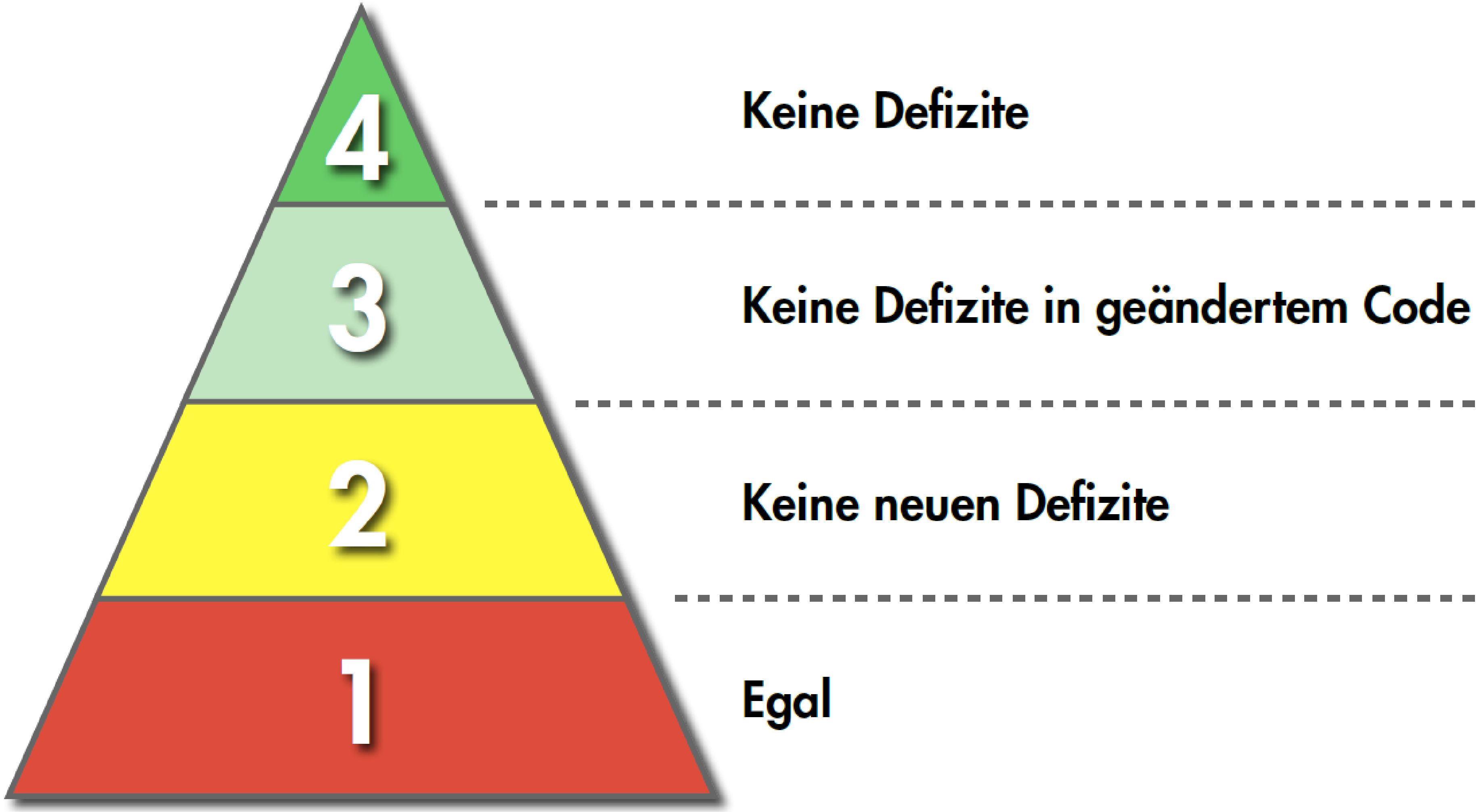


Monthly Assessments, Reports

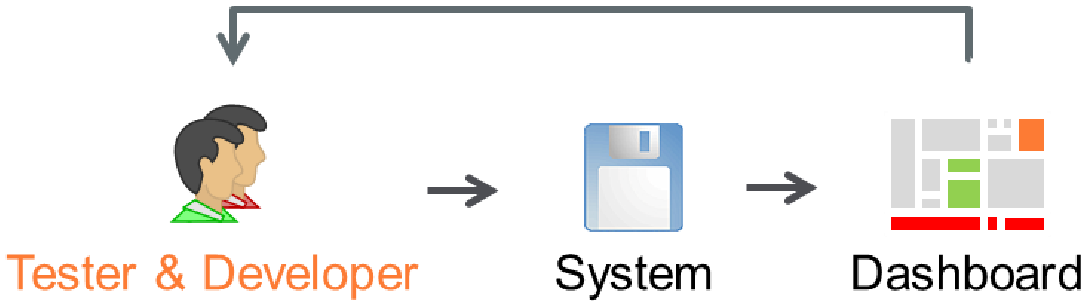



StyleCop - Microsoft Visual Studio (Administrator)
FILE EDIT VIEW PROJECT BUILD DEBUG TEAM TOOLS VISUALSVN TEST ANALYZE WINDOW HELP
Start
Debug
CodeParser.Expressions.cs
CodeParser.Preprocessor.cs
CodeParser.Statements.cs
CsParser.cs
CsToken.cs
ElementType.cs
FileHeader.cs
ICodePartExtensions.cs
QueryClauseType.cs
StyleCop.CSharp.CodeParser
GetObjectInitializerExpression(bool unsafeCode)
Document Outline
Data Sources
Toolbox
else
{
initializerValue = this.GetNextExpression(ExpressionPrecedence.None, initializerExpressionReference, unsafeCode);
}
// Create and add this initializer.
CsTokenList initializerTokens = new CsTokenList(this.tokens, identifier.Tokens.First, initializerValue.Tokens.Last);
AssignmentExpression initializerExpression = new AssignmentExpression(
initializerTokens, AssignmentExpression.Operator.Equals, identifier, initializerValue);
initializerExpressionReference.Target = initializerExpression;
initializerExpressions.Add(initializerExpression);
// Check whether we're done.
this.GetNextSymbol(expressionReference);
ol.SymbolType == SymbolType.Comma)
.tokens.Add(this.GetToken(CsTokenType.Comma, SymbolType.Comma, expressionReference));
// If the next symbol after this is the closing curly bracket, then we are done.
symbol = this.GetNextSymbol(expressionReference);
if (symbol.SymbolType == SymbolType.CloseCurlyBracket)
{
break;
}
}
else
{
break;
}
}
// Add and move past the closing curly bracket.
Bracket closingBracket = this.GetBracketToken(CsTokenType.CloseCurlyBracket, SymbolType.CloseCurlyBracket, expressionReference);
Node<CsToken> closingBracketNode = this.tokens.InsertLast(closingBracket);
Clone with 2 instances of length 9
Redundancy
Clones

Qualitätsziele



Prozess



Portfolio Overview – Links to Dashboards & Monthly Assessments

Application	Dev	Test	TQE	TGA	TSA
[illegible]	[illegible] IT1.5 [illegible]	[illegible]	2019-12		
[illegible]	[illegible] IT1.2 [illegible]	[illegible]			2019-12
[illegible]	[illegible] IT1.5 [illegible]	[illegible]	2019-12	2019-12	2019-12
[illegible]	[illegible] IT1.5 [illegible]	[illegible]			
[illegible]	[illegible] IT1.8 [illegible]	[illegible]	2019-12		2019-12

Portfolio Overview – Trends

TQE assessment trend for

Assessment	Comment	QG relevant findings	Details
2019-09	Only one small finding in changed code	1	Show Details
2019-08	No code changes.	0	
2019-07	Only minor new findings	6	Show Details
2019-06	Only a small change with no findings churn.	0	
2019-05	Only 2 small findings in modified code.	2	
2019-04	Mostly minor violations.	70	Show Details

Notable findings:

- Naming convention violations in `TmOneParam`
- Method threshold violation in method `DeleteProcessYear` of class `ProcessYearsController`
- Method threshold violation in a lambda in class `LossChartService`
- Cloning between `ReverseTriangleGrid` and `CommissionTriangleGrid` (c.f. here)

Assessment	Comment	QG relevant findings	Details
2019-03	Minor violations only.	3	
2019-02	Only 5 new findings. Remaining findings are located in code that was changed during a migration to Angular 7 and thus can be ignored for this assessment.	39	Show Details

Portfolio Overview – Trends

Application	Dev	TSA

TQE assessment trend for

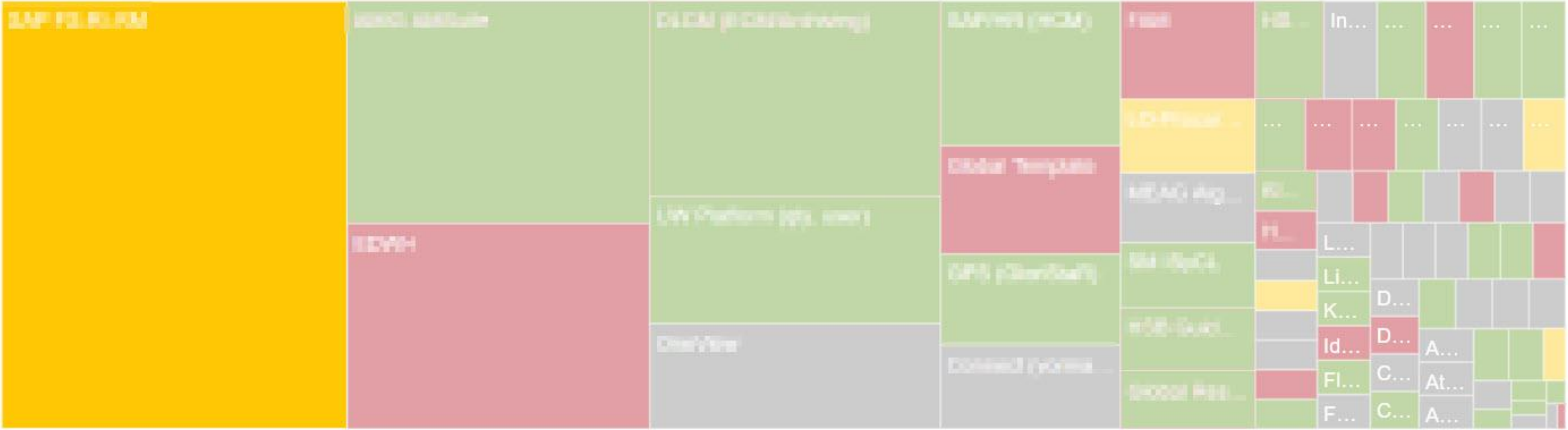
Assessment	Comment	QG relevant findings	Details
2019-09	Only one small finding in changed code	1	Show Details
2019-08	No code changes.	0	
2019-07	Only minor new findings	6	Show Details
2019-06	Only a small change with no findings churn.	0	
2019-05	Only 2 small findings in modified code.	2	
2019-04	Mostly minor violations.	70	Show Details
	Notable findings: • Naming convention violations in `TmOneParam` • Method threshold violation in method `DeleteProcessYear` of class `ProcessYearsController` • Method threshold violation in a lambda in class `LossChartService` • Cloning between `ReverseTriangleGrid` and `CommissionTriangleGrid` (c.f. here)		
2019-03	Minor violations only.	3	
2019-02	Only 5 new findings. Remaining findings are located in code that was changed during a migration to Angular 7 and thus can be ignored for this assessment.	39	Show Details

TQE assessment trend for

Assessment	Comment	QG relevant findings	Details
2019-09	Only few findings compared to the amount of change	93	Show Details
2019-08	Mostly minor violations given large amount of changes.	87	Show Details
2019-07	Mostly minor violations given large amount of changes.	131	Show Details
2019-06	Mostly minor violations.	117	
2019-05	Mostly minor violations.		Show Details
2019-04	Tolerable number of findings given amount of code changes	194	Show Details
2019-03	Tolerable but not significant amount of findings.	198	Show Details
2019-02	Tolerable but no significant amount of findings.	160	Show Details
2019-01	Tolerable but significant amount of findings.	173	Show Details
2018-12	Minor violations only.	77	Show Details
2018-11	Some findings that could have been avoided and resolved.	123	Show Details
2018-10	Minor violations only. Amount of findings tolerable with respect to code churn.	121	Show Details
2018-09	Minor violations only. Amount of findings tolerable with respect to code churn.	64	
2018-08	Minor violations only.	181	Show Details
2018-07	Duplicated code in lean and spc classes increased clone coverage significantly.	101	Show Details
2018-06	Minor violations only.	77	
2018-05	Tolerable amount of findings wrt to code change.	180	
2018-04	Tolerable amount of findings wrt to code change	99	Show Details
2018-03	Findings correspond to minor structural violations only.	97	
2018-02	New findings correspond to minor structural violations.	77	
2018-01	Goal reached. Amount of findings tolerable with respect to code churn.	122	Show Details
2017-12	Goal reached. Amount of findings tolerable with respect to code churn.	101	
2017-11	Goal reached. Amount of findings tolerable with respect to code churn.	37	
2017-10	High code churn with positive trend in most quality indicators. Architecture specification not up-to-date anymore.	80	Show Details
2017-09	Code duplications, long methods, deeply nested code and coding guideline violations	92	Show Details
2017-08	Newly duplicated code, new long methods, new coding guideline violations. However tolerable, increase of system size by 10.000 code lines.	148	Show Details
2017-07	new long methods, new deeply nested code,. OK wrt to file changes Build/ Test: builds and tests are running.	8	
2017-06	New findings tolerable, new compiler warnings may also be from last month	46	
2017-05	SLOC +5000. Several clones between Kbs and normal submission header broke up (intended?). Also some new clones. Other findings tolerable.	99	
2017-04	New cloned component (Treaty.Kbs.DataAccess)	142	
2017-03	several new clones affecting not related business entities, findings tolerable given growth of +10k SLOC	106	
2017-02	Given large amount of new code (+10k LOC), new findings ok; also resolved several old ones	40	
2017-01	excludeds review related findings	53	Show Details
2016-12	Peer review findings have been subtracted	26	Show Details
2016-11			
2016-10	Given amount of development acceptable amount of new findings.	71	
2016-09	Most findings are either architecture related or target reviewi findings or test code. The remaining ones or minor clones or minor method length violations	297	
2016-08	Most findings concern the unfinished architecture spec. However several findings in new or modified code.	386	

Close

Monthly Assessment Results Portfolio Aggregation



Name	Tool	Assessment	Comments
[illegible]	TQE	GREEN	Only few violations
[illegible]	TGA	YELLOW	Some relevant test gaps
[illegible]	TSA	GREEN	The team added 47 new test case, changed 11 test case, moved 5 test case and removed 15 test cases, which introduced 0 new findings.

Was bringt's?

```
      tierQuery.or(\orCriteria -> {
        orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, polLocation.County)
        orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, null)
      })
      tierQuery.or(\orCriteria -> {
        orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, polLocation.City)
        orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, null)
      })
      tierQuery.or(\orCriteria -> {
        orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, polLocation.PostalCode)
        orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, null)
      })

      var results = tierQuery.select()
      if(results.HasElements) {
        var result = results.firstWhere( \ elt -> elt.ZipCode == polLocation.PostalCode and
            elt.County == polLocation.County and elt.City == polLocation.City)
        var result2 = results.firstWhere( \ elt -> elt.County == polLocation.County and elt.City == p
        var result3 = results.firstWhere( \ elt -> elt.County == polLocation.County)

        if(result != null) {
```

```
        orCriteria.compareIgnoreCase(TerrorismTargetTier_Ext#County, Equals, polLocation.County)
        orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, null)
      })
      tierQuery.or(\orCriteria -> {
        orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, polLocation.City)
        orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, null)
      })
      tierQuery.or(\orCriteria -> {
        orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, polLocation.PostalCode)
        orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, null)
      })

      var results = tierQuery.select()
      if(results.HasElements) {
        var result = results.firstWhere( \ elt -> elt.ZipCode == polLocation.PostalCode and
           elt.County == polLocation.County and elt.City == polLocation.City)
        var result2 = results.firstWhere( \ elt -> elt.County == polLocation.County and elt.City == p
        //AKN - Defect 224598 - Fix for defaulting target tier value irrespective of lower and upper
        var result3 = results.firstWhere( \ elt -> elt.County.equalsIgnoreCase(polLocation.County))

        if(result != null) {
```

Project	QG	Assessment	Comment	Detailed Comment	QG-relevant Findings
[redacted]	2	Y	Four new security relevant findings should have been reviewed.	Missing authorization checks (1) at the beginning of report [redacted]_GOODSRECEIPT. (2) before CALL TRANSACTION ycl_p2p_invoice=>gc_ta_f90 in method create_post_values in class [redacted]FIXEDASSET (3) before CALL TRANSACTION gc_ta_fbd1 in method start_recurring_entry in class [redacted]INVOICE [redacted]	4

Di 29.10.2019 18:01

[redacted] Munich-MR

AW: Security Code Scan - Monthly Assessment [redacted]

To IT TQE-TGA (Pool) - Munich-MR; [redacted]

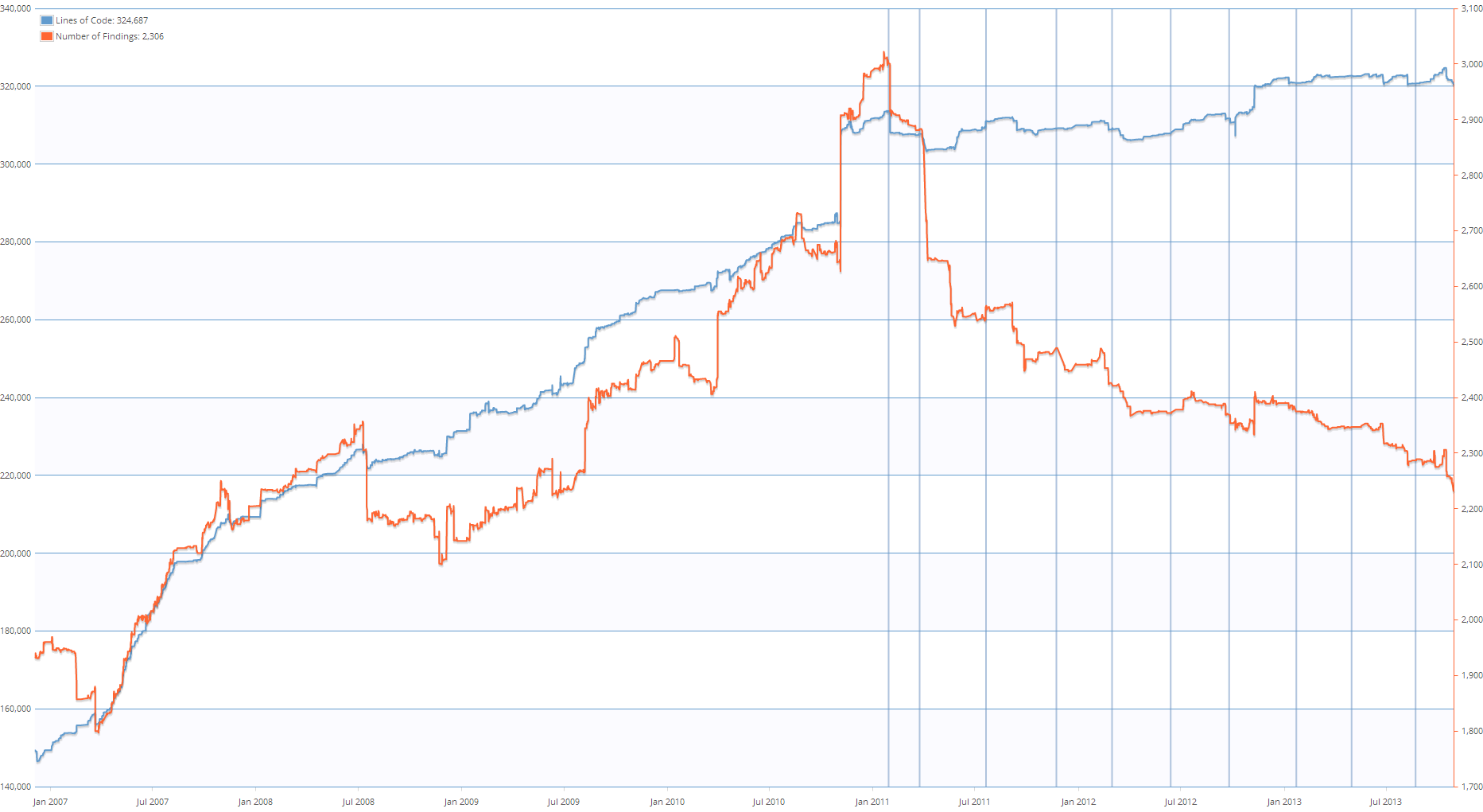
Cc Proft Uwe - Munich-MR

You replied to this message on 30.10.2019 15:06.

Hi all,

the findings down below has been fixed, next assessment status should be green again.

Best regards
[redacted]



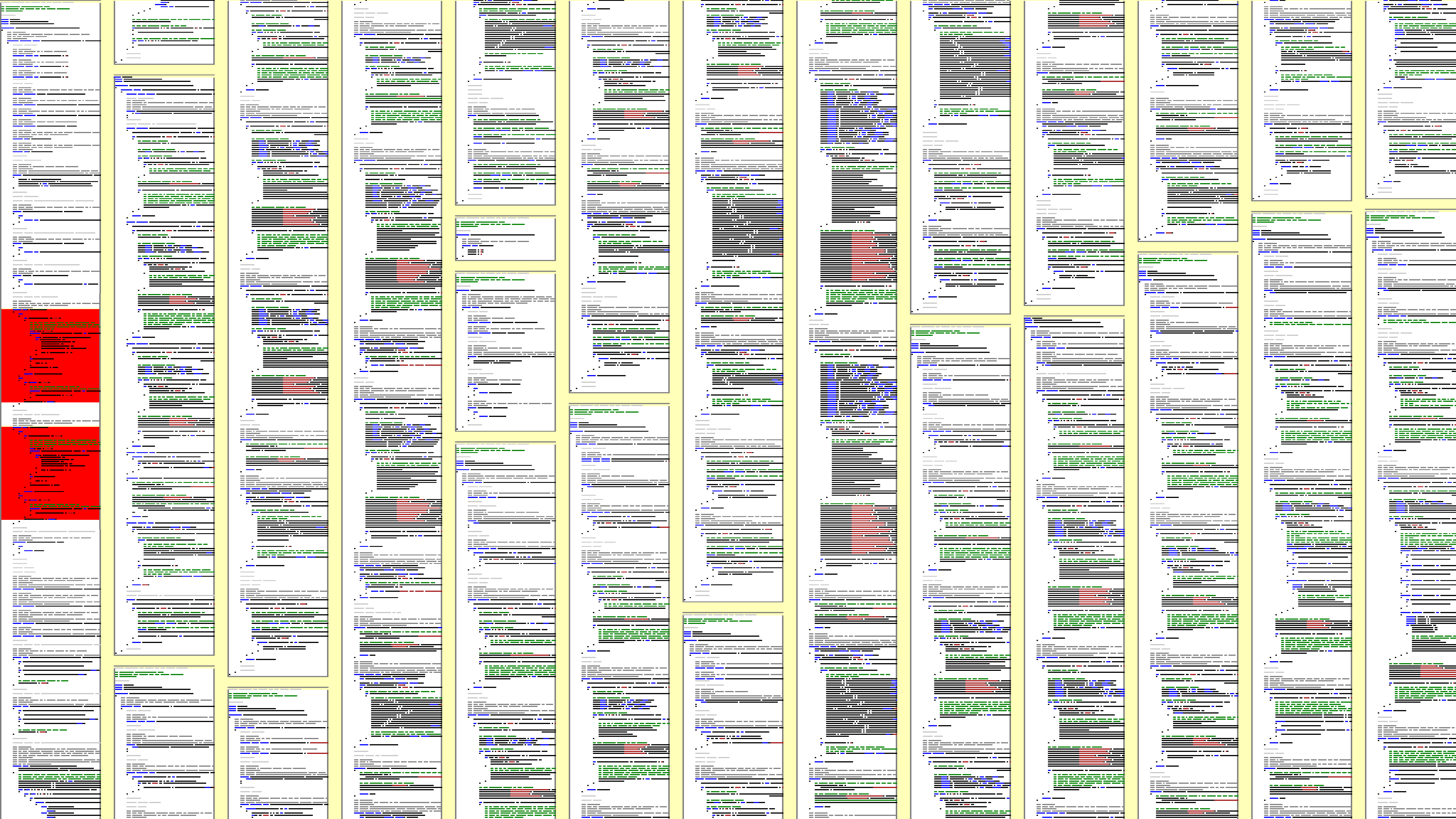
Lines of Code: 324,687
Number of Findings: 2,306
340,000
320,000
300,000
280,000
260,000
240,000
220,000
200,000
180,000
160,000
140,000
3,100
3,000
2,900
2,800
2,700
2,600
2,500
2,400
2,300
2,200
2,100
2,000
1,900
1,800
1,700
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011
Jul 2011
Jan 2012
Jul 2012
Jan 2013
Jul 2013

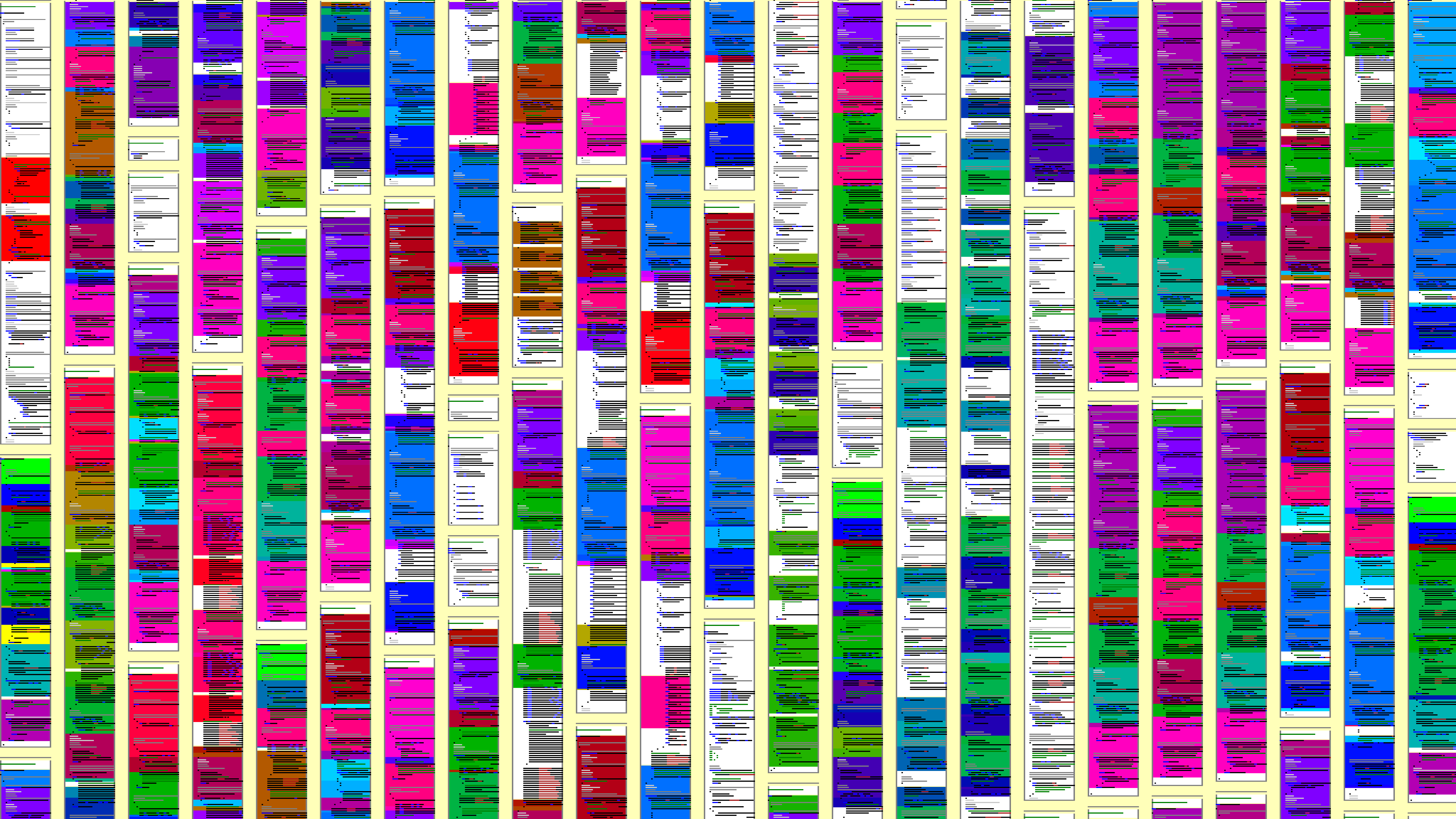
Wie können wir den Nutzen quantifizieren?

```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();
}
```

```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.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();
}
```





```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();
}
```

```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.length() > 0) {
        res.append(HTML.italics(nomsg));
    }
    return res.toString();
}
```

$$\text{Anzahl } \frac{Fehler}{Jahr} \times \text{Fehlerfolgekosten } \frac{PT}{Fehler}$$

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

#Fehler durch inkonsistente Klone

Daten aus Studie

- 3 Systeme von Munich Re analysiert
- 79 Fehler gefunden (Impact auf Funktionalität, nicht nur Wartbarkeit o.ä.)
- Systeme waren produktiv, einzelne Fehler schon durch Anwender als Tickets reportet
- 1 Produktionsfehler durch inkonsistente Klone / 17k SLOC

Bedeutung heute

- Betrachtetes Portfolio der Munich Re umfasst ca. 8,25 Millionen SLOC
- Konservative Annahme: Clone Management spart 1 Produktionsfehler pro 50k SLOC pro Jahr
- 8,25 Millionen SLOC / 50k = 165

$$165 \frac{Fehler}{Jahr} \times \text{Fehlerfolgekosten} \frac{PT}{Fehler}$$

ØFehlerfolgekosten von Fehlern in Produktion

Mögliche Auswirkungen fehlerhafter Software

- Nutzer bekommen falsche Ergebnisse
- Anwendung stürzt ab
- Daten gehen verloren
- Frustration bei Nutzern (Kunden und Mitarbeiter)

? PT

Aufwand für Reparatur

- Nutzer schreibt Ticket für Fehler
- Debugging (Nachstellen, Diagnose, …)
- Fixing
- Test
- Ggf. Deployment

? PT

ØFehlerfolgekosten von Fehlern in Produktion

Mögliche Auswirkungen fehlerhafter Software

- Nutzer bekommen falsche Ergebnisse
- Anwendung stürzt ab
- Daten gehen verloren
- Frustration bei Nutzern (Kunden und Mitarbeiter)

0 PT: bewusste Unterschätzung

Aufwand für Reparatur

- Nutzer schreibt Ticket für Fehler
- Debugging (Nachstellen, Diagnose, …)
- Fixing
- Test
- Ggf. Deployment

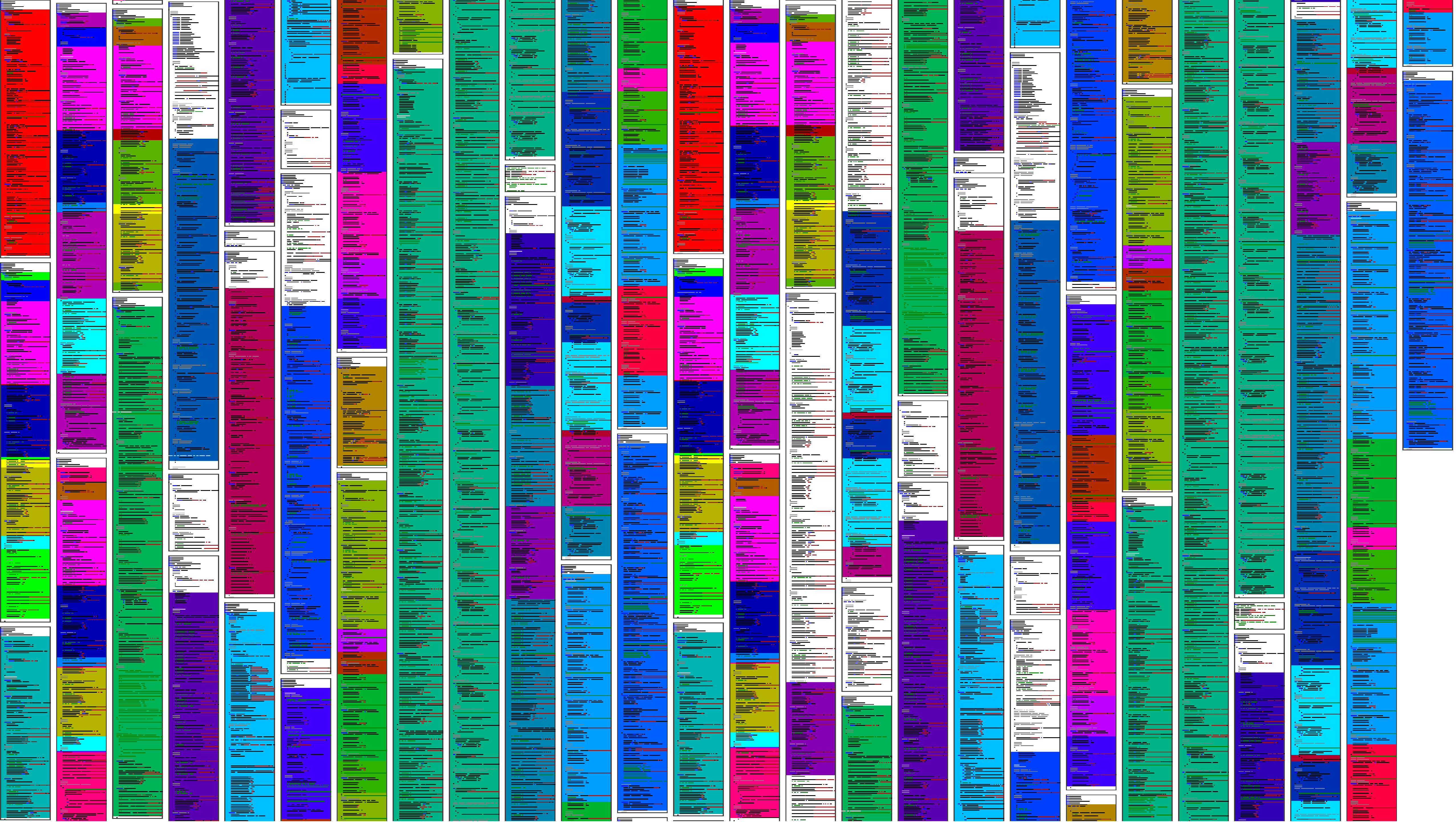
3 PT

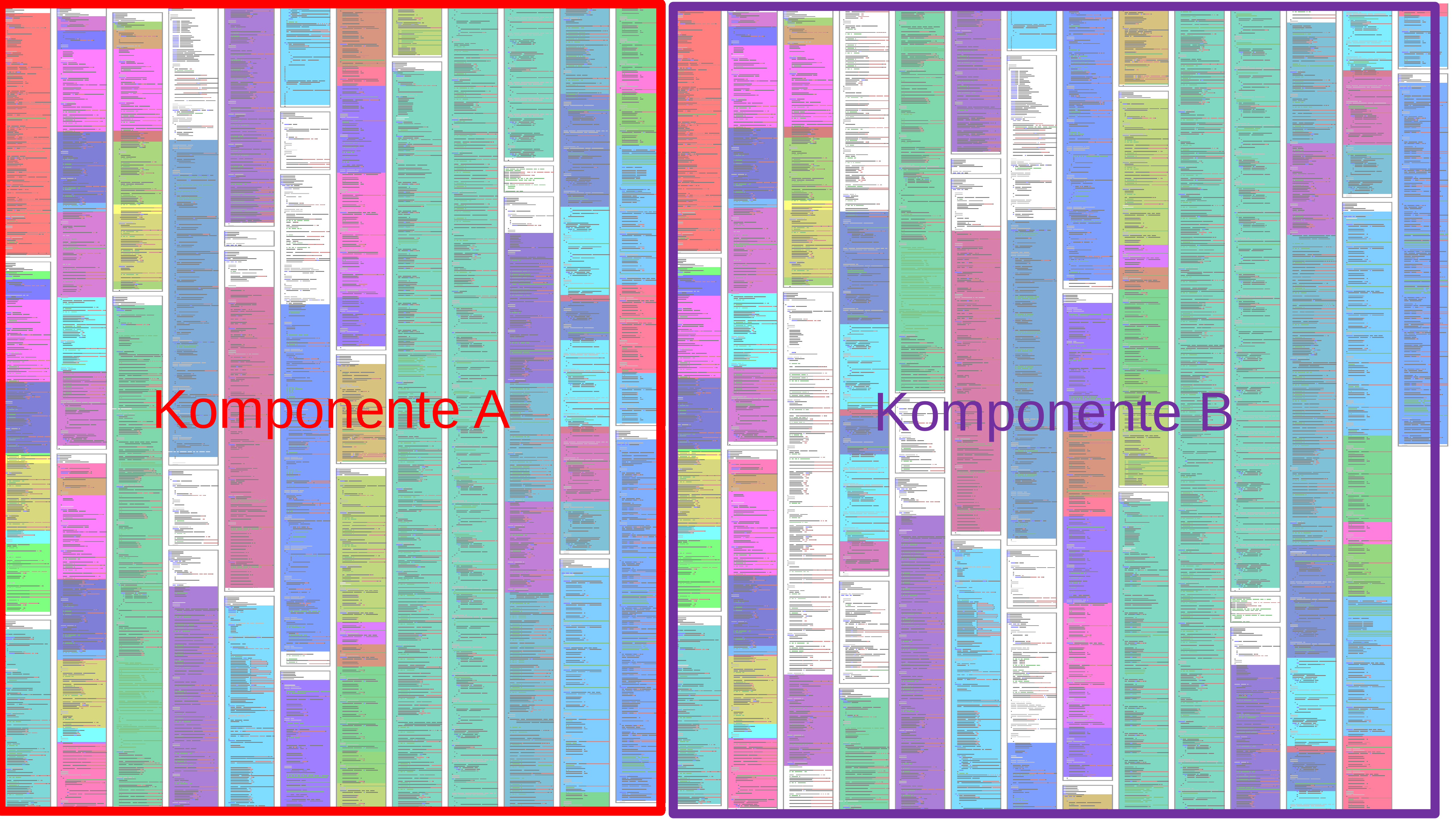
$$165\,\frac{Fehler}{Jahr} \times 3\,\frac{PT}{Fehler}$$

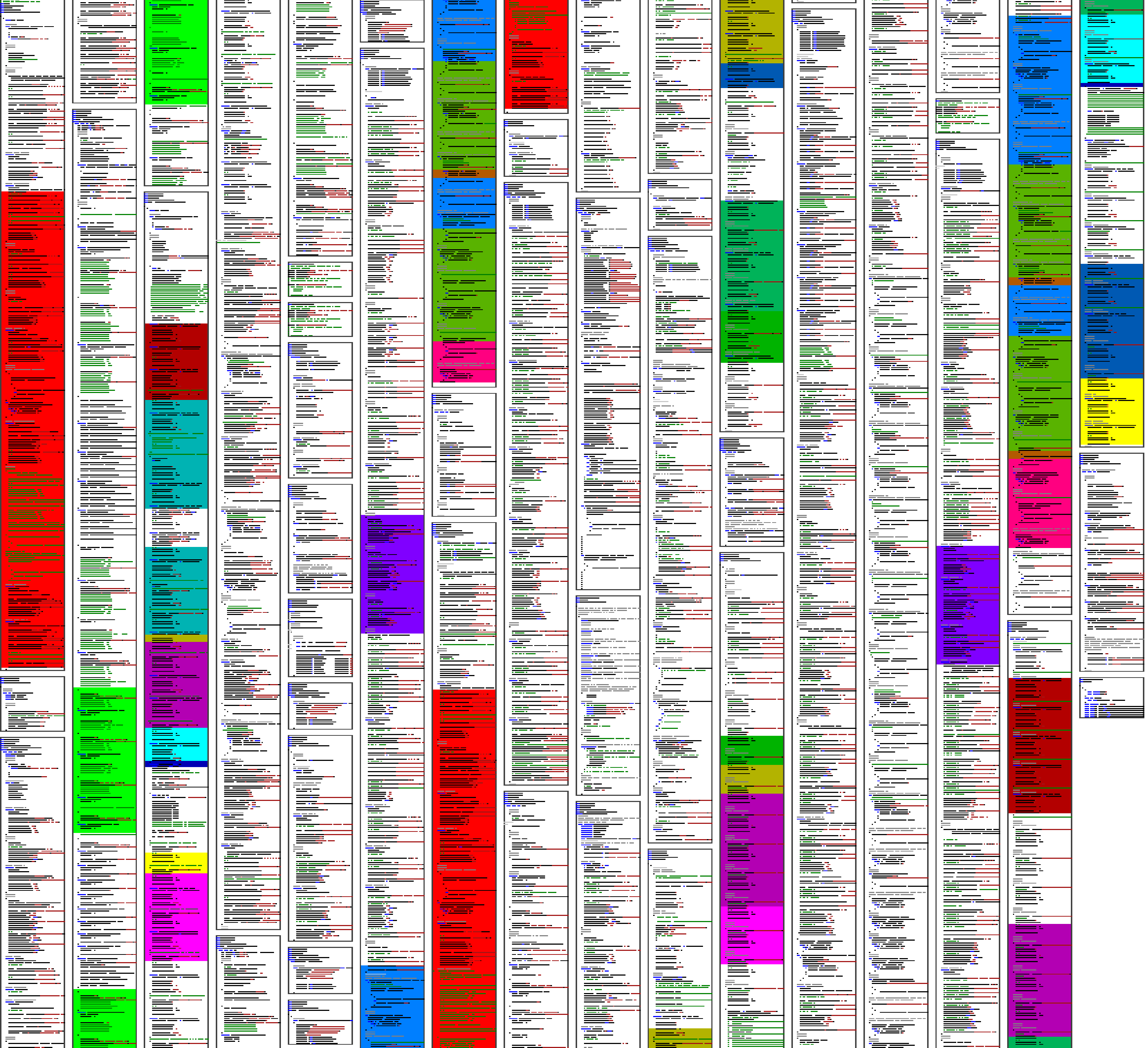
$$495\ \frac{PT}{Jahr}$$

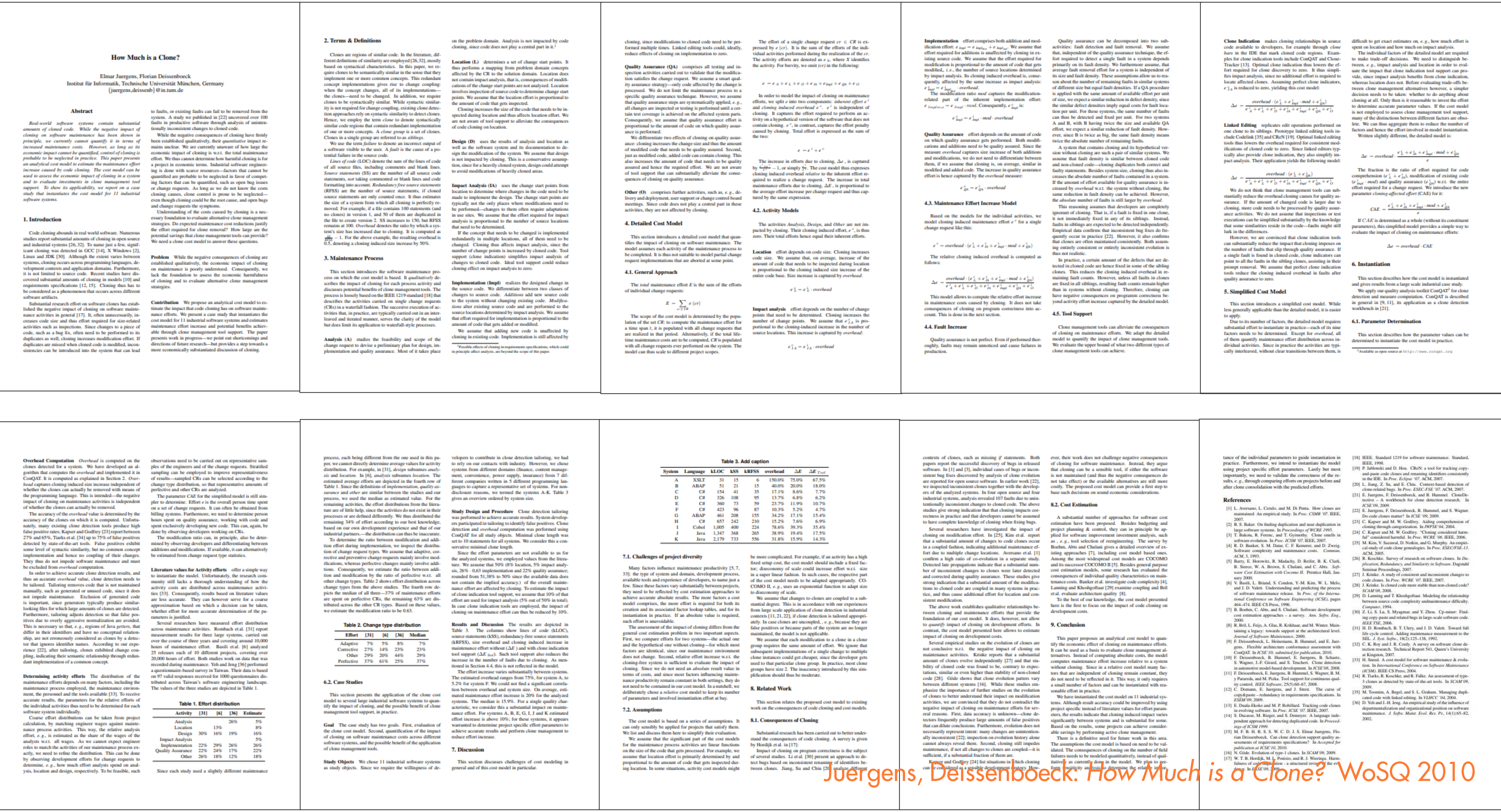
$$500\ \frac{PT}{Jahr}$$

Munich Re spart durch Einsatz von Clone Management jährlich ca. 500 PT Aufwand für Fehlerbehebung




Komponente A
Komponente B

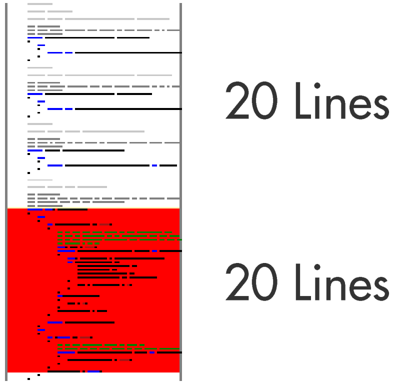




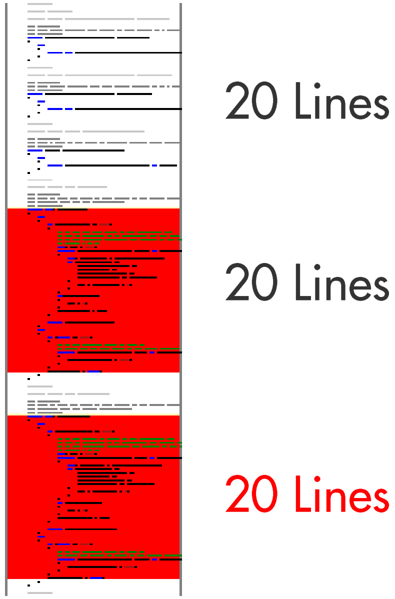
Juergens, Deissenboeck: How Much is a Clone? WoSQ 2010

$$\Delta \text{Aufwand} = \%\text{BlowUp} \times \%\text{CloneAffectedEffort}$$

Blow-Up: 0%



Blow-Up: 50%



Project	BlowUp	SLOC
[redacted]	1,54%	3658
[redacted]	9,47%	8294
[redacted]	14,77%	99852
[redacted]	25,77%	33592
[redacted]	14,21%	291411
[redacted]	7,57%	33764
[redacted]	44,78%	43092
[redacted]	0,95%	12211
[redacted]	20,90%	18910
[redacted]	15,92%	6579
[redacted]	22,49%	57939
[redacted]	0,35%	7791
[redacted]	4,62%	15942
[redacted]	12,94%	14885
[redacted]	4,19%	22239
[redacted]	14,36%	139295
[redacted]	2,60%	2046
[redacted]	4,57%	2963
[redacted]	15,11%	26633
[redacted]	54,92%	3754
[redacted]	11,01%	55582
[redacted]	24,19%	14500
[redacted]	15,72%	6496
[redacted]	6,44%	48572
[redacted]	1,67%	14759
[redacted]	24,39%	51610
[redacted]	1,62%	15432
[redacted]	5,94%	27758
[redacted]	27,07%	5171
[redacted]	0,83%	12147
[redacted]	9,01%	513884
[redacted]	20,10%	15411
[redacted]	21,91%	50805
[redacted]	7,23%	58685
[redacted]	3,47%	89744
[redacted]	9,54%	14323
[redacted]	20,07%	19498
[redacted]	4,97%	10023
[redacted]	7,65%	1033692
[redacted]	6,10%	50753
[redacted]	26,70%	133212
[redacted]	1,79%	14098
[redacted]	21,47%	25105
[redacted]	27,31%	13304
[redacted]	24,23%	21789
[redacted]	10,68%	20748
[redacted]	19,42%	138605
[redacted]	18,99%	25826
[redacted]	18,62%	216704
[redacted]	3,37%	7060
[redacted]	15,70%	230588
[redacted]	27,02%	362546
[redacted]	4,18%	269356
[redacted]	3,51%	9449
[redacted]	5,04%	48447
[redacted]	22,34%	83027
[redacted]	3,93%	192236
[redacted]	25,26%	259587
[redacted]	8,20%	8763
[redacted]	5,41%	118943
[redacted]	9,55%	337006
[redacted]	7,62%	121976
[redacted]	1,57%	34942
[redacted]	10,28%	25523
[redacted]	2,35%	66736
[redacted]	17,35%	250101
[redacted]	2,91%	76358
[redacted]	7,48%	62847
[redacted]	2,48%	16202
[redacted]	0,65%	52059
[redacted]	4,29%	123482
[redacted]	10,24%	16220
[redacted]	13,00%	212391
[redacted]	8,00%	54948
[redacted]	10,53%	243324
[redacted]	4,04%	136083
[redacted]	13,28%	112395
[redacted]	1,09%	5326
[redacted]	24,65%	16805
[redacted]	11,49%	101958
[redacted]	5,86%	34248
[redacted]	17,18%	25810
[redacted]	9,64%	908518
[redacted]	1,10%	45147
[redacted]	2,05%	5348

Blow-Up: Ø12%

Project	BlowUp	SLOC
[redacted]	1,54%	3658
[redacted]	9,47%	8294
[redacted]	14,77%	99852
[redacted]	25,77%	33592
[redacted]	14,21%	291411
[redacted]	7,57%	33764
[redacted]	44,78%	43092
[redacted]	0,95%	12211
[redacted]	20,90%	18910
[redacted]	15,92%	6579
[redacted]	22,49%	57939
[redacted]	0,35%	7791
[redacted]	4,62%	15942
[redacted]	12,94%	14885
[redacted]	4,19%	22239
[redacted]	14,36%	139295
[redacted]	2,60%	2046
[redacted]	4,57%	2963
[redacted]	15,11%	26633
[redacted]	54,92%	3754
[redacted]	11,01%	55582
[redacted]	24,19%	14500
[redacted]	15,72%	6496
[redacted]	6,44%	48572
[redacted]	1,67%	14759
[redacted]	24,39%	51610
[redacted]	1,62%	15432
[redacted]	5,94%	27758
[redacted]	27,07%	5171
[redacted]	0,83%	12147
[redacted]	9,01%	513884
[redacted]	20,10%	15411
[redacted]	21,91%	50805
[redacted]	7,23%	58685
[redacted]	3,47%	89744
[redacted]	9,54%	14323
[redacted]	20,07%	19498
[redacted]	4,97%	10023
[redacted]	7,65%	1033692
[redacted]	6,10%	50753
[redacted]	26,70%	133212
[redacted]	1,79%	14098
[redacted]	21,47%	25105
[redacted]	27,31%	13304
[redacted]	24,23%	21789
[redacted]	10,68%	20748
[redacted]	19,42%	138605
[redacted]	18,99%	25826
[redacted]	18,62%	216704
[redacted]	3,37%	7060
[redacted]	15,70%	230588
[redacted]	27,02%	362546
[redacted]	4,18%	269356
[redacted]	3,51%	9449
[redacted]	5,04%	48447
[redacted]	22,34%	83027
[redacted]	3,93%	192236
[redacted]	25,26%	259587
[redacted]	8,20%	8763
[redacted]	5,41%	118943
[redacted]	9,55%	337006
[redacted]	7,62%	121976
[redacted]	1,57%	34942
[redacted]	10,28%	25523
[redacted]	2,35%	66736
[redacted]	17,35%	250101
[redacted]	2,91%	76358
[redacted]	7,48%	62847
[redacted]	2,48%	16202
[redacted]	0,65%	52059
[redacted]	4,29%	123482
[redacted]	10,24%	16220
[redacted]	13,00%	212391
[redacted]	8,00%	54948
[redacted]	10,53%	243324
[redacted]	4,04%	136083
[redacted]	13,28%	112395
[redacted]	1,09%	5326
[redacted]	24,65%	16805
[redacted]	11,49%	101958
[redacted]	5,86%	34248
[redacted]	17,18%	25810
[redacted]	9,64%	908518
[redacted]	1,10%	45147
[redacted]	2,05%	5348

Blow-Up: Ø12%
(Das ist wenig)

Project	BlowUp	SLOC
[redacted]	1,54%	3658
	9,47%	8294
	14,77%	99852
	25,77%	33592
	14,21%	291411
	7,57%	33764
	44,78%	43092
	0,95%	12211
	20,90%	18910
	15,92%	6579
	22,49%	57939
	0,35%	7791
	4,62%	15942
	12,94%	14885
	4,19%	22239
	14,36%	139295
	2,60%	2046
	4,57%	2963
	15,11%	26633
	54,92%	3754
	11,01%	55582
	24,19%	14500
	15,72%	6496
	6,44%	48572
	1,67%	14759
	24,39%	51610
	1,62%	15432
	5,94%	27758
[redacted]	27,07%	5171
	0,83%	12147
	9,01%	513884
	20,10%	15411
	21,91%	50805
	7,23%	58685
	3,47%	89744
	9,54%	14323
	20,07%	19498
	4,97%	10023
	7,65%	1033692
	6,10%	[illegible]
	26,70%	[illegible]
	1,79%	14098
	21,47%	25105
	[illegible]	[illegible]
	24,23%	[illegible]
	10,68%	20748
	19,42%	138605
	18,99%	25826
	18,62%	216704
	3,37%	7060
	15,70%	230588
	27,02%	362546
	4,18%	269356
	3,51%	9449
	5,04%	48447
	22,34%	83027
	3,93%	192236
[redacted]	25,26%	259587
	8,20%	8763
	5,41%	118943
	9,55%	337006
	7,62%	121976
	1,57%	34942
	10,28%	25523
	2,35%	66736
	17,35%	250101
	2,91%	76358
	7,48%	62847
	2,48%	16202
	0,65%	52059
	4,29%	123482
	10,24%	16220
	13,00%	212391
	8,00%	54948
	10,53%	243324
	4,04%	136083
	13,28%	112395
	1,09%	5326
	24,65%	16805
	11,49%	101958
	5,86%	34248
	17,18%	25810
	9,64%	908518
	1,10%	45147
	2,05%	5348

$$\Delta \text{Aufwand} = \%\mathbf{12} \times \%\text{CloneAffectedEffort}$$

%CloneAffectedEffort

Aktivitäten	Aufwändiger durch Cloning
▪ Analysis	-
▪ Location	Location
▪ Design	-
▪ Impact Analysis	Impact Analysis
▪ Implementation	Implementation
▪ Quality Assurance	Quality Assurance
▪ Other	-

Detaillierte Herleitung und Berechnung im Paper.

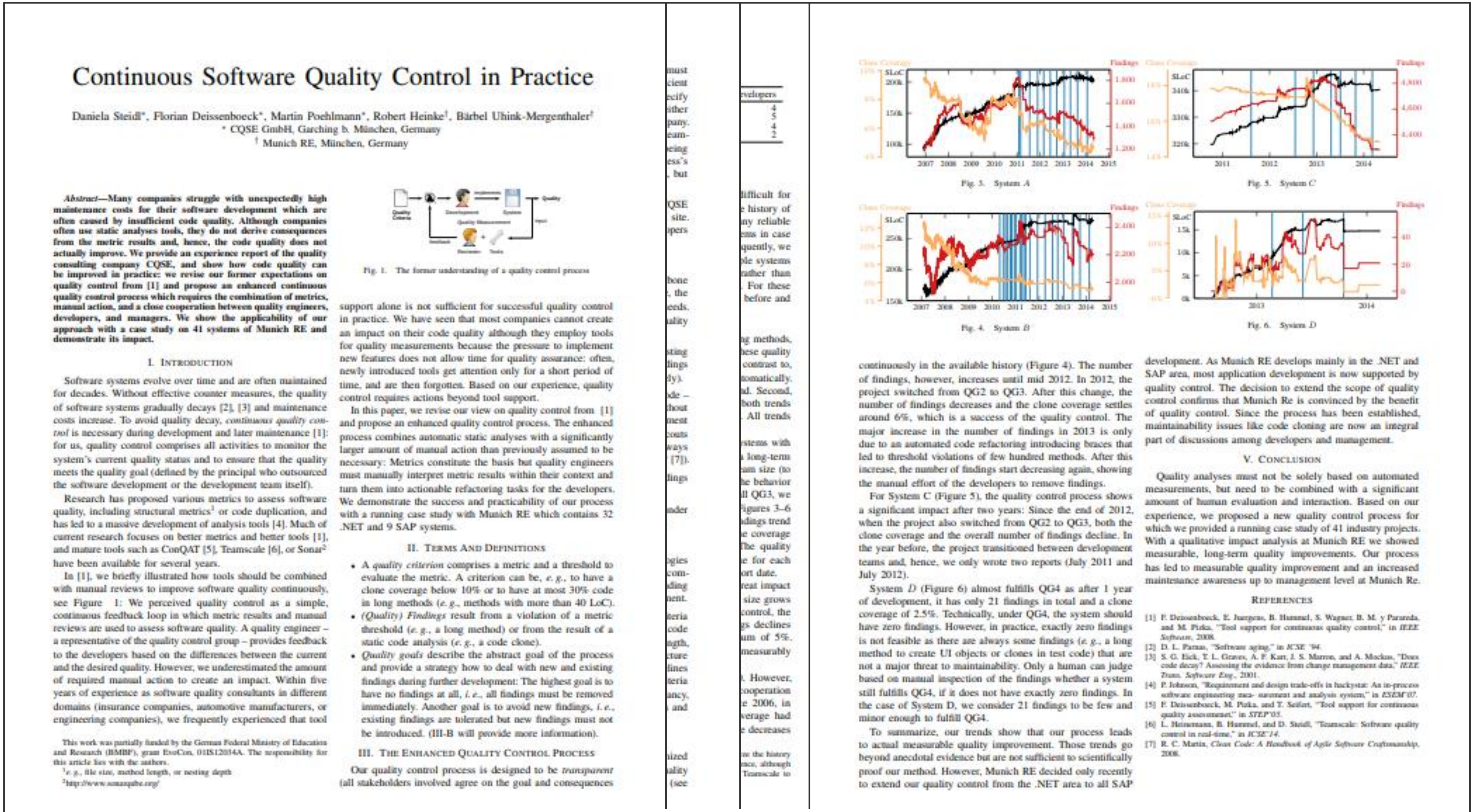
Wert für Berechnung: **51%**.

$$\Delta \text{Aufwand} = \%12 \times \mathbf{\%50}$$

$$\Delta \text{Aufwand} = \%12 \times \%50 = \mathbf{6\%}$$

Die Munich Re setzt
Clone Management seit
ca. 10 Jahren ein.

Wie sähe es ohne aus?



Continuous Software Quality Control in Practice

Daniela Steidl*, Florian Deissenboeck*, Martin Poehlmann*, Robert Heinke†, Bärbel Uhink-Mergenthaler†
* CQSE GmbH, Garching b. München, Germany
† Munich RE, München, Germany

Abstract—**Many companies struggle with unexpectedly high maintenance costs for their software development which are often caused by insufficient code quality. Although companies often use static analyses tools, they do not derive consequences from the metric results and, hence, the code quality does not actually improve. We provide an experience report of the quality consulting company CQSE, and show how code quality can be improved in practice: we revise our former expectations on quality control from [1] and propose an enhanced continuous quality control process which requires the combination of metrics, manual action, and a close cooperation between quality engineers, developers, and managers. We show the applicability of our approach with a case study on 41 systems of Munich RE and demonstrate its impact.**

I. Introduction

Software systems evolve over time and are often maintained for decades. Without effective counter measures, the quality of software systems gradually decays [2], [3] and maintenance costs increase. To avoid quality decay, *continuous quality control* is necessary during development and later maintenance [1]: for us, quality control comprises all activities to monitor the system's current quality status and to ensure that the quality meets the quality goal (defined by the principal who outsourced the software development or the development team itself).

Research has proposed various metrics to assess software quality, including structural metrics[1] or code duplication, and has led to a massive development of analysis tools [4]. Much of current research focuses on better metrics and better tools [1], and mature tools such as ConQAT [5], Teamscale [6], or Sonar[2] have been available for several years.

In [1], we briefly illustrated how tools should be combined with manual reviews to improve software quality continuously, see Figure 1: We perceived quality control as a simple, continuous feedback loop in which metric results and manual reviews are used to assess software quality. A quality engineer – a representative of the quality control group – provides feedback to the developers based on the differences between the current and the desired quality. However, we underestimated the amount of required manual action to create an impact. Within five years of experience as software quality consultants in different domains (insurance companies, automotive manufacturers, or engineering companies), we frequently experienced that tool

This work was partially funded by the German Federal Ministry of Education and Research (BMBF), grant EvoCon, 01IS12034A. The responsibility for this article lies with the authors.

[1] *e. g.*, file size, method length, or nesting depth

[2] http://www.sonarqube.org/

Fig. 1. The former understanding of a quality control process

support alone is not sufficient for successful quality control in practice. We have seen that most companies cannot create an impact on their code quality although they employ tools for quality measurements because the pressure to implement new features does not allow time for quality assurance: often, newly introduced tools get attention only for a short period of time, and are then forgotten. Based on our experience, quality control requires actions beyond tool support.

In this paper, we revise our view on quality control from [1] and propose an enhanced quality control process. The enhanced process combines automatic static analyses with a significantly larger amount of manual action than previously assumed to be necessary: Metrics constitute the basis but quality engineers must manually interpret metric results within their context and turn them into actionable refactoring tasks for the developers. We demonstrate the success and practicability of our process with a running case study with Munich RE which contains 32 .NET and 9 SAP systems.

II. Terms and Definitions

- A *quality criterion* comprises a metric and a threshold to evaluate the metric. A criterion can be, *e. g.*, to have a clone coverage below 10% or to have at most 30% code in long methods (*e. g.*, methods with more than 40 LoC).
- *(Quality) Findings* result from a violation of a metric threshold (*e. g.*, a long method) or from the result of a static code analysis (*e. g.*, a code clone).
- *Quality goals* describe the abstract goal of the process and provide a strategy how to deal with new and existing findings during further development: The highest goal is to have no findings at all, *i. e.*, all findings must be removed immediately. Another goal is to avoid new findings, *i. e.*, existing findings are tolerated but new findings must not be introduced. (III-B will provide more information).

III. The Enhanced Quality Control Process

Our quality control process is designed to be *transparent* (all stakeholders involved agree on the goal and consequences

Fig. 3. System *A*

Fig. 5. System *C*

Fig. 4. System *B*

Fig. 6. System *D*

continuously in the available history (Figure 4). The number of findings, however, increases until mid 2012. In 2012, the project switched from QG2 to QG3. After this change, the number of findings decreases and the clone coverage settles around 6%, which is a success of the quality control. The major increase in the number of findings in 2013 is only due to an automated code refactoring introducing braces that led to threshold violations of few hundred methods. After this increase, the number of findings start decreasing again, showing the manual effort of the developers to remove findings.

For System C (Figure 5), the quality control process shows a significant impact after two years: Since the end of 2012, when the project also switched from QG2 to QG3, both the clone coverage and the overall number of findings decline. In the year before, the project transitioned between development teams and, hence, we only wrote two reports (July 2011 and July 2012).

System *D* (Figure 6) almost fulfills QG4 as after 1 year of development, it has only 21 findings in total and a clone coverage of 2.5%. Technically, under QG4, the system should have zero findings. However, in practice, exactly zero findings is not feasible as there are always some findings (*e. g.*, a long method to create UI objects or clones in test code) that are not a major threat to maintainability. Only a human can judge based on manual inspection of the findings whether a system still fulfills QG4, if it does not have exactly zero findings. In the case of System D, we consider 21 findings to be few and minor enough to fulfill QG4.

To summarize, our trends show that our process leads to actual measurable quality improvement. Those trends go beyond anecdotal evidence but are not sufficient to scientifically proof our method. However, Munich RE decided only recently to extend our quality control from the .NET area to all SAP development. As Munich RE develops mainly in the .NET and SAP area, most application development is now supported by quality control. The decision to extend the scope of quality control confirms that Munich Re is convinced by the benefit of quality control. Since the process has been established, maintainability issues like code cloning are now an integral part of discussions among developers and management.

V. Conclusion

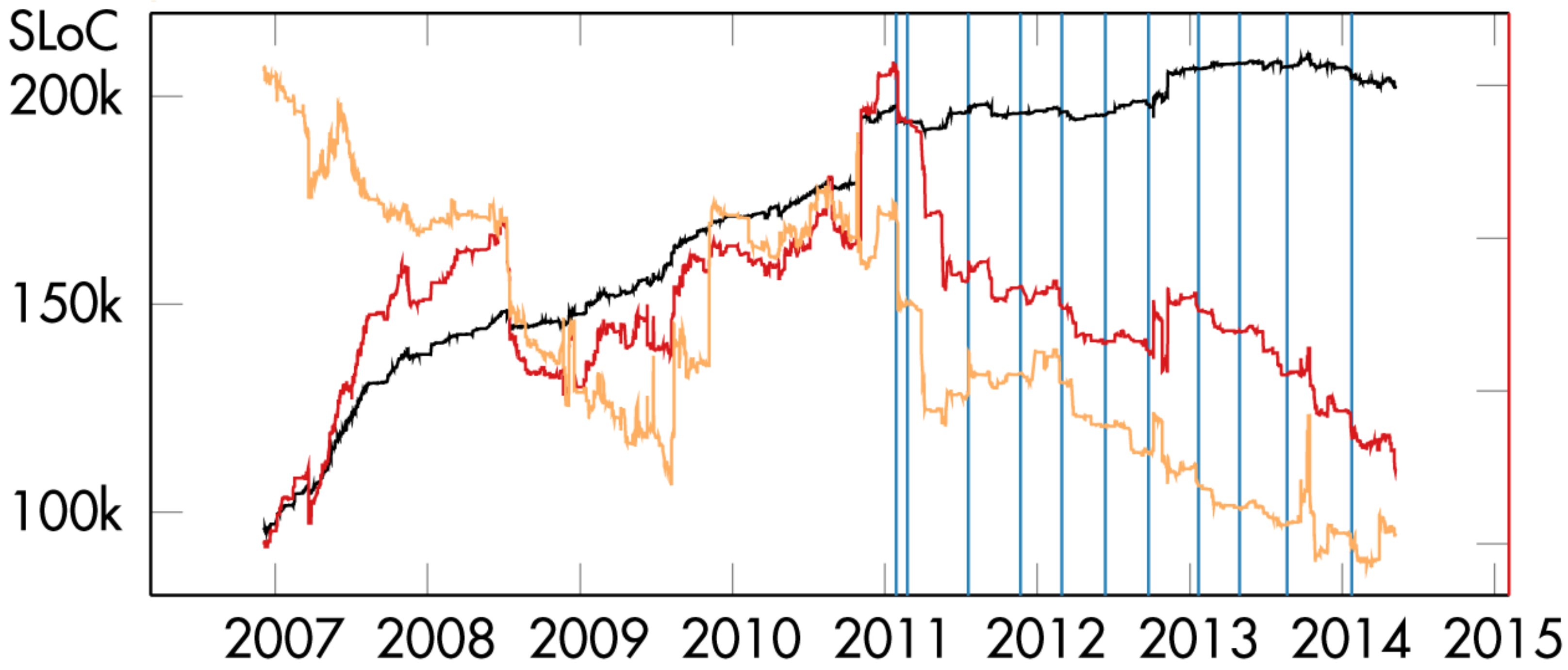
Quality analyses must not be solely based on automated measurements, but need to be combined with a significant amount of human evaluation and interaction. Based on our experience, we proposed a new quality control process for which we provided a running case study of 41 industry projects. With a qualitative impact analysis at Munich RE we showed measurable, long-term quality improvements. Our process has led to measurable quality improvement and an increased maintenance awareness up to management level at Munich Re.

References

[1] F. Deissenboeck, E. Juergens, B. Hummel, S. Wagner, B. M. y Parareda, and M. Pizka, "Tool support for continuous quality control," in *IEEE Software*, 2008.
[2] D. L. Parnas, "Software aging," in *ICSE '94*.
[3] S. G. Eick, T. L. Graves, A. F. Karr, J. S. Marron, and A. Mockus, "Does code decay? Assessing the evidence from change management data," *IEEE Trans. Software Eng.*, 2001.
[4] P. Johnson, "Requirement and design trade-offs in hackystat: An in-process software engineering mea- surement and analysis system," in *ESEM'07*.
[5] F. Deissenboeck, M. Pizka, and T. Seifert, "Tool support for continuous quality assessment," in *STEP'05*.
[6] L. Heinemann, B. Hummel, and D. Steidl, "Teamscale: Software quality control in real-time," in *ICSE'14*.
[7] R. C. Martin, *Clean Code: A Handbook of Agile Software Craftsmanship*, 2008.

Steidl, Heinke, Uhink: Continuous Software Quality Control in Practice? *ICSME* 2014

Einsparung durch Clone Detection



Menge an geklontem Code hat sich seit der Einführung von Clone Management halbiert.
Ohne Clone Management wäre der Clone Blow-Up daher vorraussichtlich doppelt so groß.

$$\text{Ersparnis Aufwand} = \mathbf{6\%}$$

Munich Re spart durch Einsatz von Clone Detection jährlich 6% Aufwand durch vermiedene Redundanz ein.

Findings

All	**6793**
Architecture	**1**
Architecture Conformance	1
Code Anomalies	**1344**
Bad practice	971
Correctness	2
Exception Handling	62
General checks (built-in)	120
Null pointer dereference	13
Performance	36
Unused code	93
Unused variable or parameter	47
Code Duplication	**988**
Cloning	101
Redundant Literals	887
Documentation	**3378**
Comment completeness	3236
Task tags	142
Formatting	**6**
Code formatting	6
Naming	**110**
Java naming conventions	110
Structure	**966**
File Size	38
Method Length	278
Nesting Depth	650

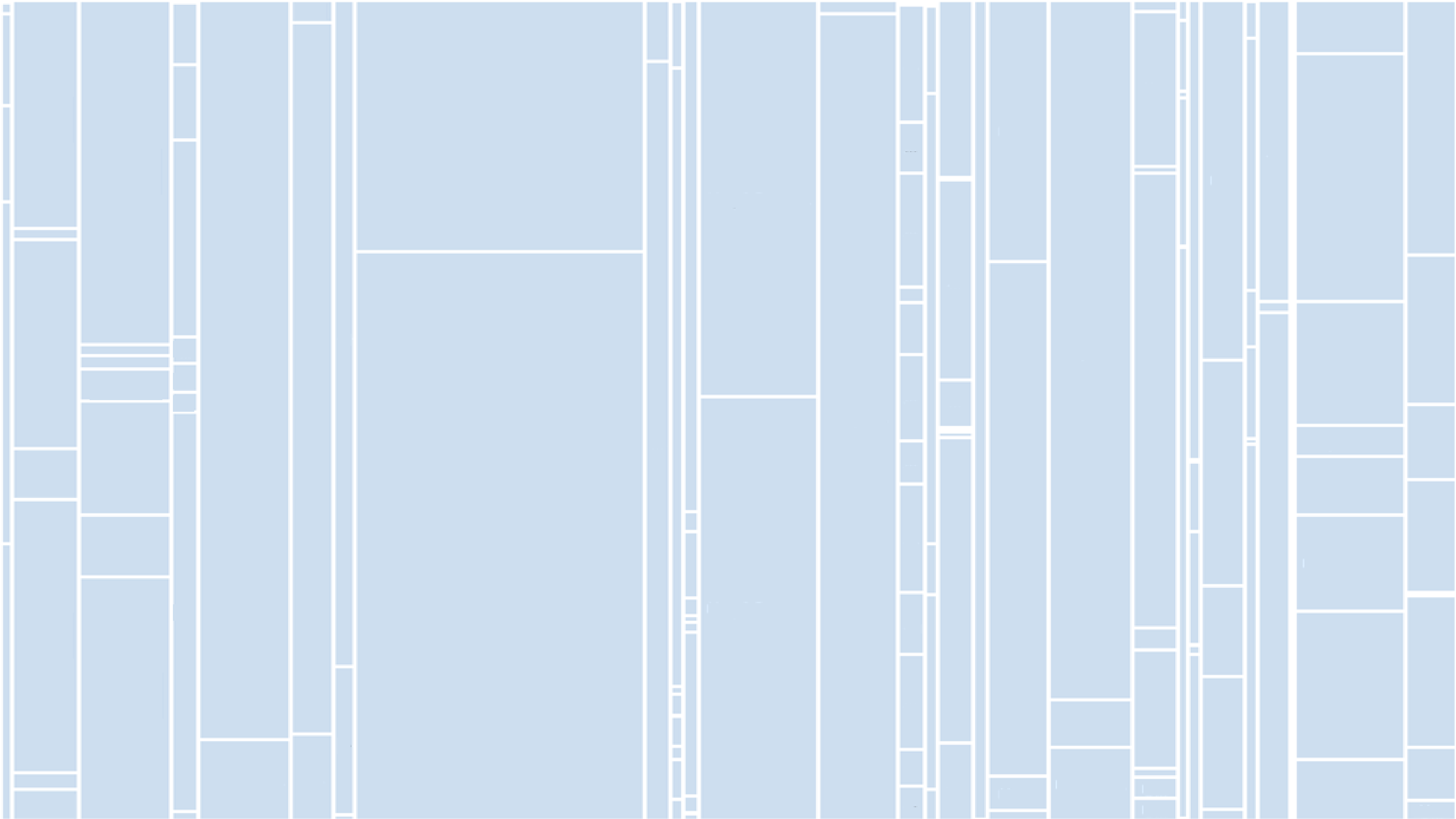
$$500 \frac{PT}{Jahr}$$

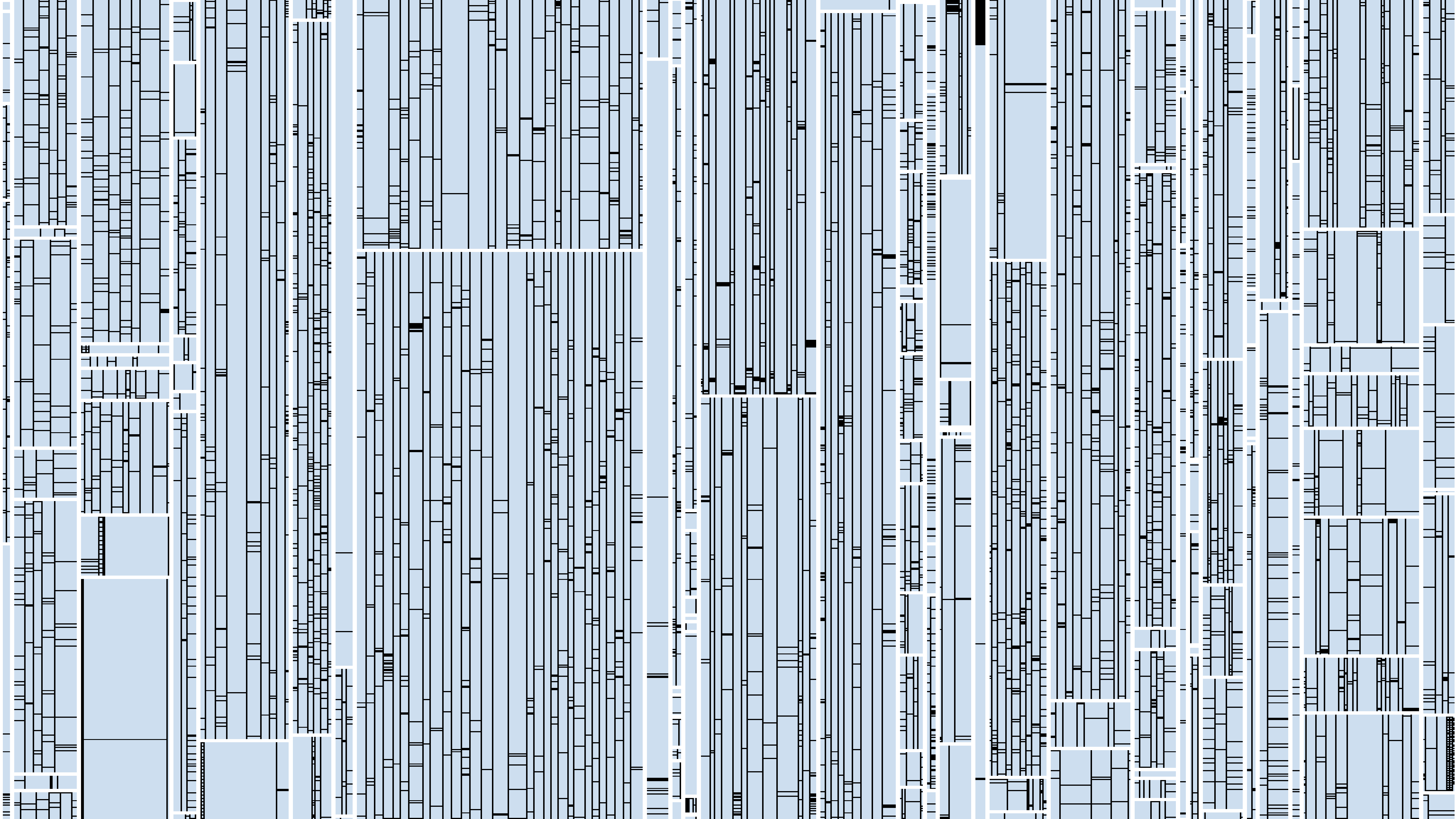
Munich Re spart durch Einsatz von Clone Detection jährlich ca. 500 PT Aufwand für Fehlerbehebung

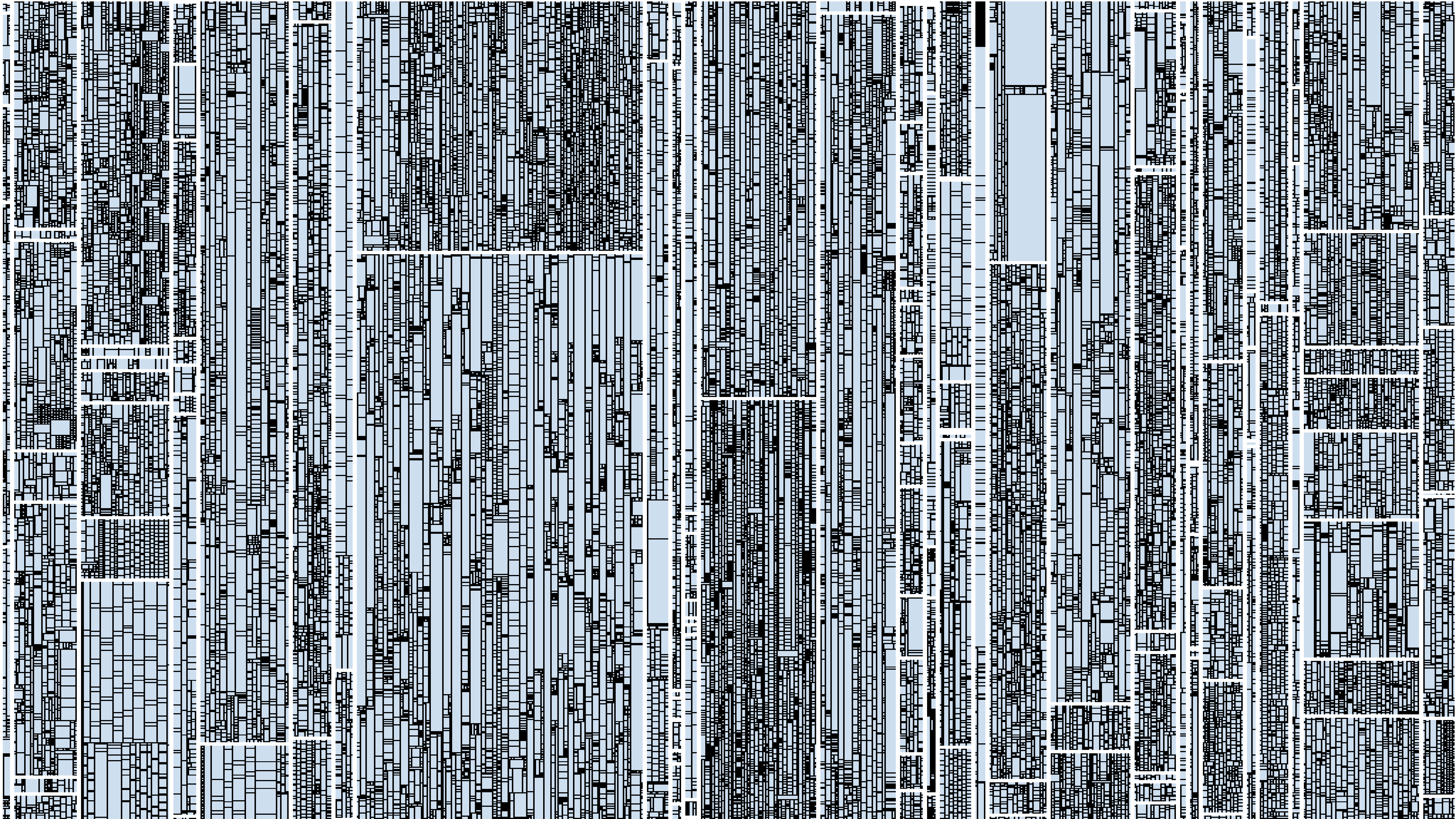
$$\text{Ersparnis Aufwand} = \mathbf{6\%}$$

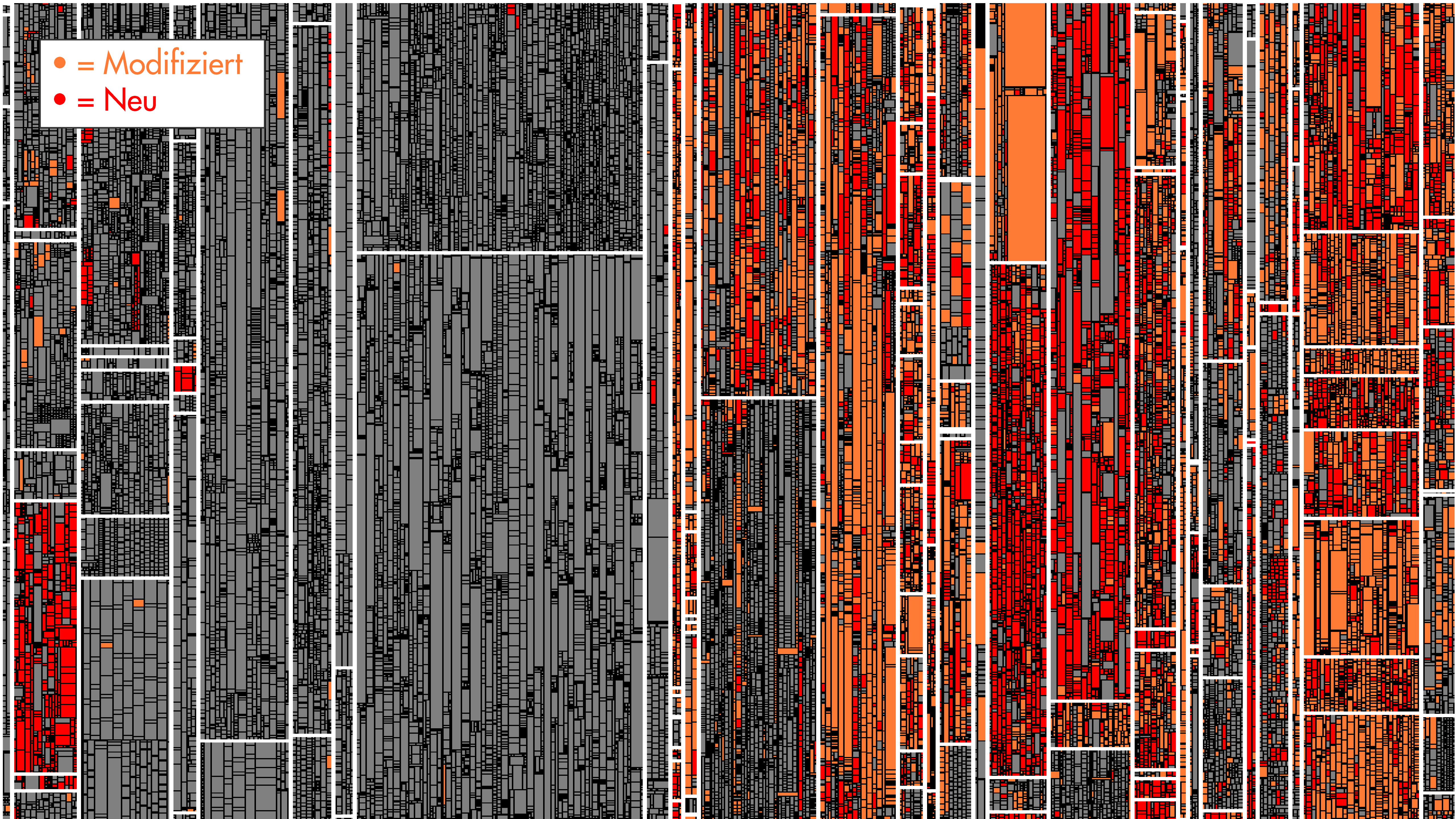
Munich Re spart durch Einsatz von Clone Detection jährlich 6% Aufwand durch vermiedene Redundanz ein.

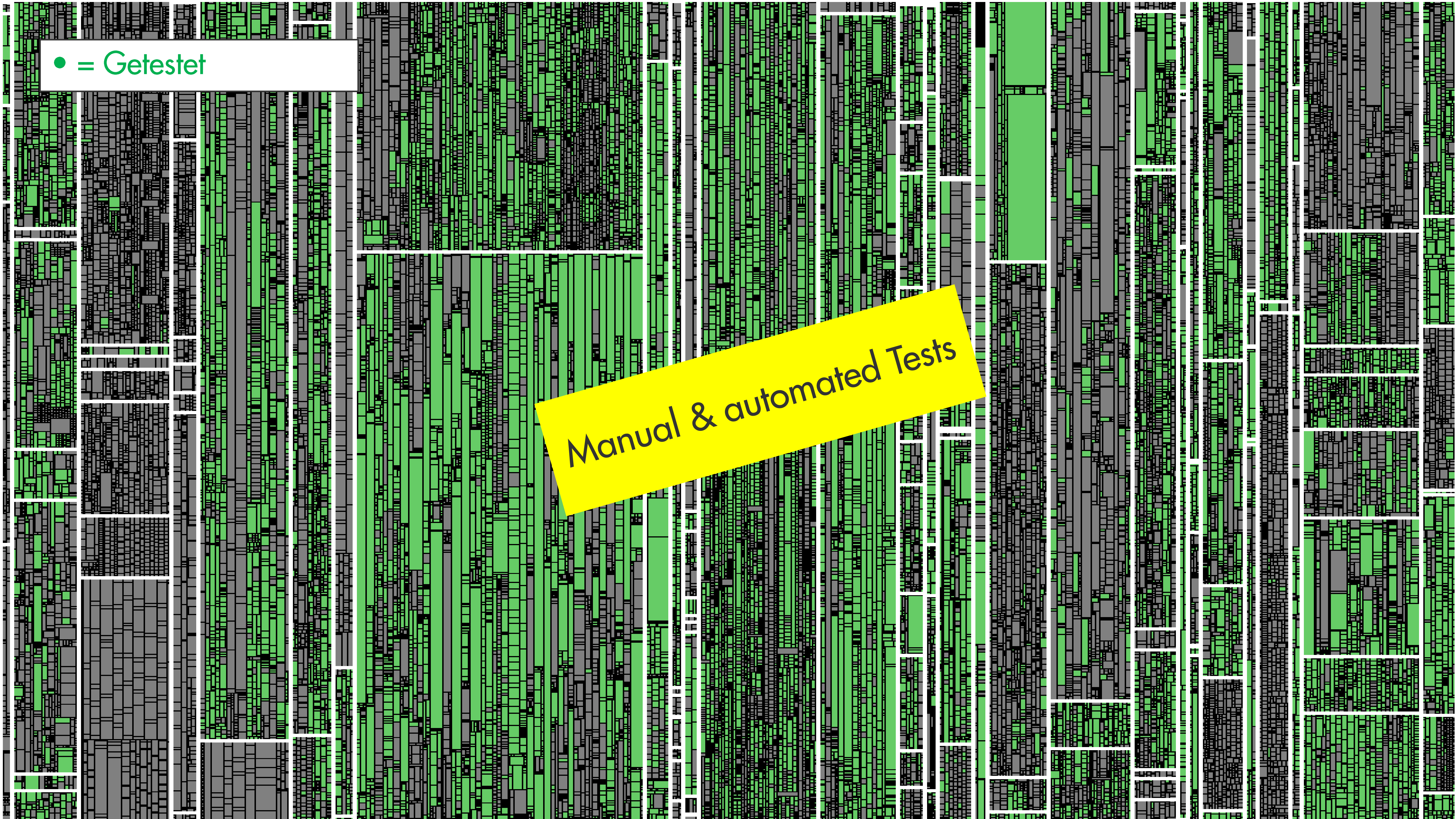
Kosten-Nutzen von Test-Gap-Analyse

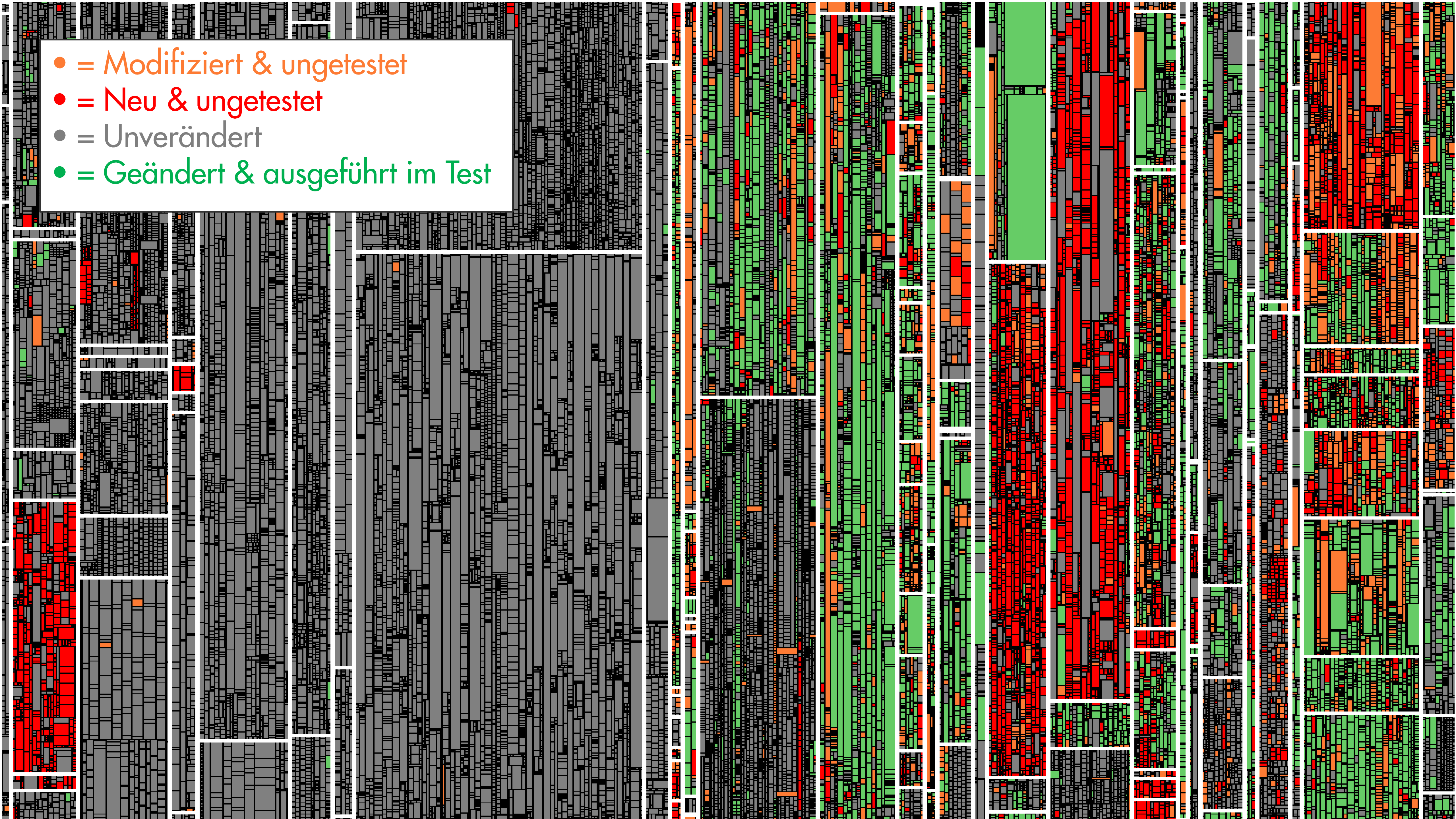




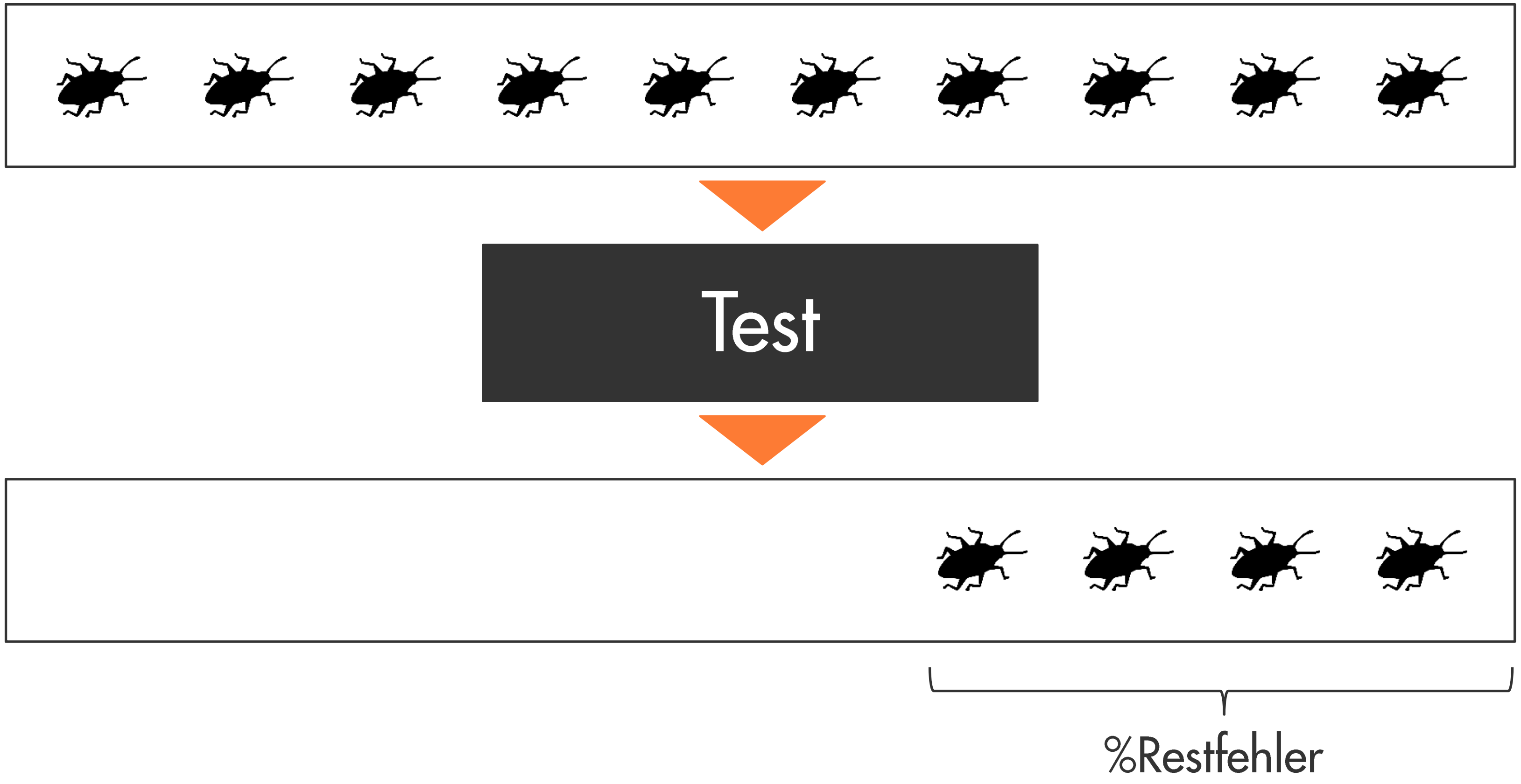



● = Modifiziert
● = Neu

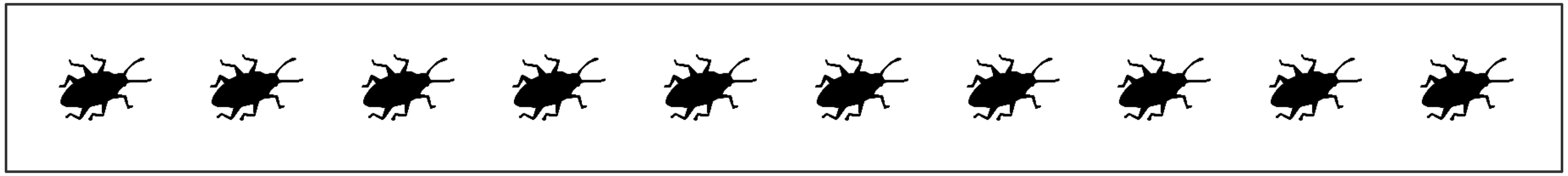

• = Getestet
Manual & automated Tests



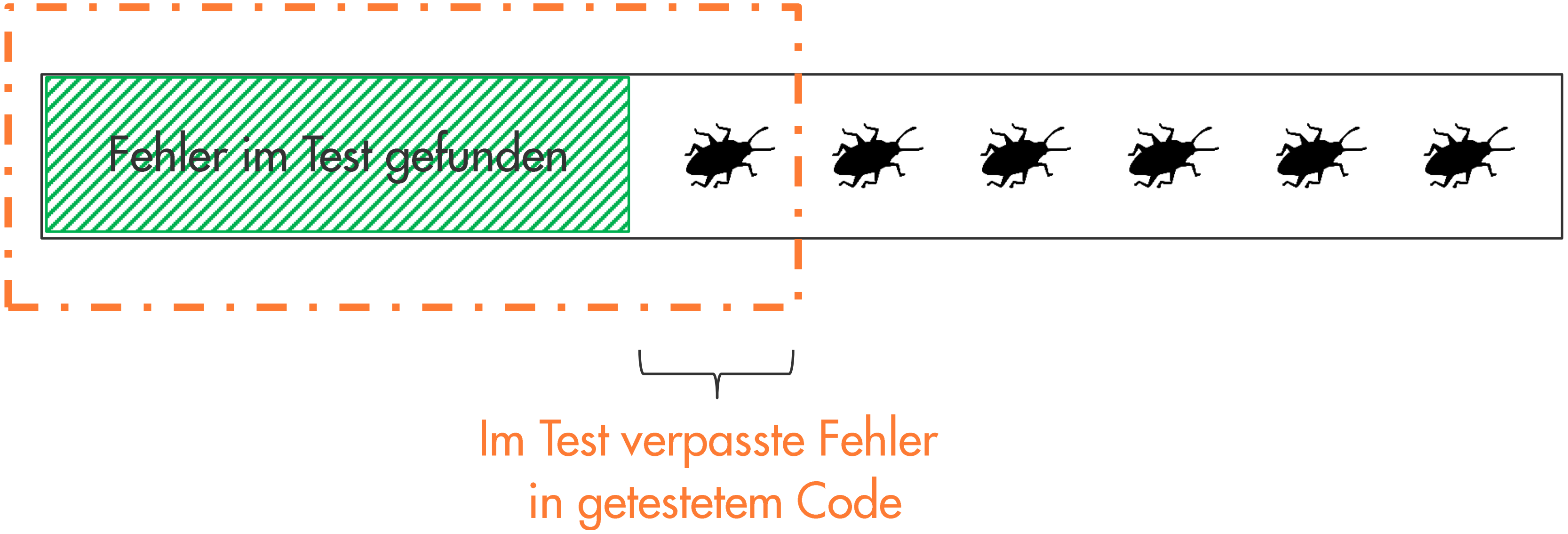
● = Modifiziert & ungetestet
● = Neu & ungetestet
● = Unverändert
● = Geändert & ausgeführt im Test


Test
%Restfehler

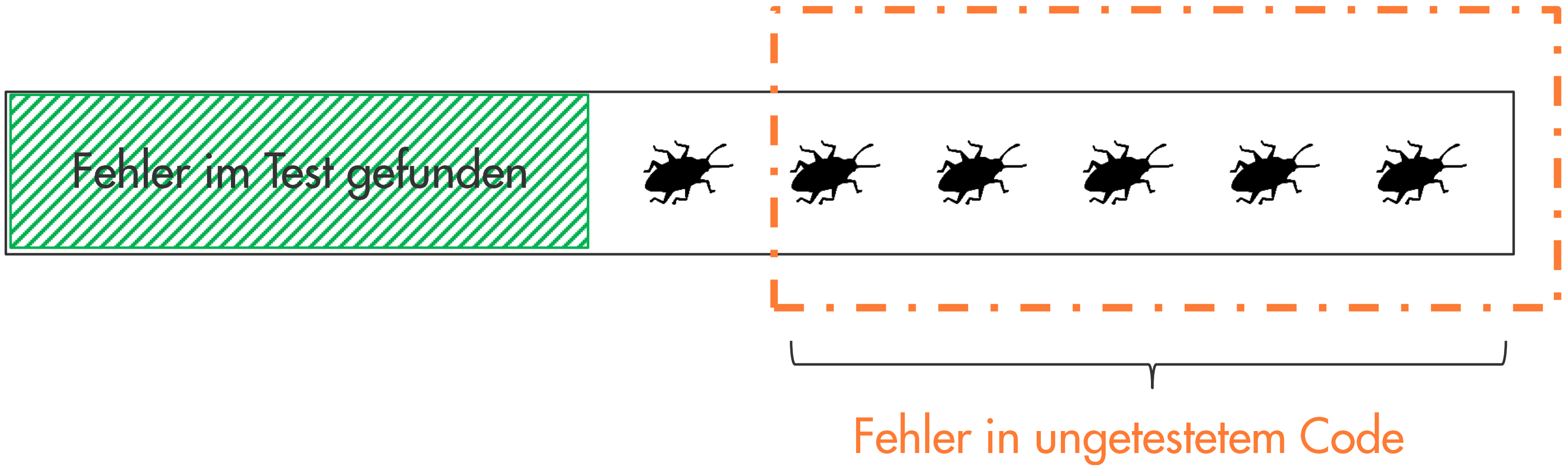
$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$

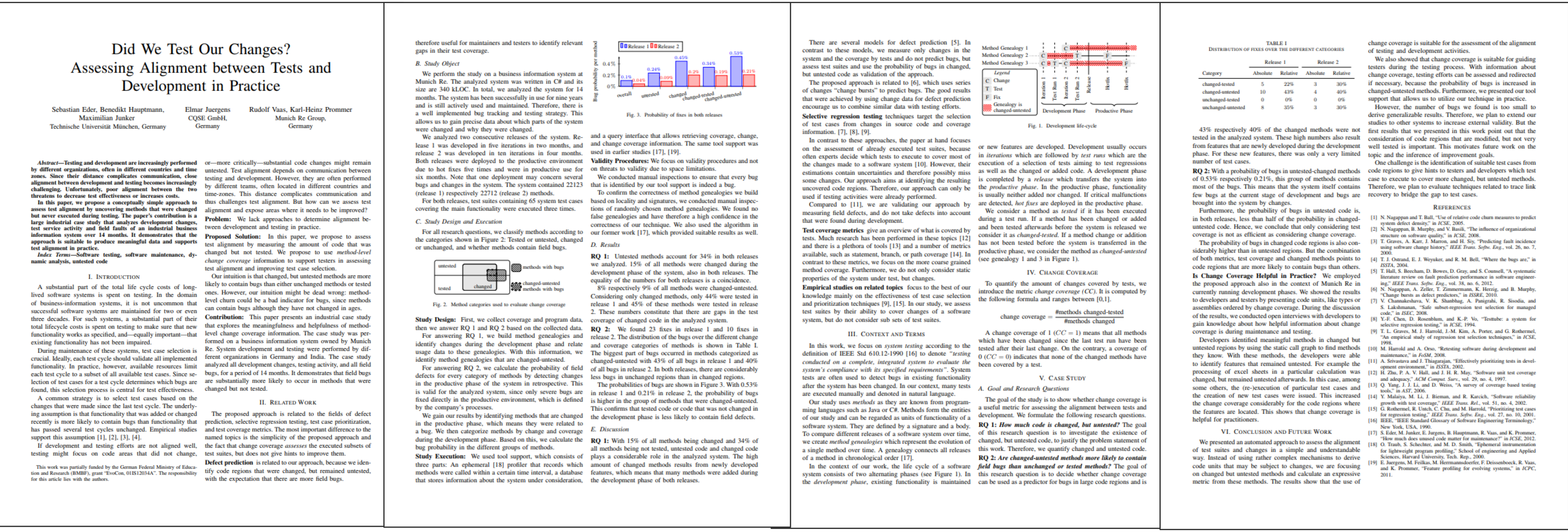


$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$





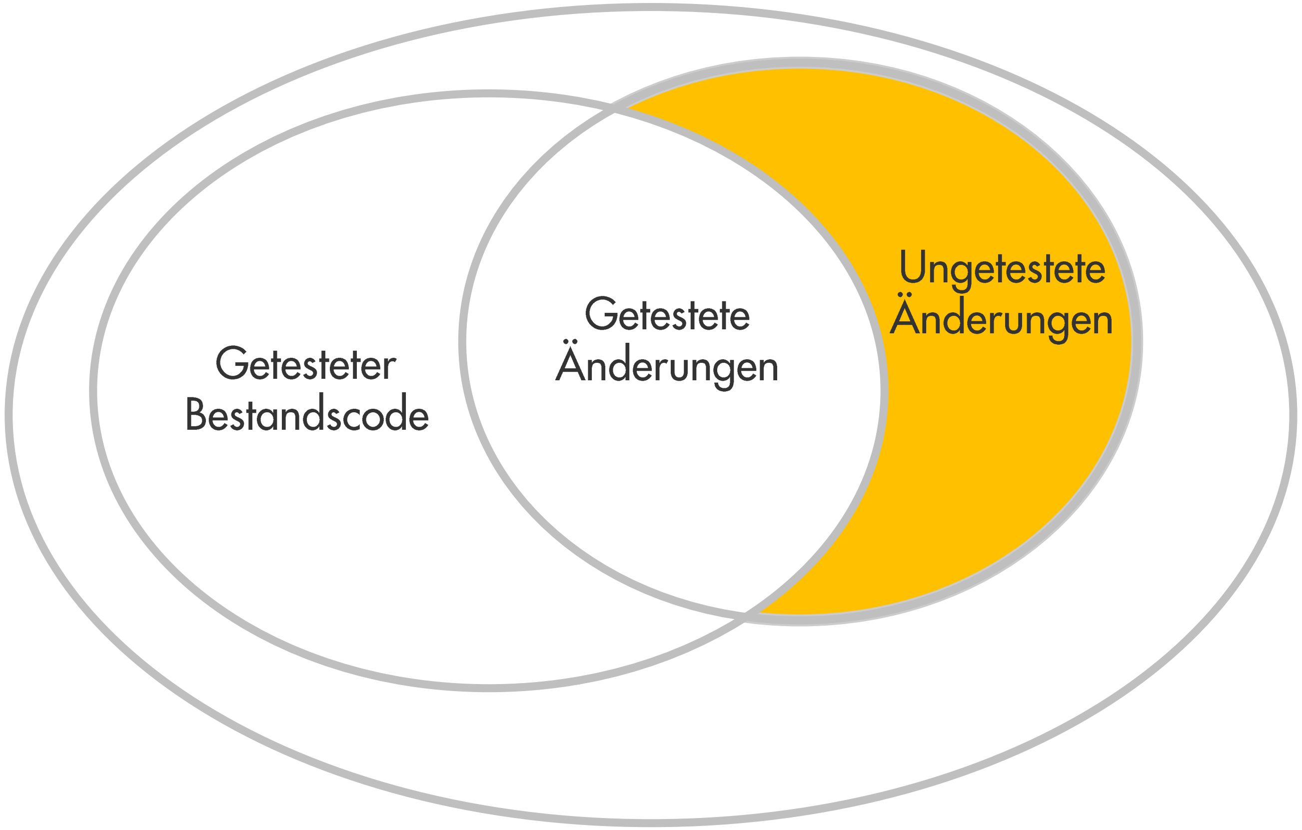
Paper published at Workshop on Automation of Software Test, 2013.

Wieviele Änderungen sind ungetestet?

Studie: C# System @ Munich Re

Release A:
15% Code neu/geändert,
>50% ungetestet

Release B:
15% Code neu/geändert,
>60% ungetestet

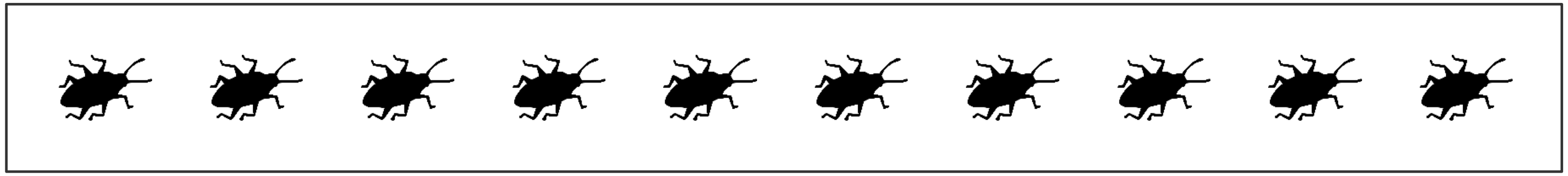


Feldfehlerwahrscheinlichkeit 5x höher für ungetestete Änderungen!

Eder, Jürgens, … *Did We Test Our Changes? Assessment btw. Tests & Development in Practice*, AST@ICSE 2013

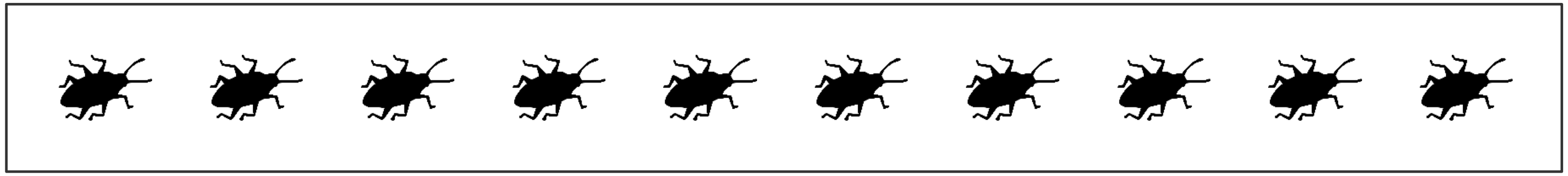
Ohne Test-Gap-Analyse

$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



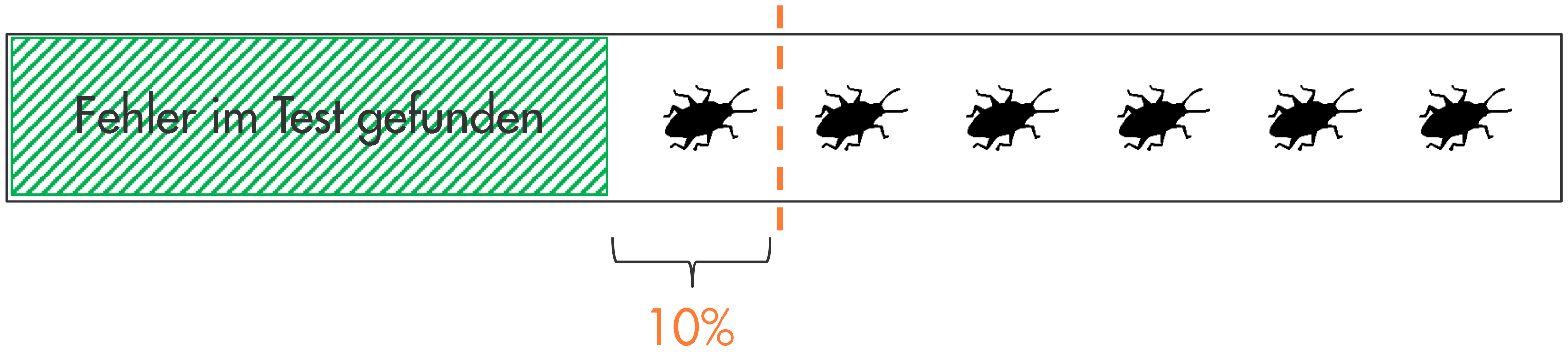
Ohne Test-Gap-Analyse

$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



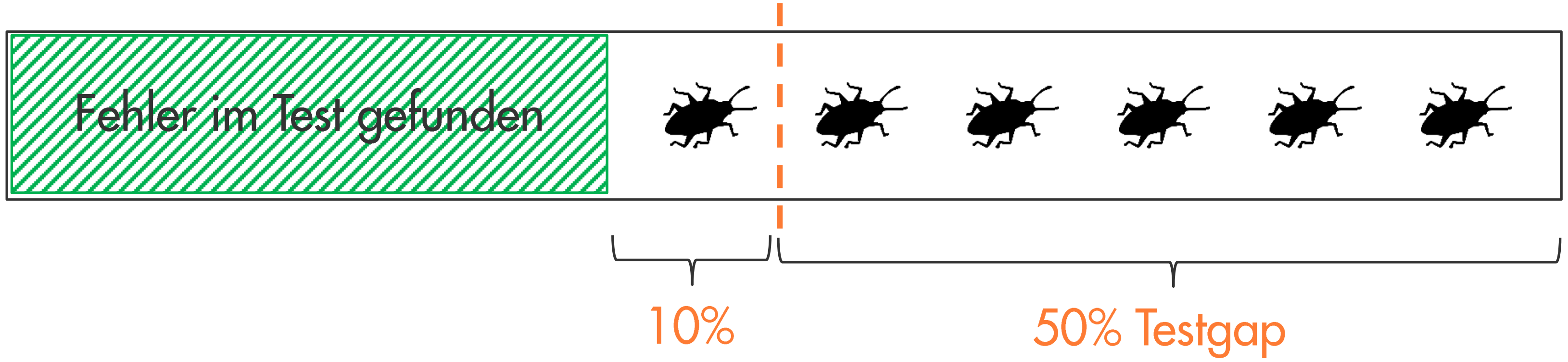
Ohne Test-Gap-Analyse

$$\%\text{Restfehler} = 10\% + \%\text{Testgap}$$



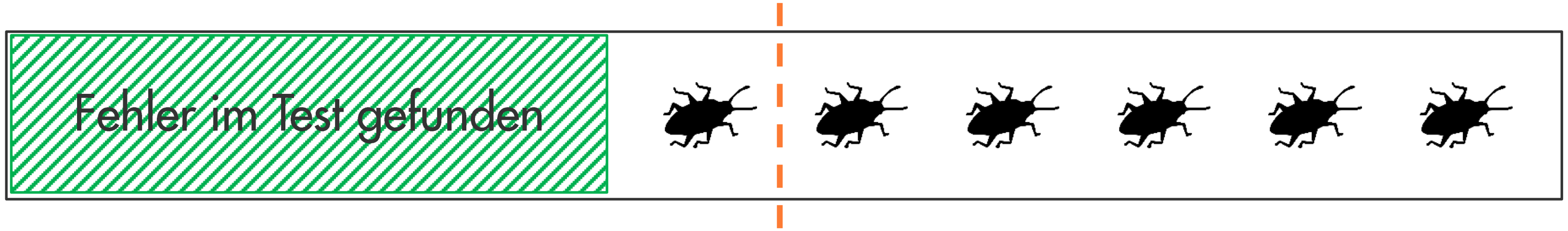
Ohne Test-Gap-Analyse

$$\%Restfehler = 10\% + 50\%$$



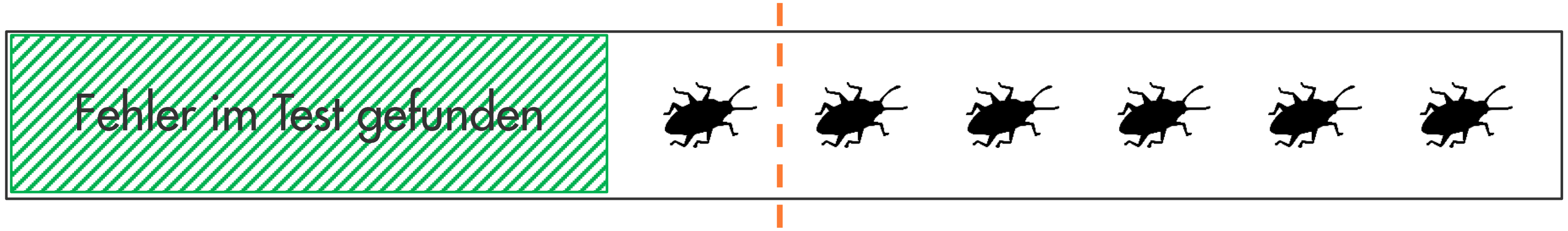
Ohne Test-Gap-Analyse

$$\%\text{Restfehler} = 60\%$$



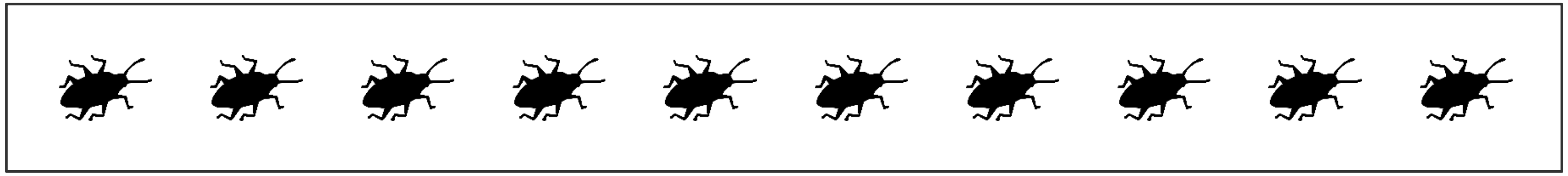
Ohne Test-Gap-Analyse

$$\%Restfehler = 60\%$$



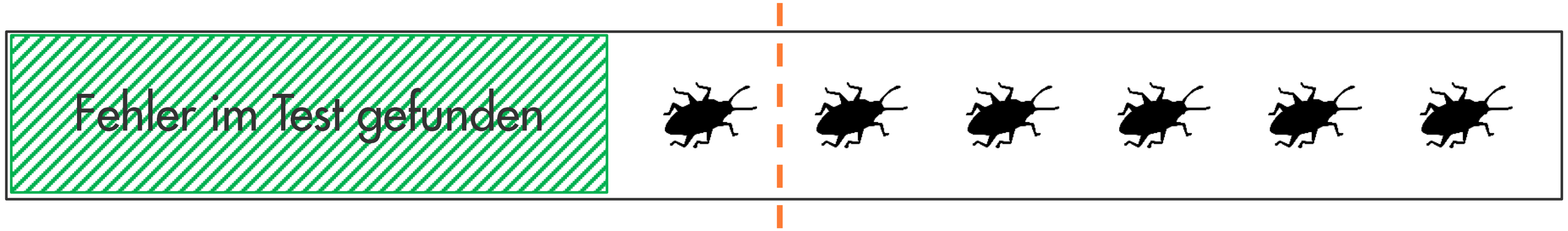
Mit Test-Gap-Analyse

$$\%Restfehler = \%Getestet * Testineffektivität + \%Testgap$$



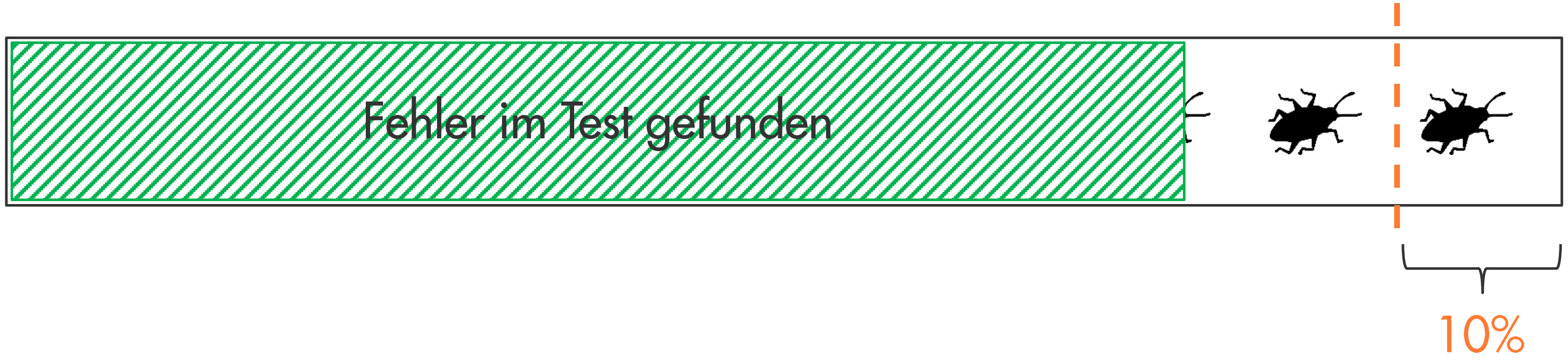
Ohne Test-Gap-Analyse

$$\%Restfehler = 60\%$$



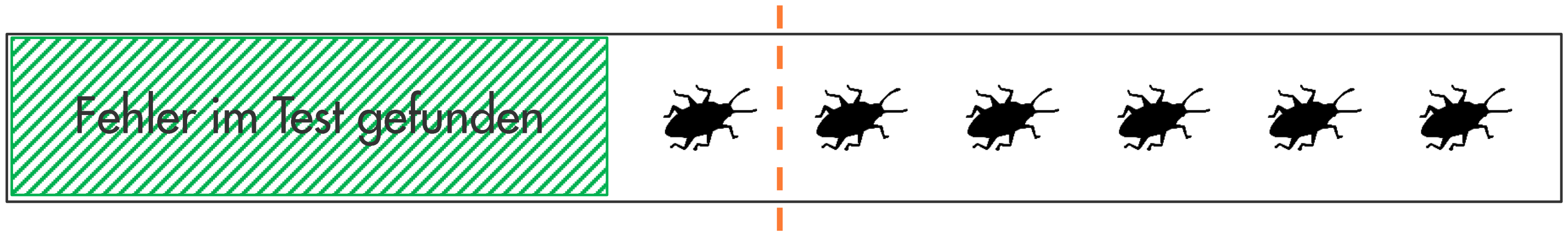
Mit Test-Gap-Analyse

$$\%Restfehler = 90\% * 20\% + 10\%$$



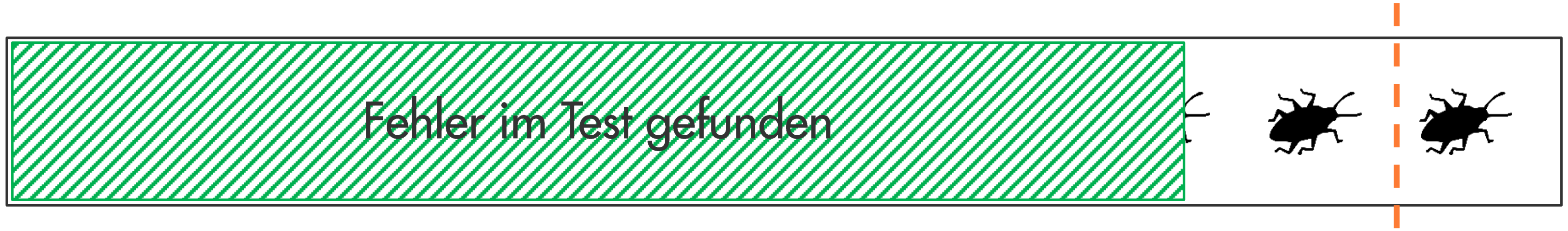
Ohne Test-Gap-Analyse

$$\%Restfehler = 60\%$$



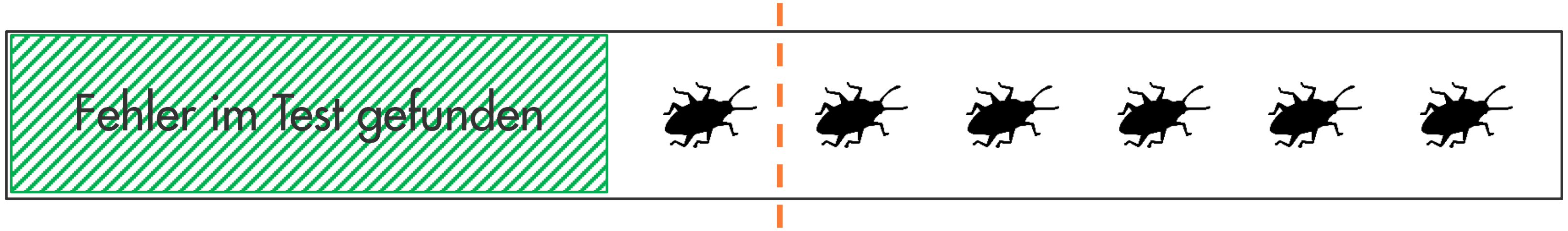
Mit Test-Gap-Analyse

$$\%Restfehler = 18\% + 10\%$$



Ohne Test-Gap-Analyse

%Restfehler = 60%



Mit Test-Gap-Analyse

%Restfehler = 28%

Reduzierte Feldfehler = **50%**

Reduzierte Feldfehler = **50%**

Test-Gap-Analyse reduziert Feldfehler in den Applikationen der Munich Re um ½

Fazit

- Conformance Costs << Costs of Non-Conformance

- Mit der Nutzenargumentation im Rücken konzentrieren uns auf umfassende Nutzung der Tools und Prozesse.

- Tools ***und*** Prozesse wichtig, etabliert und fest verankert.
- Internes Change Management („1/3“) notwendig.
- Sichtbarmachen von Qualität ist essentiell.

Vielen Dank für Ihre Teilnahme!

Feedback zum
Software Intelligence Talk 2020-1

cqse.eu/si-talks/1/feedback

Anmeldung zum
Software Intelligence Talk 2021-2
(Passwort: 20210519_SI)

cqse.eu/si-talks/2