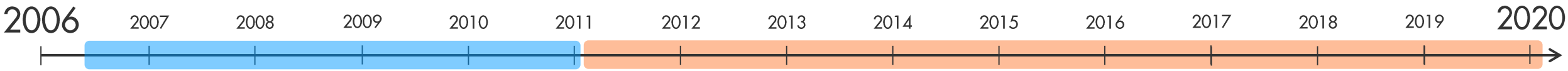
Vom Wiegen allein wird die Sau nicht fett

Erfahrungen aus einem Jahrzehnt Qualitätsanalyse in Forschung und Praxis

Dr. Elmar Jürgens

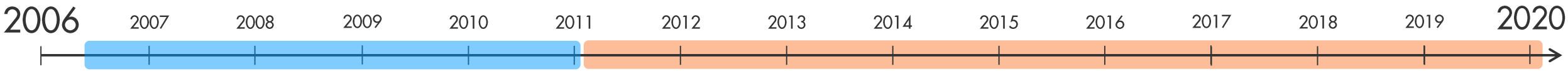
CQSE


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

- Clone Detection

- Clone Detection & Management
- Architektur-Konformitätsanalyse
- Data-Taint-Analyse
- Repository-Mining
- Test-Gap-Analyse

- Qualitäts-Audits
- Quality Control



CQSE

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

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

ICSE 2009

Studie

Munich Re LV 1871

- Über 100 Fehler in produktiver Software

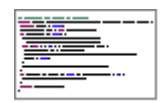
- 52% aller ungewollten Unterschiede fehlerhaft

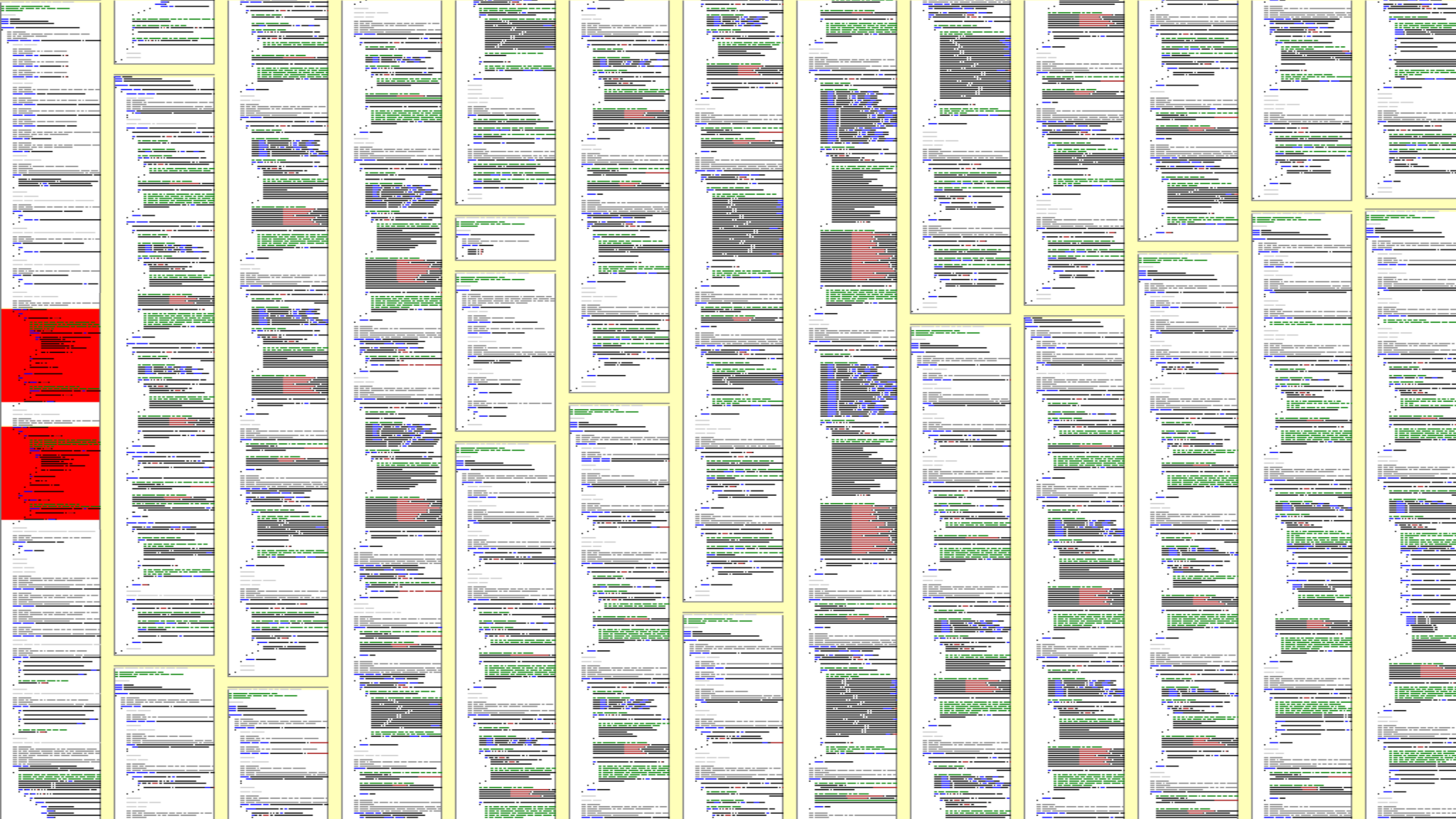
Juergens, Deissenboeck et al: *Do Code Clones Matter?* ICSE 2009

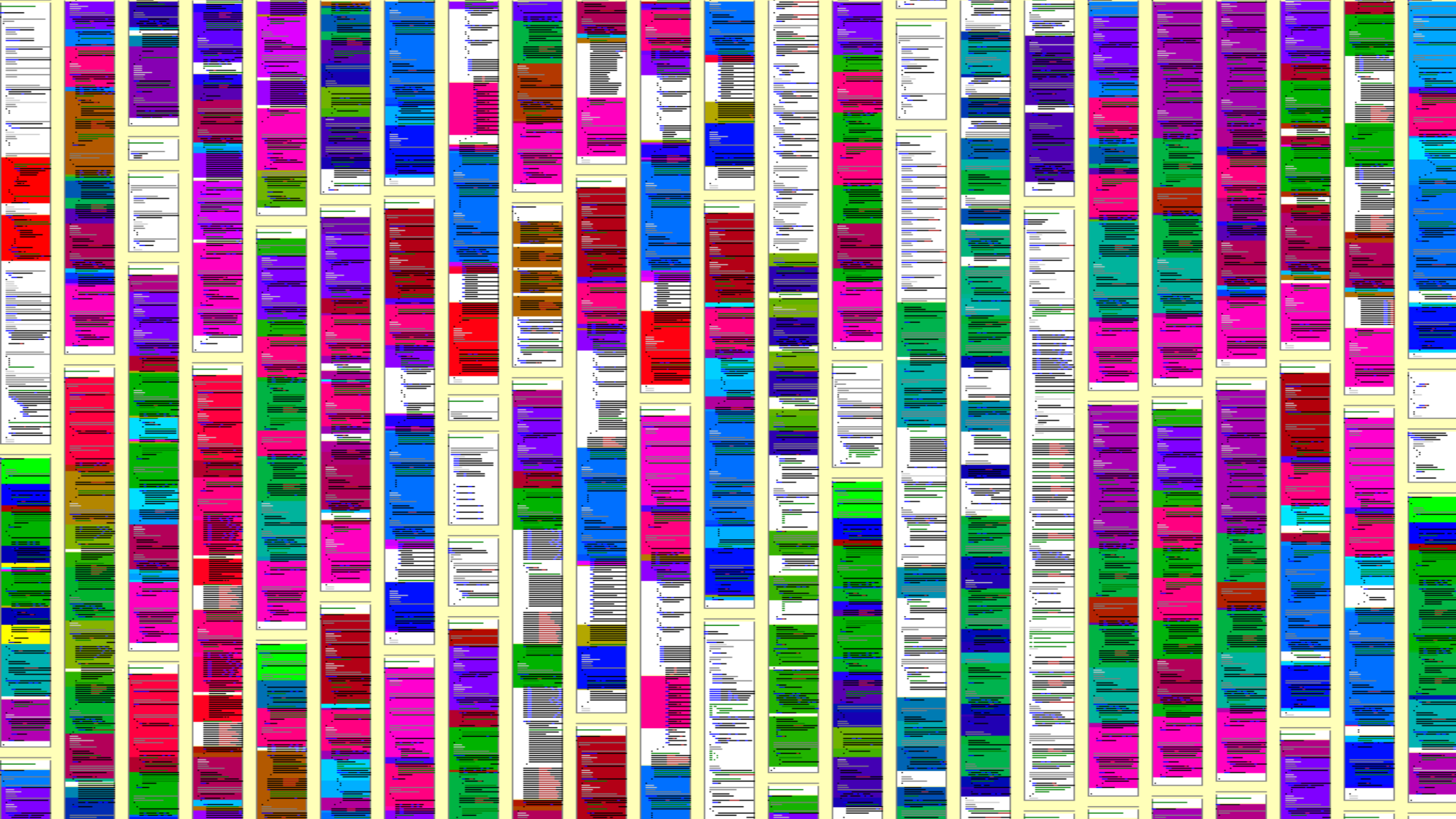
```java
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
   boolean found = false;
   StringBuffer res = new StringBuffer();
   if (elements != null) {
      Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());
      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();
}
```







Dashboard
Activity
Findings
Metrics
Tests
Issues
Tasks
Architecture
Delta
Projects
System
Admin

jenkins/test/src/main/java/org/jvnet/hudson/test/HudsonTestCase.java (revision 12d96a56...)

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

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
  assertEquals(lhs.size(), rhs.size());
```

jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.java (revis)

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

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
```

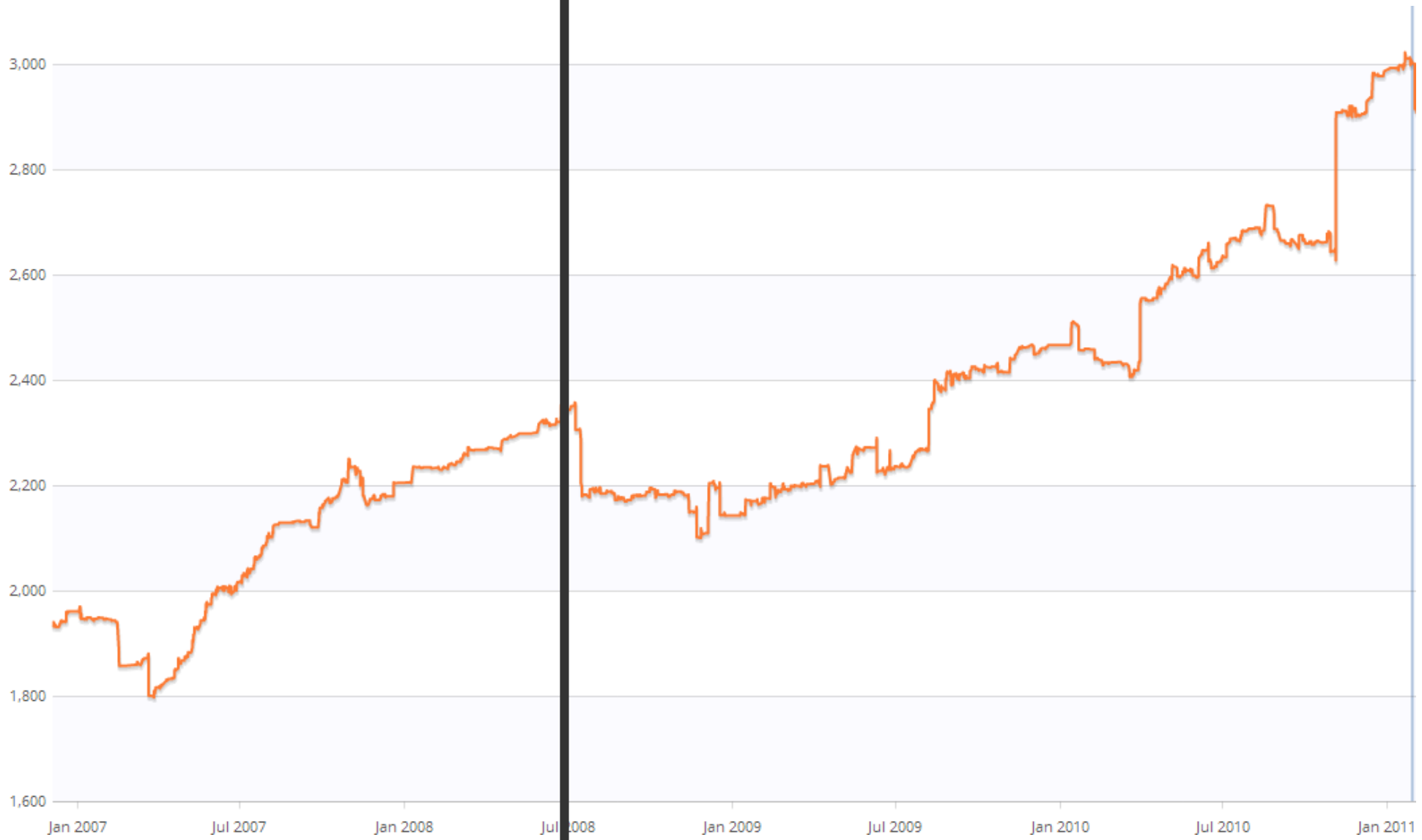
Collapse

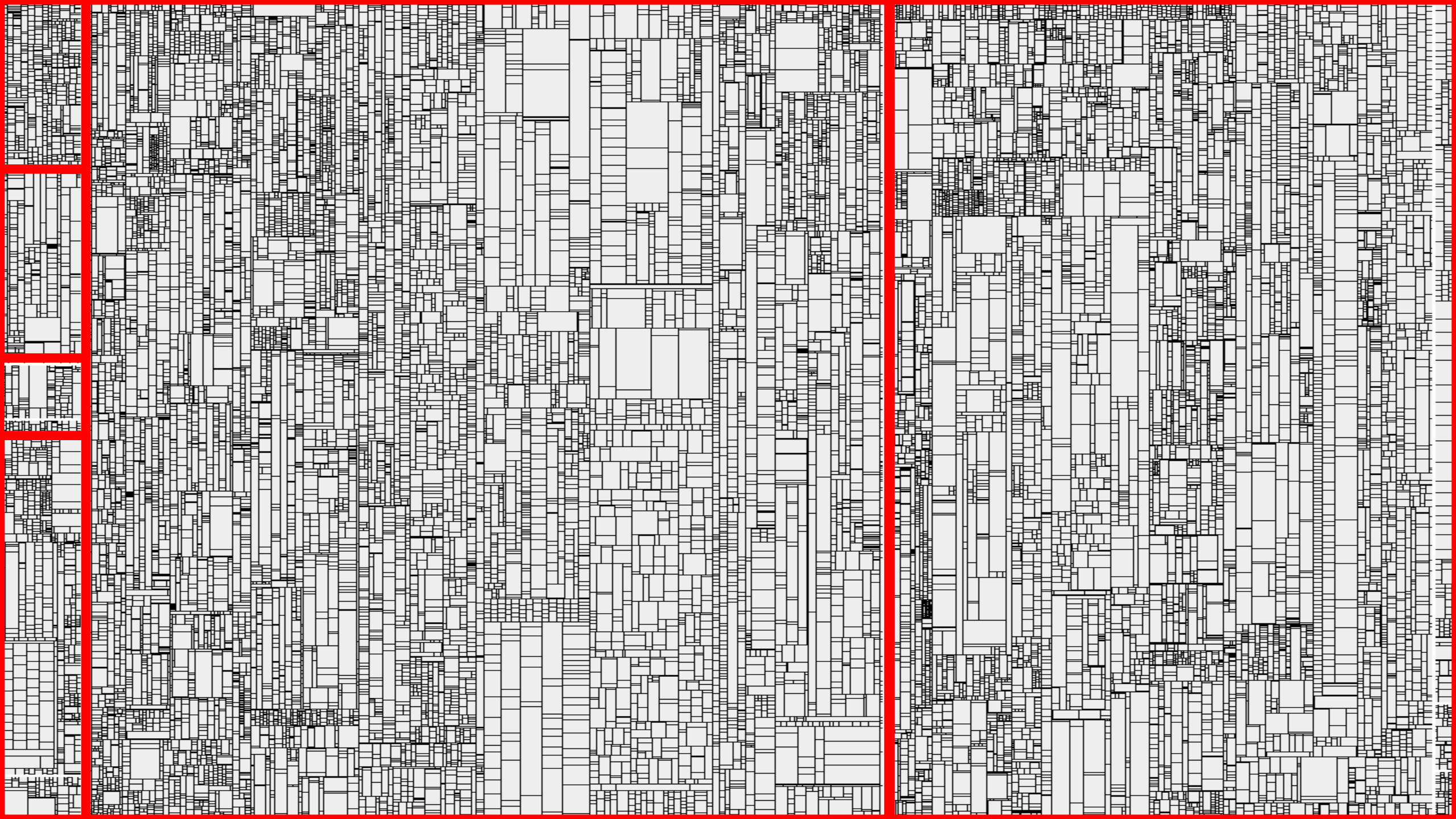
C/C++

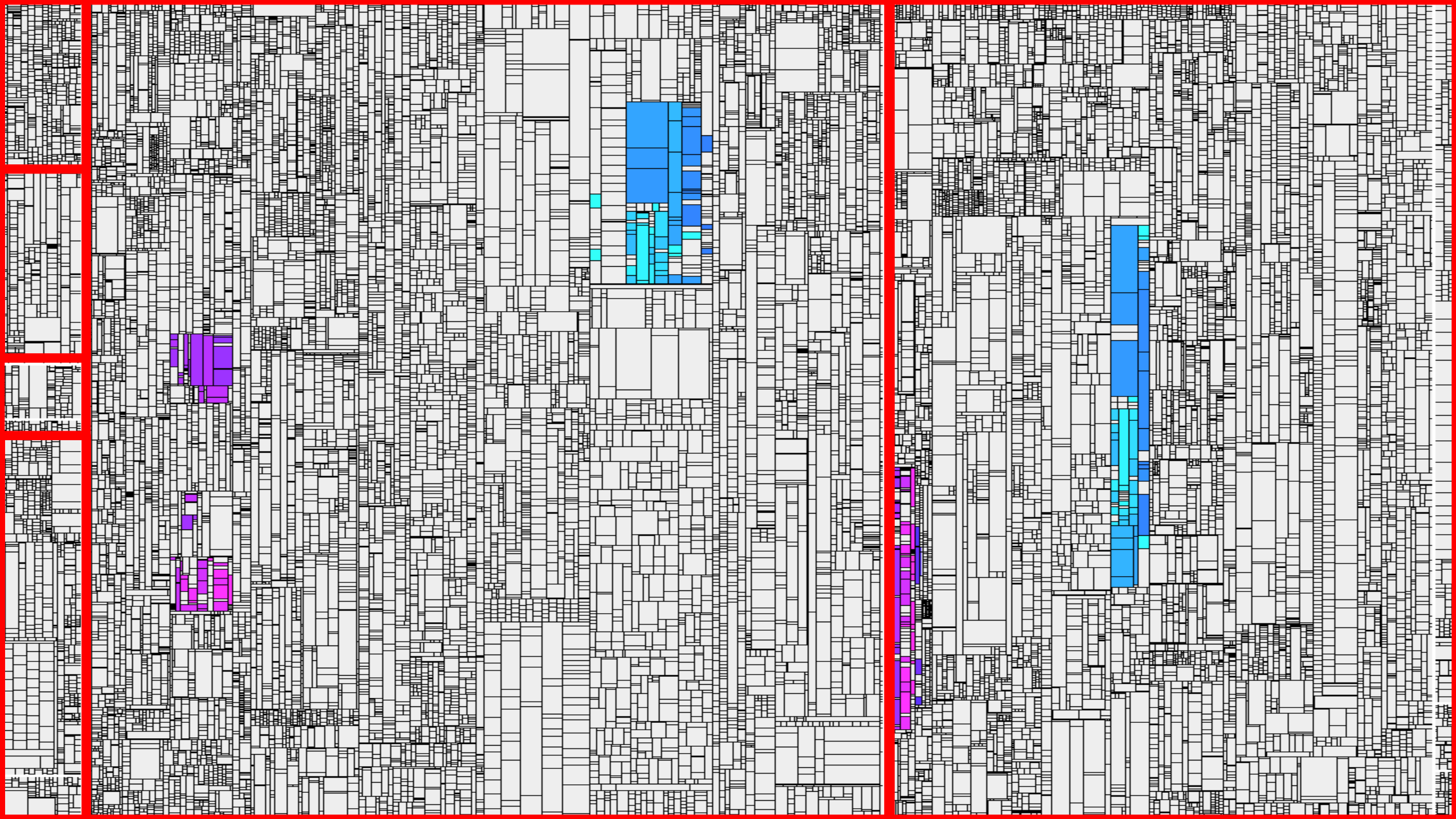
```
    /* Process the input string received prior to the
    newline. */
    do
    {
        /* Pass the string to FreeRTOS+CLI. */
        xMoreDataToFollow = FreeRTOS_CLIProcessCommand( cInputString, cO

        /* Send the output generated by the command's
        implementation. */
        sendto( xSocket, cOutputString,  strlen( cOutputString ), 0, ( S

    } while( xMoreDataToFollow != pdFALSE ); /* Until the command does n

    /* All the strings generated by the command processing
    have been sent.  Clear the input string ready to receive
    the next command. */
    cInputIndex = 0;
    memset( cInputString, 0x00, cmdMAX_INPUT_SIZE );

    /* Transmit a spacer, just to make the command console
    easier to read. */
    sendto( xSocket, "\r\n",  strlen( "\r\n" ), 0, ( SOCKADDR * ) &xClie
}
else
{
    if( cInChar == '\r' )
    {
        /* Ignore the character.  Newlines are used to
        detect the end of the input string. */
    }
    else if( cInChar == '\b' )
    {
        /* Backspace was pressed.  Erase the last character
        in the string - if any. */
        if( cInputIndex > 0 )
        {
            cInputIndex--;
            cInputString[ cInputIndex ] = '\0';
        }
    }
    else
    {
```

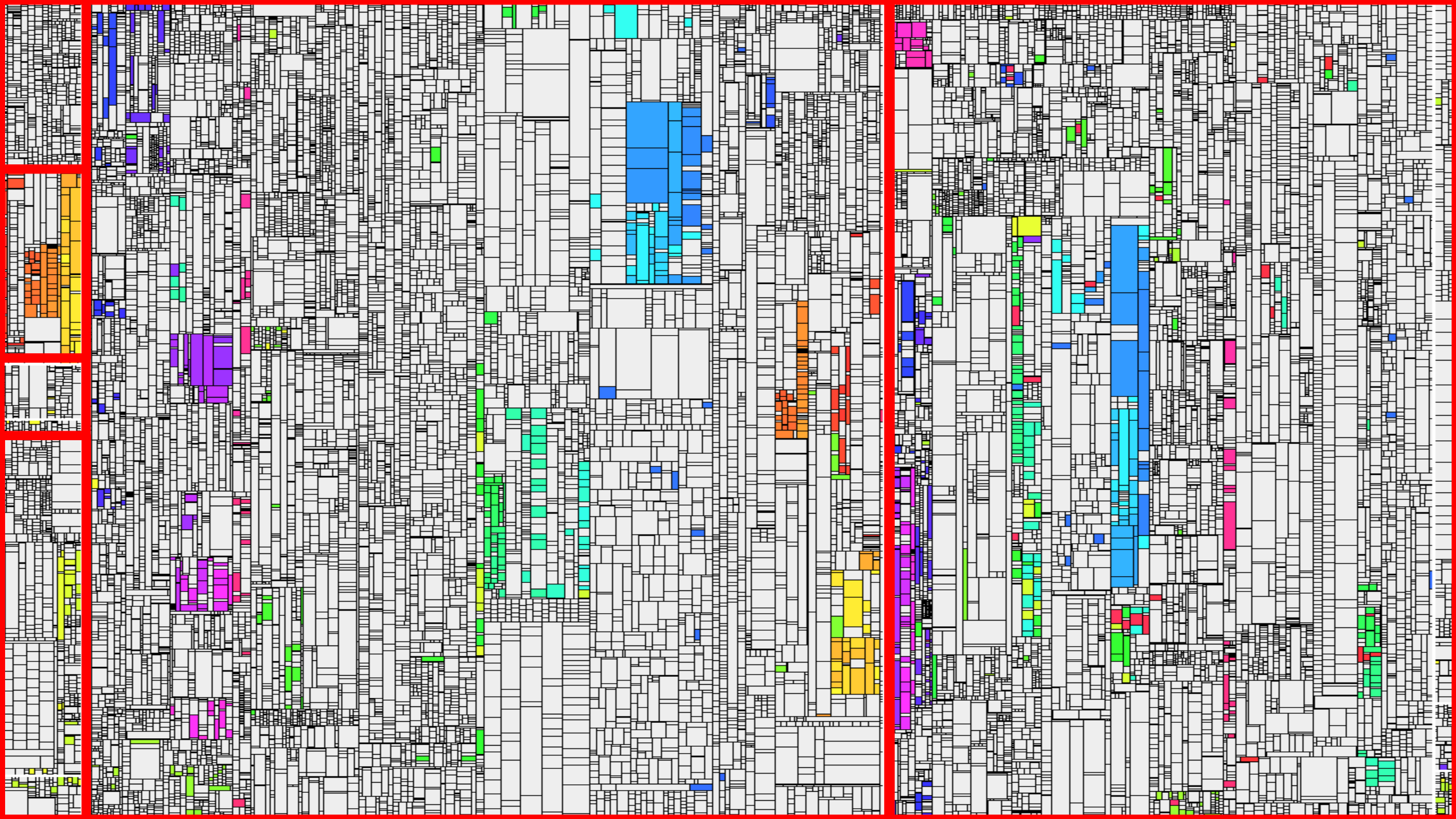
```
    /* Process the input string received prior to the
    newline. */
    do
    {
        /* Pass the string to FreeRTOS+CLI. */
        xMoreDataToFollow = FreeRTOS_CLIProcessCommand( cInputString,

        /* Send the output generated by the command's
        implementation. */
        sendto( xSocket, cOutputString,  strlen( cOutputString ), 0,

    } while( xMoreDataToFollow != pdFALSE ); /* Until the command doe

    /* All the strings generated by the command processing
    have been sent.  Clear the input string ready to receive
    the next command. */
    cInputIndex = 0;
    memset( cInputString, 0x00, cmdMAX_INPUT_SIZE );

    /* Transmit a spacer, just to make the command console
    easier to read. */
    sendto( xSocket, "\r\n",  strlen( "\r\n" ), 0, ( SOCKADDR * ) &xC
}
else
{
    if( cInChar == '\r' )
    {
        /* Ignore the character.  Newlines are used to
        detect the end of the input string. */
    }
    else if( ( cInChar == '\b' ) || ( cInChar == cmdASCII_DEL ) )
    {
        /* Backspace was pressed.  Erase the last character
        in the string - if any. */
        if( cInputIndex > 0 )
        {
            cInputIndex--;
            cInputString[ cInputIndex ] = '\0';
        }
    }
    else
    {
```

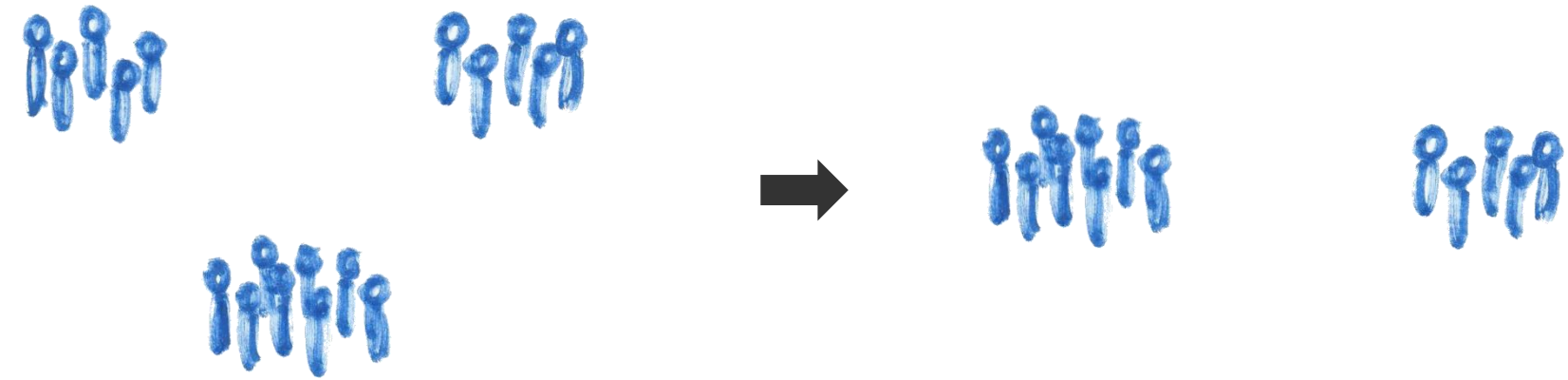
2006 2007 2008 2009 2010 2011 2012 2013 2014 2015 2016 2017 2018 2019 2020

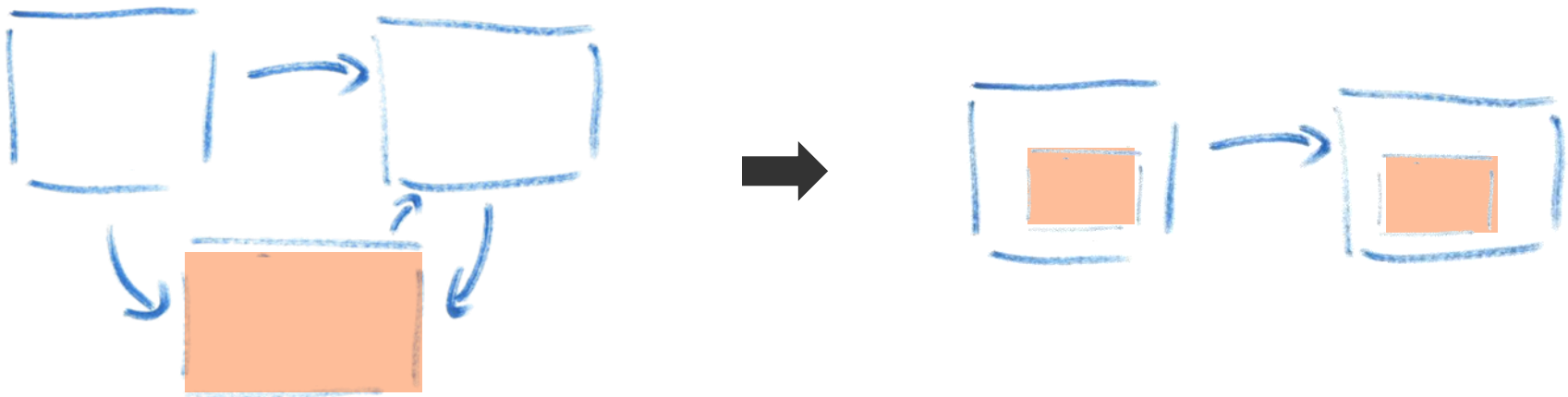

3,000
2,800
2,600
2,400
2,200
2,000
1,800
1,600
Jan 2007
Jul 2007
Jan 2008
Jul 2008
Jan 2009
Jul 2009
Jan 2010
Jul 2010
Jan 2011

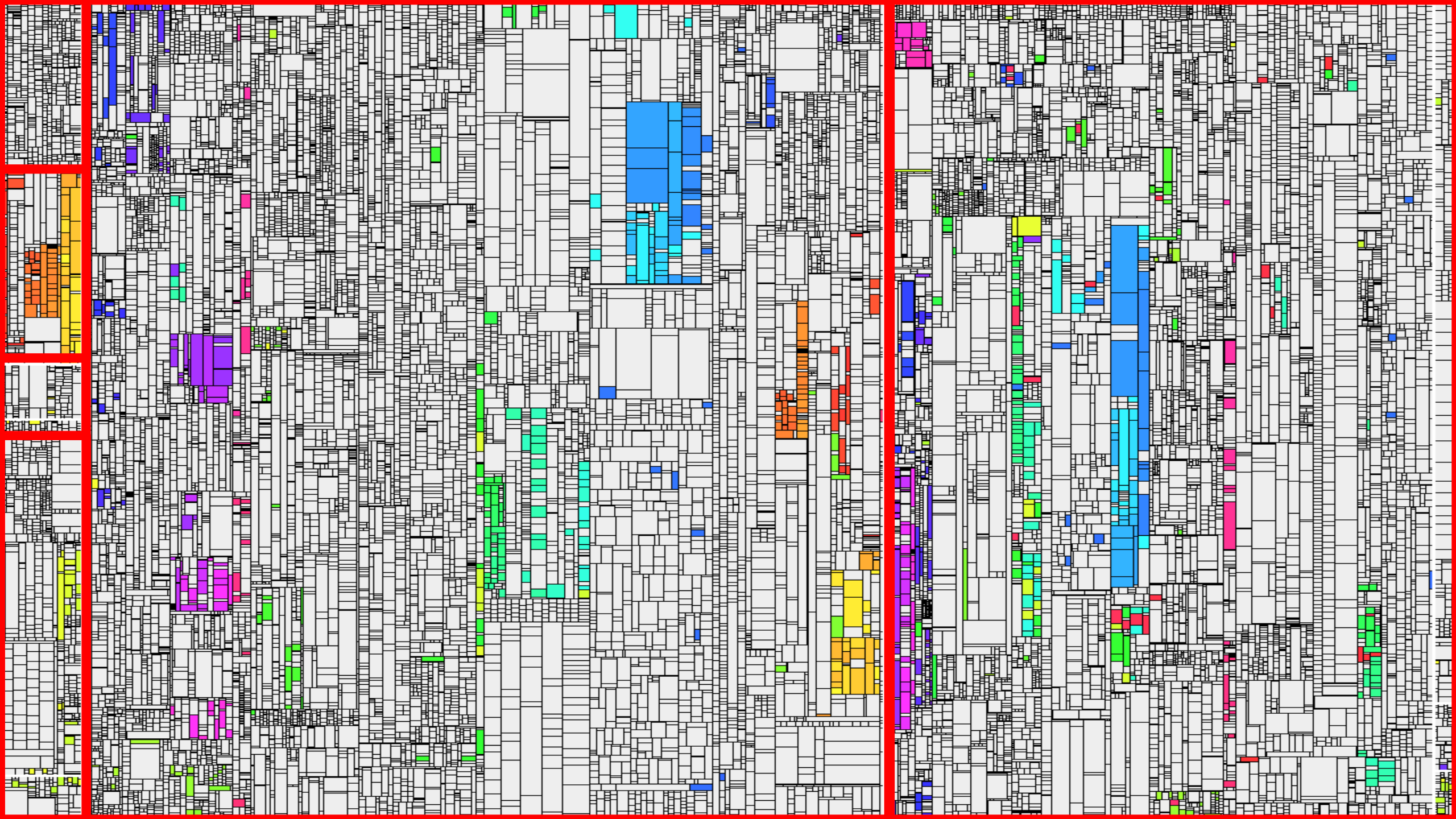


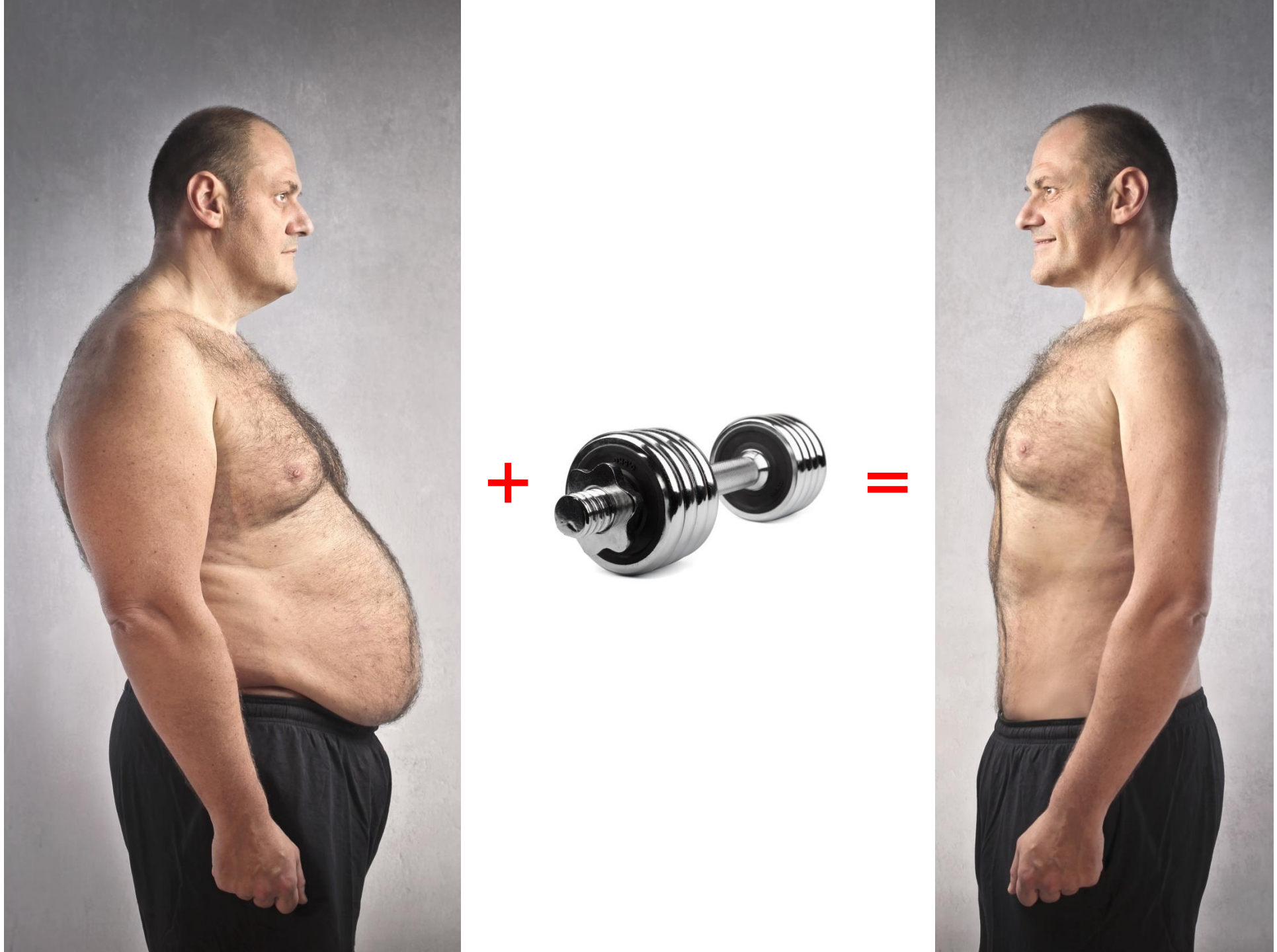




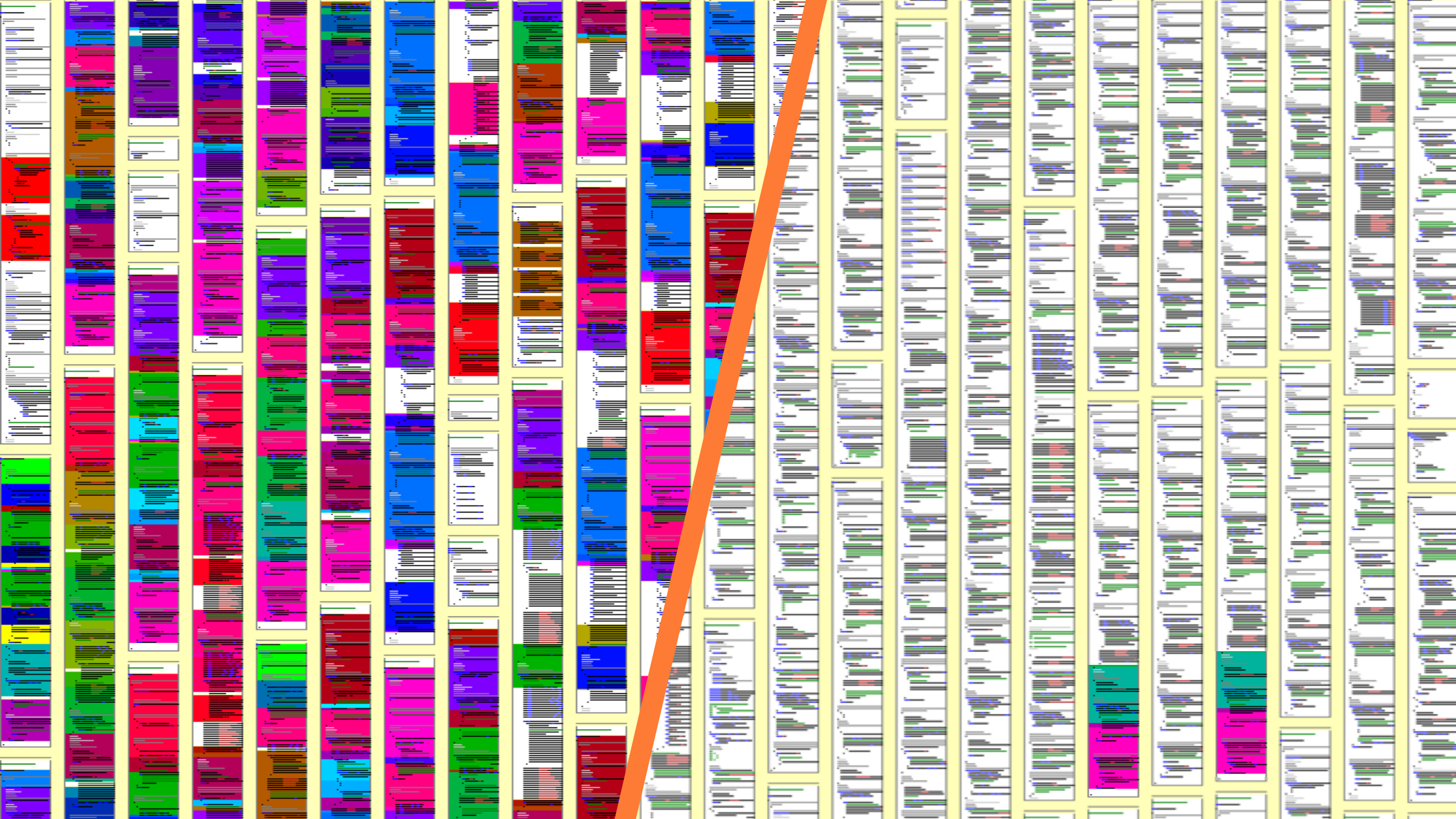


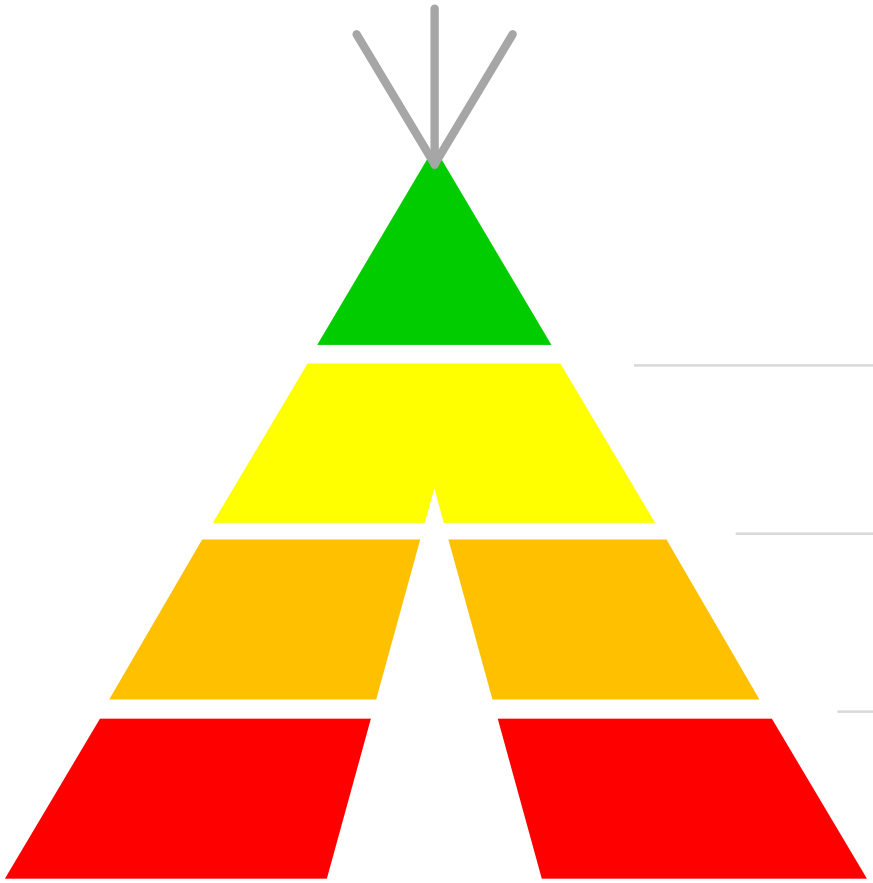







soft cloths
soft cloths
BANKERS BOX
33250
33250
SOLE F80



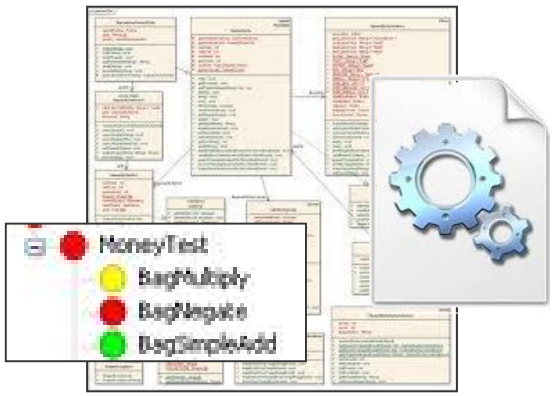


Keine Defizite

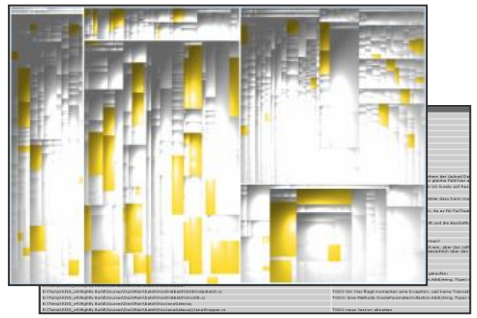
Keine Defizite in geändertem Code

Keine neuen Defizite

Egal

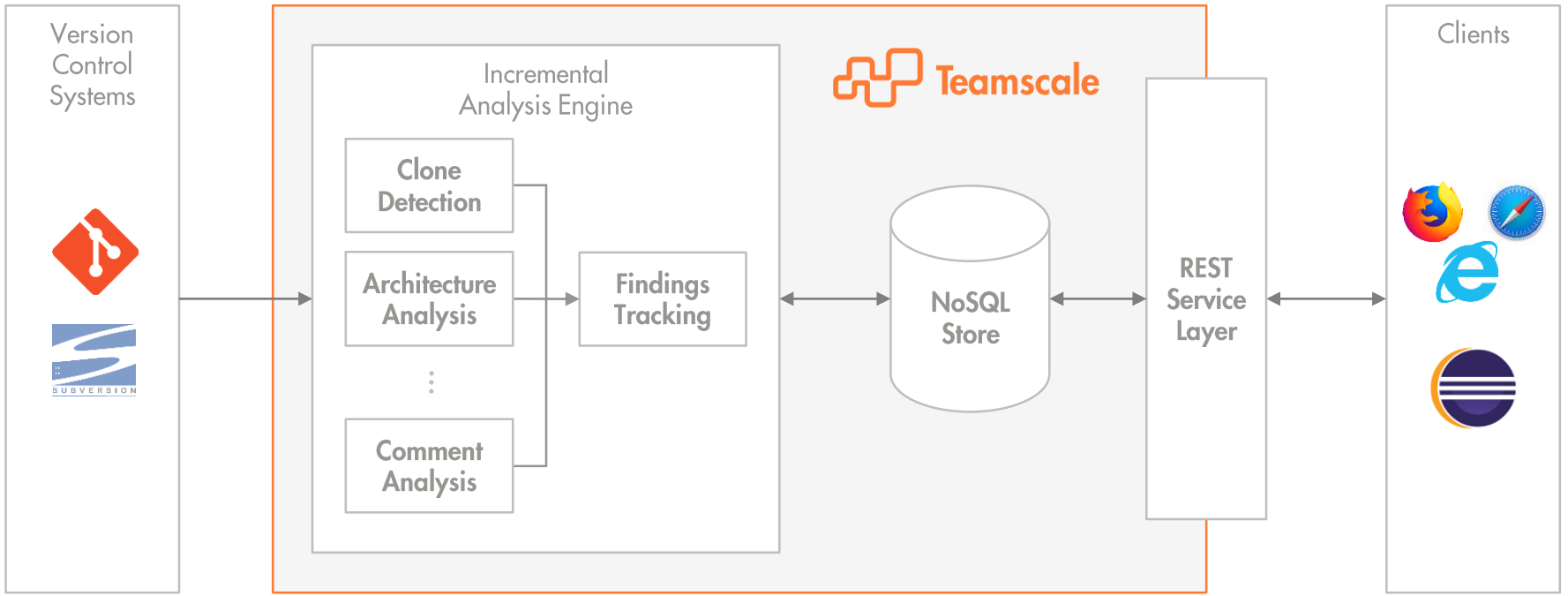


conQAT




Baseli
MoneyTest
BagMultiply
BagNegate
BagSimpleAdd
Latest V
MoneyTest
aseli
oard
nboard
+ Delta
Laufzeit?
Findings-Tracking?
Feature-Branches?
Config Änderungen?
...?

ICSM 2012


Version Control Systems
SUBVERSION
Incremental Analysis Engine
Clone Detection
Architecture Analysis
Comment Analysis
Findings Tracking
Teamscale
NoSQL Store
REST Service Layer
Clients

Teamscale | Dashboard | Activity | Findings | Metrics | Tests | Issues | Tasks | Audit | Architecture | Delta | More | CQSE All

Project: CQSE All **Branch:** master **Timetravel:** All Commits.

CR#14434 Fix Artifactory test
by Andi Scharfstein in revision a32daf8a in branch cr/14434_unify_default_branch (Gitlab)
Files: 2 changed
Issue: TS-14434
Findings: 0

Mar 13 2018
14:15

Merge branch 'teamscale/v4.1.x'
by Lars Heinemann in revision c4f01a6e in branch master (Gitlab)
Merged from cr/14098_connection_profile_branch 3
Files: 1 added, 20 changed
Findings: 0

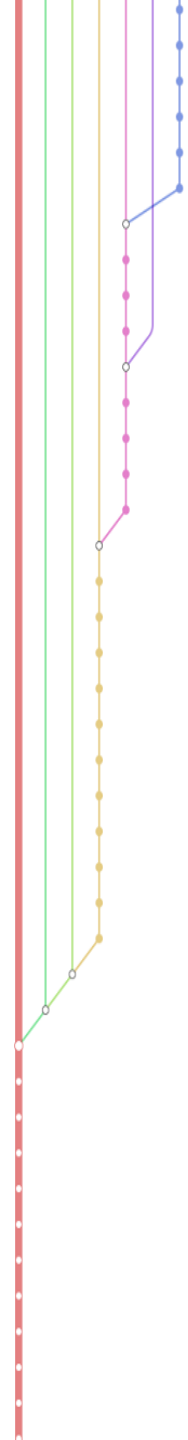
Mar 13 2018
14:08

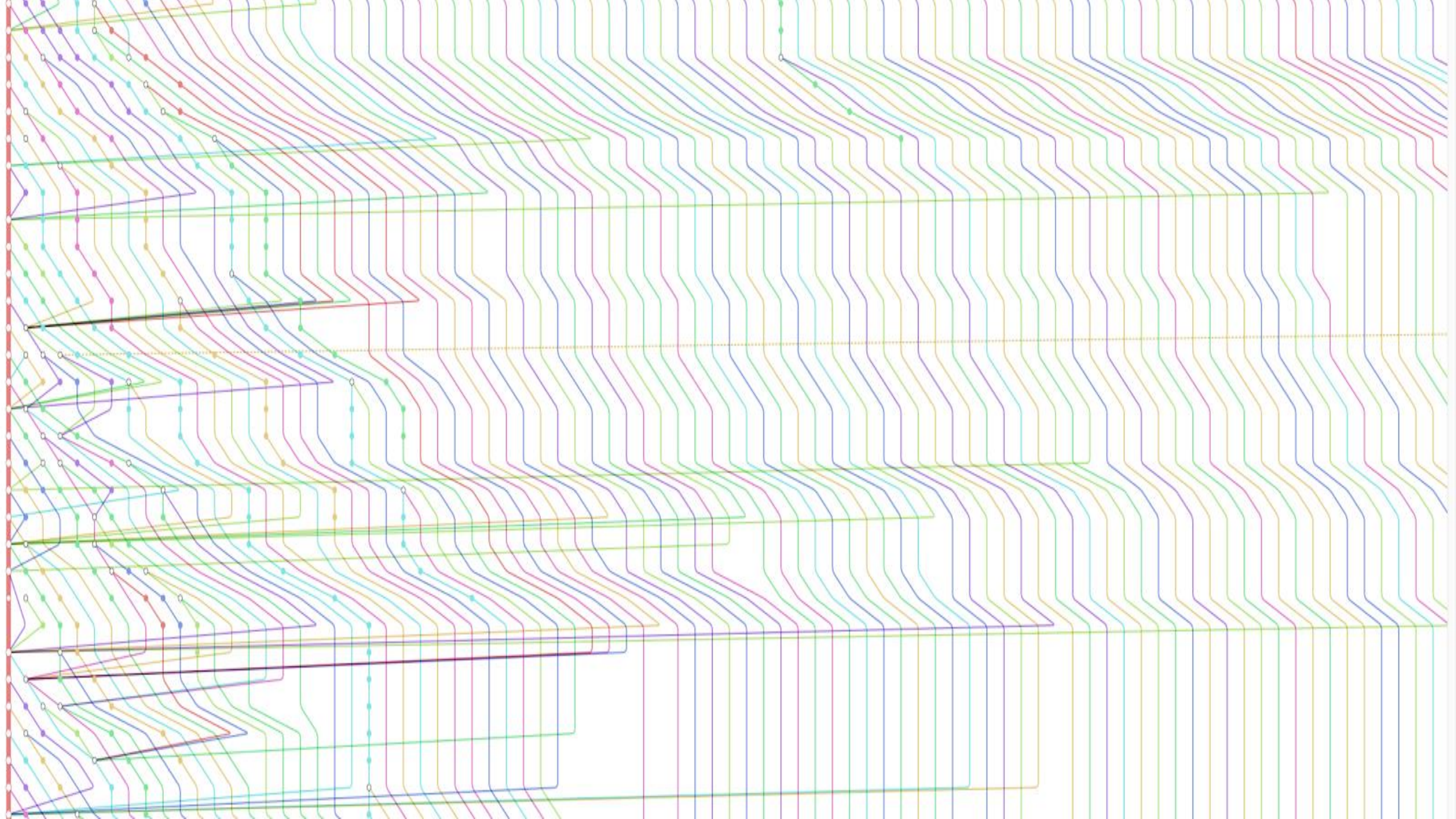
CR#14434 Fix test cases
by Andi Scharfstein in revision 2aa45de3 in branch cr/14434_unify_default_branch (Gitlab)
Files: 6 changed
Issue: TS-14434
Findings: 1

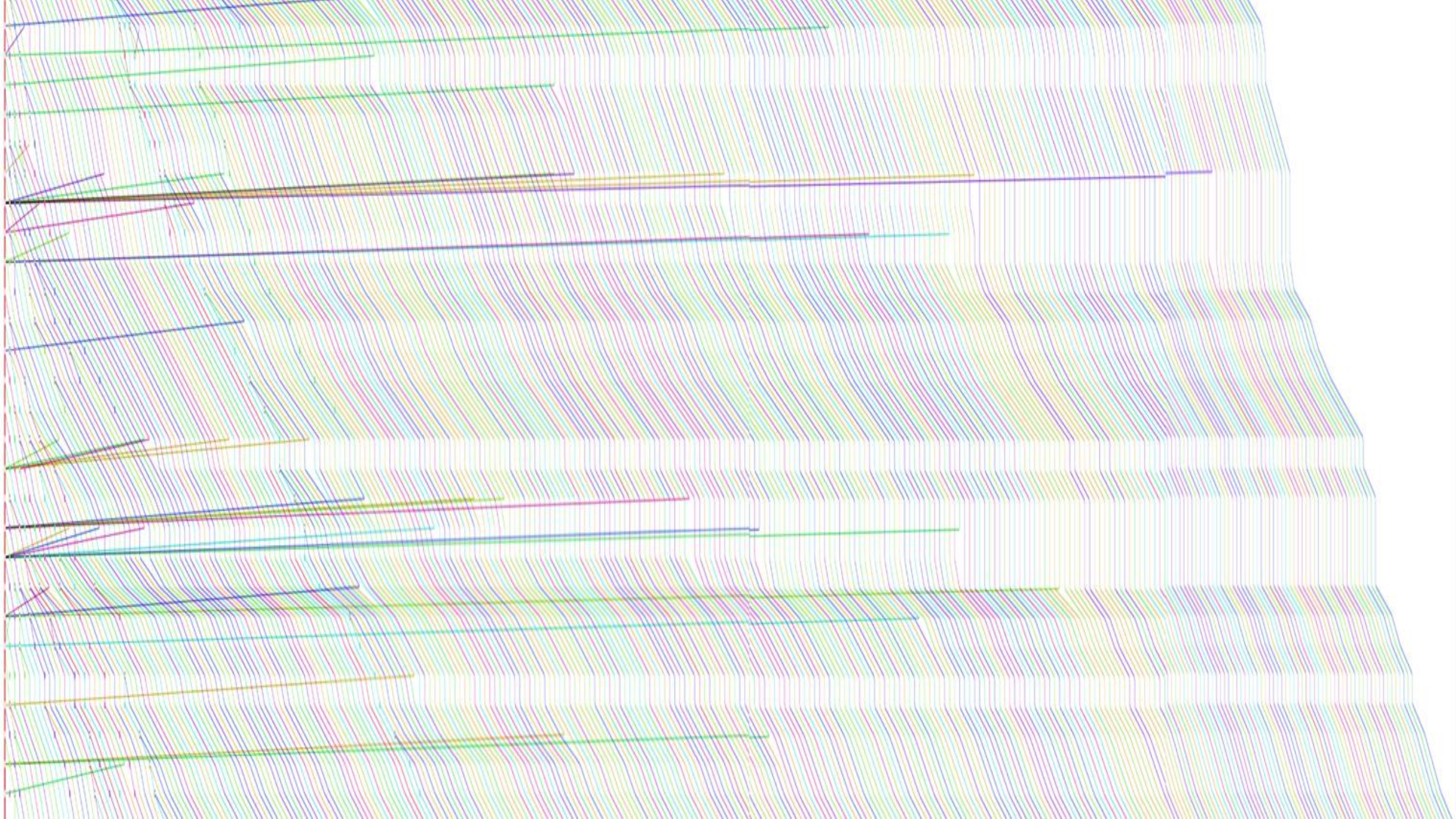
Mar 13 2018
14:00

CR#14448: disable flickering test
by Lars Heinemann in revision 47f12dde in branch cr/14098_connection_profile_branch (Gitlab)
Files: 1 changed
Issue: TS-14448
Findings: 1

Mar 13 2018
13:28







Teamscale | Dashboard | Activity | Findings | Metrics | Tests | Issues | Tasks | Audit | Architecture | Delta | More | CQSE All

Project: CQSE All **Branch:** master **Timetravel:** All Commits.

CR#14434 Fix Artifactory test
by Andi Scharfstein in revision a32daf8a in branch cr/14434_unify_default_branch (Gitlab)
Files: 2 changed
Issue: TS-14434
Findings: 0

Mar 13 2018
14:15

Merge branch 'teamscale/v4.1.x'
by Lars Heinemann in revision c4f01a6e in branch master (Gitlab)
Merged from cr/14098_connection_profile_branch 3
Files: 1 added, 20 changed
Findings: 0

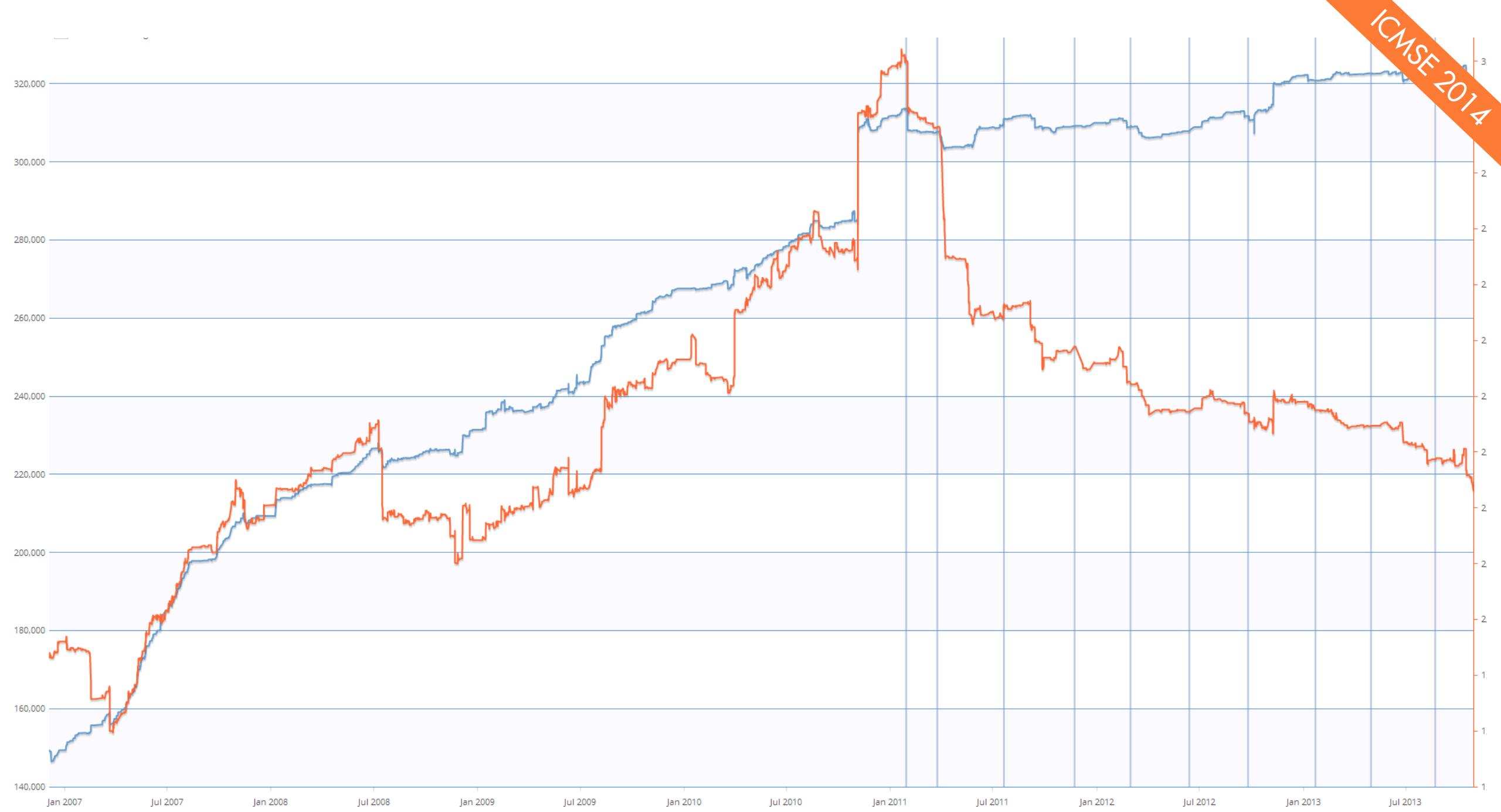
Mar 13 2018
14:08

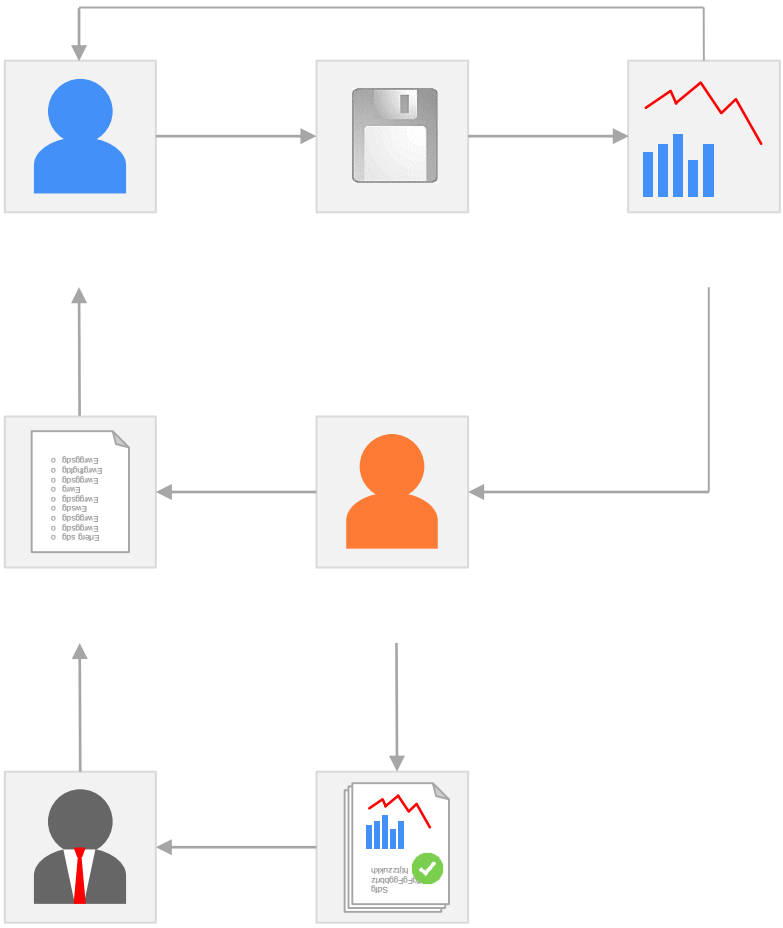
CR#14434 Fix test cases
by Andi Scharfstein in revision 2aa45de3 in branch cr/14434_unify_default_branch (Gitlab)
Files: 6 changed
Issue: TS-14434
Findings: 1

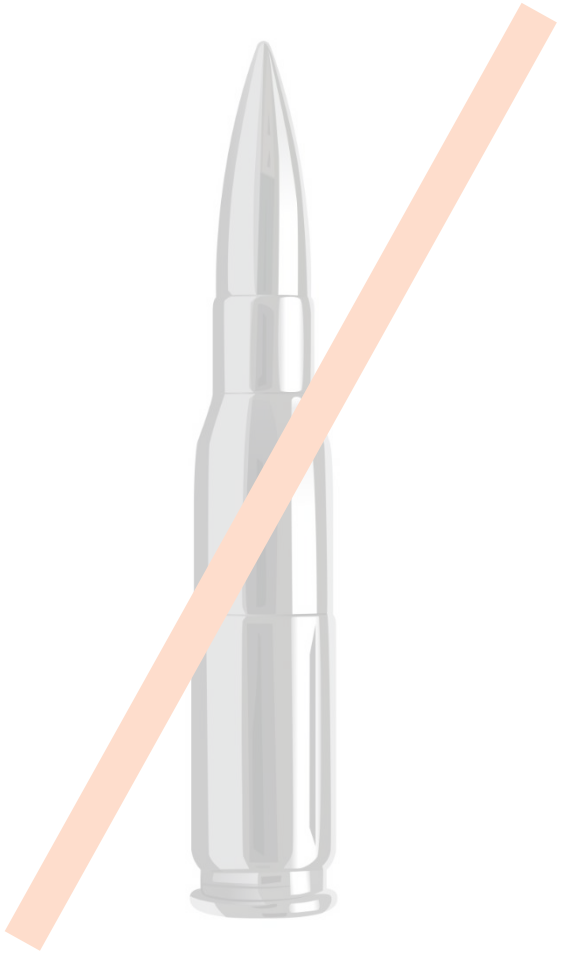
Mar 13 2018
14:00

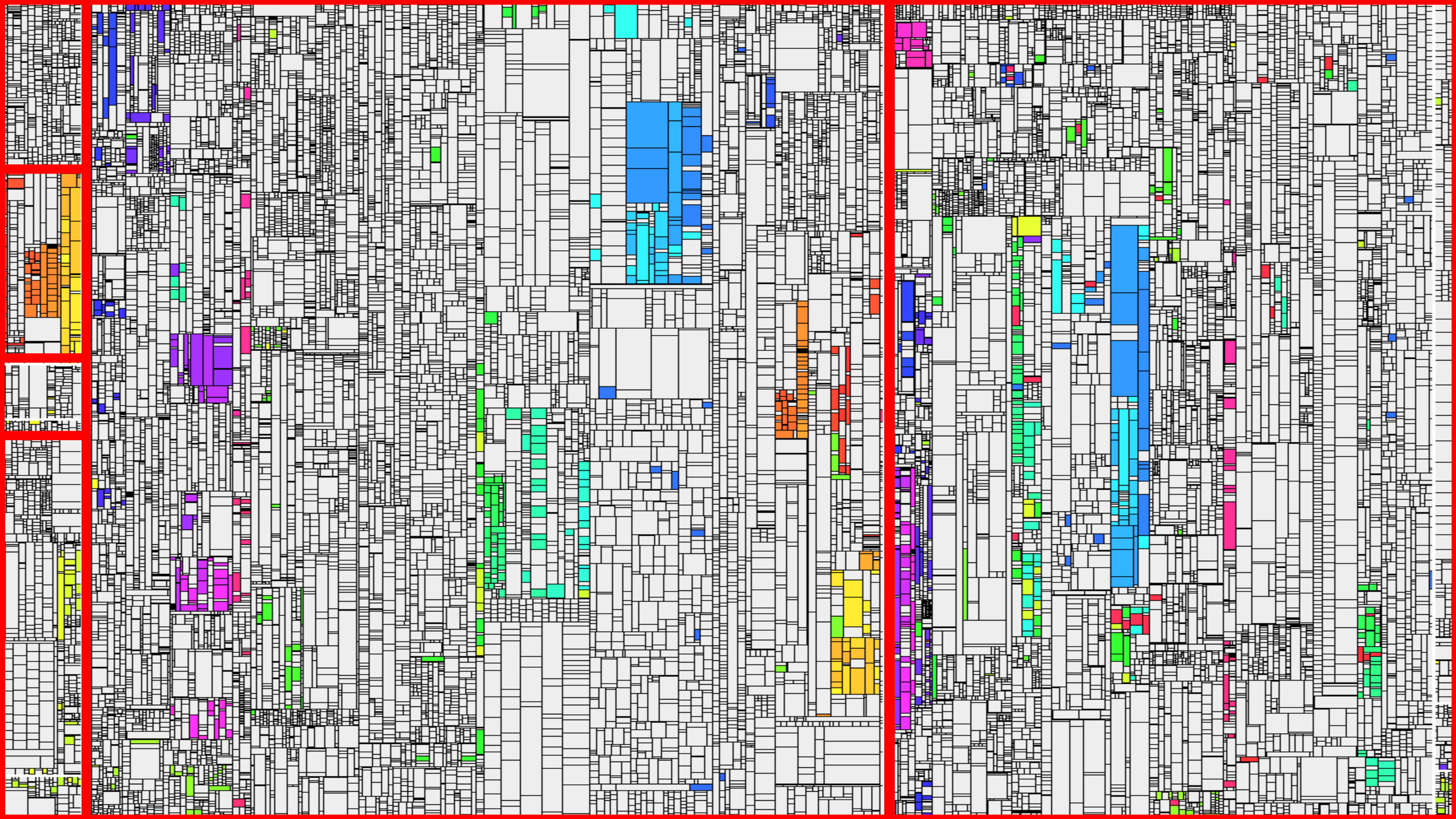
CR#14448: disable flickering test
by Lars Heinemann in revision 47f12dde in branch cr/14098_connection_profile_branch (Gitlab)
Files: 1 changed
Issue: TS-14448
Findings: 1

Mar 13 2018
13:28


ICMSE 2014
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







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

Dashboard
Activity
Findings
Metrics
Tests
Issues
Tasks
Architecture
Delta
Projects
System
Admin

Collapse

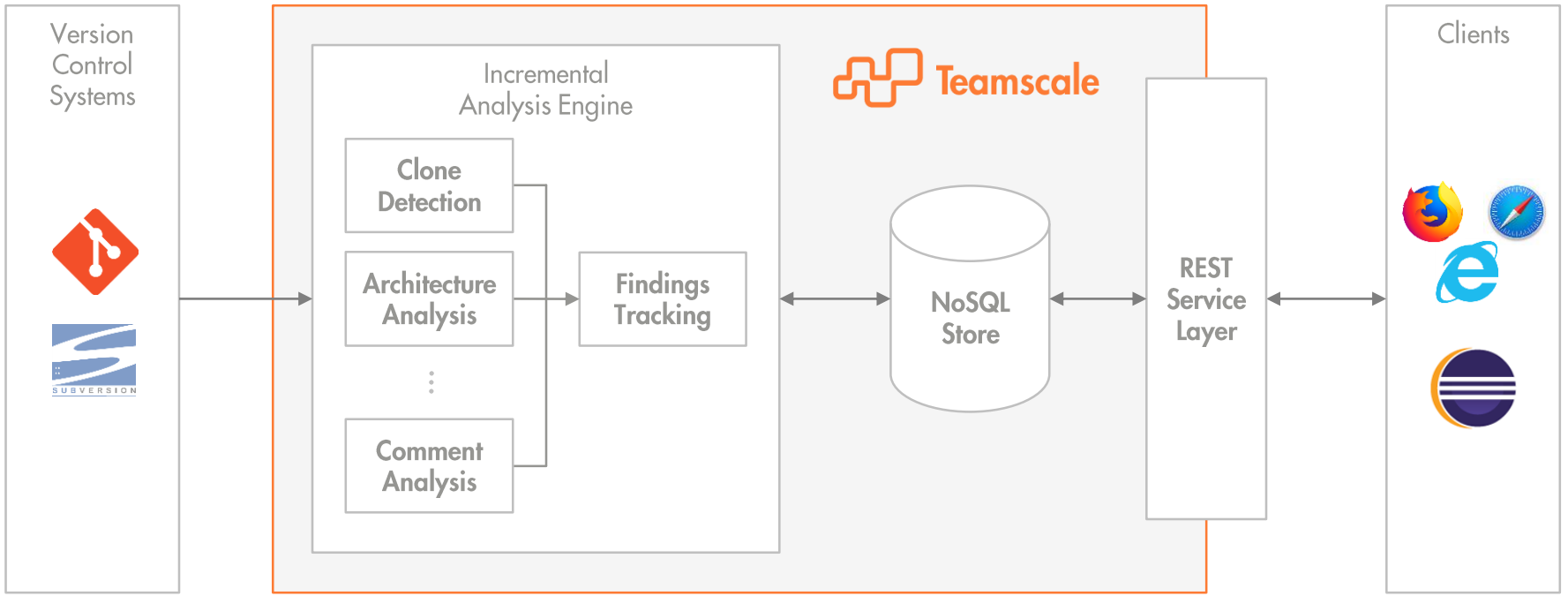
jenkins/test/src/main/java/org/jvnet/hudson/test/HudsonTestCase.java (revision 12d96a56...)

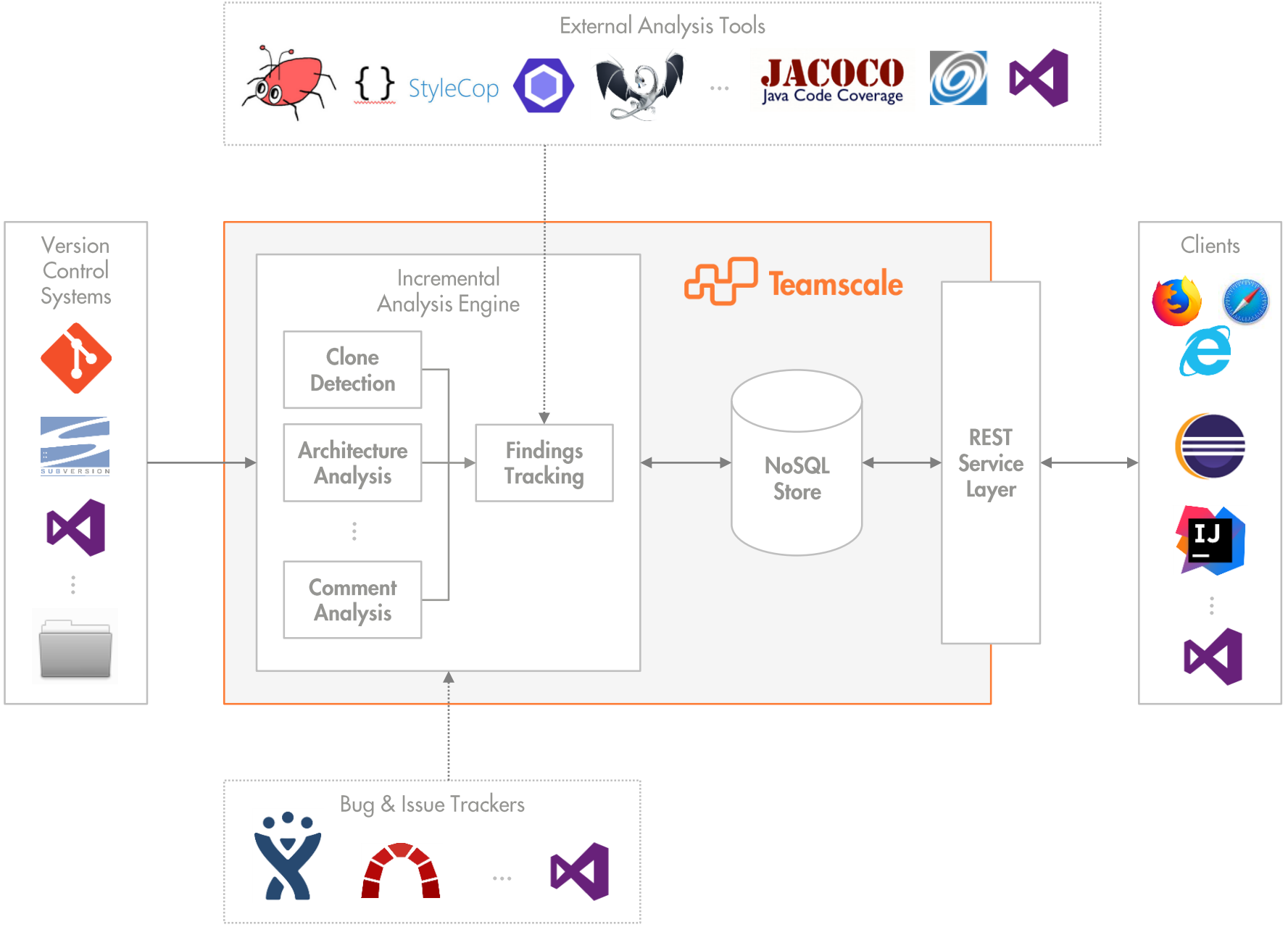
```
 if (lhs==null && rhs==null)     return;
 if (lhs==null)      fail("lhs is null while rhs="+rhs);
 if (rhs==null)      fail("rhs is null while lhs="+lhs);

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
 assertEquals(lhs.size(), rhs.size());
```

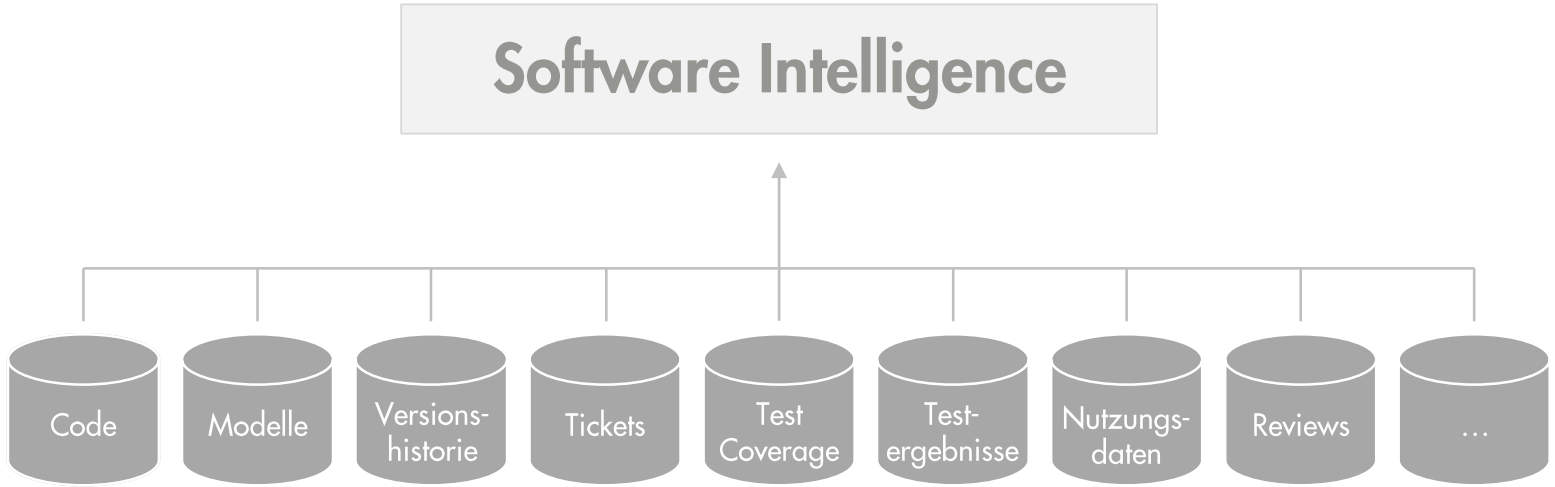
jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.java (revision 3909f5ac...)

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

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
```

ICSM 2012

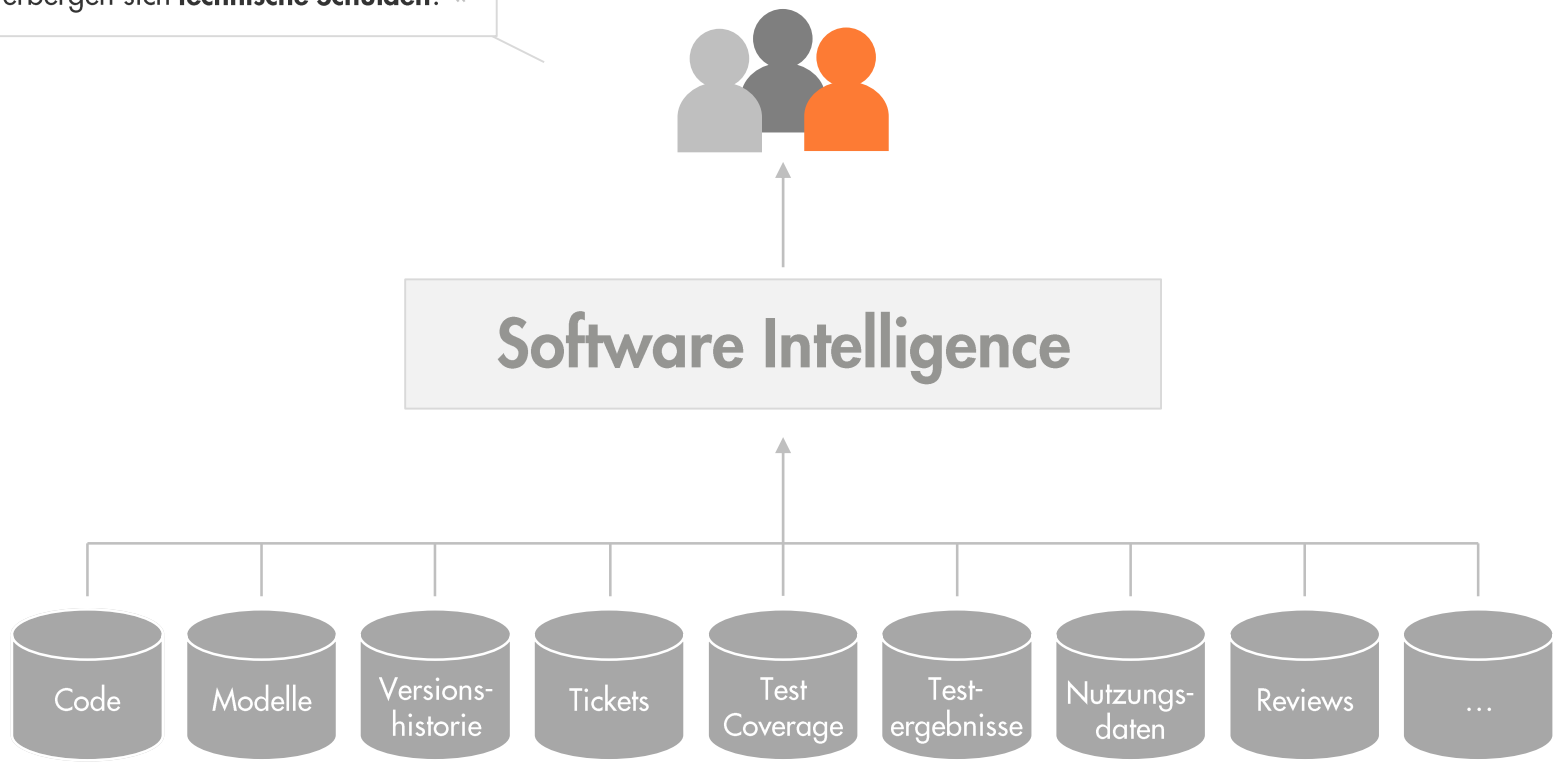

Version Control Systems
SUBVERSION
Incremental Analysis Engine
Clone Detection
Architecture Analysis
Comment Analysis
Findings Tracking
Teamscale
NoSQL Store
REST Service Layer
Clients

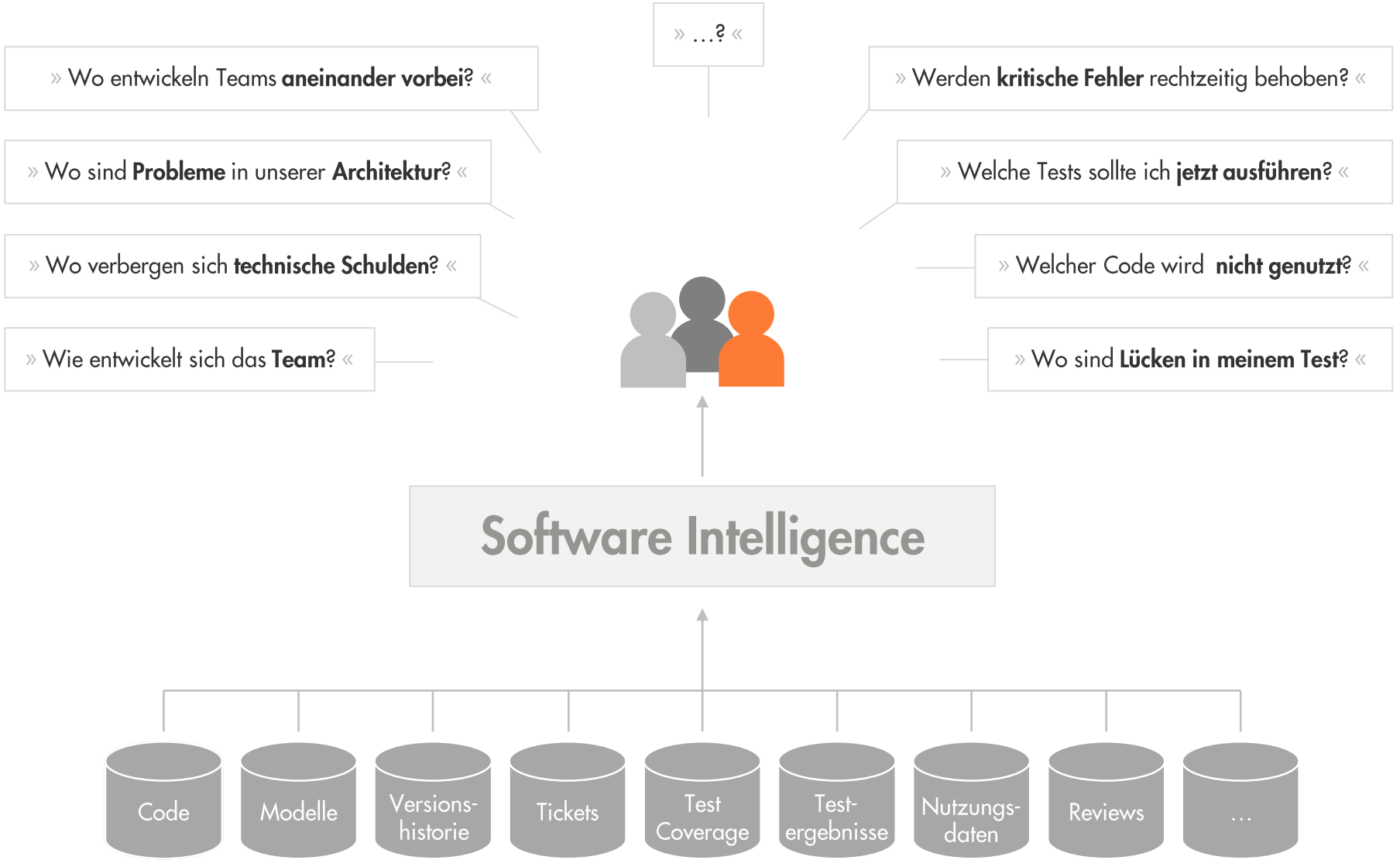

External Analysis Tools
StyleCop
JACOCO
Java Code Coverage
Version Control Systems
SUBVERSION
Incremental Analysis Engine
Clone Detection
Architecture Analysis
Comment Analysis
Findings Tracking
Teamscale
NoSQL Store
REST Service Layer
Clients
IJ
Bug & Issue Trackers

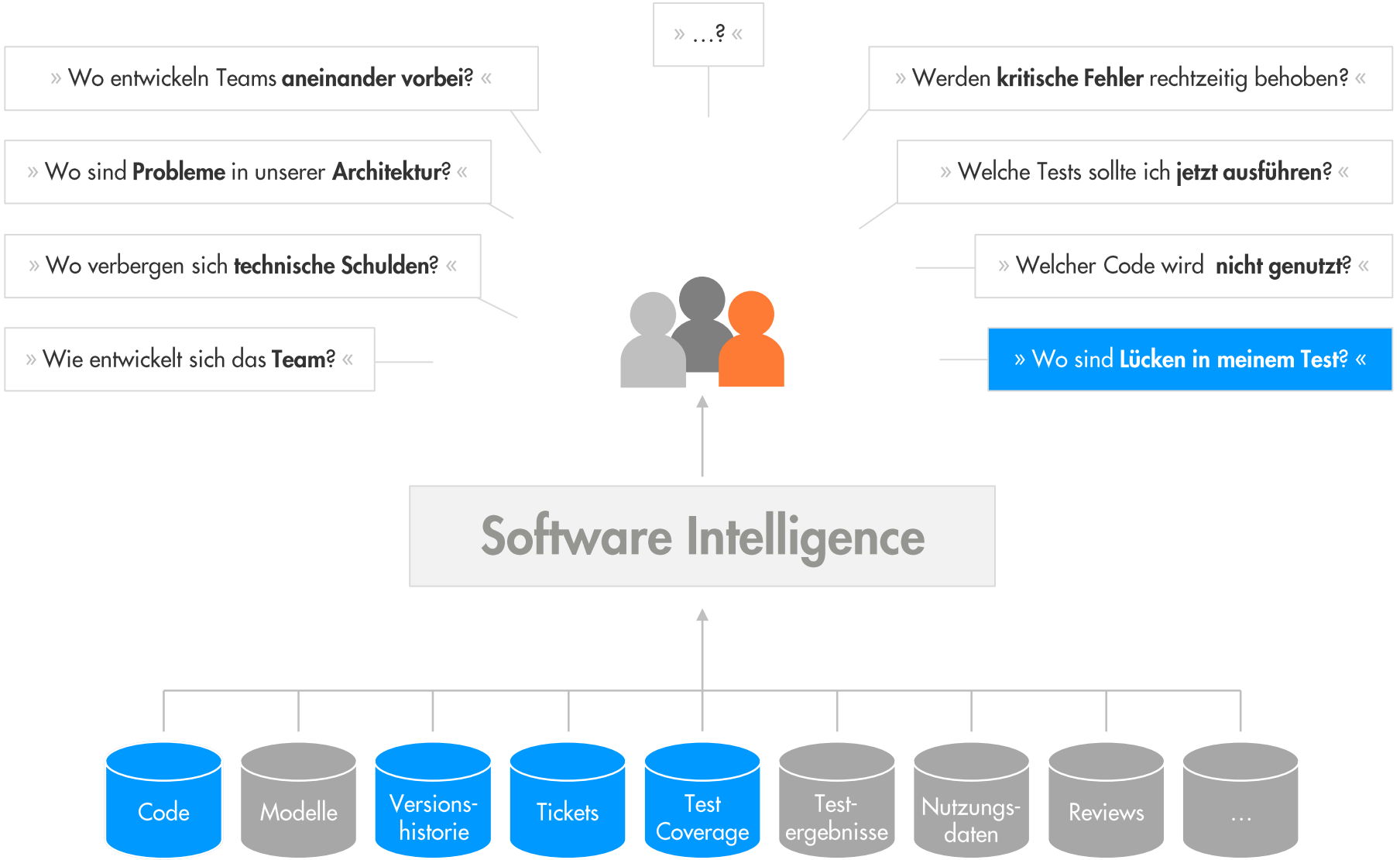

Software Intelligence
Code
Modelle
Versions-
historie
Tickets
Test
Coverage
Test-
ergebnisse
Nutzungs-
daten
Reviews
…

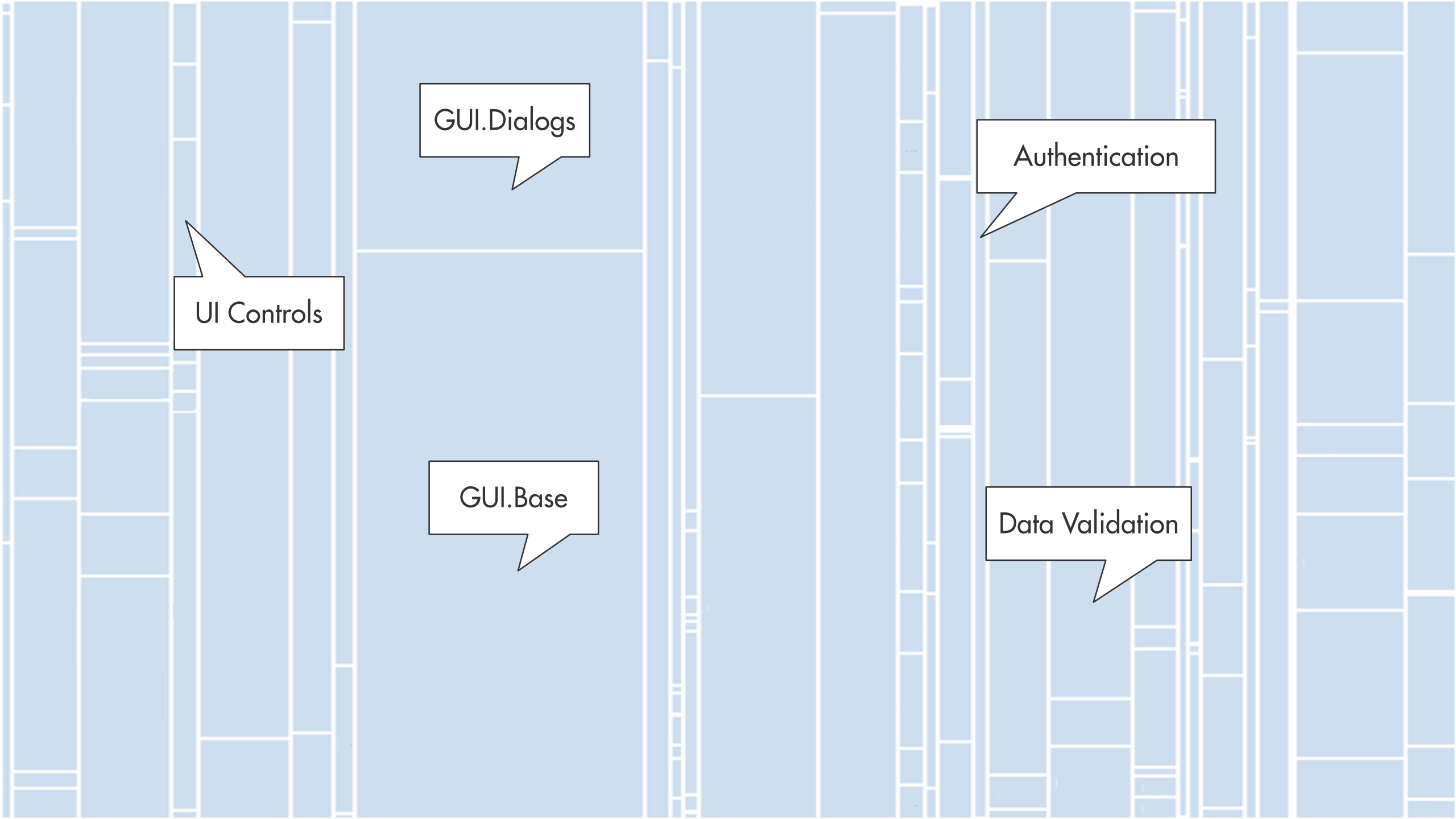
» Wo sind **Probleme** in unserer **Architektur**? «

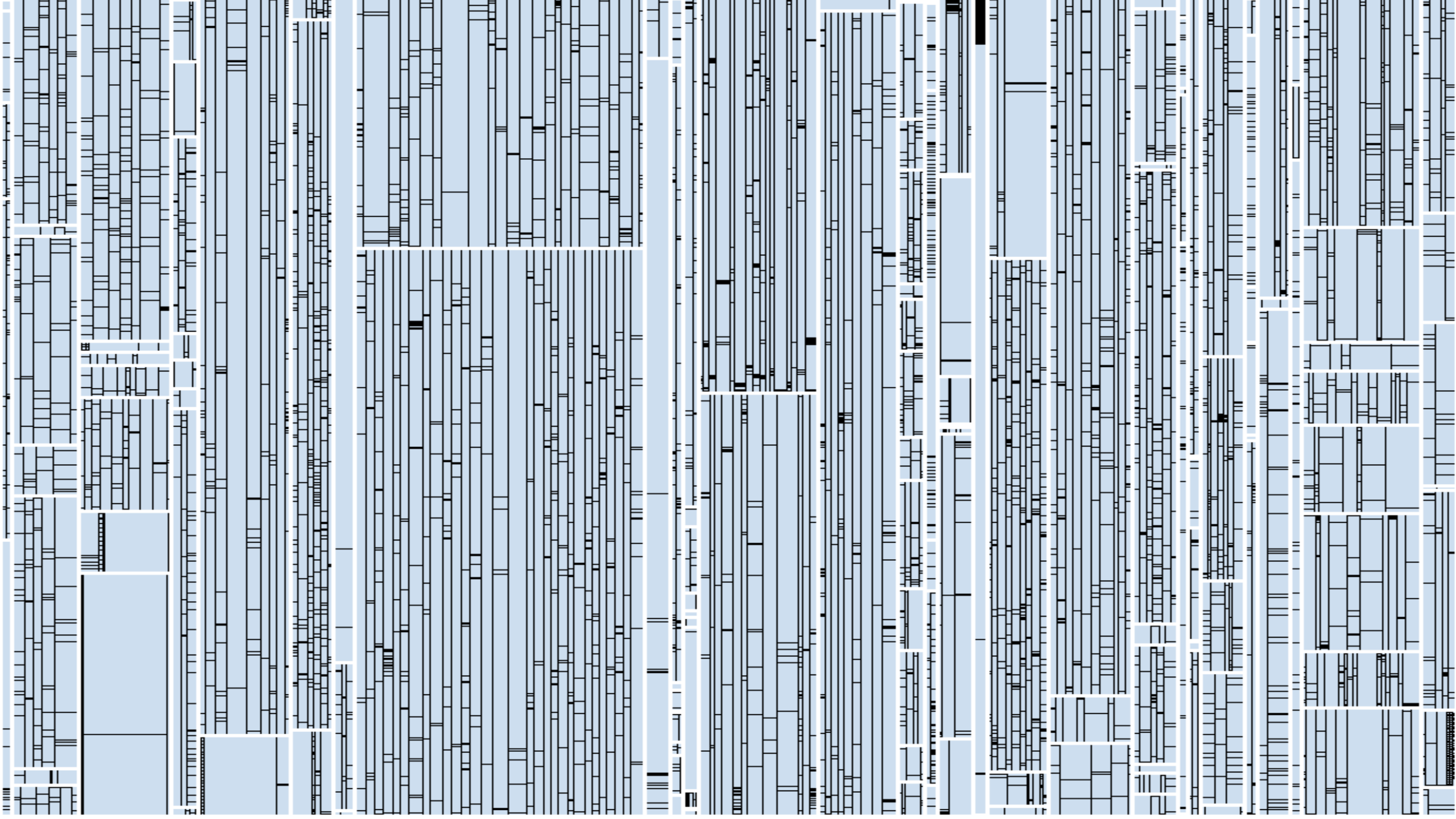
» Wo verbergen sich **technische Schulden**? «

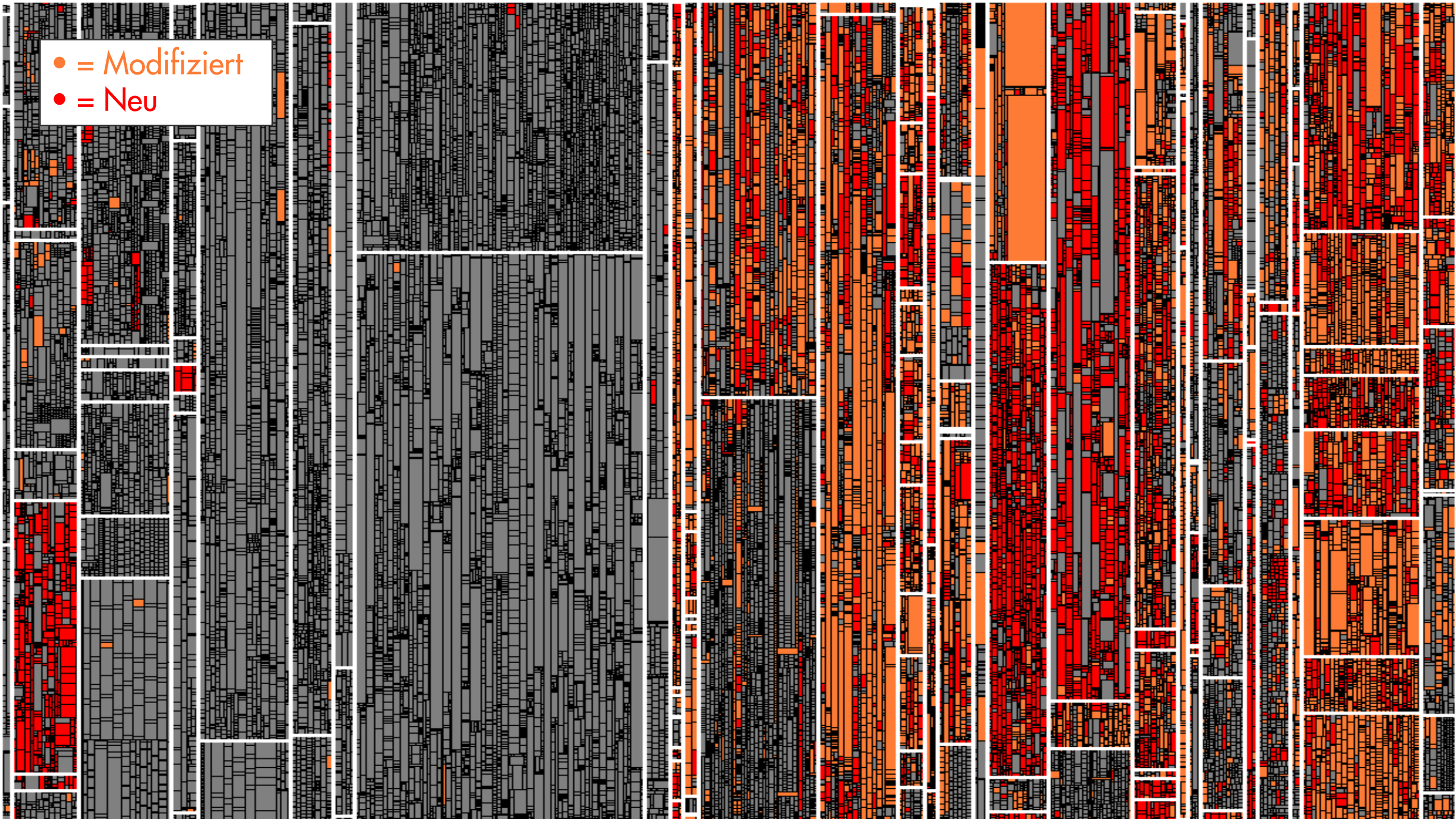


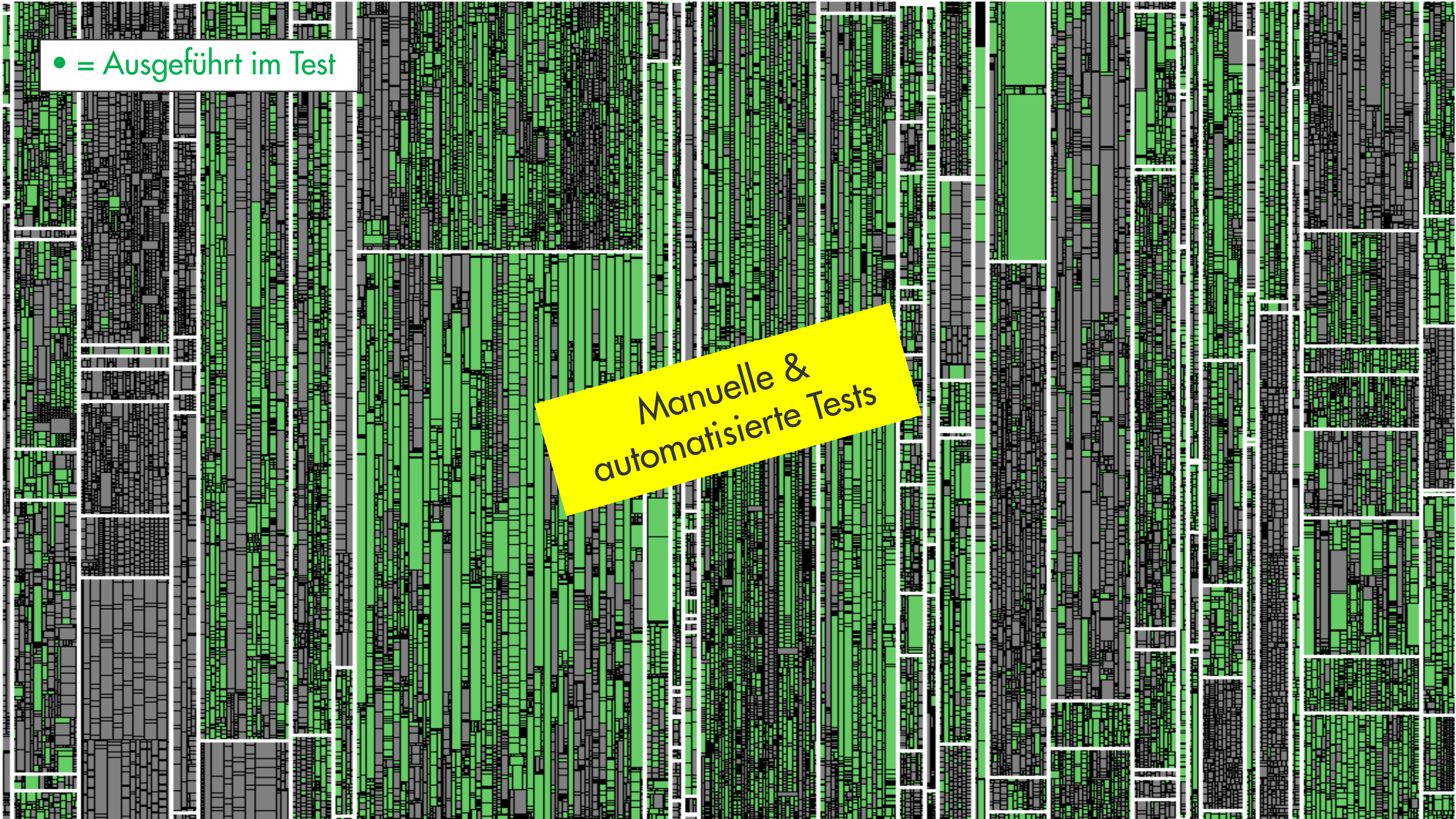

» …? «
» Wo entwickeln Teams **aneinander vorbei**? «
» Werden **kritische Fehler** rechtzeitig behoben? «
» Wo sind **Probleme** in unserer **Architektur**? «
» Welche Tests sollte ich **jetzt ausführen**? «
» Wo verbergen sich **technische Schulden**? «
» Welcher Code wird **nicht genutzt**? «
» Wie entwickelt sich das **Team**? «
» Wo sind **Lücken in meinem Test**? «
Software Intelligence
Code
Modelle
Versions-historie
Tickets
Test Coverage
Test-ergebnisse
Nutzungs-daten
Reviews
…

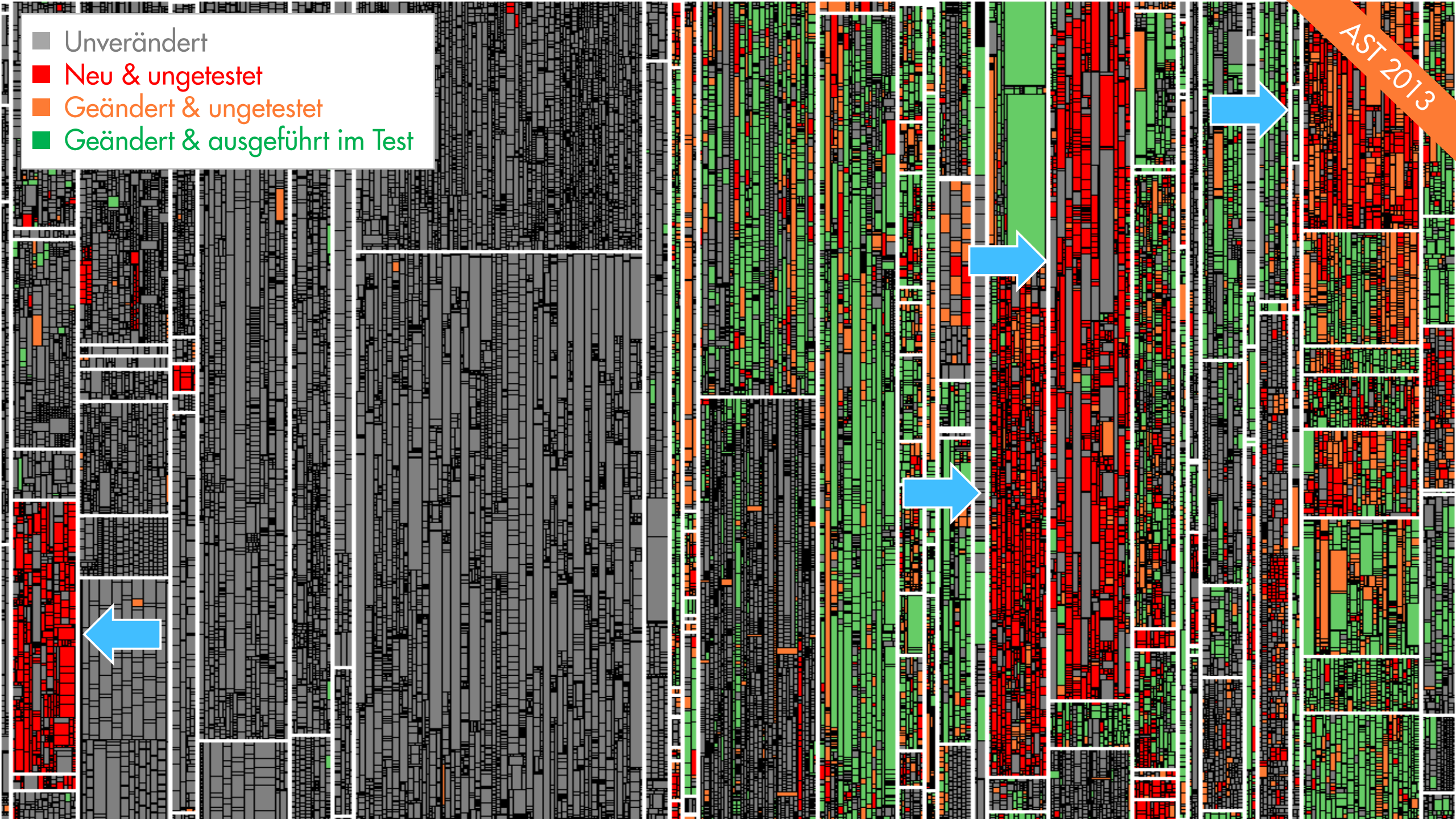

» …? «
» Wo entwickeln Teams **aneinander vorbei**? «
» Werden **kritische Fehler** rechtzeitig behoben? «
» Wo sind **Probleme** in unserer **Architektur**? «
» Welche Tests sollte ich **jetzt ausführen**? «
» Wo verbergen sich **technische Schulden**? «
» Welcher Code wird **nicht genutzt**? «
» Wie entwickelt sich das **Team**? «
» Wo sind **Lücken in meinem Test**? «
Software Intelligence
Code
Modelle
Versions-historie
Tickets
Test Coverage
Test-ergebnisse
Nutzungs-daten
Reviews
…

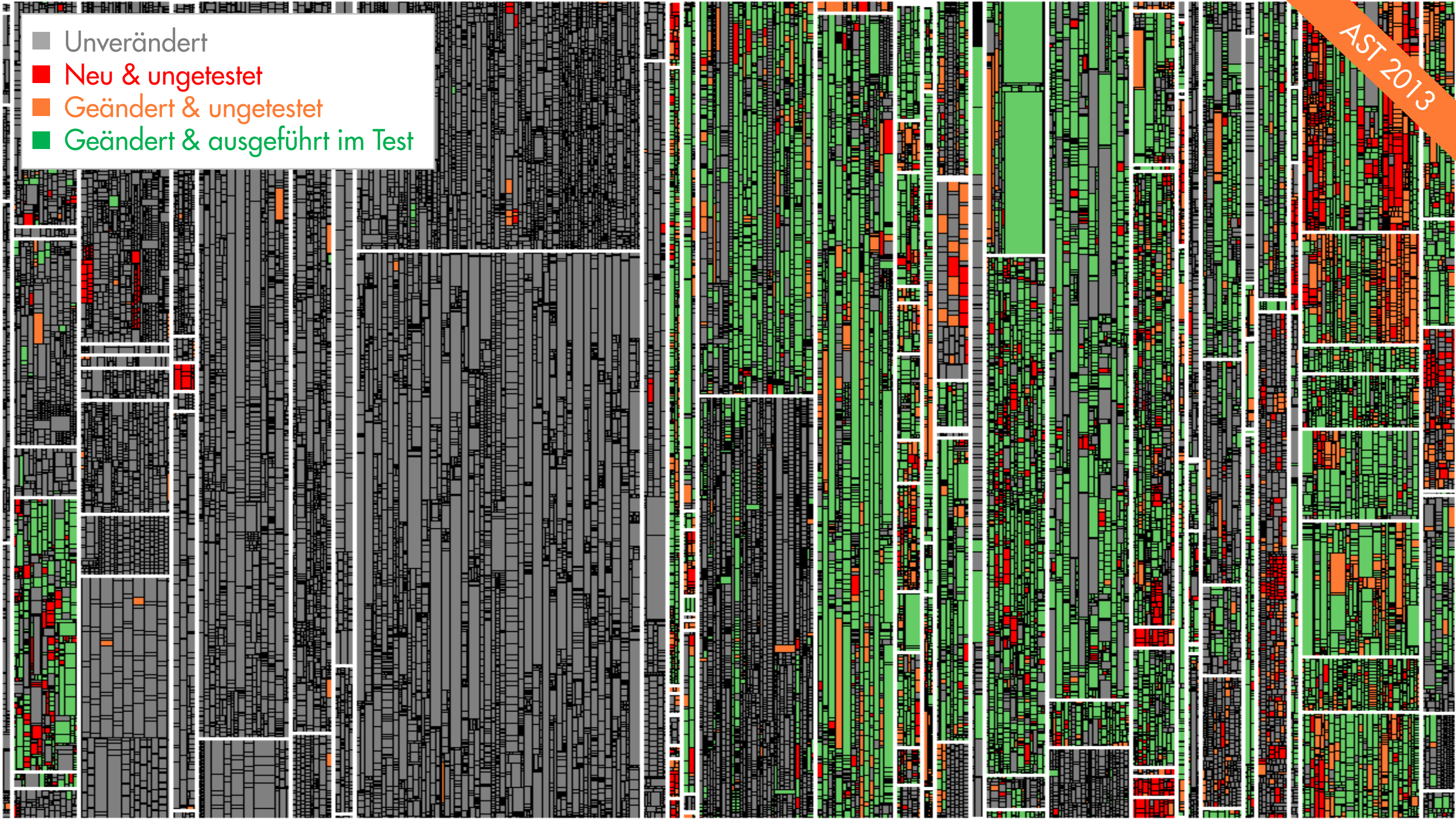

GUI.Dialogs
Authentication
UI Controls
GUI.Base
Data Validation




• = Modifiziert
• = Neu


• = Ausgeführt im Test
Manuelle &
automatisierte Tests


Unverändert
Neu & ungetestet
Geändert & ungetestet
Geändert & ausgeführt im Test
AST 2013


Unverändert
Neu & ungetestet
Geändert & ungetestet
Geändert & ausgeführt im Test
AST 2013

Issue #	Subject			Test Gap
TS-10549	Undo/Redo for web-based architecture editor	Done		0%
TS-10784	Fix long method finding in TaintAnalysisRunner	Done		0%
TS-10923	Implement metric 'Nesting Depth' for Simulink	Done		29%
TS-11364	External findings are not registered during first upload	Done		14%
TS-11942	Manual test coverage upload during development	Done		43%
TS-12050	Tool for transferring findings blacklists and tasks	Done		50%
TS-12262	Cannot set or alter alias without reanalysis	Done		0%
TS-13151	Fetch parent relationship of TFS work items	Done		0%

Issue #	Subject		Test Gap
TS-14421	Get rid of TestGapSynchronizer block	Done	0%
TS-14733	Remove Dataflow blocks	Done	22%

Done **Issue TS-14733 - Remove Dataflow blocks**

Creator: (on Apr 06 2018 19:44)

Last update: Aug 24 2018 09:32

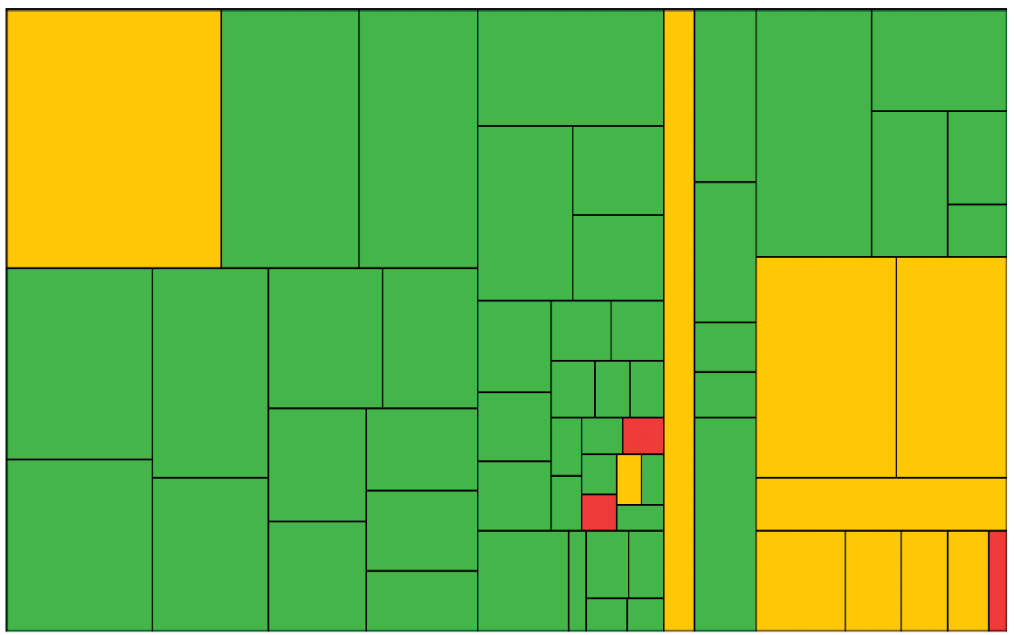
Assignee:

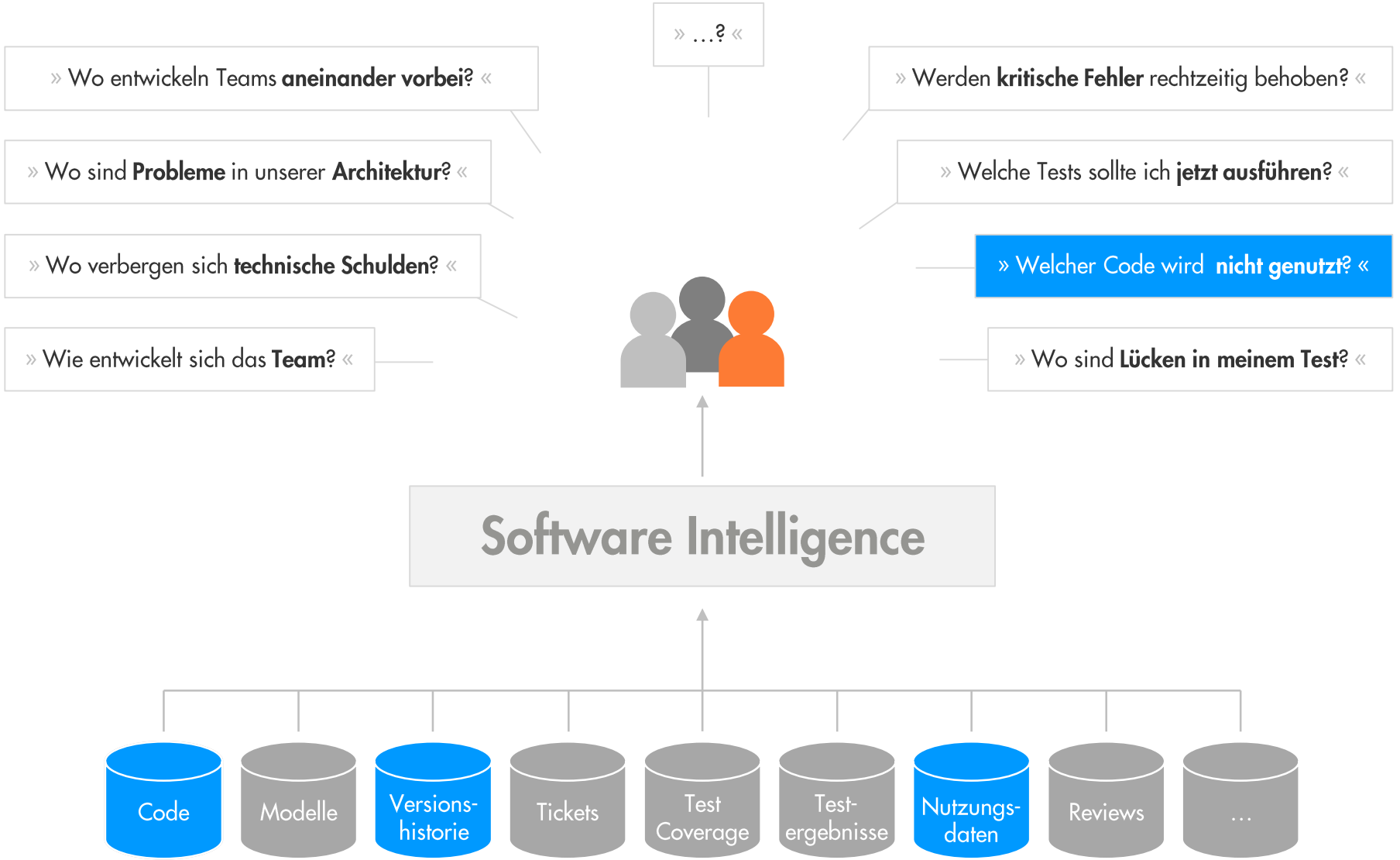
Project	Type	Priority	Resolution	Fix Version
TS	Maintenance	Normal	Green	Teamscale 4.5

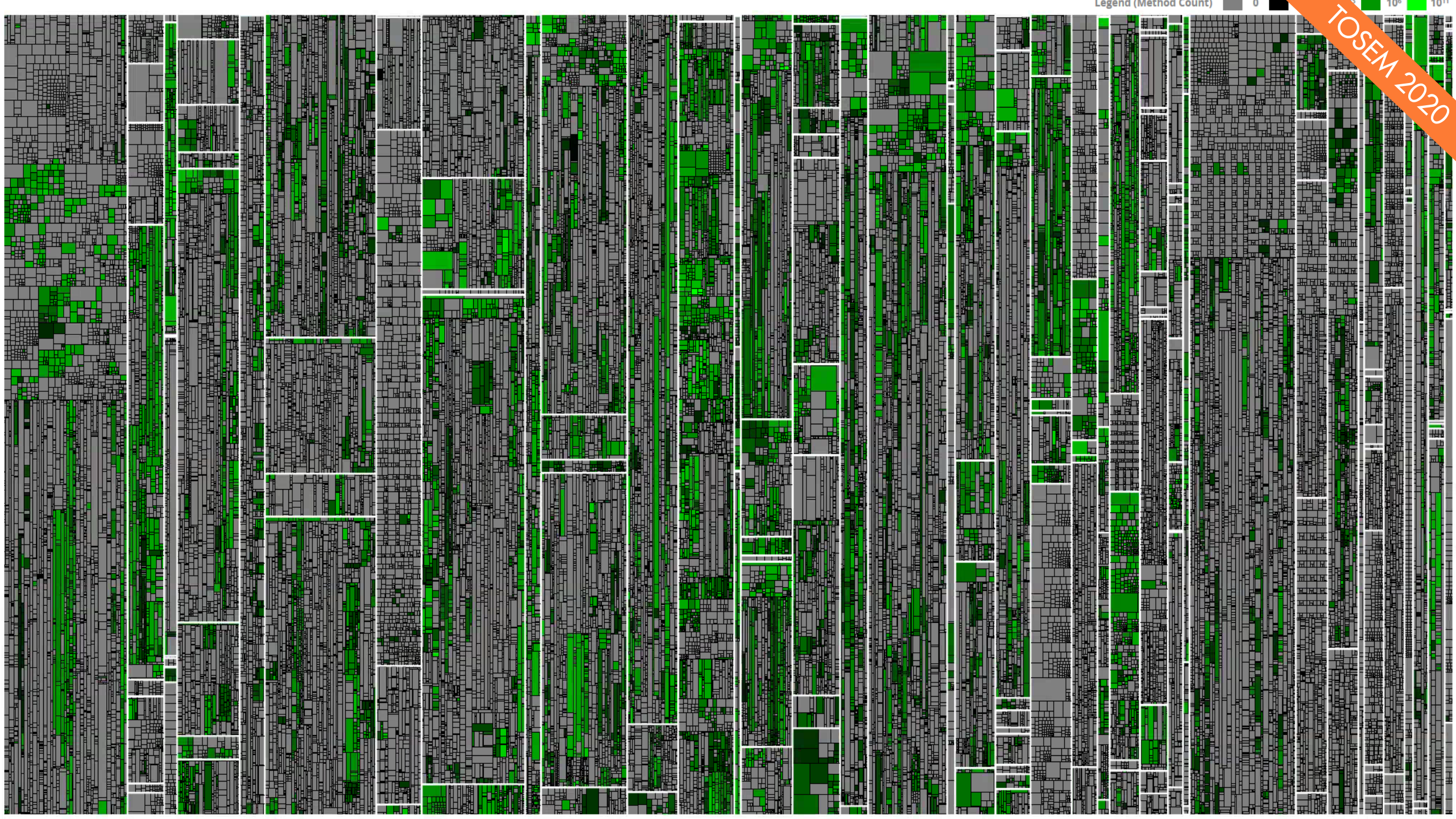
Component	Labels	Affected Version	Customer	Customer Issue
Backend	Performance			

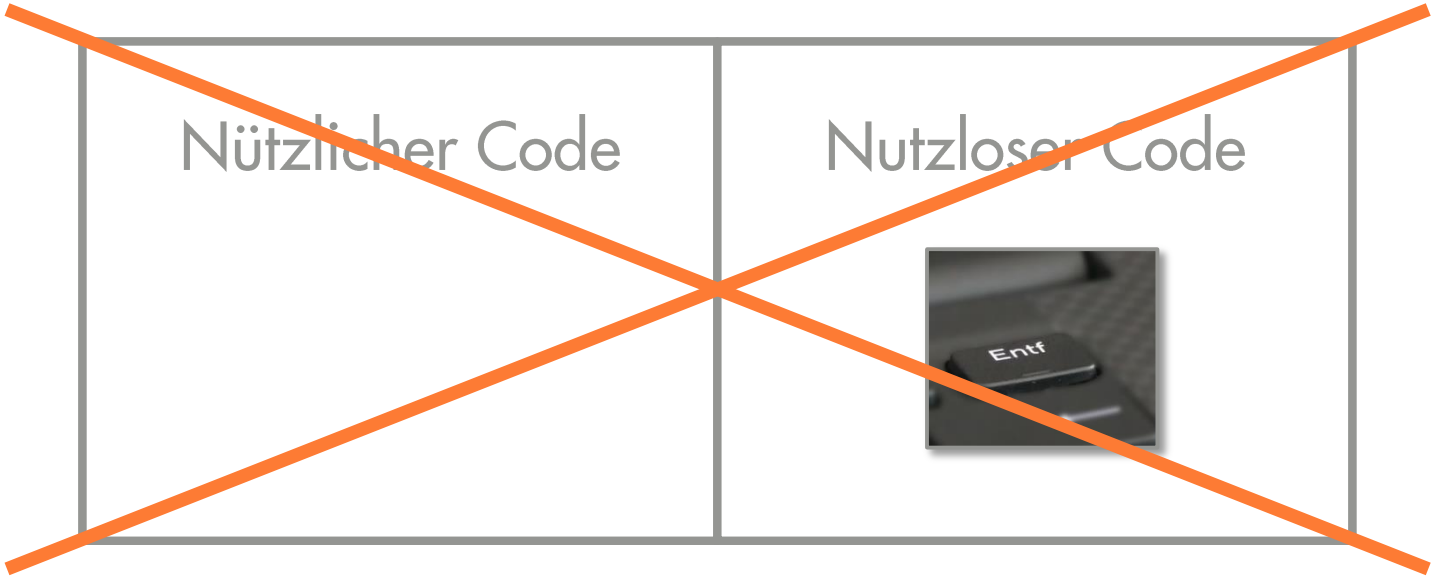
Epic Name	Freshdesk URL	Merge Request
		https://git.cqse.eu/cqse/teamscale/3621

Aug 15 2018 12:37–Now | Test Gap: 22%



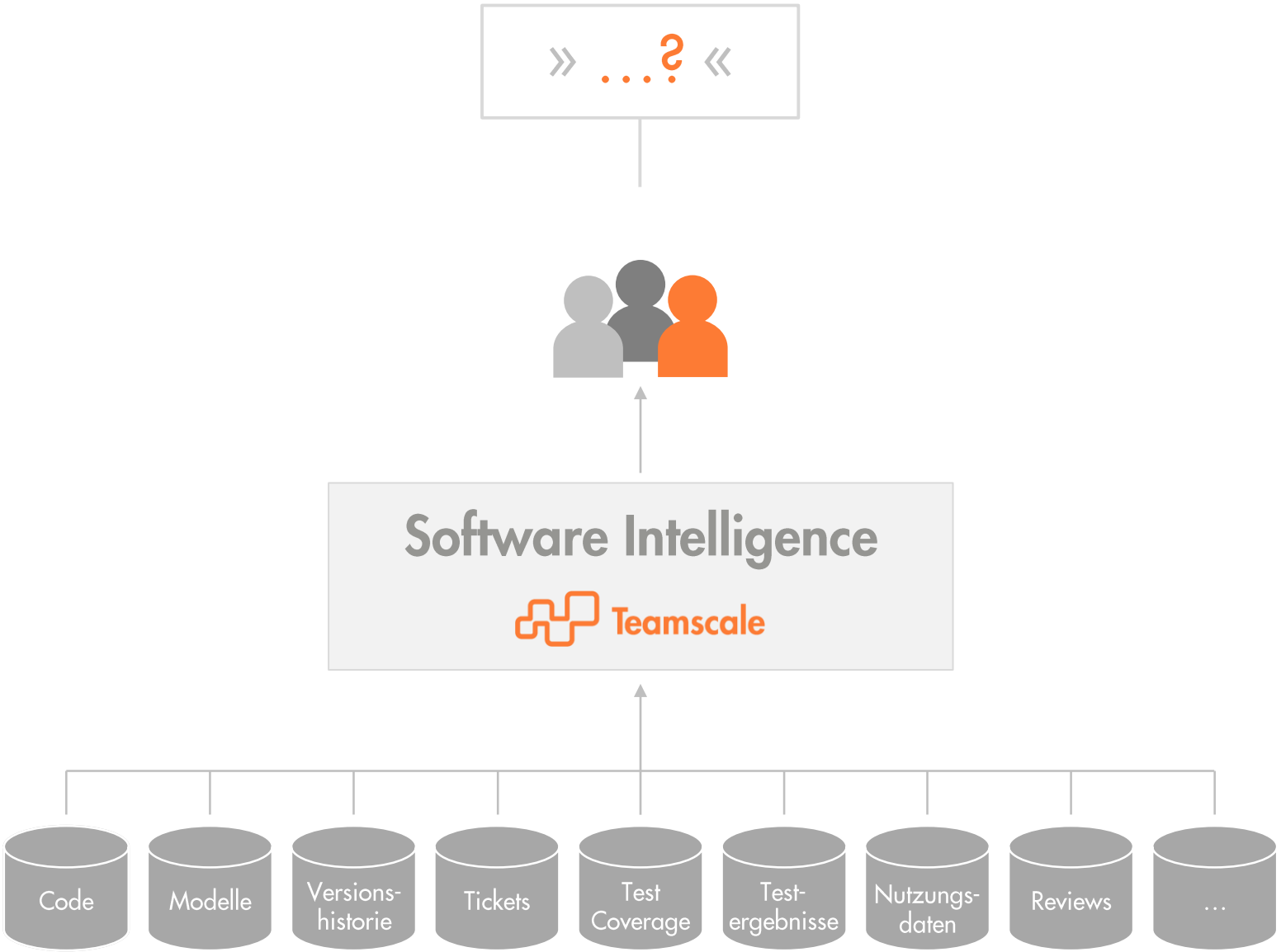

» …? «
» Wo entwickeln Teams **aneinander vorbei**? «
» Werden **kritische Fehler** rechtzeitig behoben? «
» Wo sind **Probleme** in unserer **Architektur**? «
» Welche Tests sollte ich **jetzt ausführen**? «
» Wo verbergen sich **technische Schulden**? «
» Welcher Code wird **nicht genutzt**? «
» Wie entwickelt sich das **Team**? «
» Wo sind **Lücken in meinem Test**? «
Software Intelligence
Code
Modelle
Versions-historie
Tickets
Test Coverage
Test-ergebnisse
Nutzungs-daten
Reviews
…


Legend (Method Count)
0
10^6
10^{11}
TOSEM 2020


Nützlicher Code
Nutzloser Code
Entf


Nützlicher Code
Vermutlich nutzloser Code
Nutzloser Code
Entf

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


» ...? «
Software Intelligence
Teamscale
Code
Modelle
Versions-historie
Tickets
Test Coverage
Test-ergebnisse
Nutzungs-daten
Reviews
...

3 Clone class (46 token-sequences) with 2 instances found at
D:/Source/[illegible]/Gui.Dialogs/UserControls/NavigationTrees/Trees/TreeAdministrationQuarterly.cs (94)
length: 145 lines; transformed length: 46
D:/Source/[illegible]/Gui.Dialogs/UserControls/NavigationTrees/Trees/TreeAdministrationYearly.cs (107)
length: 145 lines; transformed length: 46

4 Clone class (43 token-sequences) with 2 instances found at
D:/Source/[illegible]/Gui.Dialogs/UserControls/[illegible]/UserControlGeneralMainSegment.cs (889)
length: 128 lines; transformed length: 43
D:/Source/[illegible]/Gui.Dialogs/UserControls/[illegible]/UserControlGeneralMainSegmentCMC.cs (654)
length: 122 lines; transformed length: 43

5 Clone class (42 token-sequences) with 2 instances found at
D:/Source/[illegible]/Gui.Dialogs/UserControls/NavigationTrees/TreeAndFilter/TreeAndFilterAdministrationQuarterly.cs
(127)
length: 98 lines; transformed length: 42
D:/Source/[illegible]/Gui.Dialogs/UserControls/NavigationTrees/TreeAndFilter/TreeAndFilterAdministrationYearly.cs
(130)
length: 98 lines; transformed length: 42

6 Clone class (35 token-sequences) with 2 instances found at
D:/Source/[illegible]/AppCore.BusinessObjects/BackEndObjects/BeoBranchHistory.cs (130)
length: 34 lines; transformed length: 35
D:/Source/[illegible]/AppCore.BusinessObjects/BackEndObjects/BeoHistory.cs (117)
length: 34 lines; transformed length: 35

7 Clone class (33 token-sequences) with 2 instances found at
D:/Source/[illegible]AppCore.BusinessObjects/History/BranchHistory.cs (582)
length: 68 lines; transformed length: 33
D:/Source/[illegible]/AppCore.BusinessObjects/History/History.cs (488)
length: 70 lines; transformed length: 33

8 Clone class (33 token-sequences) with 2 instances found at
D:/Source/[illegible]/Application/[illegible] (250)
length: 69 lines; transformed length: 33
D:/Source/[illegible]/Gui.Test/TestForm.cs (1080)
length: 69 lines; transformed length: 33

TreeAdministrationQuarterly.cs

```
93      /// </summary>
94      protected override bool DoLazyLoading(
            UltraTreeNode node)
95      {
96       // Some types of segments always have the
             expand icon
97       if ( node.Tag is GaSegment ||
98         node.Tag is MainSegment ||
99         node.Tag is GeneralMainSegment ||
100        node.Tag is Branch )
101       return true;
102      // All others only if there are child nodes
103      return false;
104     }

106     /// <summary>
107     /// Structure segments return only loss or
            premium segments according to
108     /// the mode.
109     /// </summary>
110     protected override IList GetChildSegments(
            ISegment parent, ref bool alreadySorted )
111     {
112      if ( parent is StructureSegment )
113      {
114       // Show either preminum or loss beyond
              structure segment
115       StructureSegment strucSeg = (StructureSegment
              )parent;
116       if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
              TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
              )
117        return strucSeg.GetLossProcessingSegments();
118       else
119        return strucSeg.GetPremiumProcessingSegments
              ();
```

TreeAdministrationYearly.cs

```
106     /// </summary>
107     protected override bool DoLazyLoading(
            UltraTreeNode node)
108     {
109      // Some types of segments always have the
             expand icon
110      if ( node.Tag is GaSegment ||
111        node.Tag is MainSegment ||
112        node.Tag is GeneralMainSegment ||
113        node.Tag is Branch )
114       return true;
115      // All others only if there are child nodes
116      return false;
117     }

119     /// <summary>
120     /// Structure segments return only loss or
            premium segments according to
121     /// the mode.
122     /// </summary>
123     protected override IList GetChildSegments(
            ISegment parent, ref bool alreadySorted )
124     {
125      if ( parent is StructureSegment )
126      {
127       // Show either preminum or loss beyond
              structure segment
128       StructureSegment strucSeg = (StructureSegment
              )parent;
129       if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
              TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
              )
130        return strucSeg.GetLossProcessingSegments();
131       else
132        return strucSeg.GetPremiumProcessingSegments
              ();
```

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 already Sorted )
        {
          if ( parent is StructureSegment )
          {
            // Show either premium or loss beyond
                structure segment
            StructureSegment strucSeg = (StructureSegment
                )parent;
            if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
                TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
                )
              return strucSeg.GetLossProcessingSegments();
            else
              return strucSeg.GetPremiumProcessingSegments
                  ();
          }
          else if ( parent is MainSegment )
          {
            // Speed up loading GaSegments beyond
                MainSegments
            try
            {
              // Apply filter before expanding GASes
              return ApplicationServices.Segmentation.
                  GetFilteredGaSegments(
                false,
                (MainSegment)parent,
                (Responsibility) GetFilterValue( typeof(
                    Responsibility) ),
                null, null,
                (TypeOfBusinessEnum) GetFilterValue( typeof
                    (TypeOfBusinessEnum) ),
                (NatureOfTreaty) GetFilterValue( typeof(
                    NatureOfTreaty) ) );
            }
            catch ( ObjectsDirtyException )
            {
              return base.GetChildSegments (parent, ref
                  already Sorted);
            }
          }
          else if ( parent is Branch )
          {
            Branch br = parent as Branch;
            // Filter set for main Segment
            if ( HierachyFilter != null && HierachyFilter
                .IsFiltered( typeof(MainSegment) ) )
            {
              MainSegment filteredMS = HierachyFilter.
                  FilteredSegment( typeof(MainSegment) )
                  as MainSegment;
              // Show only filtered main segment
              if ( br.GetChildSegments(typeof(MainSegment)
                  ).Contains( filteredMS ) )
                // HistoryFilter
                if ( ApplicationServices.History.
                    IsValidBranchInActiveHistory(
                    filteredMS.Branch, false ) )
                  return new MainSegment[]{filteredMS};
              // or none (empty list)
              return new ArrayList();
            }
```

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 already Sorted )
        {
          if ( parent is StructureSegment )
          {
            // Show either premium or loss beyond
                structure segment
            StructureSegment strucSeg = (StructureSegment
                )parent;
            if ( (TypeOfSgmtEnum) GetFilterValue( typeof(
                TypeOfSgmtEnum) ) == TypeOfSgmtEnum.Loss
                )
              return strucSeg.GetLossProcessingSegments();
            else
              return strucSeg.GetPremiumProcessingSegments
                  ();
          }
          else if ( parent is MainSegment )
          {
            // Speed up loading GaSegments beyond
                MainSegments
            try
            {
              // Apply filter before expanding GASes
              return ApplicationServices.Segmentation.
                  GetFilteredGaSegments(
                false,
                (MainSegment)parent,
                (Responsibility) GetFilterValue( typeof(
                    Responsibility) ),
                null, null,
                (TypeOfBusinessEnum) GetFilterValue( typeof
                    (TypeOfBusinessEnum) ),
                (NatureOfTreaty) GetFilterValue( typeof(
                    NatureOfTreaty) ) );
            }
            catch ( ObjectsDirtyException )
            {
              return base.GetChildSegments (parent, ref
                  already Sorted);
            }
          }
          else if ( parent is Branch )
          {
            Branch br = parent as Branch;
            // Filter set for main Segment
            if ( HierachyFilter != null && HierachyFilter
                .IsFiltered( typeof(MainSegment) ) )
            {
              MainSegment filteredMS = HierachyFilter.
                  FilteredSegment( typeof(MainSegment) )
                  as MainSegment;
              // Show only filtered main segment
              if ( br.GetChildSegments(typeof(MainSegment)
                  ).Contains( filteredMS ) )
                // HistoryFilter
                if ( ApplicationServices.History.
                    IsValidBranchInActiveHistory(
                    filteredMS.Branch, false ) )
                  return new MainSegment[]{filteredMS};
              // or none (empty list)
              return new ArrayList();
            }
```

TreeAdministrationQuarterly.cs

```
          // Filter for type of business
          return br.FilteredMainSegments( (
              TypeOfBusinessEnum) GetFilterValue(
              typeof(TypeOfBusinessEnum) ) );
        }
        // Default behaviour
        return base.GetChildSegments (parent, ref
            already Sorted);
      }

      /// <summary>
      /// Overriden so that detailsegments and
          mainsegments return the
      /// correct parent.
      /// </summary>
      protected override ISegment GetParentSegment(
          ISegment child )
      {
        if (child is DetailSegment)
        {
          // Returns the loss or premium procSeg,
              depending on the mode
          DetailSegment detSeg = (DetailSegment)child;
          StructureSegment strSeg = detSeg.
              StructureSegment;
          if ( strSeg == null )
            return null;
          return strSeg.GetProcessingSegment(
            detSeg.YearType,
            (TypeOfSgmtEnum) GetFilterValue( typeof(
                TypeOfSgmtEnum) ) );
        }
        else if (child is MainSegment)
        {
          return ((MainSegment)child).Branch;
        }
        else
        {
          return base.GetParentSegment (child);
        }
      }


      /// <summary>
      /// Overriden to expand StructureSegment nodes
      /// </summary>
      protected override void AfterChildrenCreated(
          UltraTreeNode parent )
      {
        base.AfterChildrenCreated( parent );
        if (parent.Tag is StructureSegment && parent.
            HasNodes)
        {
          parent.Expanded = true;
        }
      }

      /// <summary>
      /// Provide a matching segment for premium/loss
          PS
      /// </summary>
      /// <param name="template">template to match</
          param>
      /// <returns>matching segment</returns>
      protected override ISegment MatchingSegment(
          ISegment template)
      {
        // Special case for processing segments
        if ( template is ProcessingSegment )
        {
          // Try to provide corresponding segment
          ProcessingSegment pSeg = (ProcessingSegment)
              template;
          ProcessingSegment correspondingSegment = null
              ;

          // get corresponding premium/loss segment
          if ( pSeg.TypeOfSgmt == TypeOfSgmtEnum.Loss )
            correspondingSegment = (ProcessingSegment)
                pSeg.GetPremiumSegments()[0];
          else
            correspondingSegment = pSeg.GetLossSegment()
                ;
```

TreeAdministrationYearly.cs

```
          // Filter for type of business
          return br.FilteredMainSegments( (
              TypeOfBusinessEnum) GetFilterValue(
              typeof(TypeOfBusinessEnum) ) );
        }
        // Default behaviour
        return base.GetChildSegments (parent, ref
            already Sorted);
      }

      /// <summary>
      /// Overriden so that detailsegments and
          mainsegments return the
      /// correct parent.
      /// </summary>
      protected override ISegment GetParentSegment(
          ISegment child )
      {
        if (child is DetailSegment)
        {
          // Returns the loss or premium procSeg,
              depending on the mode
          DetailSegment detSeg = (DetailSegment)child;
          StructureSegment strSeg = detSeg.
              StructureSegment;
          if ( strSeg == null )
            return null;
          return strSeg.GetProcessingSegment(
            detSeg.YearType,
            (TypeOfSgmtEnum) GetFilterValue( typeof(
                TypeOfSgmtEnum) ) );
        }
        else if (child is MainSegment)
        {
          return ((MainSegment)child).Branch;
        }
        else
        {
          return base.GetParentSegment (child);
        }
      }


      /// <summary>
      /// Overriden to expand StructureSegment nodes
      /// </summary>
      protected override void AfterChildrenCreated(
          UltraTreeNode parent )
      {
        base.AfterChildrenCreated( parent );
        if (parent.Tag is StructureSegment && parent.
            HasNodes)
        {
          parent.Expanded = true;
        }
      }

      /// <summary>
      /// Provide a matching segment for premium/loss
          PS
      /// </summary>
      /// <param name="template">template to match</
          param>
      /// <returns>matching segment</returns>
      protected override ISegment MatchingSegment(
          ISegment template)
      {
        // Special case for processing segments
        if ( template is ProcessingSegment )
        {
          // Try to provide corresponding segment
          ProcessingSegment pSeg = (ProcessingSegment)
              template;
          ProcessingSegment correspondingSegment = null
              ;

          // get corresponding premium/loss segment
          if ( pSeg.TypeOfSgmt == TypeOfSgmtEnum.Loss )
            correspondingSegment = (ProcessingSegment)
                pSeg.GetPremiumSegments()[0];
          else
            correspondingSegment = pSeg.GetLossSegment()
                ;
```

Dashboard
Activity
Findings
Metrics
Tests
Issues
Tasks
Architecture
Delta
Projects
System
Admin

Collapse

jenkins/test/src/main/java/org/jvnet/hudson/test/HudsonTestCase.java (revision 12d96a56...)

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

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
 assertEquals(lhs.size(), rhs.size());
```

jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.java (revision 3909f5ac...)

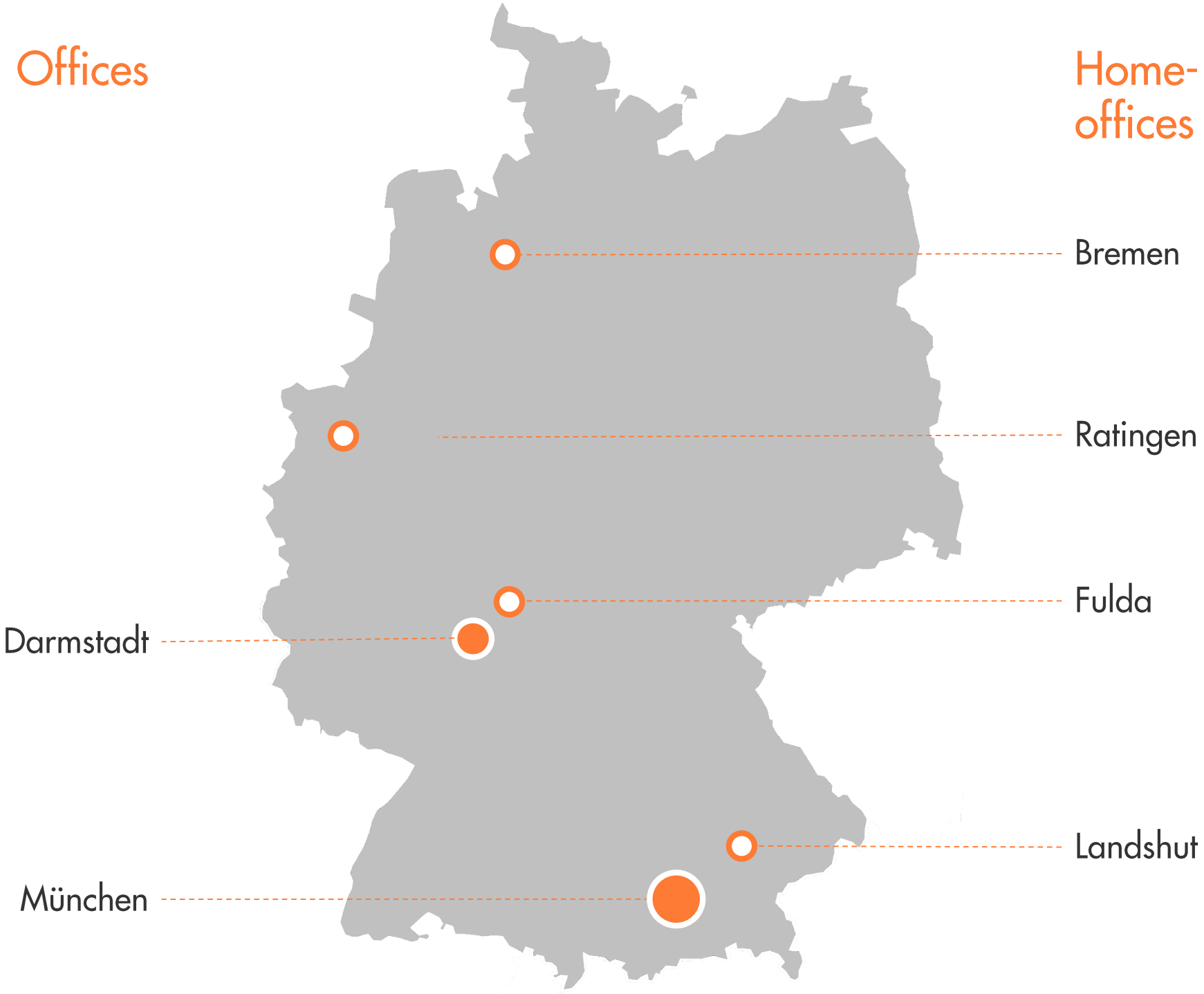
```
 if (lhs==null && rhs==null)     return;
 if (lhs==null)      fail("lhs is null while rhs="+rhs);
 if (rhs==null)      fail("rhs is null while lhs="+lhs);

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

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

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

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

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

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


*
 Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Objec
/
blic void assertEqualDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
```

Teamscale ist frei für Forschung & Lehre

Gerne bei mir melden: juergens@cqse.eu

Offices

Home-offices




CQSE NA
CQSE

Kontakt – Ich freue mich auf Diskussionen ☺

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

CQSE GmbH
Lichtenbergstraße 8
85748 Garching bei München
www.cqse.eu

CQSE