

Clone Detection & Management für Architekten



Dr. Elmar Jürgens



1975

A man with a beard and a striped jacket is smiling and looking towards the camera. He is holding a small, flat object in his hands. He is wearing a black strap over his shoulder and a lanyard around his neck. The background is a plain wall.

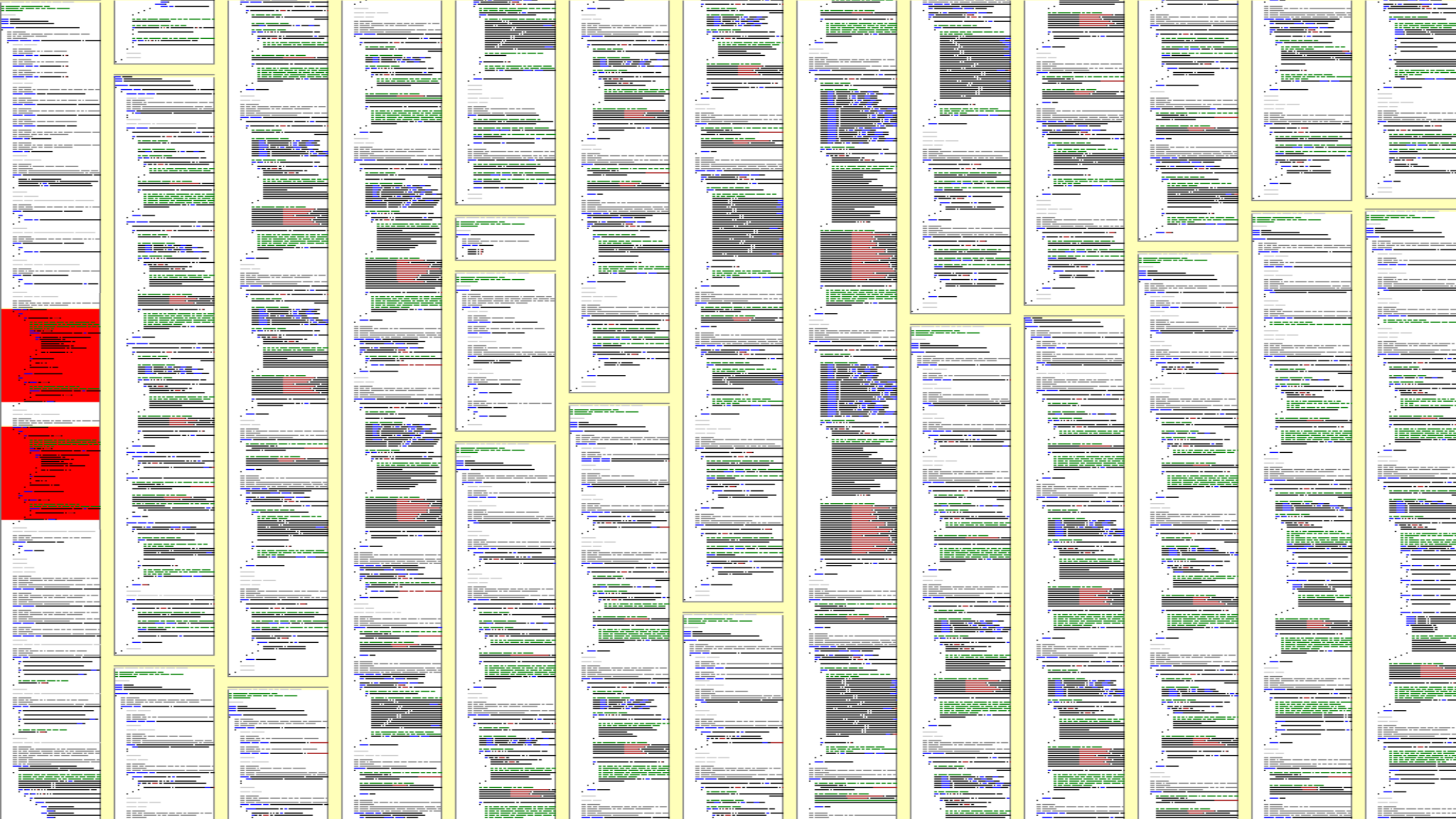
#1 Code Smell (1999)

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

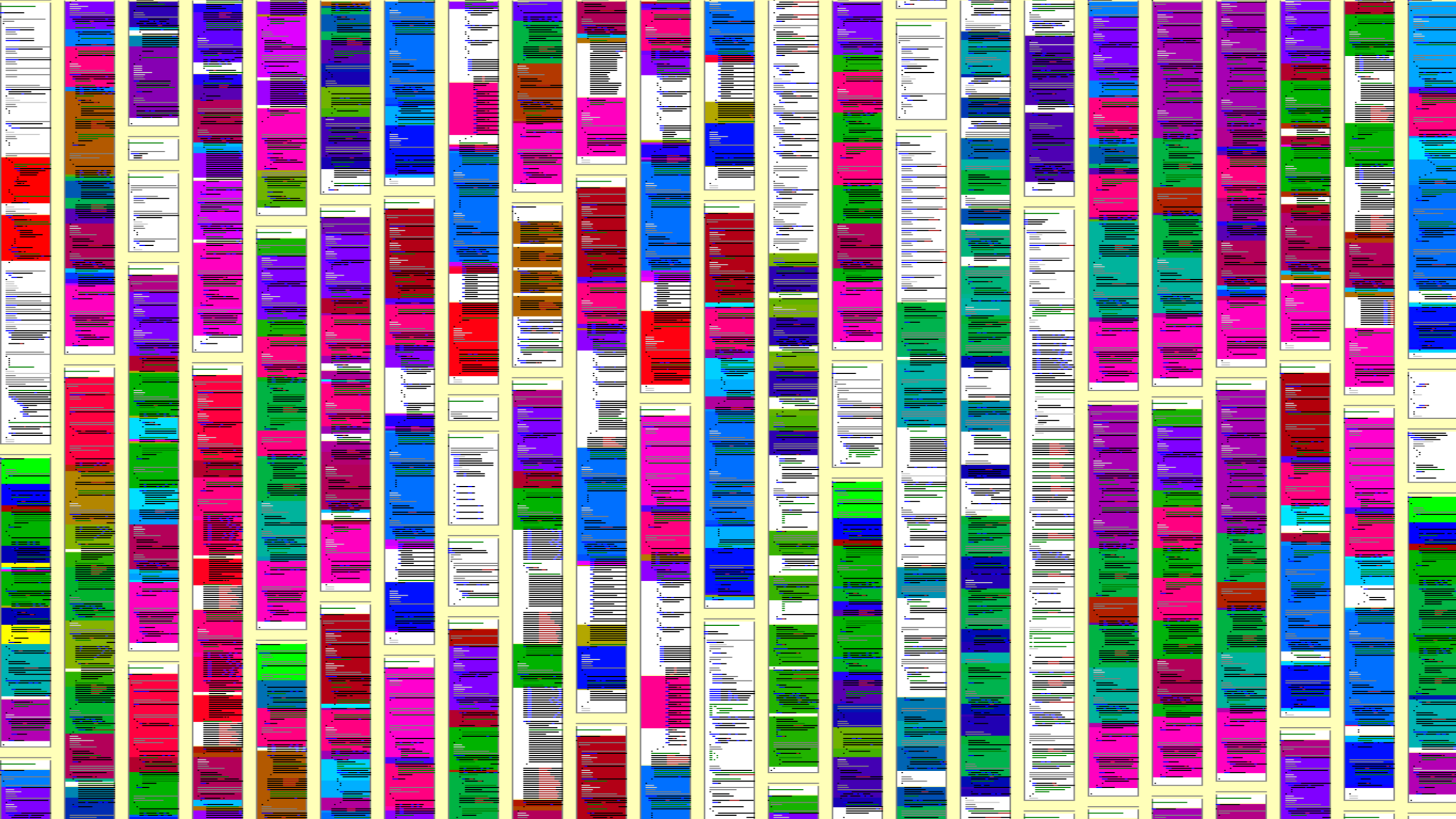
```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());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

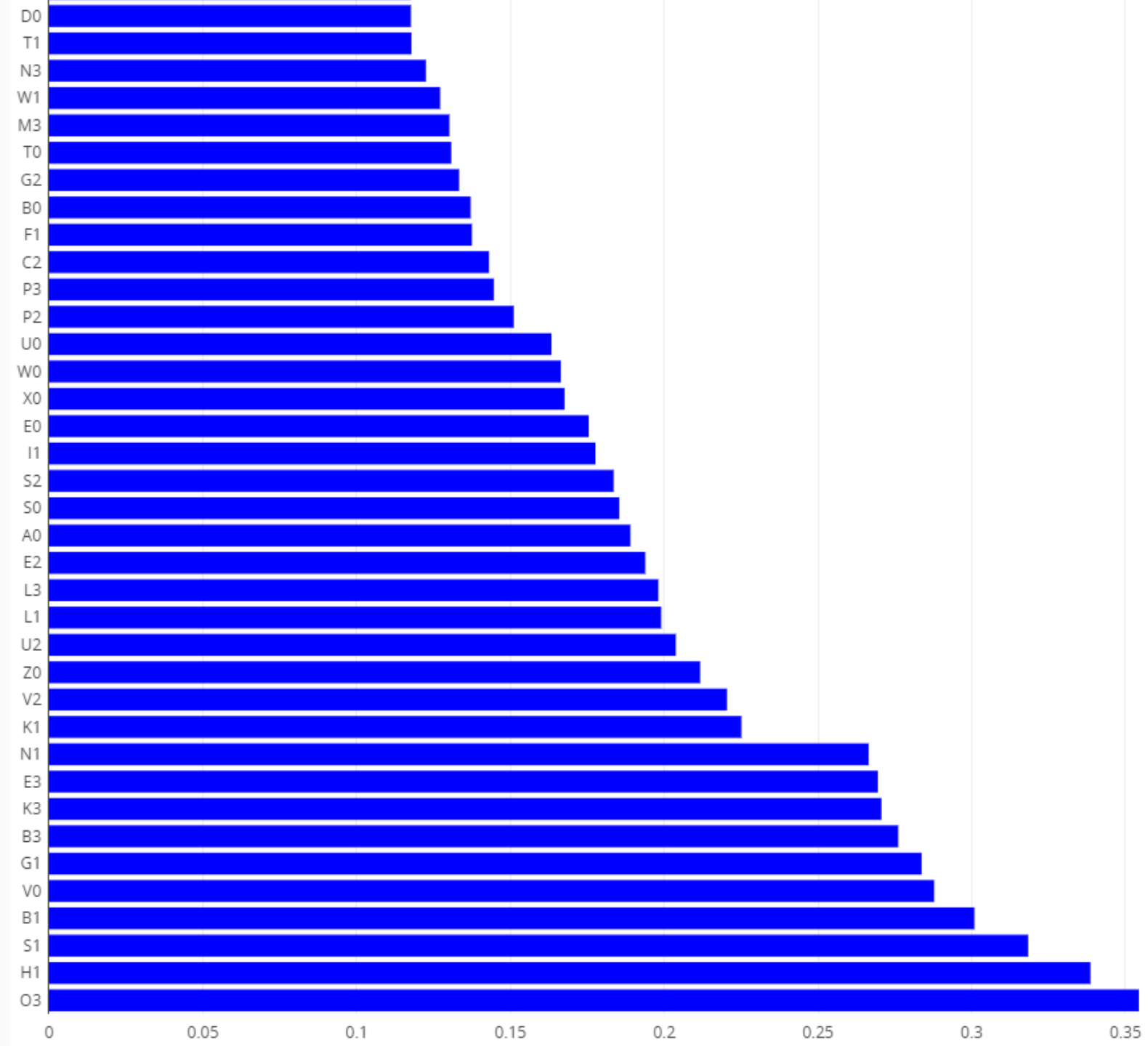
```
// Utilities for arrays of elements
public String showElements(ModelElement[] elements, String nomsg) {
    boolean found = false;
    StringBuffer res = new StringBuffer();
    if (elements != null) {
        Index.getInstance().setCurrentRenderer(
            FlatReferenceRenderer.getInstance());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









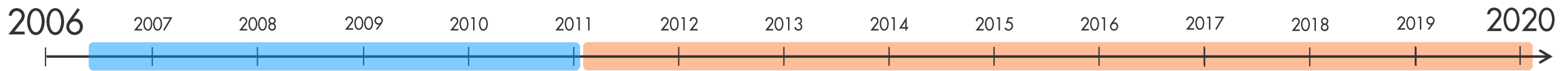




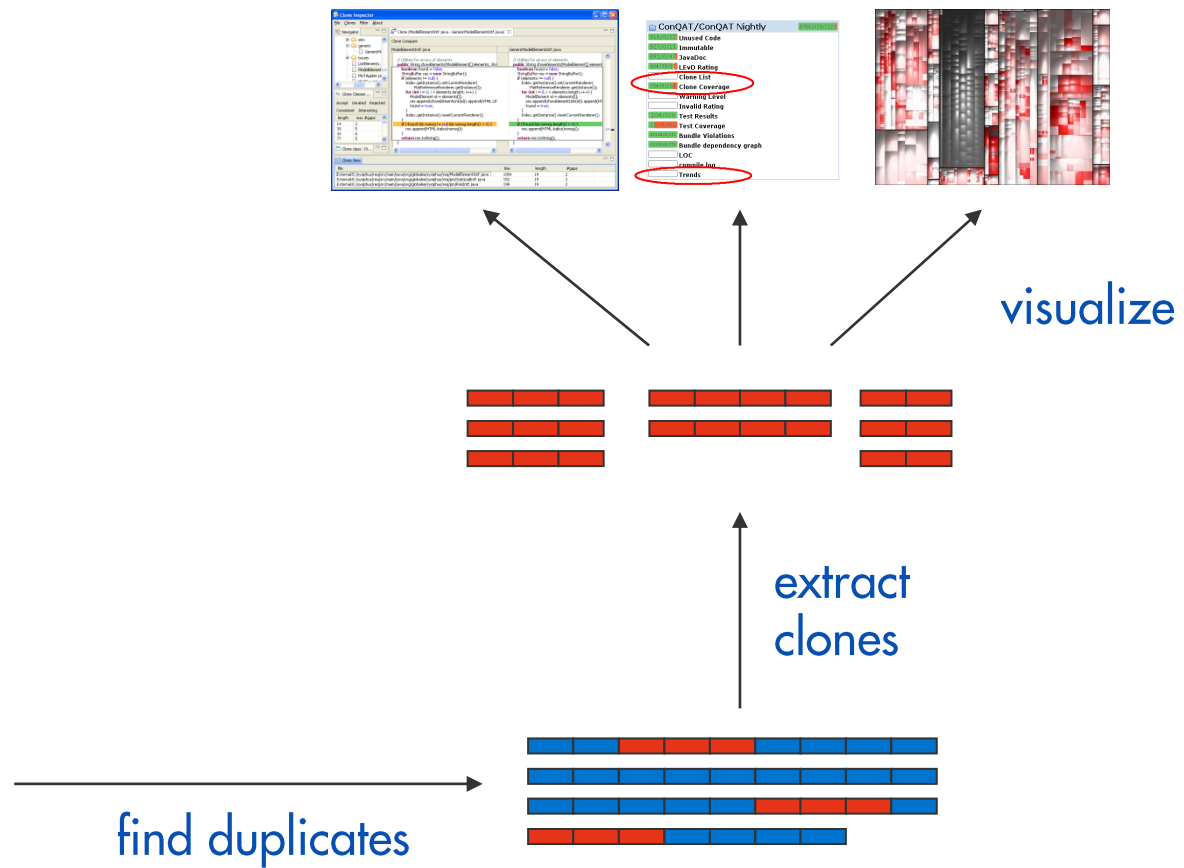
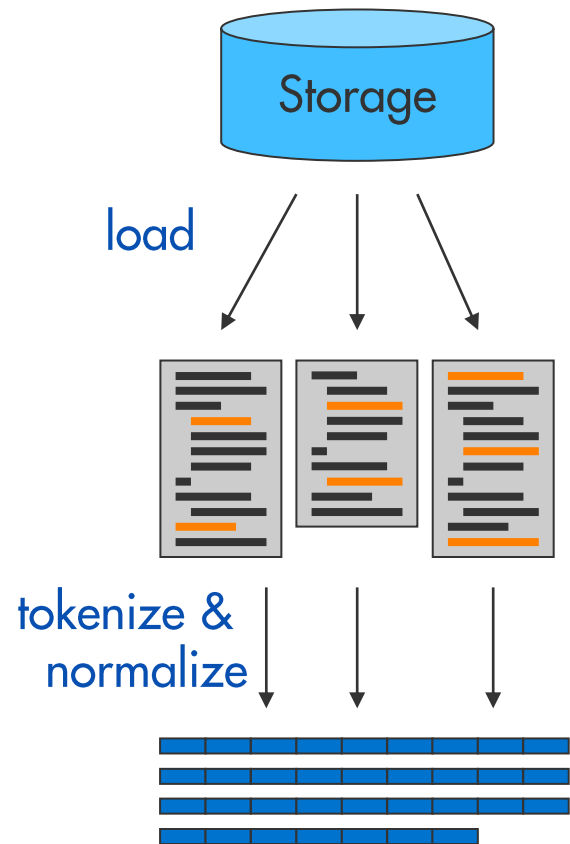
TUM

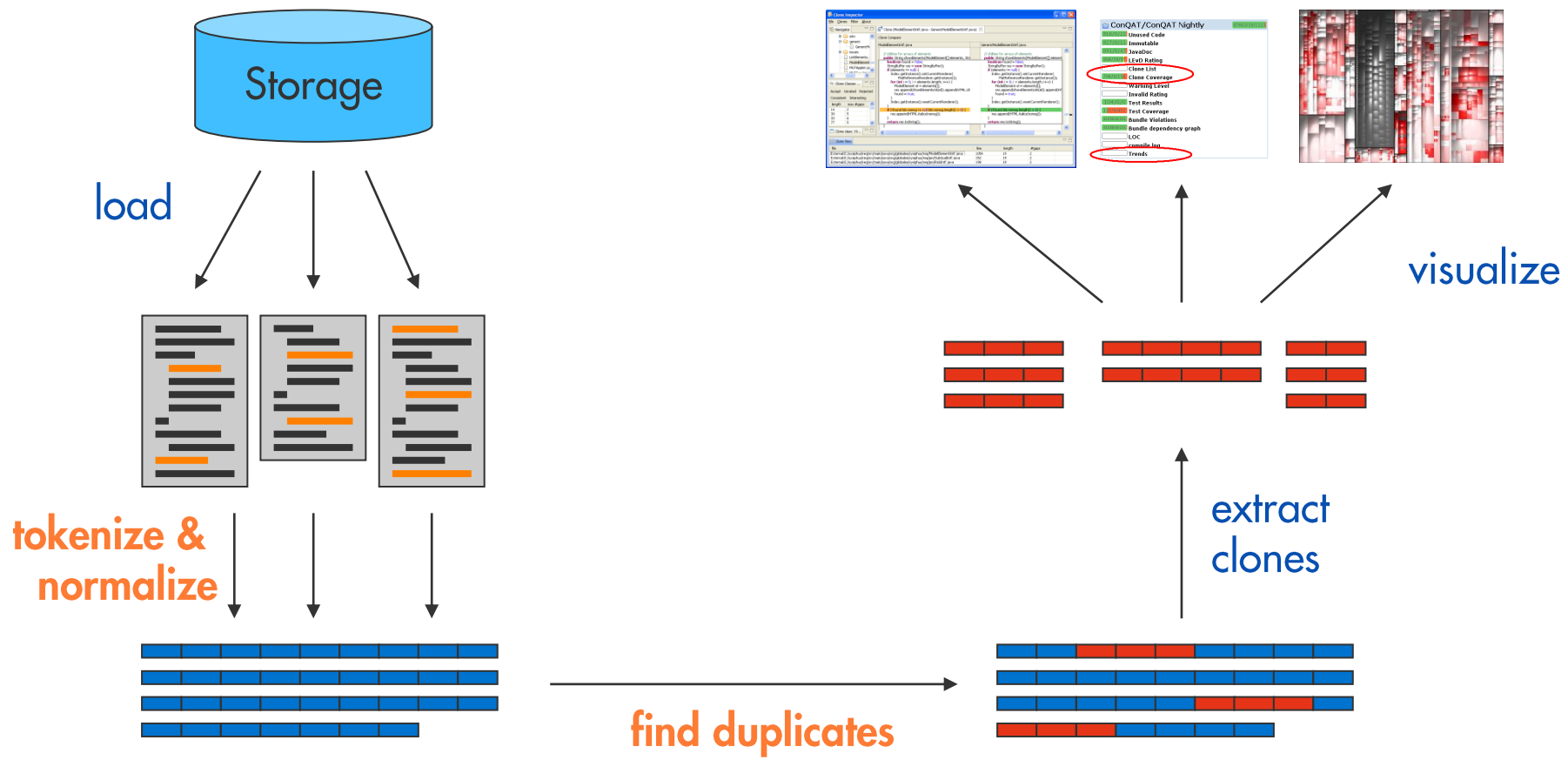


CQSE



Wie funktioniert Clone Detection?





Normalisierung

```
String readFileUtf8(File file) {  
    FileInputStream in = new FileInputStream(file);  
    byte[] buffer = new byte[file.length()];  
    in.read(buffer); in.close();  
    return new String(buffer, „UTF-8“);  
}
```

```
String readFileUtf16(File file) {  
    FileInputStream in = new FileInputStream(file);  
    byte[] buffer = new byte[file.length()];  
    in.read(buffer); in.close();  
    return new String(buffer, „UTF-16“);  
}
```

```
id0 id1(id2 id3) {  
    id0 id2 = new id0(id4);  
    id0[] id1 = new id0[id2.id3()];  
    id0.id1(id2); id0.id3();  
    return new id0(id1, lit0);  
}
```

```
id0 id1(id2 id3) {  
    id0 id2 = new id0(id4);  
    id0[] id1 = new id0[id2.id3()];  
    id0.id1(id2); id0.id3();  
    return new id0(id1, lit0);  
}
```


Normalisierung

```
String readFileUtf8(File file) {  
    FileInputStream in = new FileInputStream(file);  
    byte[] buffer = new byte[file.length()];  
    in.read(buffer); in.close();  
    return new String(buffer, „UTF-8“);  
}
```

```
String readFileUtf16(File file) {  
    FileInputStream in = new FileInputStream(file);  
    byte[] buffer = new byte[file.length()];  
    in.read(buffer); in.close();  
    return new String(buffer, „UTF-16“);  
}
```

```
id0 id1(id2 id3) {  
    id0 id2 = new id0(id4);  
    id0[] id1 = new id0[id2.id3()];  
    id0.id1(id2); id0.id3();  
    return new id0(id1, lit0);  
}
```

```
id0 id1(id2 id3) {  
    id0 id2 = new id0(id4);  
    id0[] id1 = new id0[id2.id3()];  
    id0.id1(id2); id0.id3();  
    return new id0(id1, lit0);  
}
```

```

BusinessRuleTaskXMLConverter.java
DefinitionsParser.java
model.setTargetNamespace(xtr.getAttributeValue(null, TARGET_NAMESPACE_ATTRIBUTE));
for (int i = 0; i < xtr.getNamespaceCount(); i++) {
    String prefix = xtr.getNamespacePrefix(i);
    if (StringUtils.isEmpty(prefix)) {
        model.addNamespace(prefix, xtr.getNamespaceURI(i));
    }
}

for (int i = 0; i < xtr.getAttributeCount(); i++) {
    ExtensionAttribute extensionAttribute = new ExtensionAttribute();
    extensionAttribute.setName(xtr.getAttributeLocalName(i));
    extensionAttribute.setValue(xtr.getAttributeValue(i));
    if (StringUtils.isEmpty(xtr.getAttributeNamespace(i)))
        extensionAttribute.setNamespace(xtr.getAttributeNamespaces(i));
    if (StringUtils.isEmpty(xtr.getAttributePrefix(i)))
        extensionAttribute.setNamespacePrefix(xtr.getAttributePrefixes(i));
    if (!BpmnXMLUtil.isBlacklisted(extensionAttribute, defaultAttributes))
        model.addDefinitionsAttribute(extensionAttribute);
}
}
}

```

Clone with 2 instances of length 10

Problems @ Javadoc Declaration Findings Orphans View

4 items

Description	Group
Clone with 2 instances of length 10	Code Duplication
Interface comment missing	Documentation
Interface comment missing	Documentation
Name violates naming convention: defaultAttributes. Should be	Naming

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

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

*
Makes sure that two collections are identical via {@link #assertEqualsDataBoundBeans(Object, Object)}
/
public void assertEqualsDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
assertEquals(lhs.size(), rhs.size());
}

```

jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.java

```

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]);

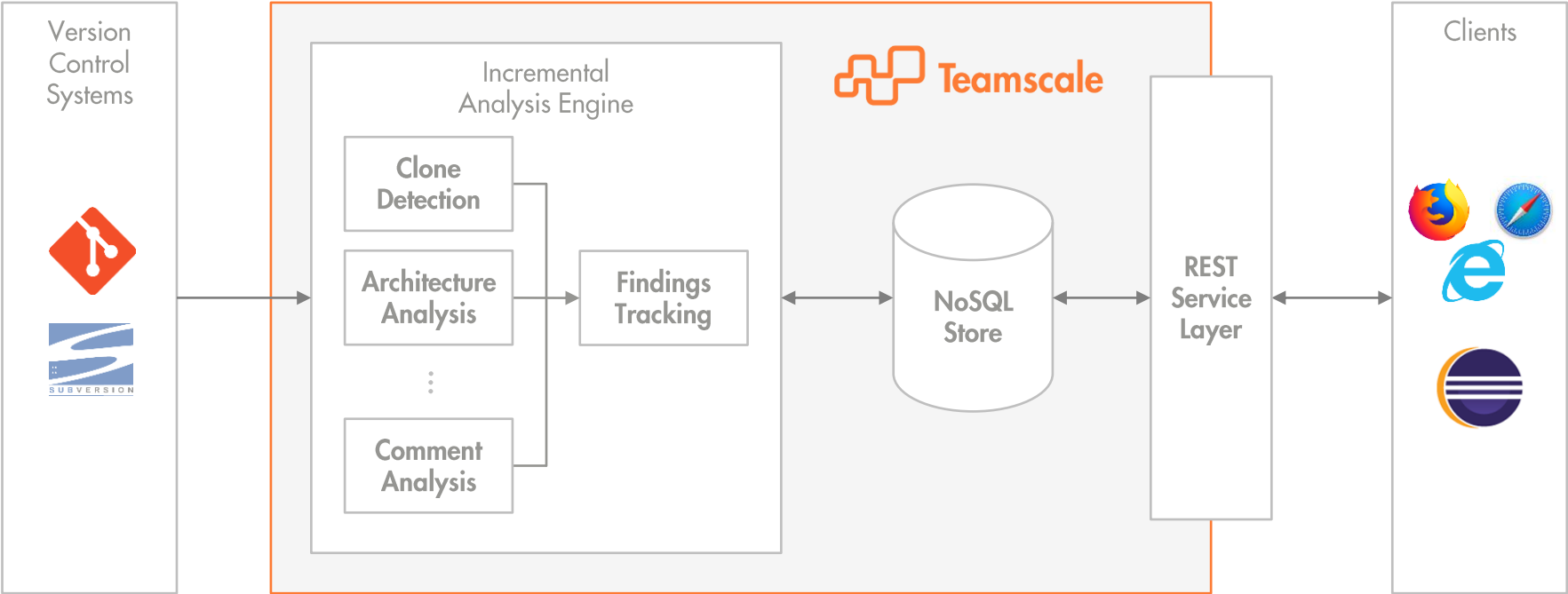
    if (lv != null && rv != null && Iterable.class.isAssignableFrom(types[i])) {
        Iterable lcol = (Iterable) lv;
        Iterable rcol = (Iterable) rv;
        Iterator ltr,rtr;
        for (ltr=lcol.iterator(), rtr=rcol.iterator(); ltr.hasNext() && rtr.hasNext();)
            Object litem = ltr.next();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

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

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

*
Makes sure that two collections are identical via {@link #assertEqualsDataBoundBeans(Object, Object)}
/
public void assertEqualsDataBoundBeans(List<?> lhs, List<?> rhs) throws Exception {
assertEquals(lhs.size(), rhs.size());
}

```



**Sind Clones ein Problem
für die Weiterentwicklung?**

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

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

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

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

jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.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());
assertThat("Data bound constructor mismatch. Different type?", (Constructor)rc, is((Cons

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

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

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

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

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

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



```

/* 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 * ) &xCl

}
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
    {

```

```

if i_kart-life_value < 0.
  write '50'          to it_down-newbs.
  i_kart-life_value = i_kart-life_value * -1.
  write i_kart-life_value to it_down-wrbtr
                        currency i_kost_ges-currency.

endif.
it_down-xblnr      = w_xblnr.
it_down-zuonr     = w_zuonr.
it_down-sgtxt     = w_sgtxt.
it_down-kostl     = i_kost_ges-fi_life_cost_center.
it_down-zzvber    = '10'.
it_down-bukrs     = i_kost_ges-booking.
*
if it_down-bukrs ne '0001'.
  write 'SA'        to it_down-blart.
  clear it_down-zzvber.
endif.
*
replace all occurrences of '.' in it_down-wrbtr with ''.
append it_down.
endloop.
*
describe table it_down lines l_it_down.
it_down-blkz = 'E'.
modify it_down index l_it_down transporting blkz.
endloop.
*
*--- Non-Life Quartal
loop at i_kost_ges where period_type = 'q' and
                        non_life_value ne 0.
                        and fehler ne 'X'.
*
if i_kost_ges-fibu_type = 'Y'.
  d_fibu_type = text-k01.

```

```

if i_kart-non_life_value < 0.
  write '50'          to it_down-newbs.
  i_kart-non_life_value = i_kart-non_life_value * -1.
  write i_kart-non_life_value to it_down-wrbtr
                        currency i_kost_ges-currency.

endif.
*
it_down-xblnr      = w_xblnr.
it_down-zuonr     = w_zuonr.
it_down-sgtxt     = w_sgtxt.
it_down-kostl     = i_kost_ges-fi_non_life_cost_center.
it_down-zzvber    = '10'.
it_down-bukrs     = i_kost_ges-booking.
*
if it_down-bukrs ne '0001'.
  write 'SA'        to it_down-blart.
  clear it_down-zzvber.
endif.
*
append it_down.
*
endloop.
describe table it_down lines l_it_down.
it_down-blkz = 'E'.
modify it_down index l_it_down transporting blkz.
endloop.
*
*--- Monatlich Health
loop at i_kost_ges where period_type = 'm' and
                        health_value ne 0.
                        fehler ne 'X'.
*
if i_kost_ges-fibu_type = 'Y'.
  d_fibu_type = text-k01.

```

larger/source/.../PriceElementUtil.plsql (revision default:1...)

```

223     AND NVL(t.valid_from, Database_SYS.Get_First_C
224     AND NVL(t.valid_to, Database_SYS.Get_Last_Cale
225
226 BEGIN
227     OPEN get_price_interval;
228     FETCH get_price_interval INTO valid_from_, valid_ur
229     CLOSE get_price_interval;
230     IF valid_from_ > Site_API.Get_Site_Date(contract_)
231         valid_date_ := valid_from_;
232     ELSE
233         valid_date_ := LEAST(NVL(valid_until_, TRUNC(Sit
234     END IF;
235
236     -- only calculate unit base (False) but no metal su
237     temp1_ :=0;
238
239     OPEN get_pl_currency_code;
240     FETCH get_pl_currency_code INTO pl_currency_code_;
241     CLOSE get_pl_currency_code;
242
243     FOR rec_ IN get_all_conn_unit_base_false LOOP

```

larger/source/.../PriceElementUtil.plsql (revision default:1...)

```

346 BEGIN
347     OPEN get_price_interval;
348     FETCH get_price_interval INTO valid_from_, valid_ur
349     CLOSE get_price_interval;
350     IF valid_from_ > Site_API.Get_Site_Date(contract_)
351         valid_date_ := valid_from_;
352     ELSE
353         -- Change for Ticket #1234 (START)
354         --valid_date_ := LEAST(NVL(valid_until_, TRUNC($
355         valid_date_ := LEAST(NVL(valid_until_, Site_API.
356         -- Change for Ticket #1234 (FINISH)
357     END IF;
358
359     -- calculate only metal surcharge without unit base
360     temp3_ :=0;
361
362     OPEN get_pl_currency_code;
363     FETCH get_pl_currency_code INTO pl_currency_code_;
364     CLOSE get_pl_currency_code;
365
366     FOR rec_ IN get_all_metal_surcharge LOOP

```

```

54     orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, polLocation.County)
55     orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, null)
56 })
57 tierQuery.or(\orCriteria -> {
58     orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, polLocation.City)
59     orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, null)
60 })
61 tierQuery.or(\orCriteria -> {
62     orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, polLocation.PostalCode)
63     orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, null)
64 })
65
66 var results = tierQuery.select()
67 if(results.HasElements) {
68     var result = results.firstWhere( \ elt -> elt.ZipCode == polLocation.PostalCode and
69         elt.County == polLocation.County and elt.City == polLocation.City)
70     var result2 = results.firstWhere( \ elt -> elt.County == polLocation.County and elt.City == polLocation.City)
71     var result3 = results.firstWhere( \ elt -> elt.County == polLocation.County)
72
73     if(result != null) {

```

```

54     orCriteria.compareIgnoreCase(TerrorismTargetTier_Ext#County, Equals, polLocation.County)
55     orCriteria.compare(TerrorismTargetTier_Ext#County, Equals, null)
56 })
57 tierQuery.or(\orCriteria -> {
58     orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, polLocation.City)
59     orCriteria.compare(TerrorismTargetTier_Ext#City, Equals, null)
60 })
61 tierQuery.or(\orCriteria -> {
62     orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, polLocation.PostalCode)
63     orCriteria.compare(TerrorismTargetTier_Ext#ZipCode, Equals, null)
64 })
65
66 var results = tierQuery.select()
67 if(results.HasElements) {
68     var result = results.firstWhere( \ elt -> elt.ZipCode == polLocation.PostalCode and
69         elt.County == polLocation.County and elt.City == polLocation.City)
70     var result2 = results.firstWhere( \ elt -> elt.County == polLocation.County and elt.City == polLocation.City)
71     // - Defect 224 - Fix for defaulting target tier value irrespective of lower and upper
72     var result3 = results.firstWhere( \ elt -> elt.County.equalsIgnoreCase(polLocation.County))
73
74     if(result != null) {

```

Study

Munich Re 

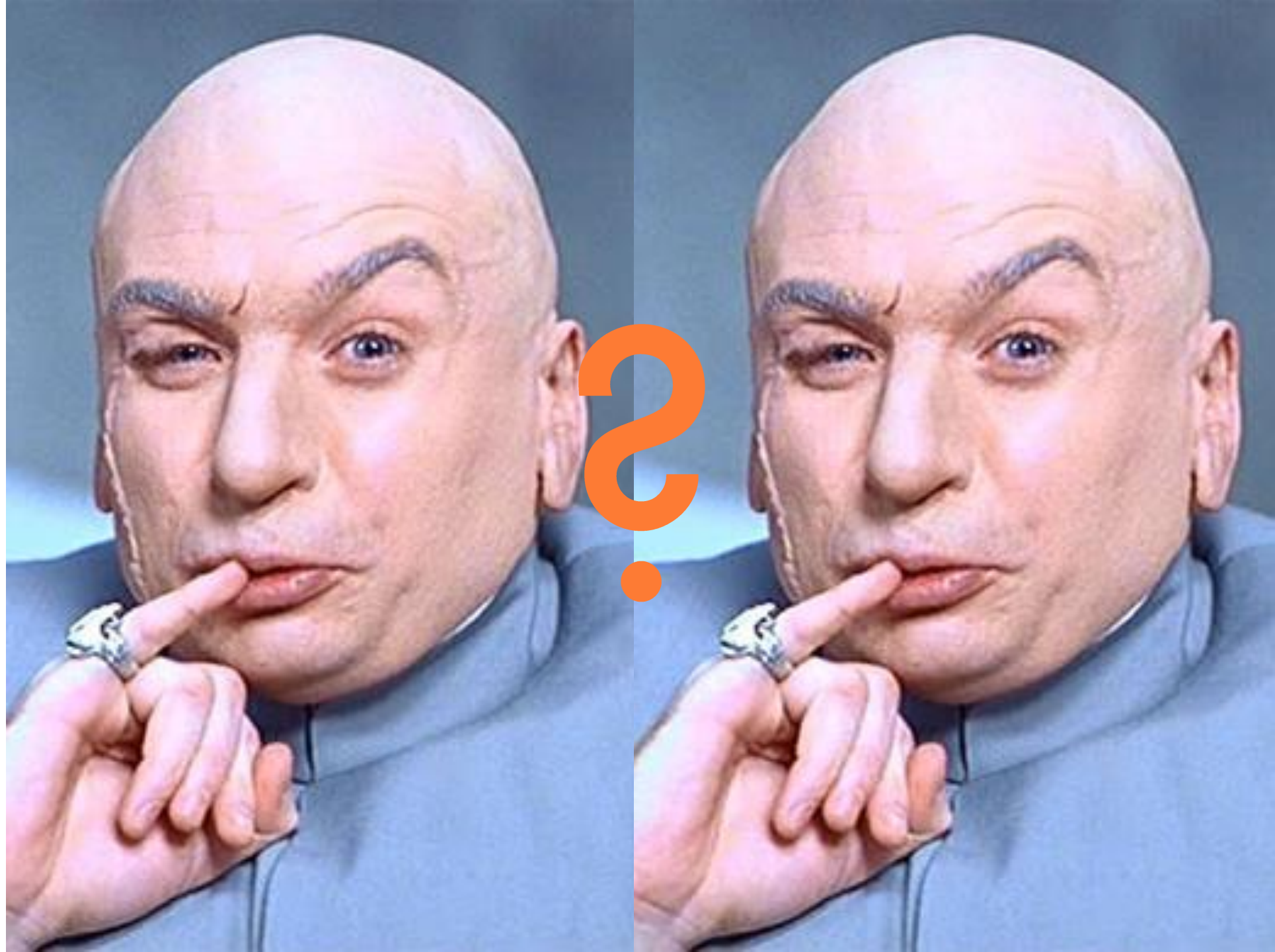
- Over 100 Bugs in produktive Software



- 52% of all unintentional inconsistencies are buggy

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

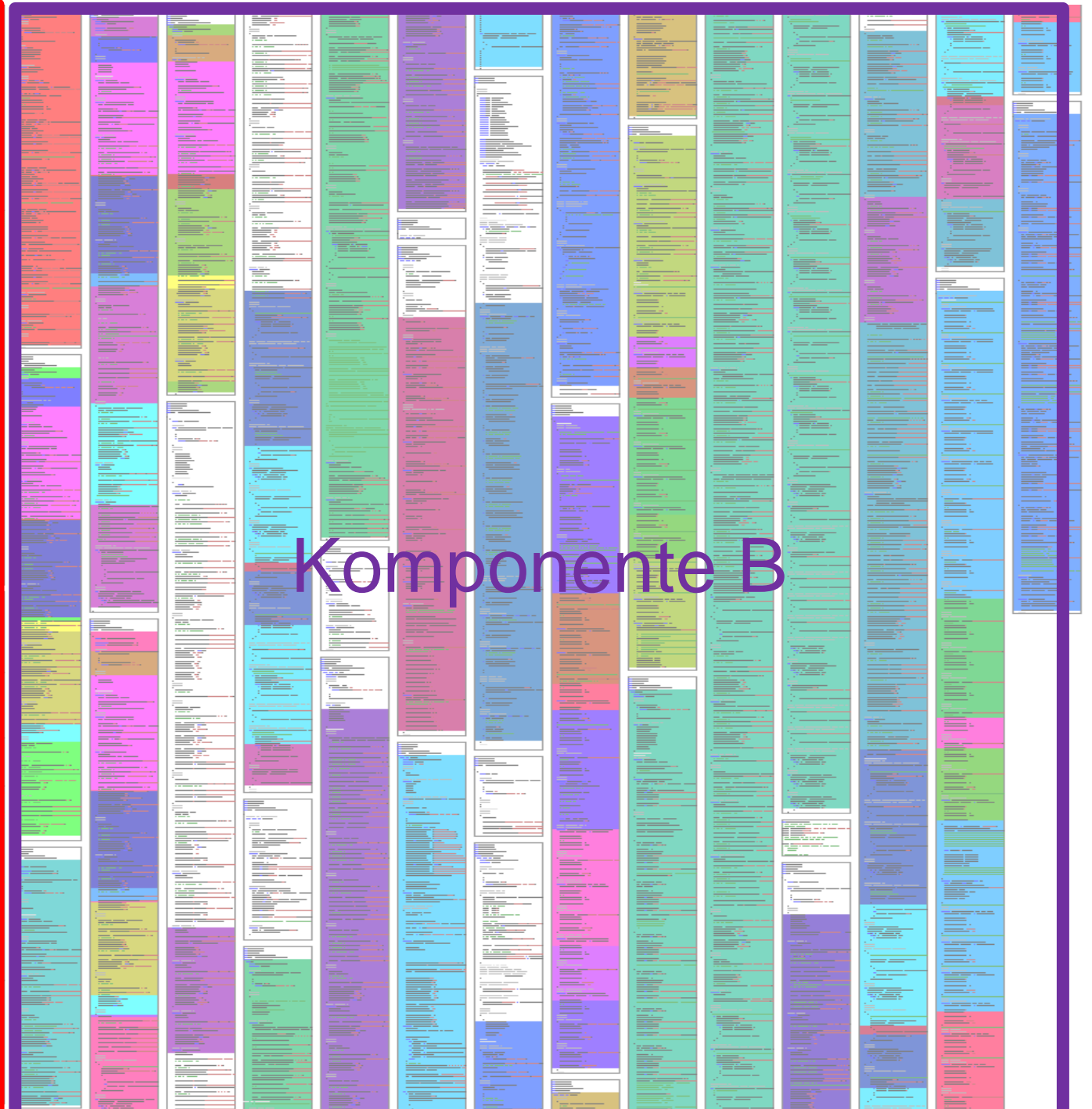
Was tun?



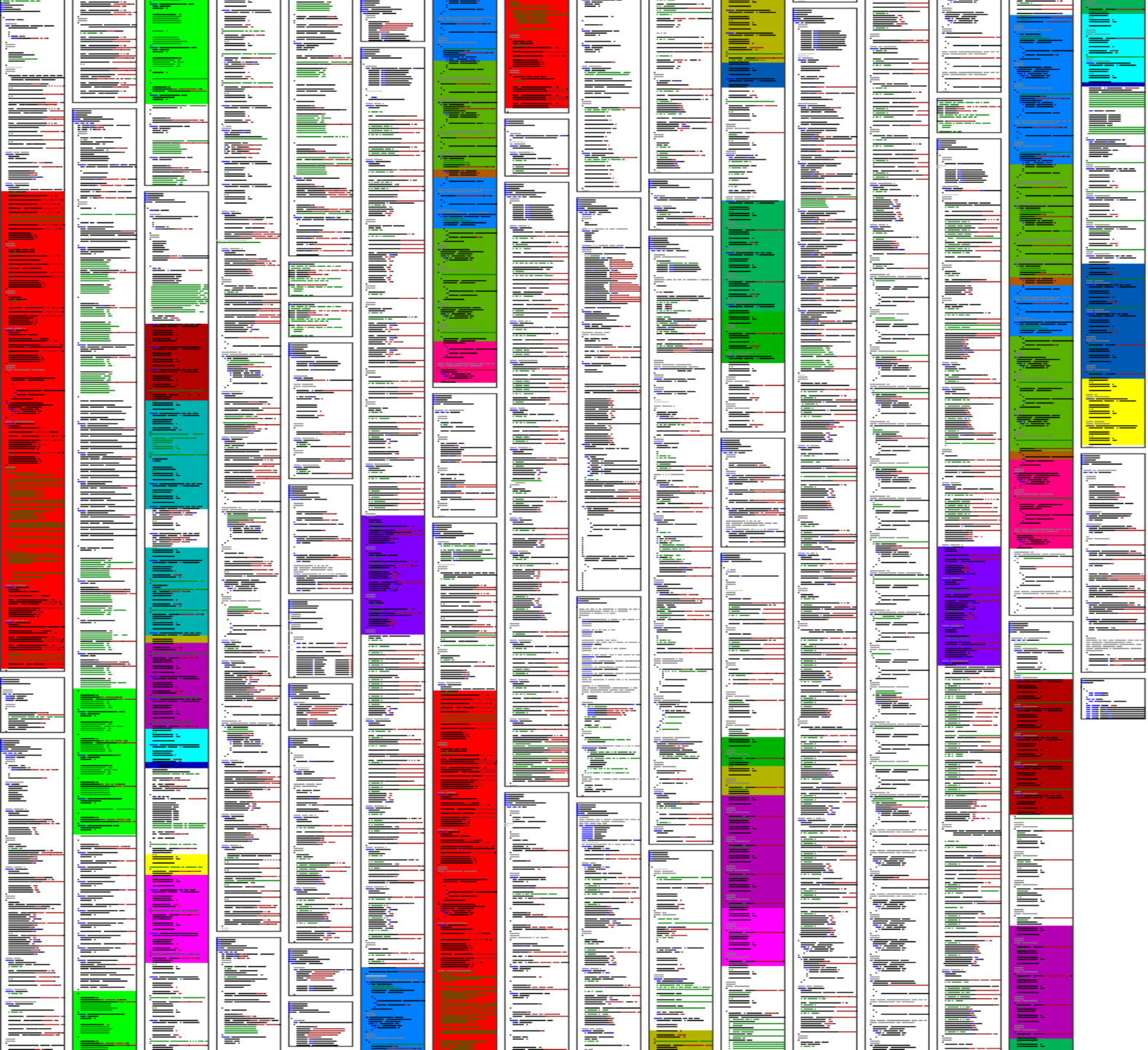




Komponente A



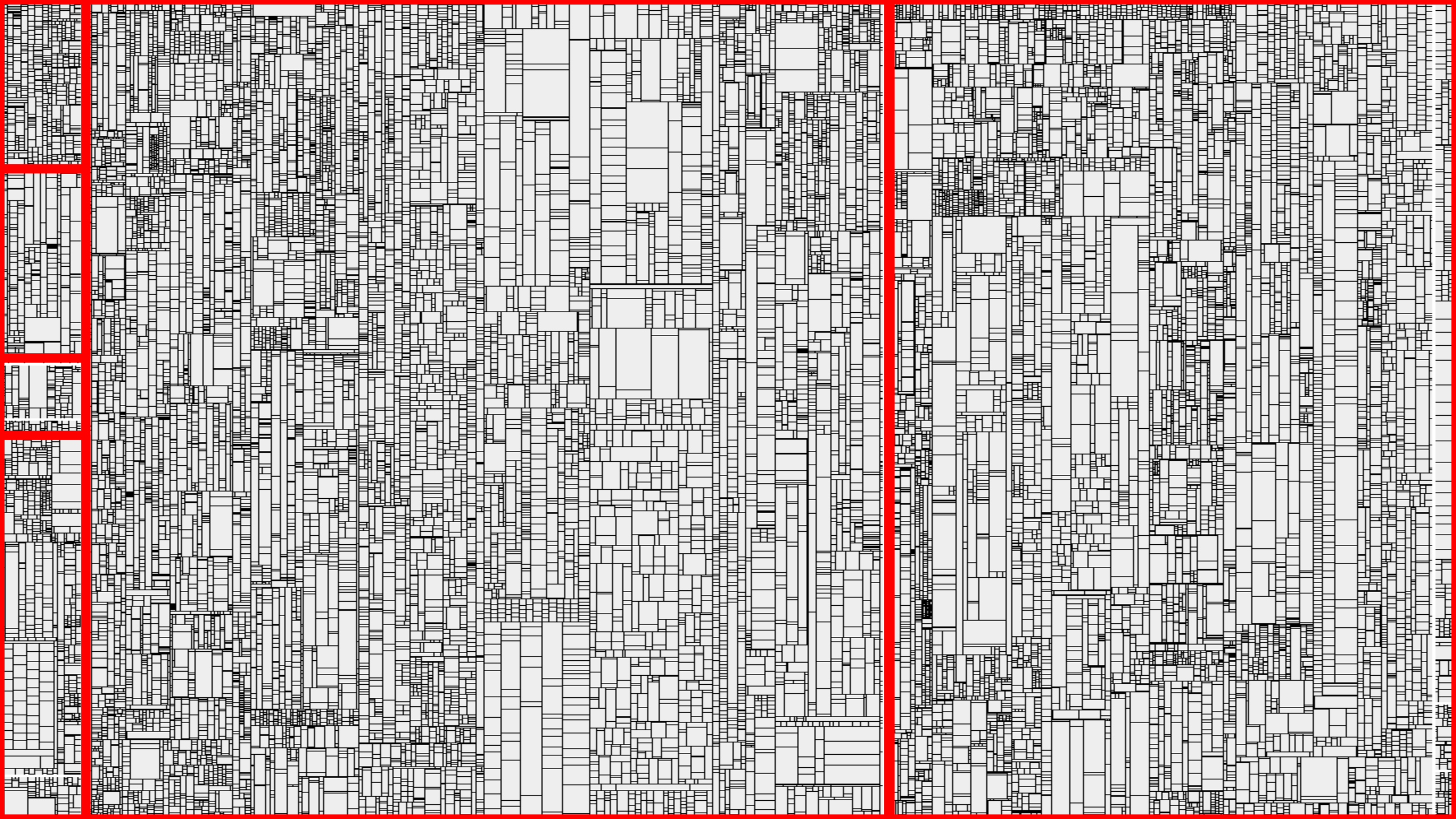
Komponente B

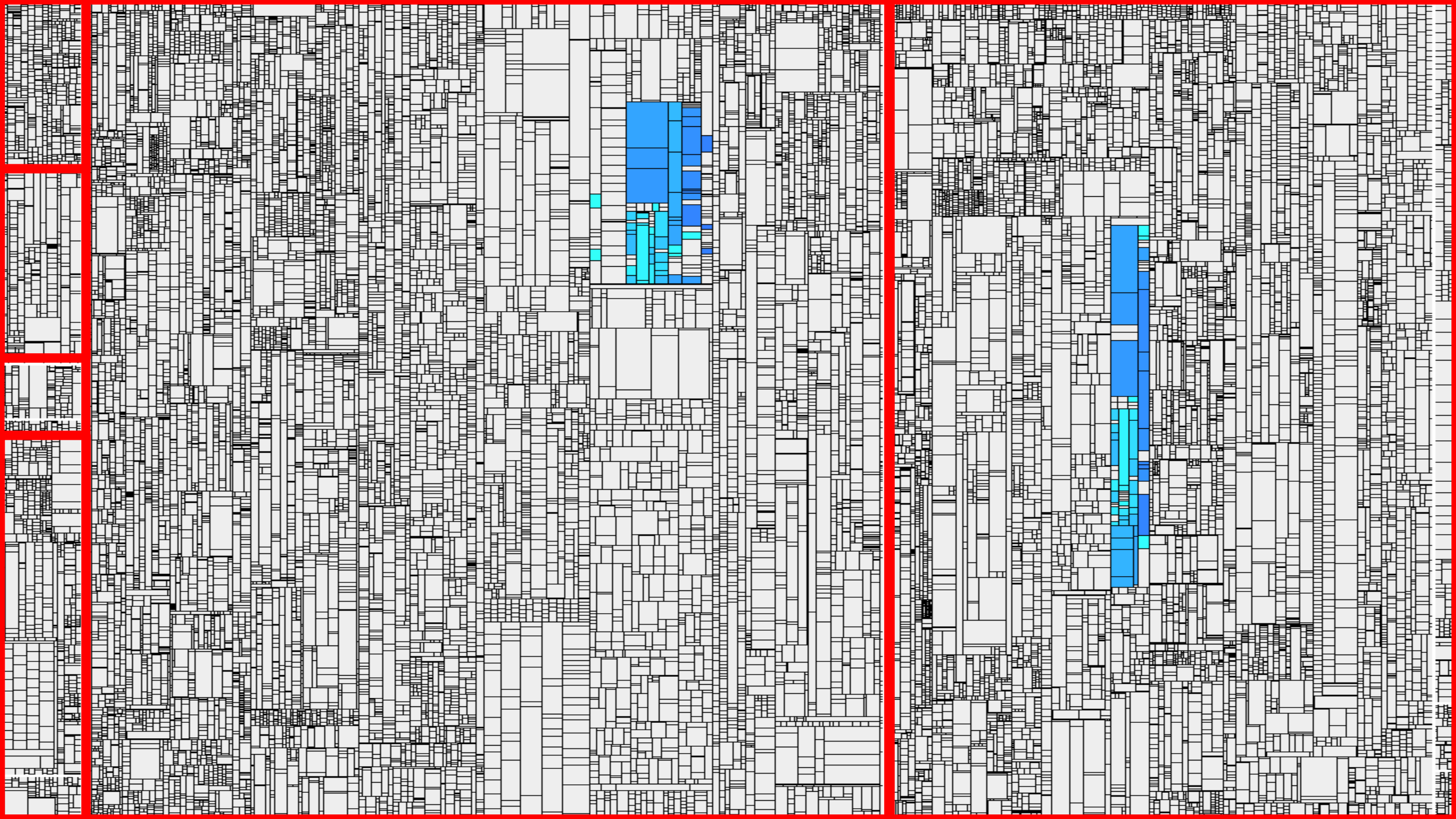


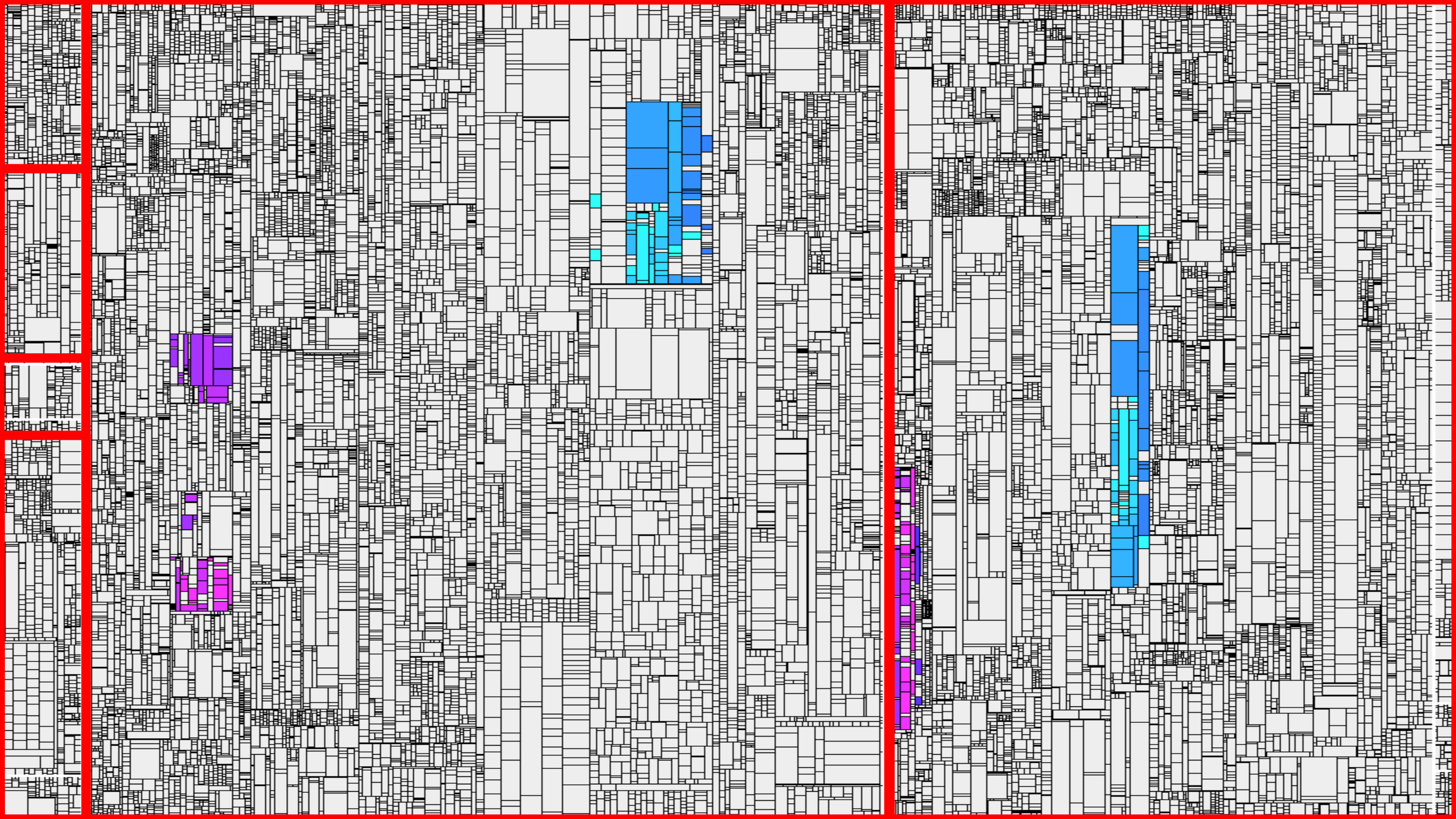
Ad-Hoc Reuse in a Build-System

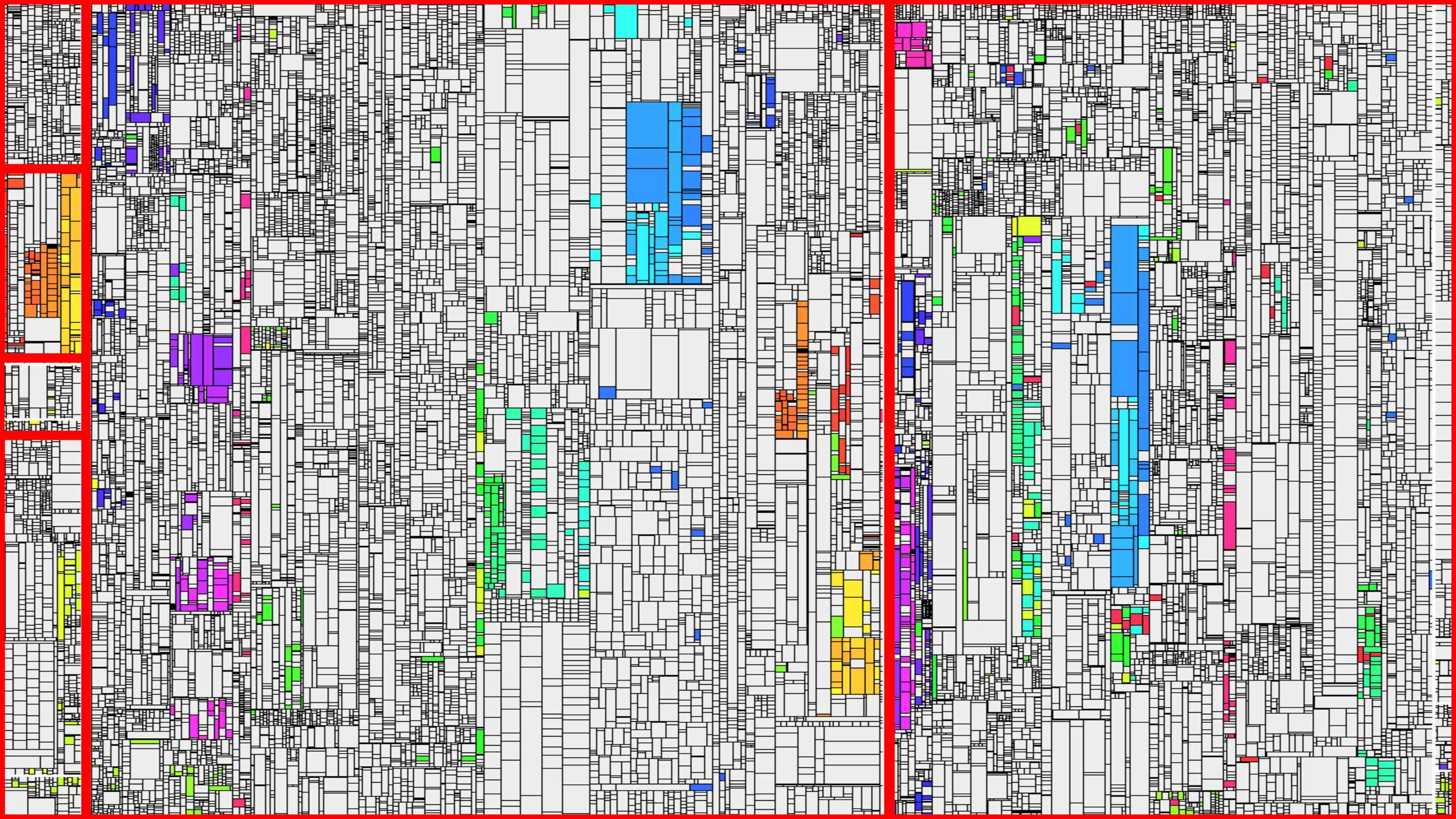


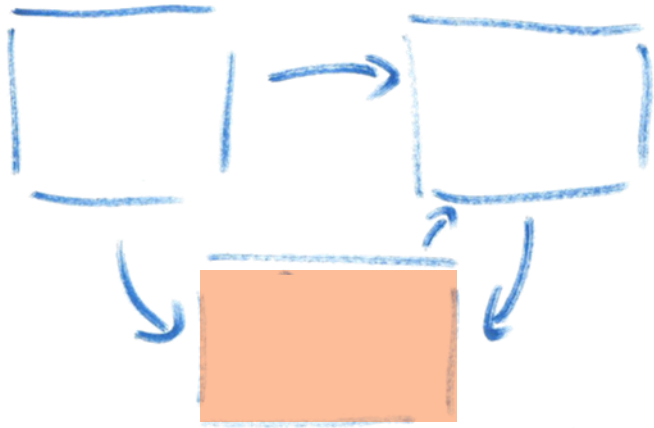
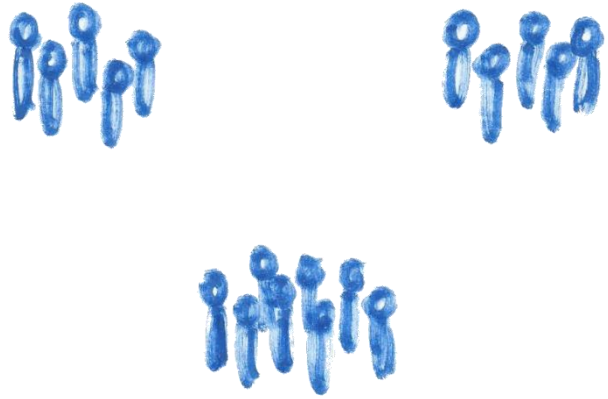
	Ad hoc Reuse
Build Definitions	81
Lines of Code	457.388
Clone Coverage	97,5%
Blow-Up	929%

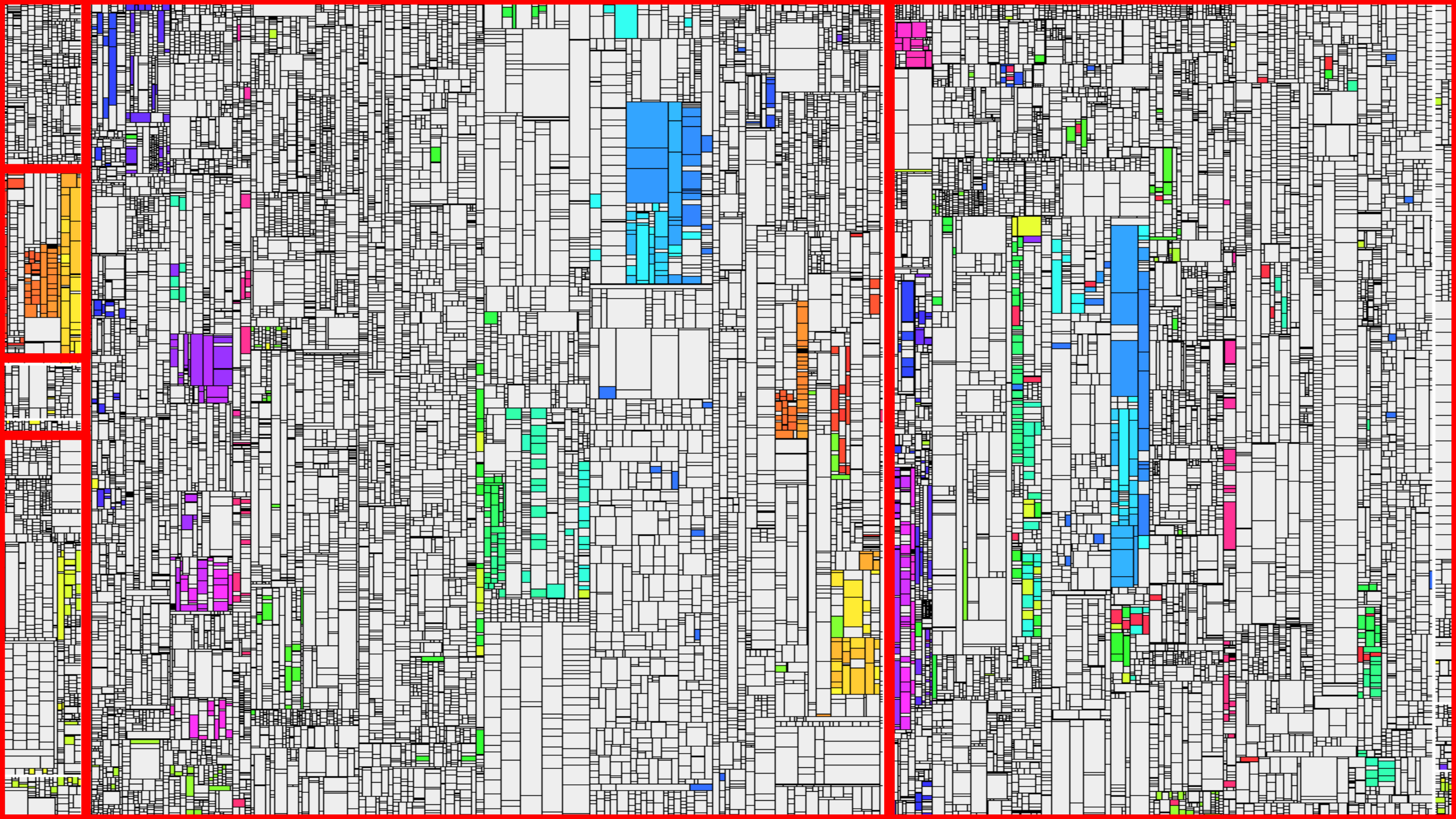












- Dashboard
- Activity
- Findings
- Metrics
- Tests
- Issues
- Tasks
- Architecture
- Delta
- Projects
- System
- Admin

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

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

*
Makes sure that two collections are identical via {@link #assertEqualDataBoundBeans(Object
/
public void assertEqualsDataBoundBeans(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) {
                assertEqualsDataBoundBeans(litem,ritem);
            } else {
                assertThat(ritem, is(litem));
            }
        }
        assertThat("collection size mismatch between " + lhs + " and " + rhs, ltr.hasNext()
            is(false));
    } else
    if (findDataBoundConstructor(types[i])!=null || (lv!=null && findDataBoundConstructo
        // recurse into nested databound objects
        assertEqualsDataBoundBeans(lv,rv);
    } else {
        primitiveProperties.add(names[i]);
    }
}

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

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



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](#)

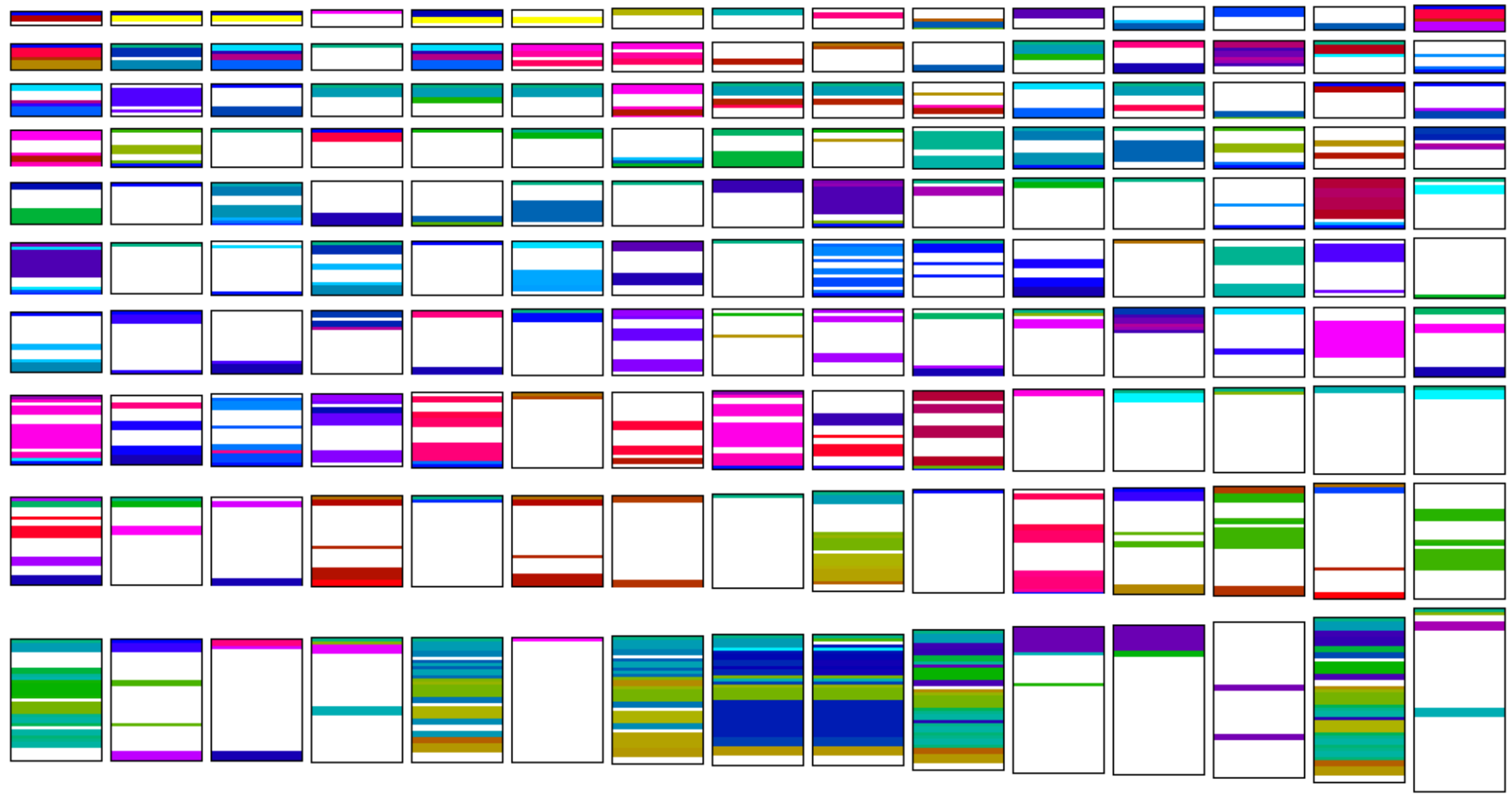
Code Duplication / Cloning

[Clone with 2 instances of length 10](#)

[src/java/net/sf/.../RightClickMenu.java:340-354](#)

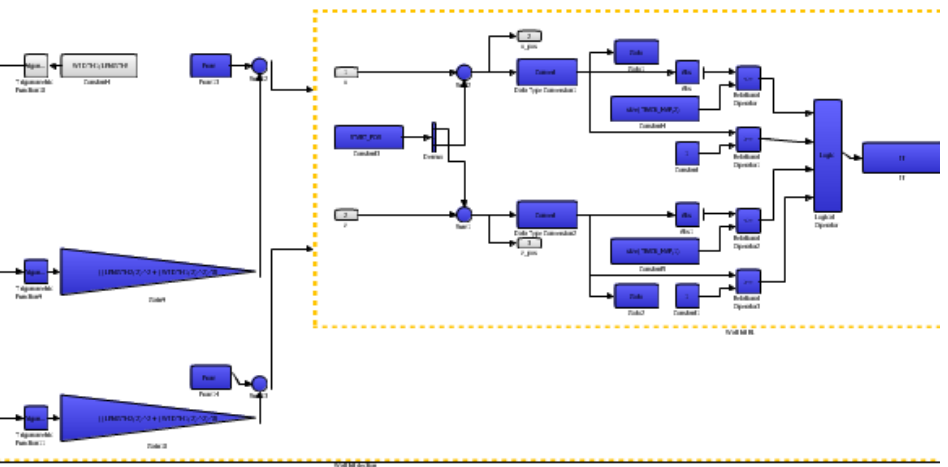
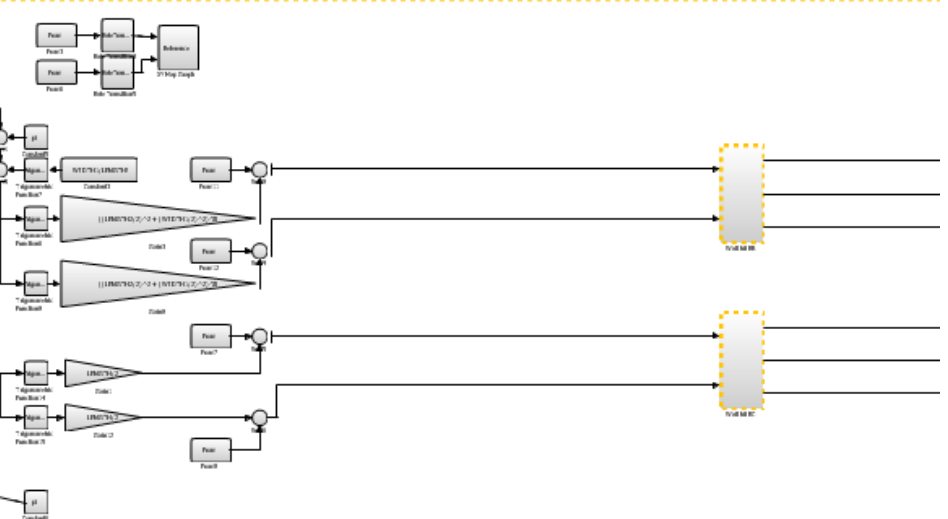
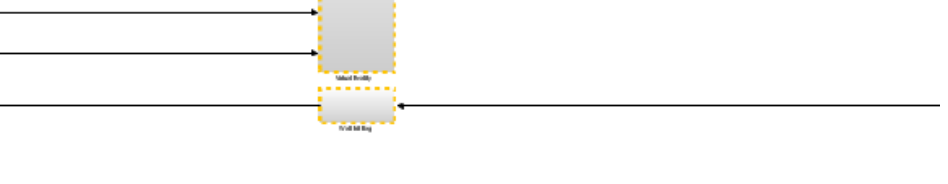
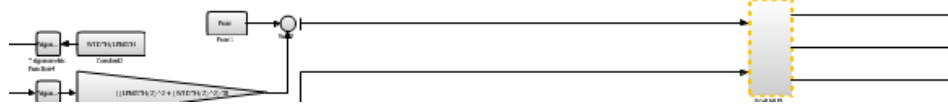
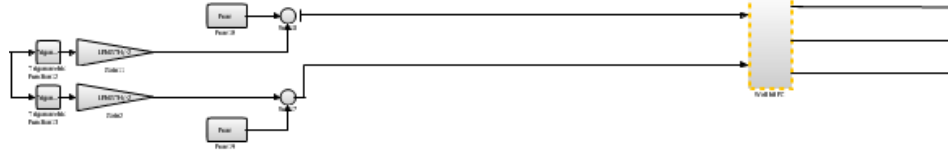
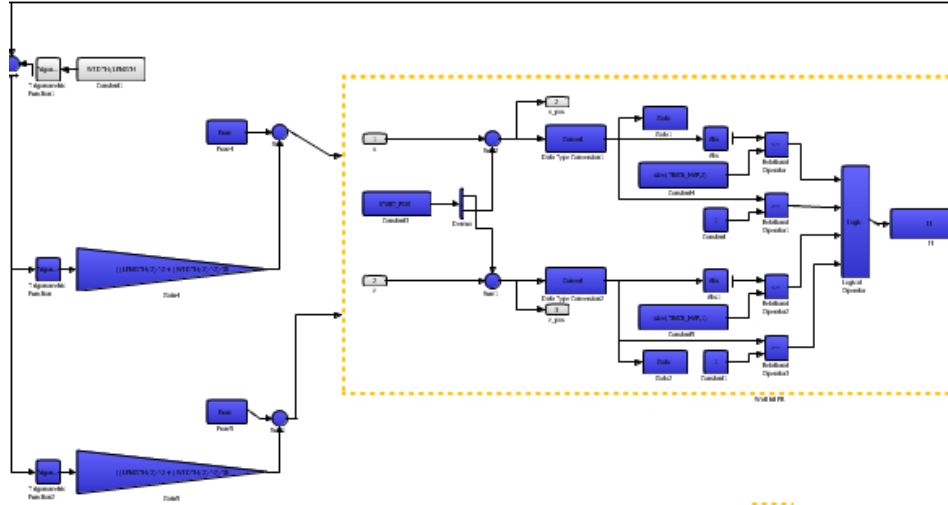
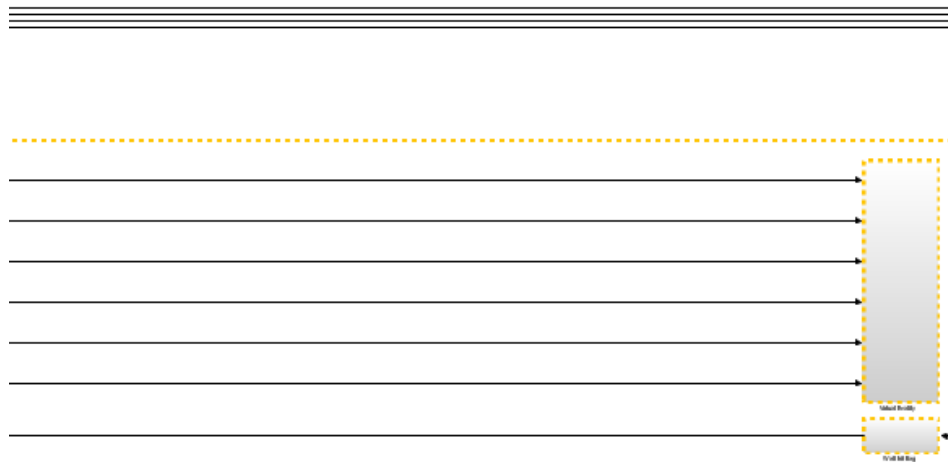
Code Duplication / Cloning

Software ist mehr als Code

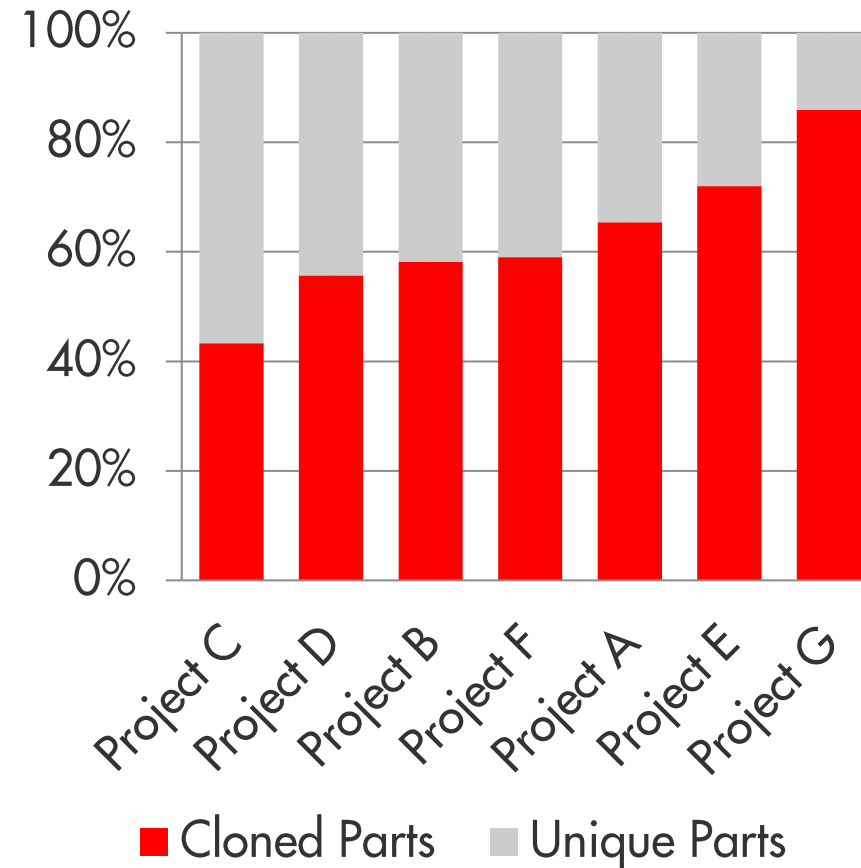


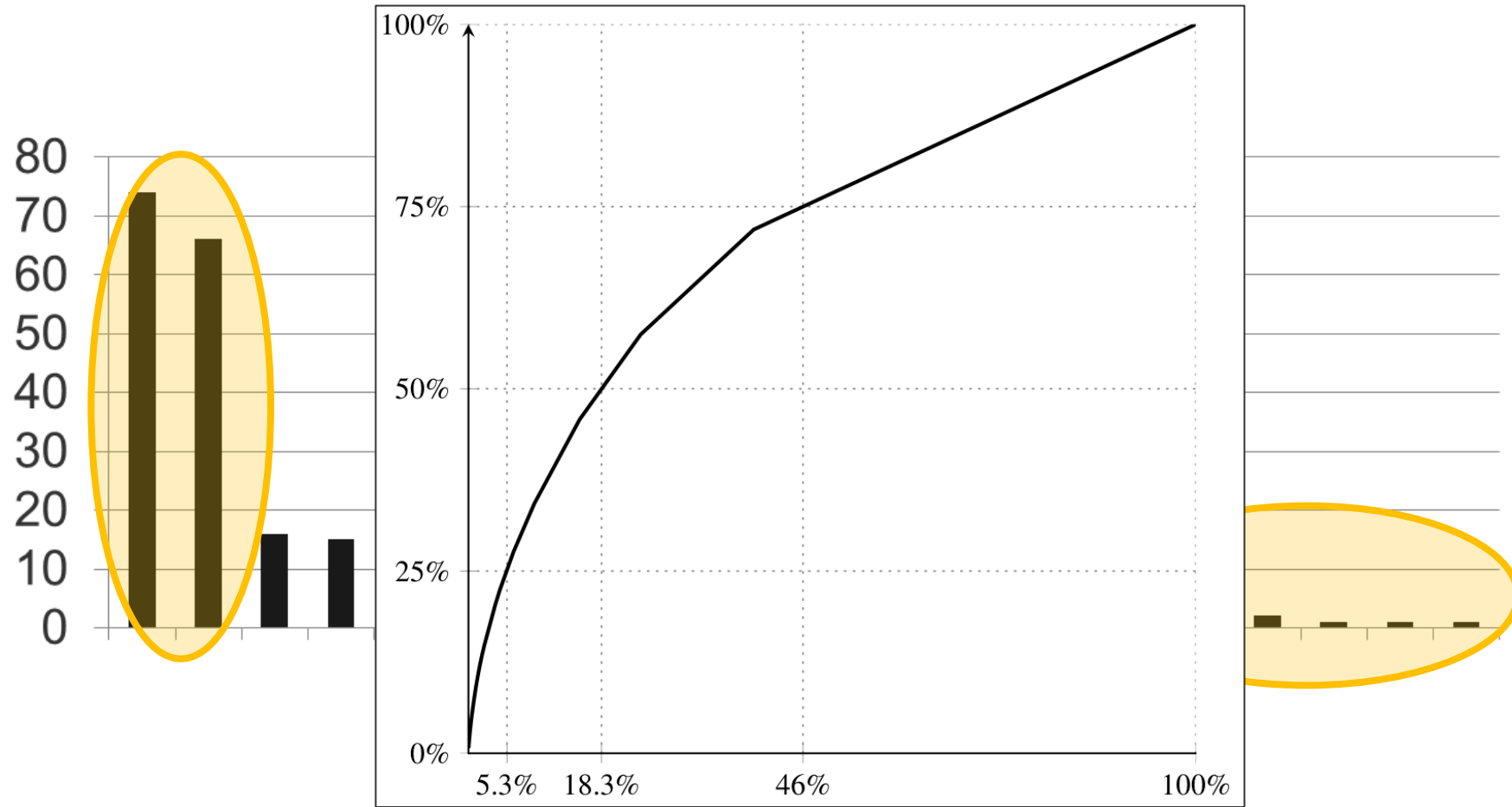
176 use cases in total, 150 contain cloning

Spec	Pages	Words	Clone cov.	Clone groups	Clones	Blow-up relative	Blow-up words
A	517	41,482	35.0%	259	914	32.6%	10,191
B	1,013	130,968	8.9%	265	639	5.3%	6,639
C	133	18,447	18.5%	37	88	11.5%	1,907
D	241	37,969	8.1%	105	479	6.9%	2,463
E	185	37,056	0.9%	6	12	0.4%	161
F	42	7,662	51.1%	50	162	60.6%	2,890
H	160	19,632	71.6%	71	360	129.6%	11,083
X	158	19,679	12.4%	21	45	6.8%	1,253
Y	235	49,425	21.9%	181	553	18.2%	7,593
Z	-	13,807	19.6%	50	117	14.2%	1,718
AB	3,100	274,489	12.1%	635	1818	8.7%	21,993
AC	696	81,410	5.4%	65	148	3.2%	2,549
Avg			13.6%			13.5%	
Σ	8,667	1,242,765		2,631	7,669		100,178



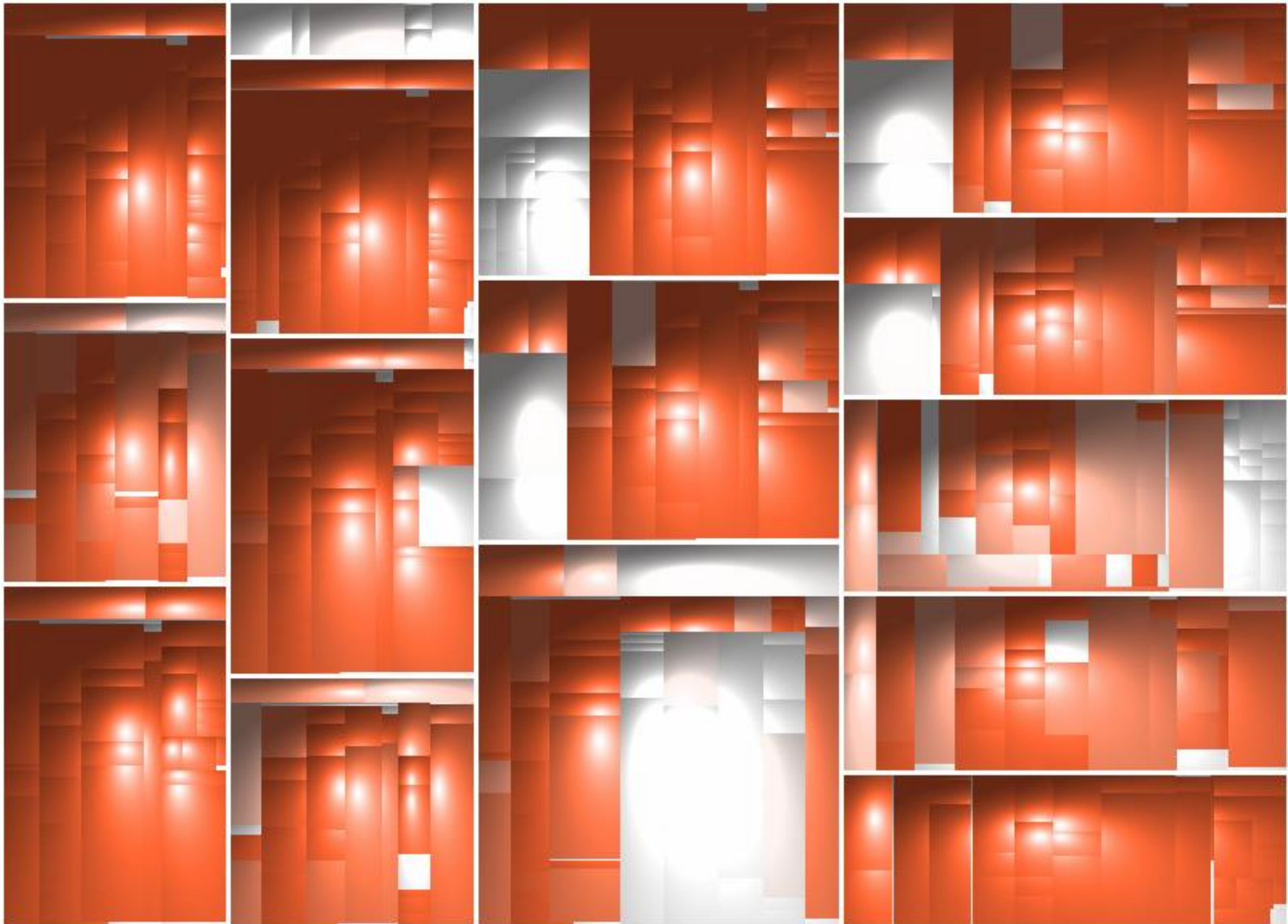
	System under Test	Test Suite
		# Tests
System A	330 kLoC	266
System B	580 kLoC	1,059
System C	150 kLoC	72
System D	430 kLoC	180
System E	760 kLoC	1,804
System F	1,400 kLoC	135
System G	160 kLoC	605



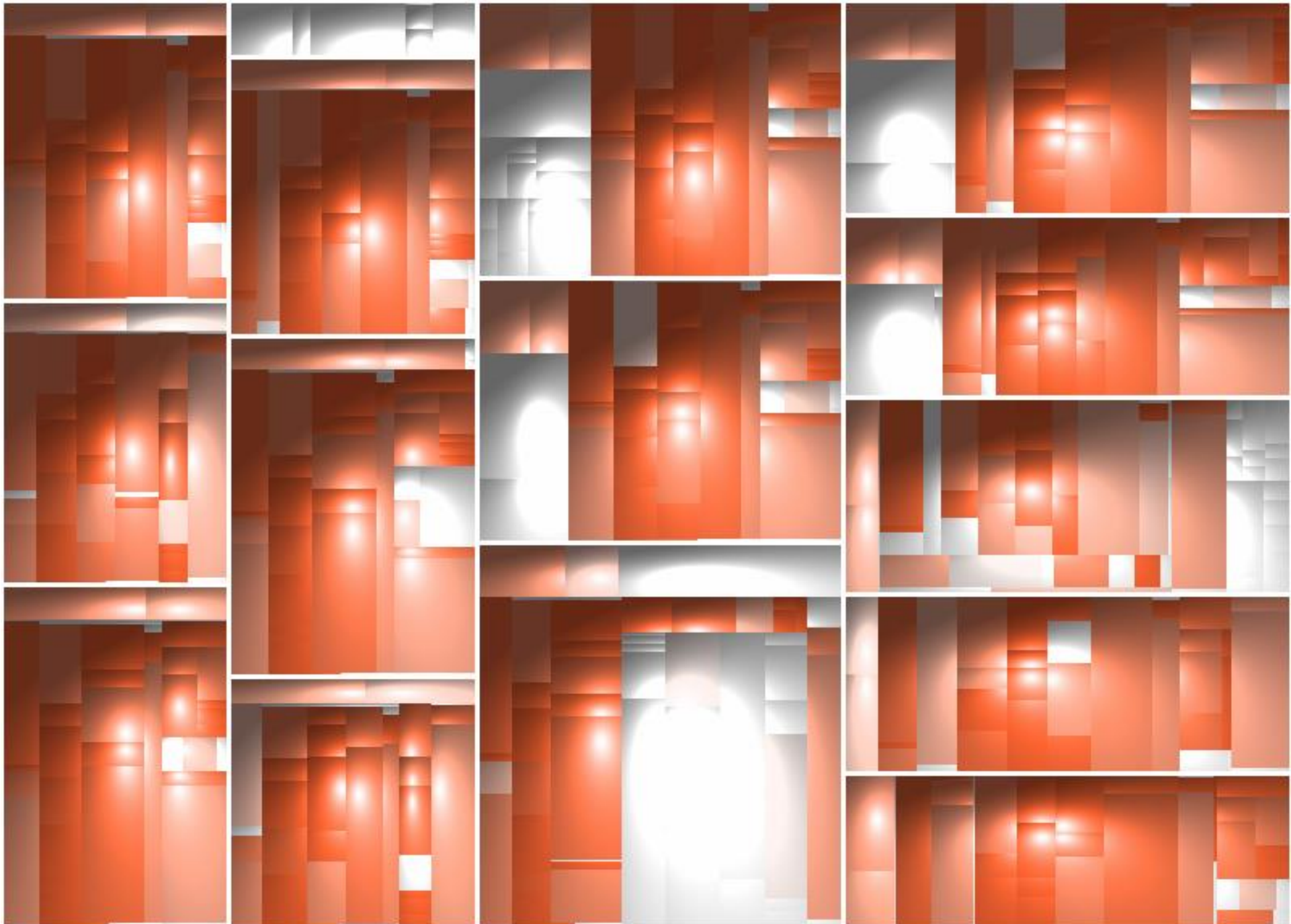


Evolution von Produktlinien

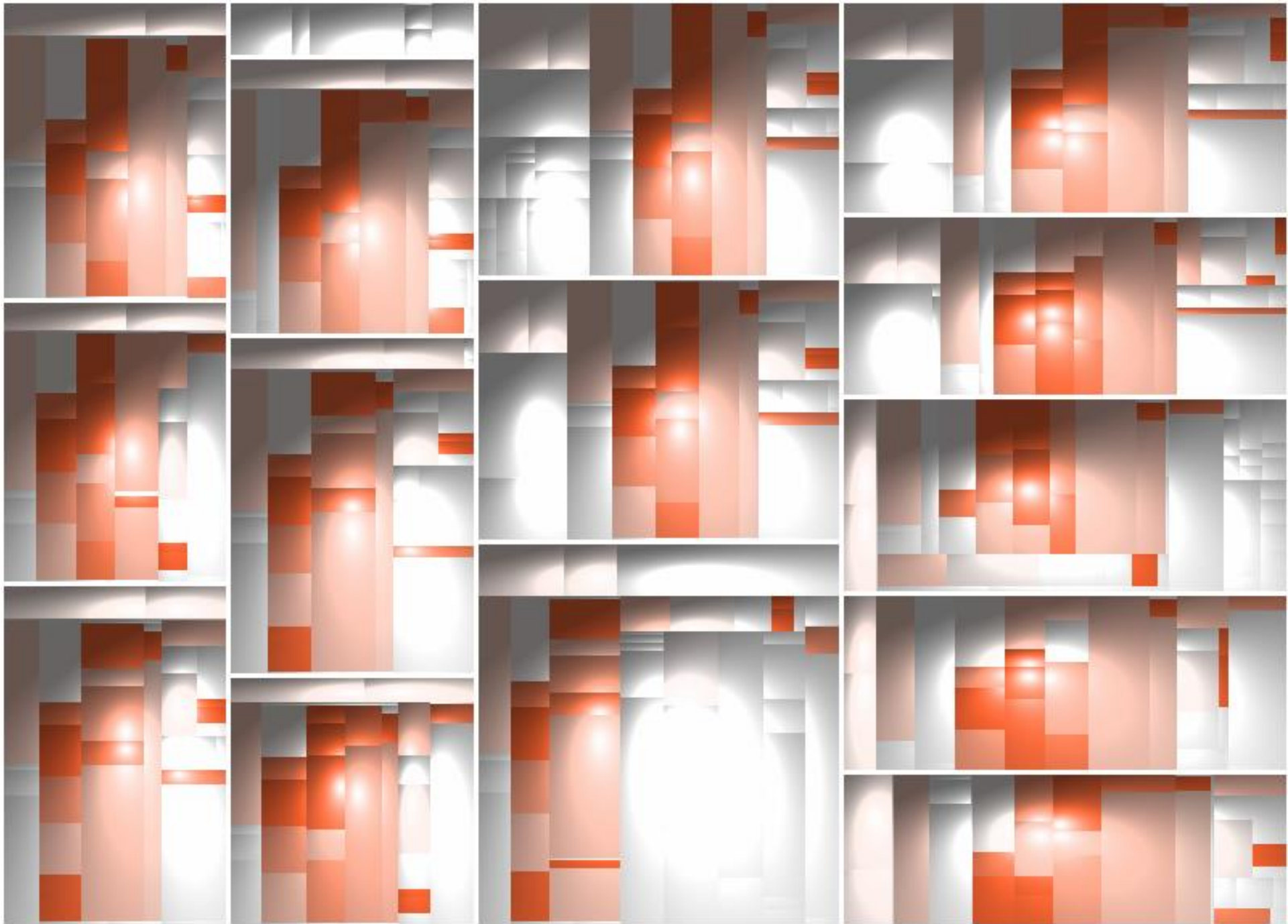




5 Projects



10 Projects



14 Projects

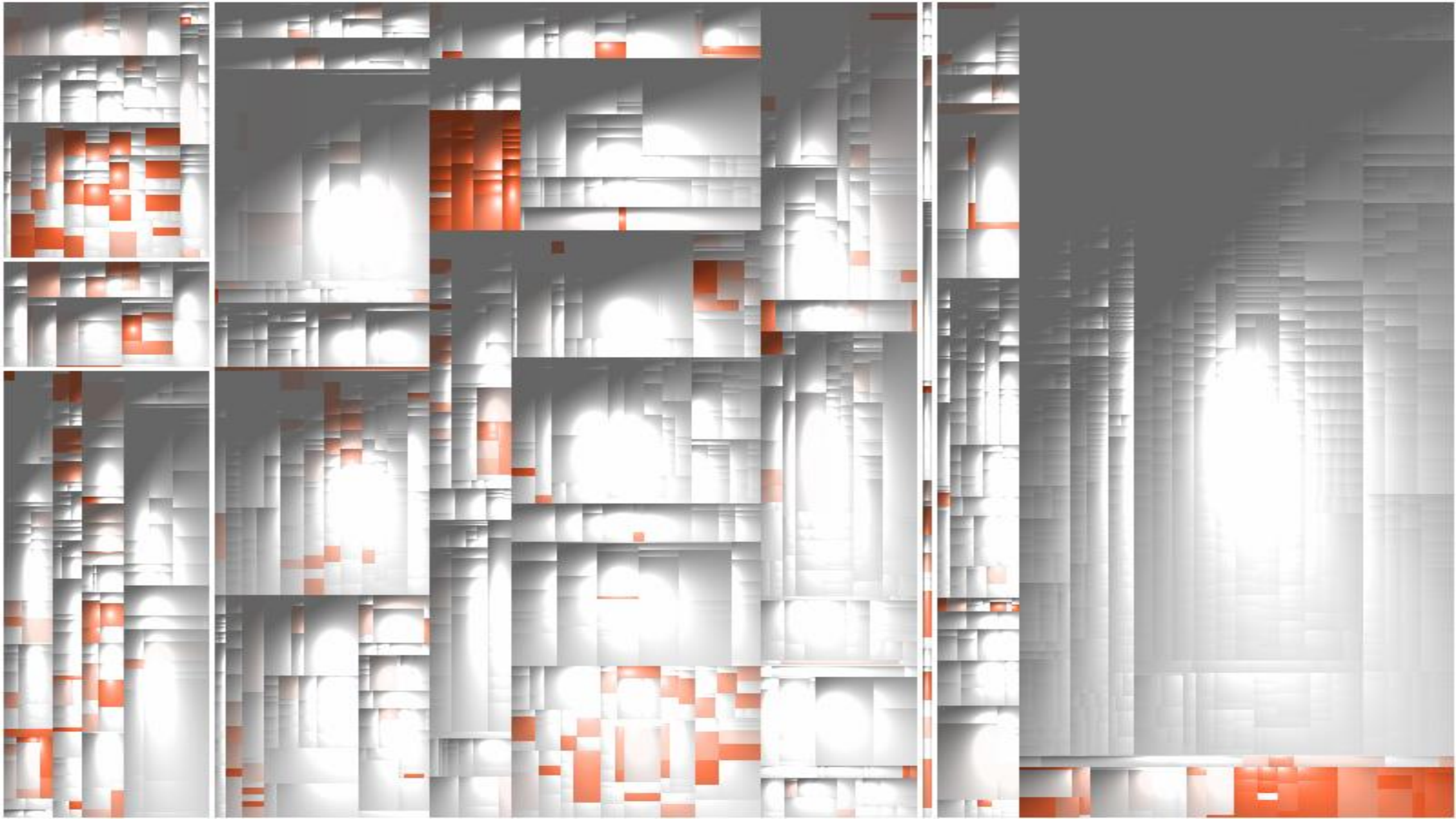
Product 1

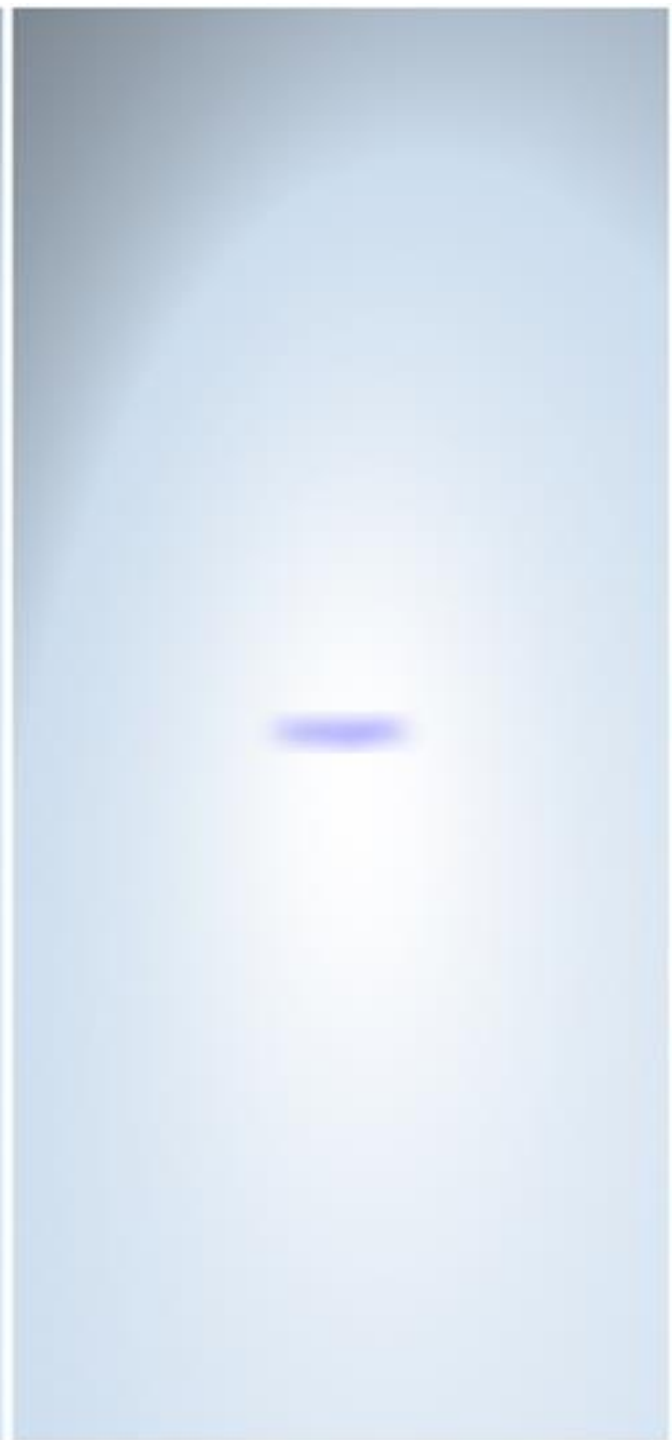
Product 2

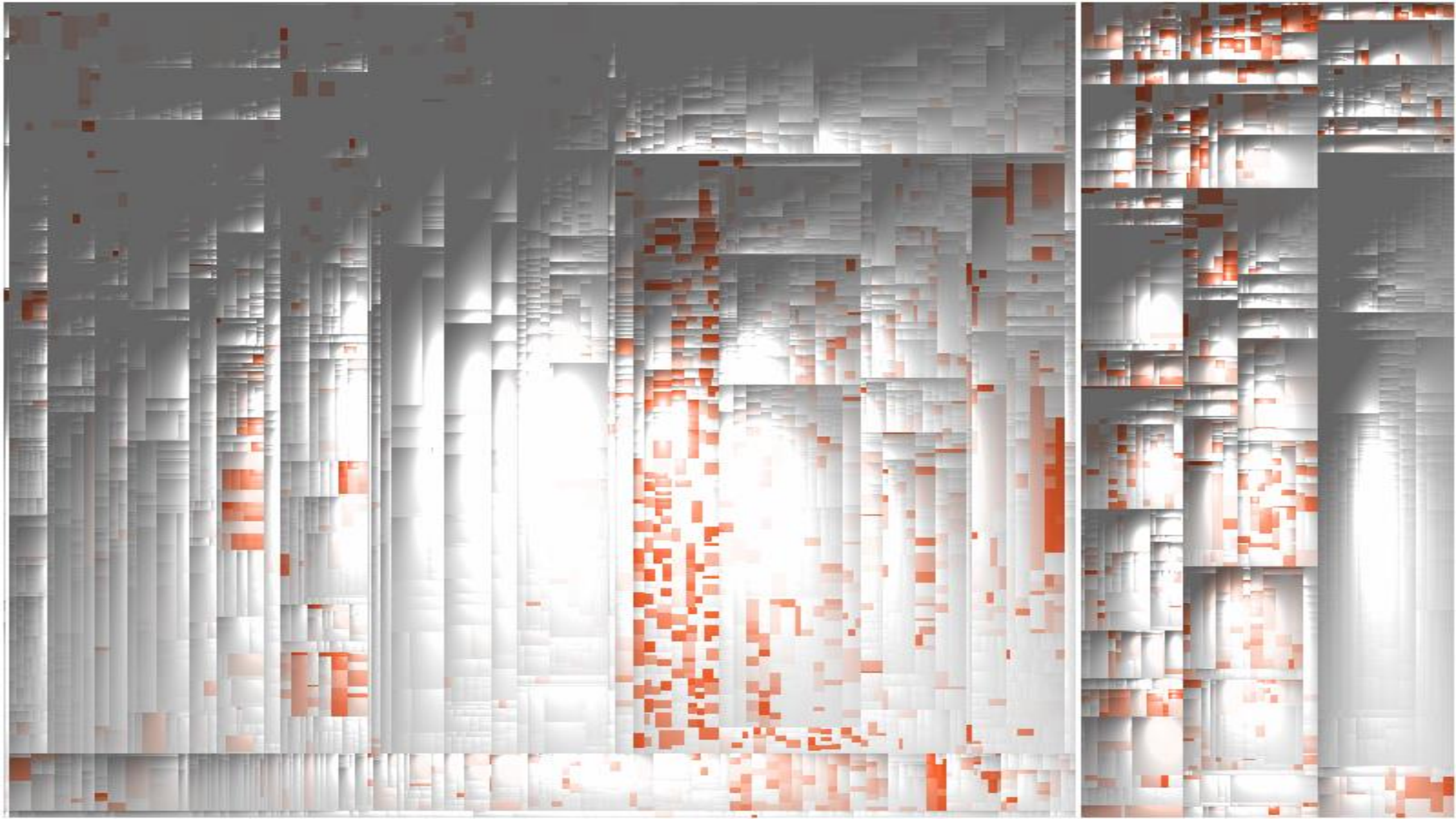
Product 3

Product 4

Product 5







**Was bringt Clone
Management langfristig?**

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


Do Code Clones Matter?

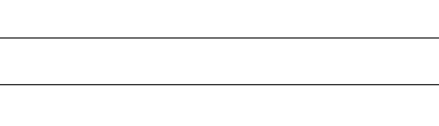
Elmar Juergens, Florian Deissenboeck, Benjamin Hummel, Stefan Wagner
Institut für Informatik, Technische Universität München
Boltzmannstr. 3, 85748 Garching b. München, Germany
{juergens,deissenboeck,hummel,wagner}@iit.tu-mun.de

Abstract

Code cloning is not only assumed to inflate maintenance costs but also identified as a source of software faults. Consequently, the identification of duplicated code, clone detection, has been a very active area of research in recent years. Up to now, however, no substantial investigation of the consequences of code cloning on program correctness has been carried out. In this paper, we analyze this connection and present the results of a large-scale case study that we undertook to find out if incoherent clones lead to code replication faults. For the analyzed commercial and open source systems we not only found that incoherent clones in source systems were very frequent but also identified a significant number of faults induced by such clones. The clone detection tool used in the case study implements a novel algorithm for the detection of incoherent clones. It is available as an open source to enable other researchers to use it as a basis for further investigations.

1. Clones and correctness

Research in software maintenance has shown that many programs contain a significant amount of duplicated (cloned) code. Such cloned code is considered harmful for two reasons: (1) multiple, possibly unnecessary, duplicate code increase maintenance costs and, (2) incoherent clones to cloned code can create faults and, hence, lead to incorrect program behavior [19, 21]. While clone detection has been a very active area of research in recent years up to now, there is no thorough understanding of the degree of harmfulness of code cloning. In fact, some researchers even started to doubt the harmfulness of cloning at all [16]. This is because the clones that are identified by the algorithms do not necessarily cause faults. Second, we present a novel static, rule-based algorithm for the detection of incoherent clones. In contrast to other algorithms for the detection of incoherent clones, our tool suite is made available for other researchers as open source.



Stylus: The open source system StylusSM is developed at the Technische Universität München (TUM) but none of the authors who executed the detection were involved in its development. It constitutes a collaboration environment for distributed software development projects. The inclusion of an IDE in the system is motivated by the fact that, as the clone detection tool is also freely available, the results can be externally replicated. This is not possible with the desktop tool to handle the 5.6MLOC of Eclipse as for 3 hours, which is fast enough to be executed within a night's hold.

Table 1. Summary of the analyzed systems

System	Deployment	Lines (K)	Files	Size (MLOC)
A	March Re	Ca	6	317
B	March Re	Ca	4	324
C	March Re	Ca	2	495
D	LV 1871	Code	2	197
Stylus	TUM	Java	8	281

Stylus: The open source system StylusSM is developed at the Technische Universität München (TUM) but none of the authors who executed the detection were involved in its development. It constitutes a collaboration environment for distributed software development projects. The inclusion of an IDE in the system is motivated by the fact that, as the clone detection tool is also freely available, the results can be externally replicated. This is not possible with the desktop tool to handle the 5.6MLOC of Eclipse as for 3 hours, which is fast enough to be executed within a night's hold.

LV 1871: The Lebensversicherung von 1871 a.G. is a Munich-based life insurance company. The LV 1871 developers developed several commercial software systems for maintenance and PCs. In this study, we analyze a maintenance system containing several hundred software systems in Cobol/DO2 compiled by about 150 users.

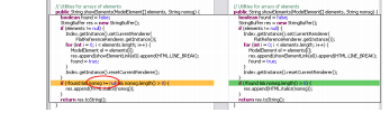


Figure 1. Missing null check on right side can cause exception (Sylphus).

2. Terms and definitions

The literature provides a wide variety of different definitions of clones and clone related terms [19, 28]. To avoid ambiguity, we describe the terms as used in this paper. A clone is interpreted as a sequence of units, which for example might be characters or submitted statements (in procedural code). Thus our definition of a clone is purely syntactical. The reason to allow normalization of units in this step is, that often pieces of code are considered equal despite differences in comments or naming, which can be resolved by the normalization. An exact clone is then a (constructive) substring of the code that appears at least twice in the (normalized) code. Thus our definition of a clone is syntactical, but catches exactly the idea of copy/paste, while not catching clones that are semantically equivalent.

3. Related work

A substantial amount of research has been dedicated to code cloning in recent years. The detailed survey by Kirsch [19] or Roy and Cordy [28] provide a comprehensive overview of existing work. Since this paper targets consequences of cloning and detection of incoherent clones, we detail existing work in these areas.

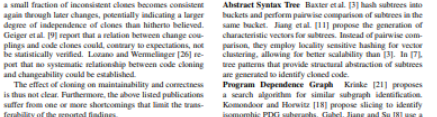


Figure 2. The clone detection pipeline used

4. Detecting incoherent clones

This section explains the approach used for detecting incoherent clones in large amounts of code. Our approach works on a high level, which usually is sufficient for finding copy-pasted code, while at the same time being efficient. The algorithm works by constructing a suffix tree of the code and then for each possible suffix an approximate search based on the edit distance in this tree is performed. Our clone detector is organized as a pipeline, which is depicted in Figure 2. The files under analysis are loaded and then fragmented by the analyzer, yielding a stream of tokens, which is filtered to exclude comments and generate code recognized by our provided patterns. From the token stream, which consist of single keywords, identifiers, operators, and so on, the normalizer normalizes statements. This stage performs normalization, such that

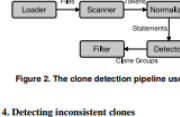


Figure 3. Outline of approximate clone detection algorithm

4.2. Detection algorithm

The task of the detection algorithm is to find clones in the stream of units provided by the normalizer. Stated differently, we want to find common substrings in the sequence formed by all units of the stream, where common substrings are not expected to be exactly identical (after normalization), but may have an edit distance below some threshold. This problem is related to the approximate string matching problem [13, 22], which is also investigated extensively in bioinformatics [30]. The main difference is that we are not interested in an approximation of only a single given word in the suffix, but rather are looking for a subsequence appearing more than once in the entire sequence.



Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

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

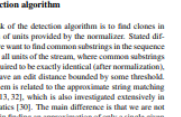


Figure 4. Runtime of incoherent clone detection on Eclipse source

4.3. Post-processing and filtering

During and after detection, the clones that are reported are subject to filtering. Filtering is usually performed as early as possible, so no memory is wasted with storing clone groups that are not considered relevant. Using these filters, we discard clone groups whose clones overlap with each other and groups whose clones are contained in other clone groups. Additionally, we exclude not only an absolute size, i.e., we filter clone groups where the number of incoherencies in the clones relative to the clone's length exceeds a certain amount. Moreover, we merge clone groups which share a common clone. While this leads to clone groups with non-related clones (so our definition of an incoherent clone is not transitive), for practical purposes it is preferred to know of these indirect relationships, too.

4.4. Tool support

To be able to experiment with the detection of incoherent clones, our algorithm and filters have been implemented as part of CloneDetectorSM [14] which is based on CoreDT [7]. The result is a highly configurable and extensible platform for clone detection on the syntactic level. As one cloning pipeline could reuse a major portion of the CloneDetector code, we consider an open platform essential for future experiments, it also allows researchers to extend the tool to support other languages. CloneDetector also offers a front-end to visualize and analyze the clones found, and thus supports the rapid review of a large number of clone groups.

4.5. Scalability and performance

Despite its high implementation details, the worst case complexity is hard to analyze. Additionally, for practical reasons, we do not analyze the development repositories of the systems in order to determine if the incoherencies really have been introduced by incoherent clones to the system and not by random similarities of unrelated code. This has two reasons: (1) we want to analyze all incoherent clones, also the ones that have been introduced directly by copy and modification in a single context. Those might not be visible in the repository. (2) The industrial systems do not have complete development histories. We conducted this study by manually analyzing each potential incoherent clone.

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

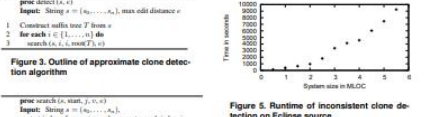


Figure 5. Runtime of incoherent clone detection on Eclipse source

4.3. Post-processing and filtering

During and after detection, the clones that are reported are subject to filtering. Filtering is usually performed as early as possible, so no memory is wasted with storing clone groups that are not considered relevant. Using these filters, we discard clone groups whose clones overlap with each other and groups whose clones are contained in other clone groups. Additionally, we exclude not only an absolute size, i.e., we filter clone groups where the number of incoherencies in the clones relative to the clone's length exceeds a certain amount. Moreover, we merge clone groups which share a common clone. While this leads to clone groups with non-related clones (so our definition of an incoherent clone is not transitive), for practical purposes it is preferred to know of these indirect relationships, too.

4.4. Tool support

To be able to experiment with the detection of incoherent clones, our algorithm and filters have been implemented as part of CloneDetectorSM [14] which is based on CoreDT [7]. The result is a highly configurable and extensible platform for clone detection on the syntactic level. As one cloning pipeline could reuse a major portion of the CloneDetector code, we consider an open platform essential for future experiments, it also allows researchers to extend the tool to support other languages. CloneDetector also offers a front-end to visualize and analyze the clones found, and thus supports the rapid review of a large number of clone groups.

4.5. Scalability and performance

Despite its high implementation details, the worst case complexity is hard to analyze. Additionally, for practical reasons, we do not analyze the development repositories of the systems in order to determine if the incoherencies really have been introduced by incoherent clones to the system and not by random similarities of unrelated code. This has two reasons: (1) we want to analyze all incoherent clones, also the ones that have been introduced directly by copy and modification in a single context. Those might not be visible in the repository. (2) The industrial systems do not have complete development histories. We conducted this study by manually analyzing each potential incoherent clone.

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

Studie: Über 1.100 Fehler in produktiver Software

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

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

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

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

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

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

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

#Fehler durch inkonsistente Klone @ Munich Re

Daten aus Studie

- 3 Systeme von Munich Re analysiert
- 79 Fehler gefunden (Impact auf Funktionalität, nicht nur Wartbarkeit o.ä.)
- System 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

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

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

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

Ø Fehlerfolgekosten 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

Ø Fehlerfolgekosten 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

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

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

495 $\frac{PT}{Jahr}$

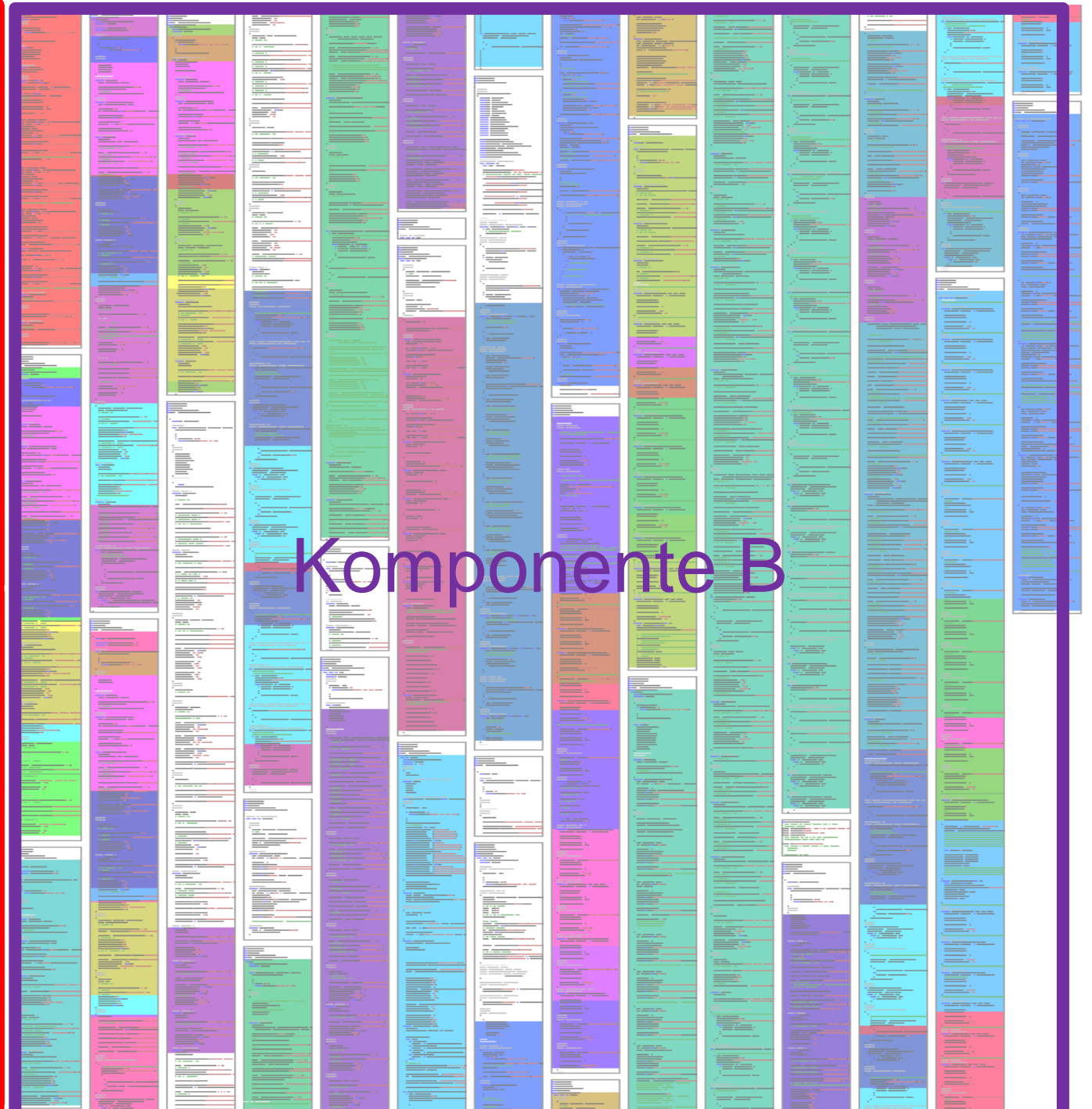
$$500 \frac{PT}{Jahr}$$

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

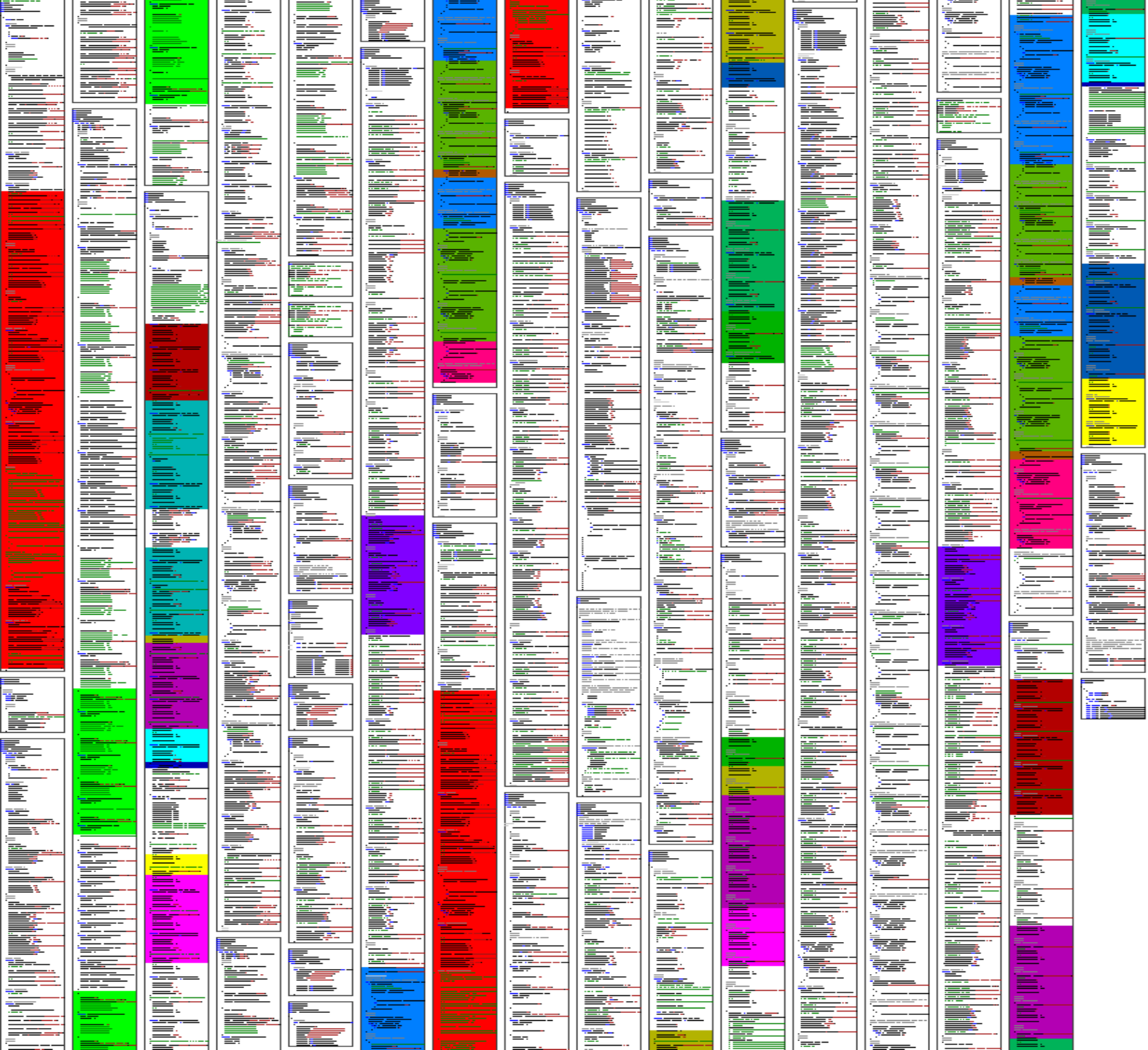




Komponente A



Komponente B



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

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

Blow-Up: 0%



20 Lines

20 Lines

Blow-Up: 50%



20 Lines

20 Lines

20 Lines

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

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

%CloneAffectedEffort

Aktivitäten

- Analysis
- Location
- Design
- Impact Analysis
- Implementation
- Quality Assurance
- Other

Aufwändiger durch Cloning

-
- Location
-
- Impact Analysis
- Implementation
- Quality Assurance
-

Detaillierte Herleitung und Berechnung im Paper.

Wert für Berechnung: **50%**.

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

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

**Die Munich Re setzt
Clone Detection seit ca.
10 Jahren ein.**

Wie sähe es ohne aus?

Continuous Software Quality Control in Practice

Daniela Steidl*, Florian Deissenboeck*, Martin Pöhlmann*, 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 metrics¹ 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 Sonar² 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 (BMBWF), grant EcoCon, 01IS12034A. The responsibility for this article lies with the authors.

¹e.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

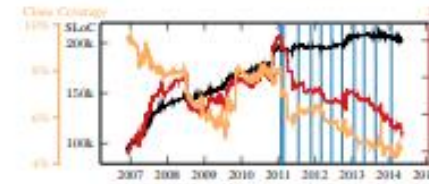


Fig. 3. System A

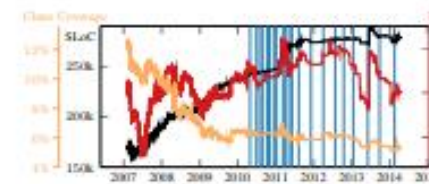


Fig. 4. System B

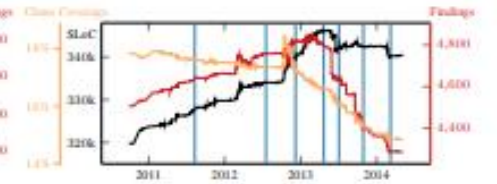


Fig. 5. System C

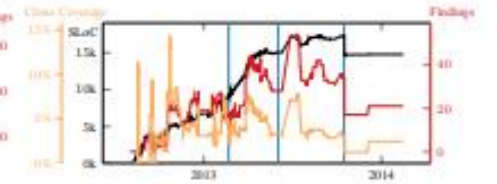


Fig. 6. System D

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 prove 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.

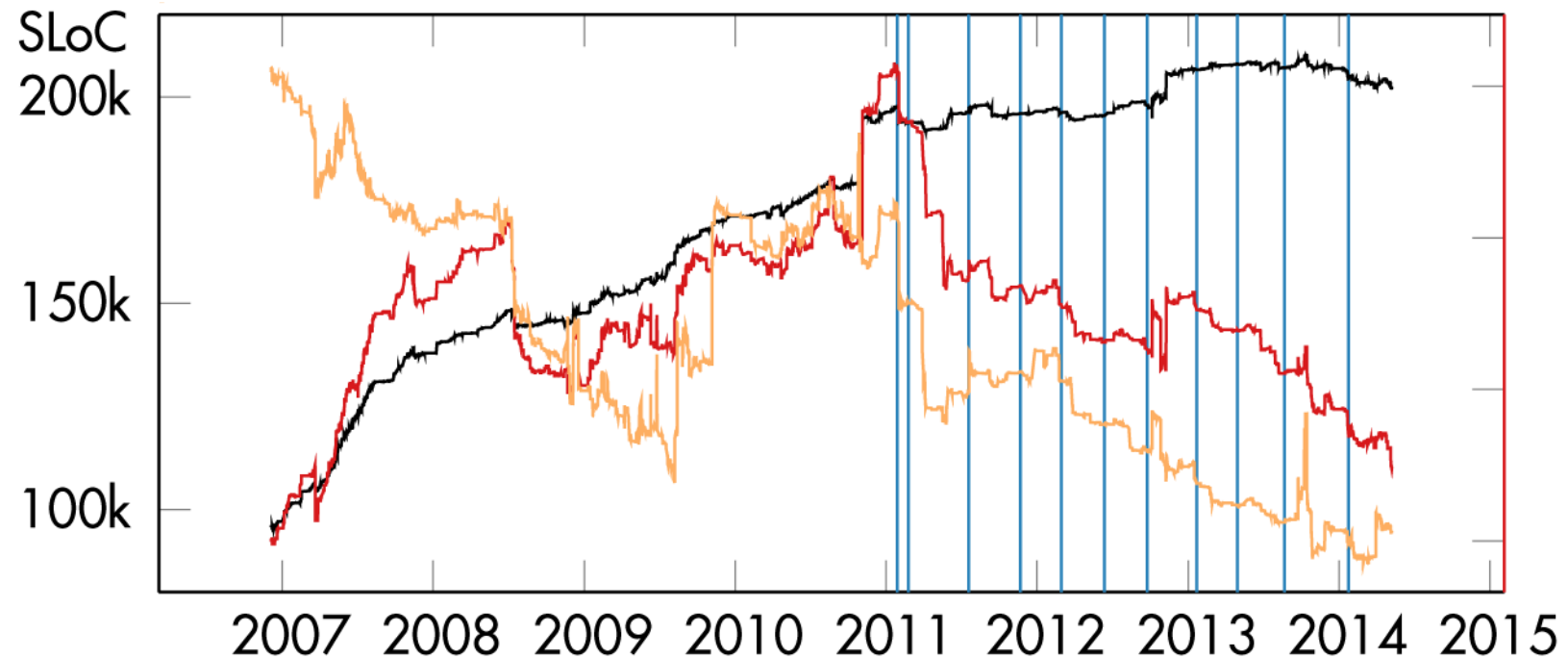
V. CONCLUSION

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. Jungers, B. Hummel, S. Wagner, B. M. y Parareda, and M. Pirka, "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 backstair: An in-process software engineering meta-statement and analysis system," in *ESEM'07*.
- [5] F. Deissenboeck, M. Pirka, and T. Seifart, "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.

Einsparung durch Clone Detection



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

Ersparnis Aufwand = 6%

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

Findings

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

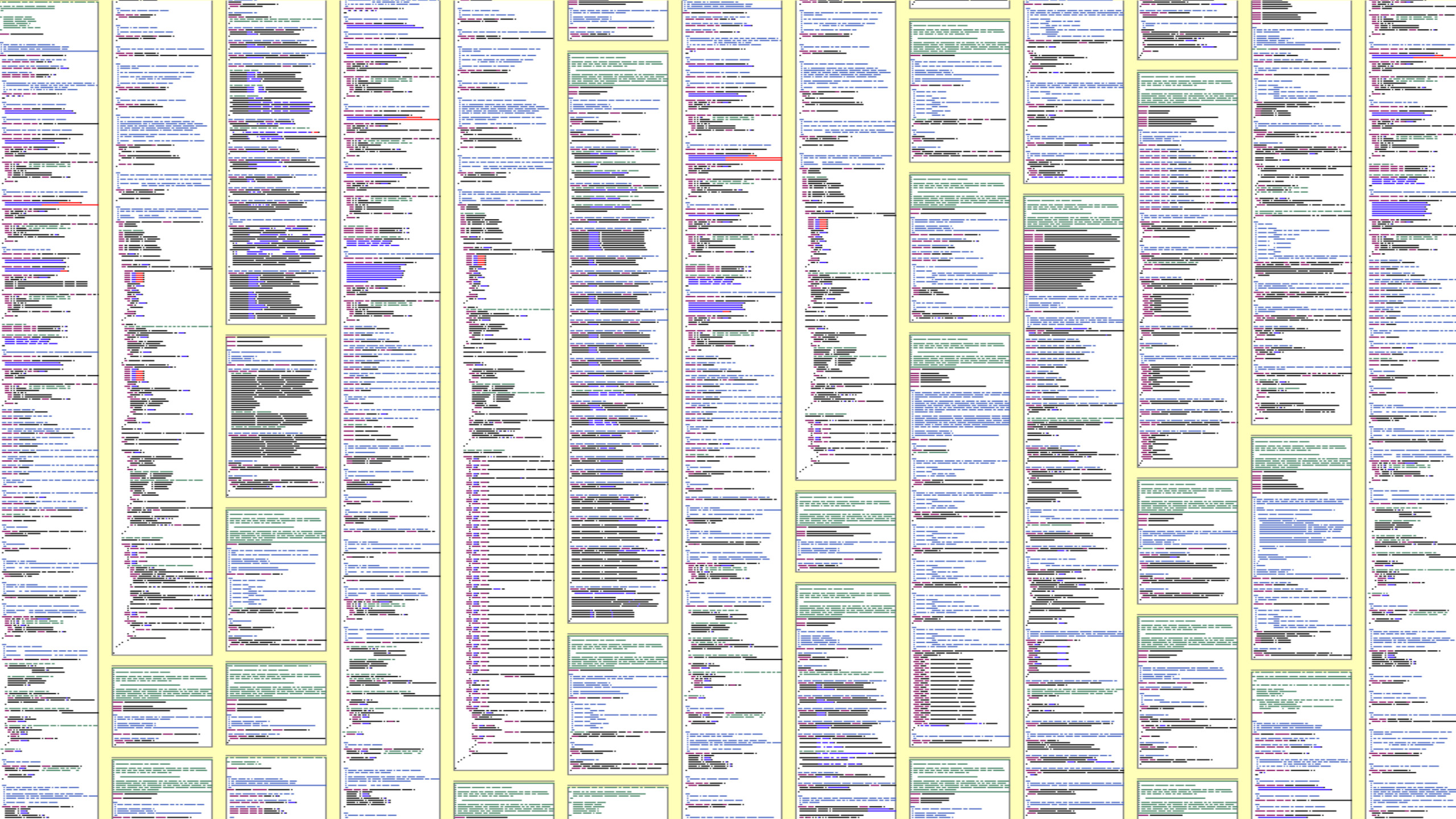
500 $\frac{PT}{Jahr}$

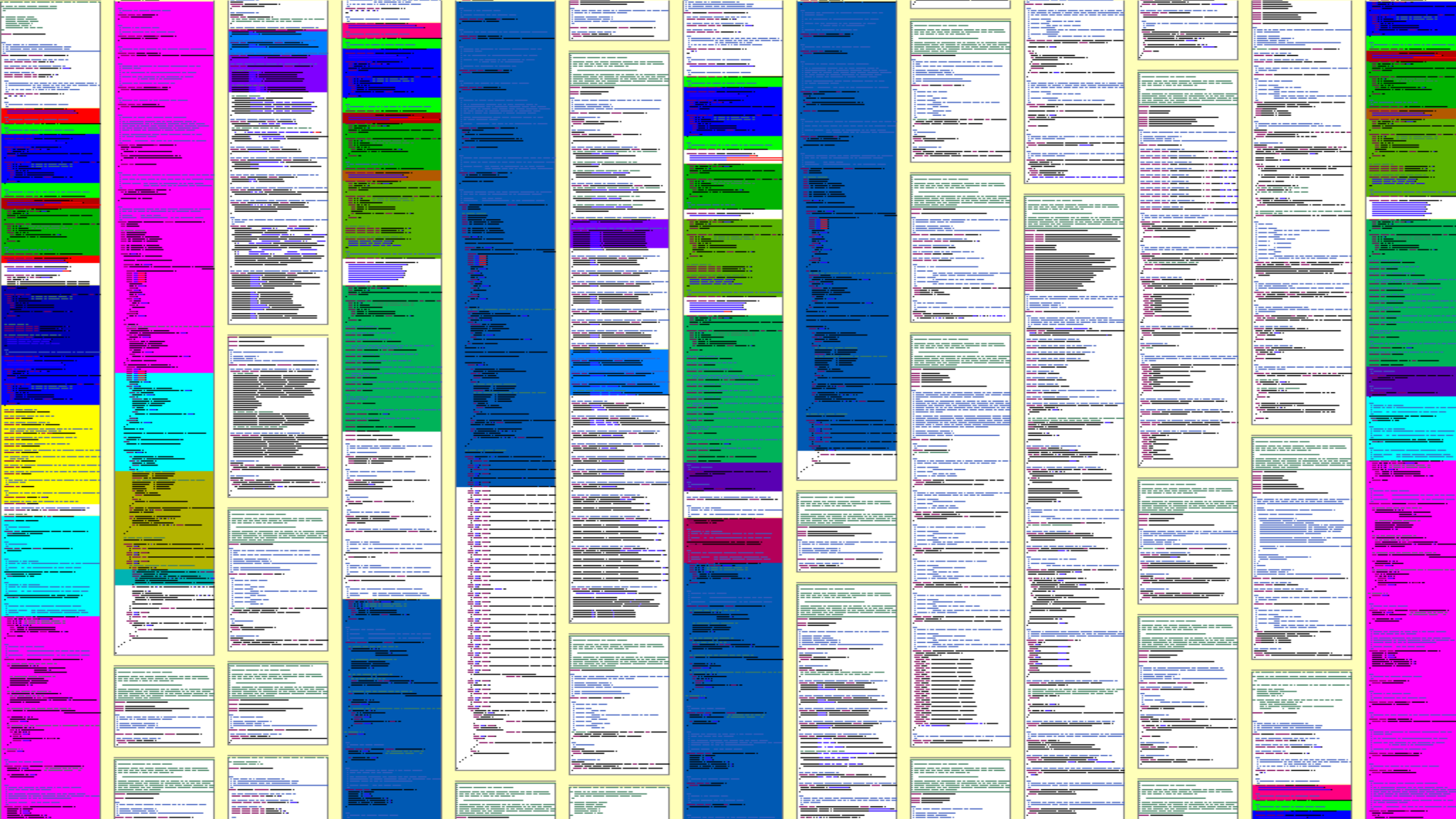
Munich Re spart durch Einsatz von Clone Management jährlich ca. 500 PT Aufwand für Fehlerbehebung

Ersparnis Aufwand = 6%

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

Wie ausprobieren?









CQSE Workshop

Clone Management

Copy, Paste und dann ...?



09. November
10:30-12:00 Uhr
cqse.eu/clone-22-11-saat



Wie zukunftssicher ist Ihr Softwaresystem?

Unser Vorgehen und Erfahrungen aus 10 Jahren Softwareaudits



10. November
10:30-12:00 Uhr
cqse.eu/audit-22-11-saat



Kontakt – Freude 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