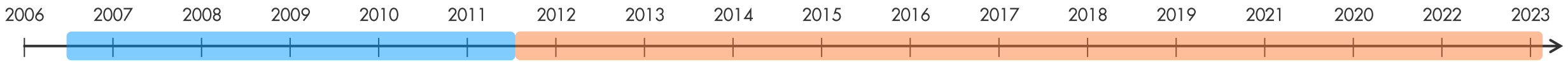
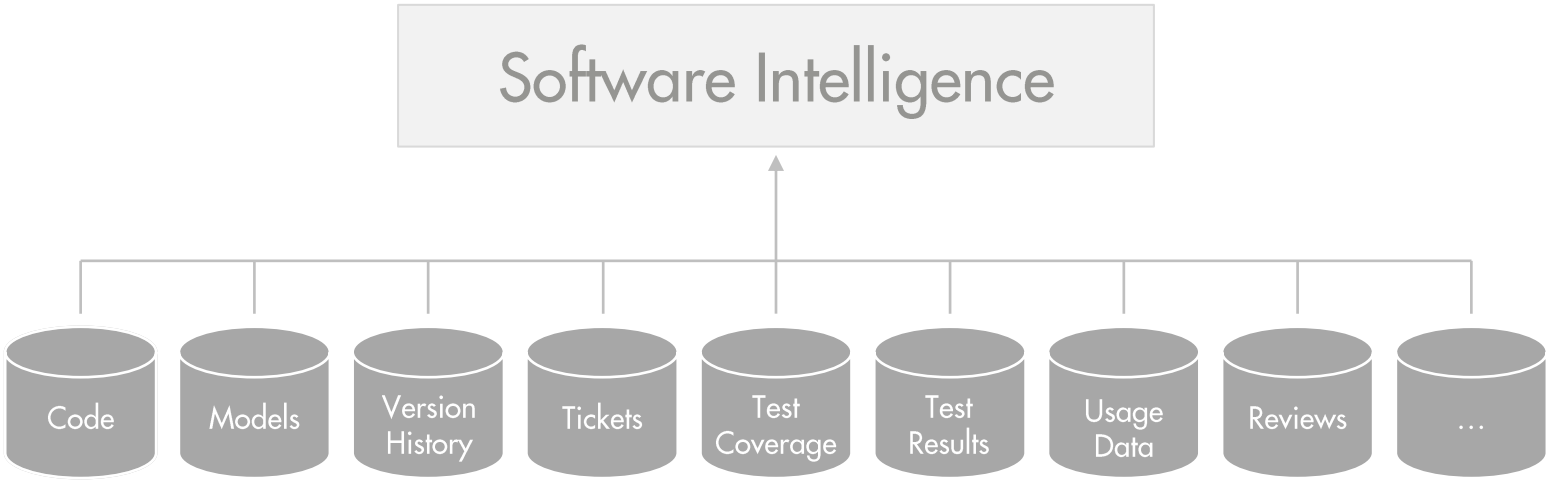
Software Intelligence

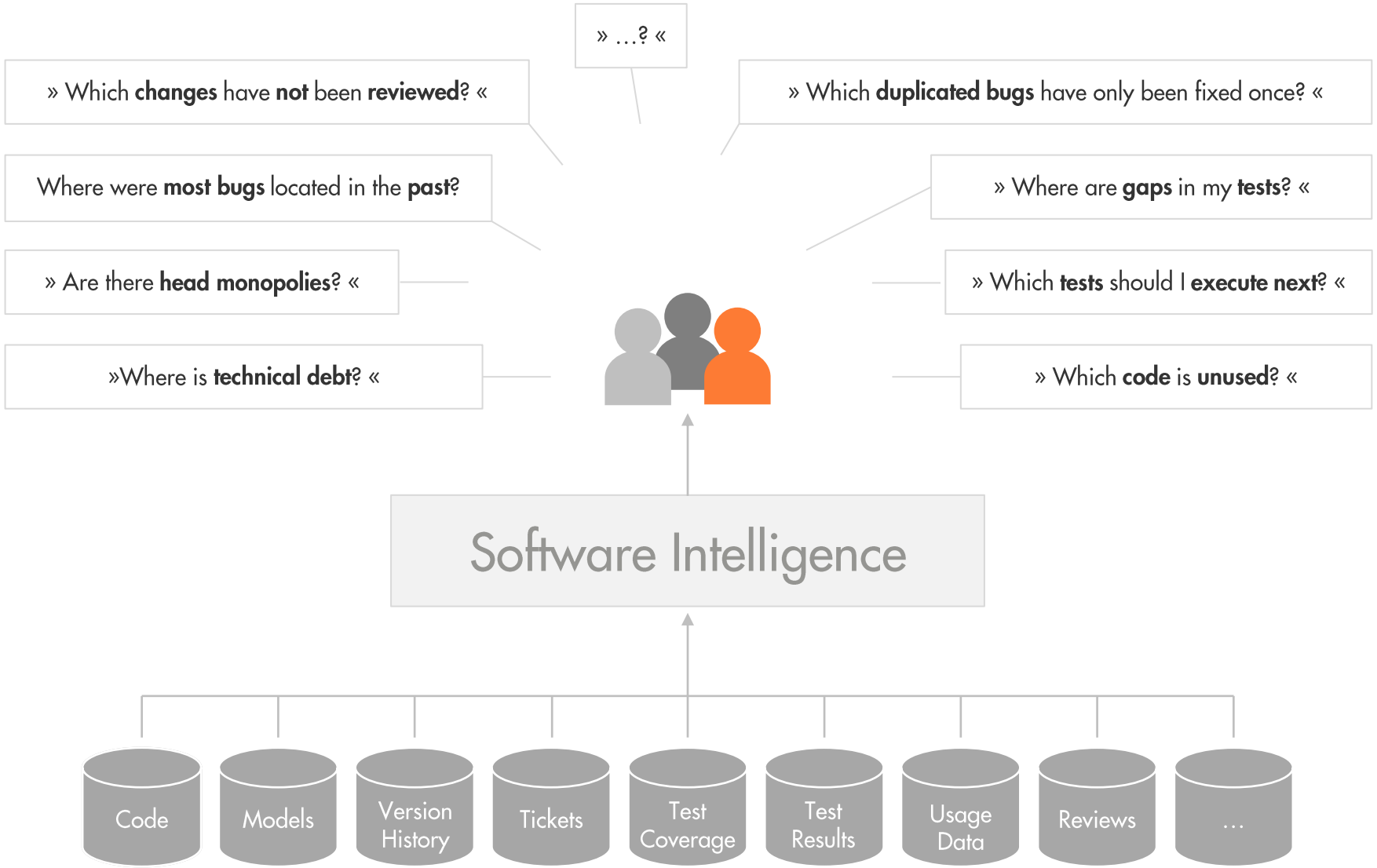
Was verraten unsere Entwicklungsdaten über Probleme in unserer Software?

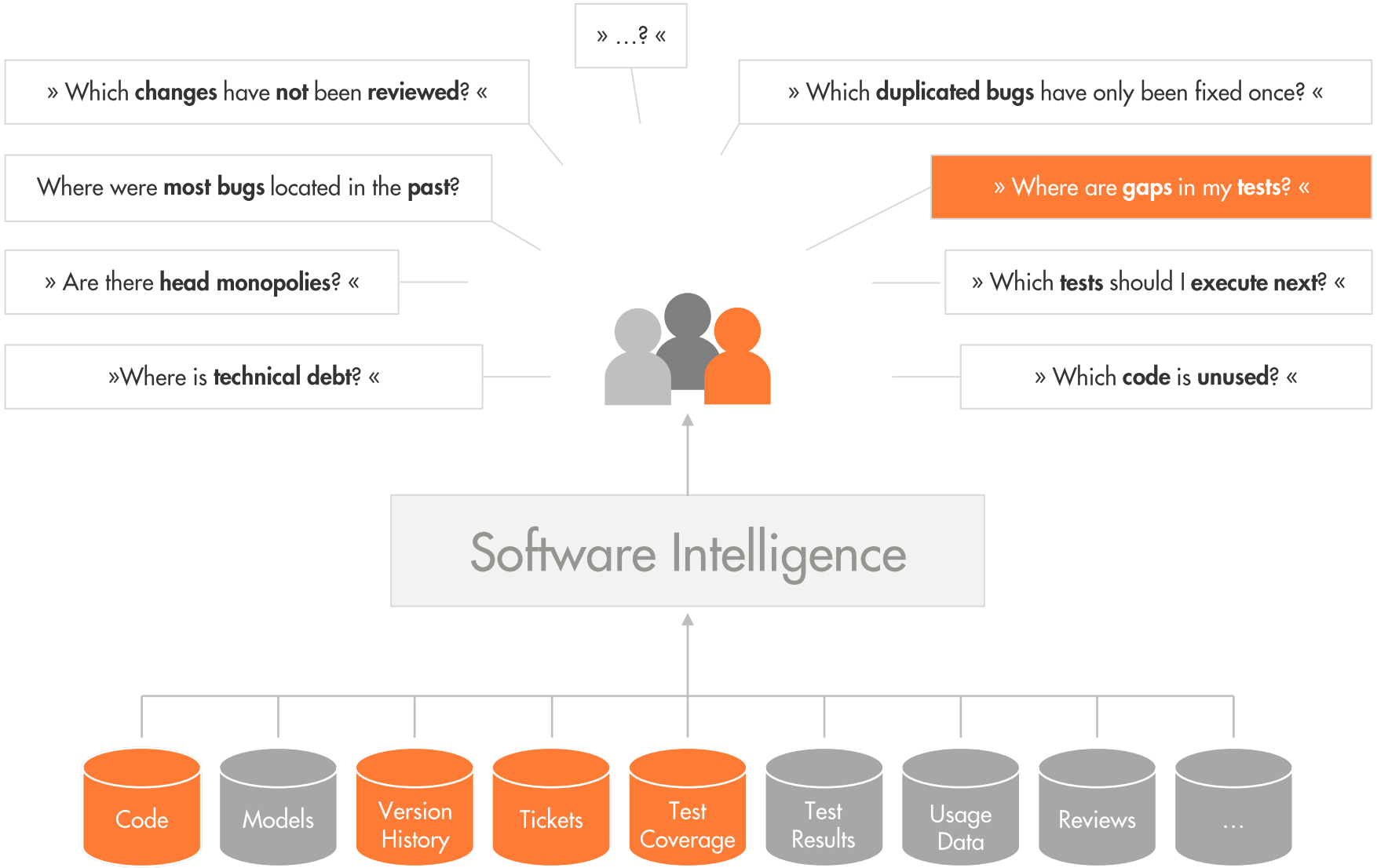
Dr. Elmar Jürgens

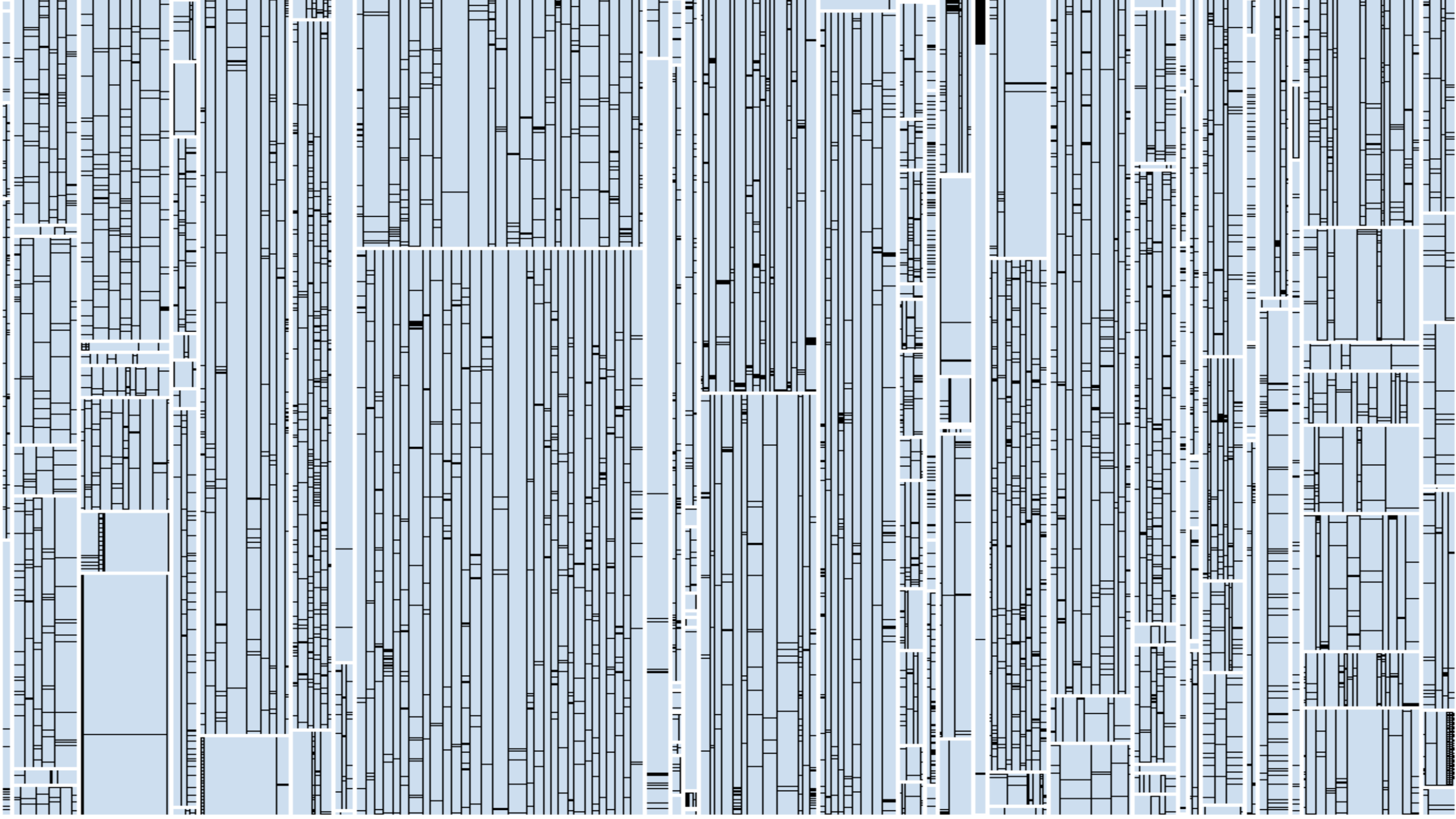
CQSE

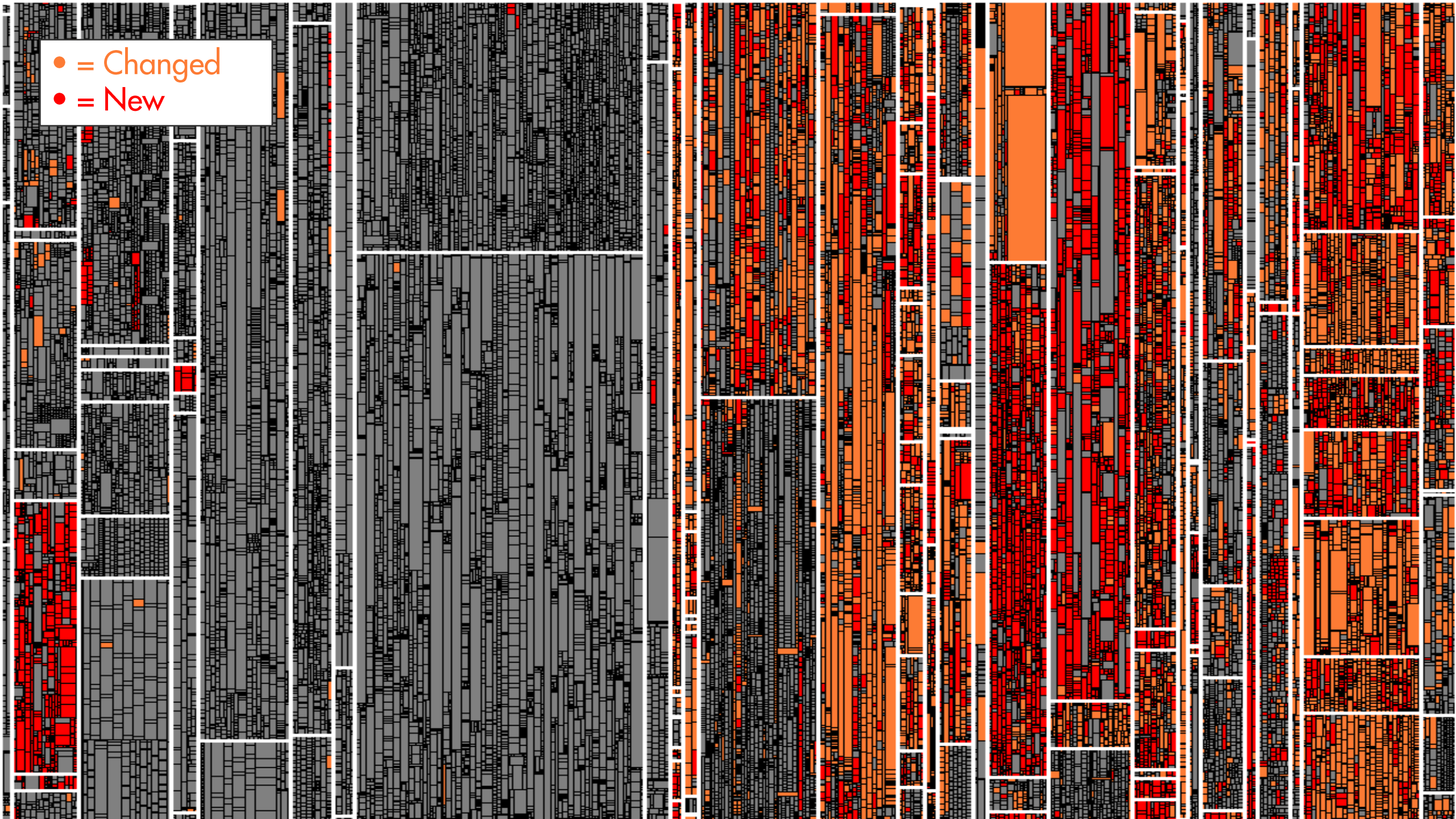

2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2021
2020
2022
2023

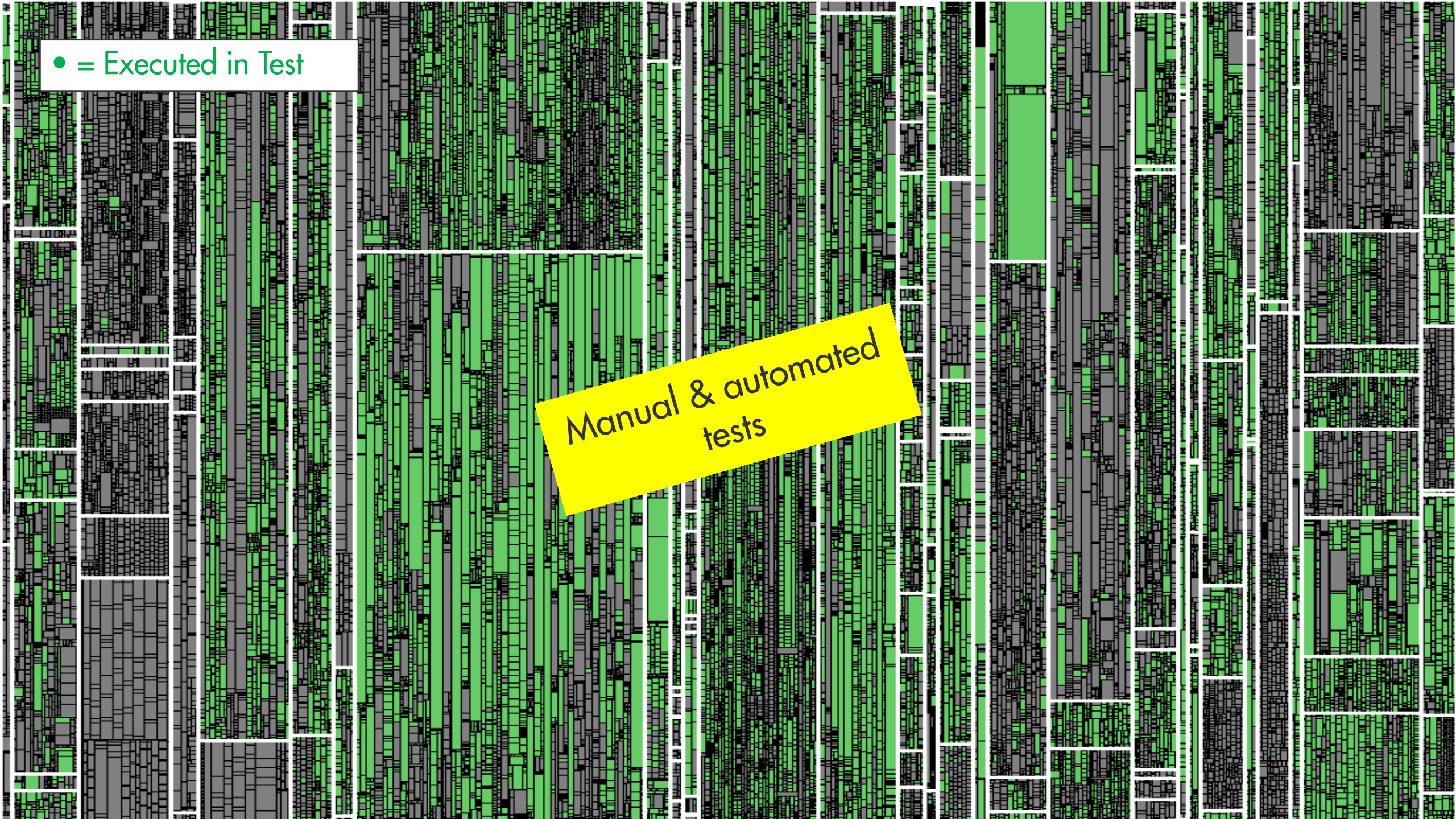

Software Intelligence
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…

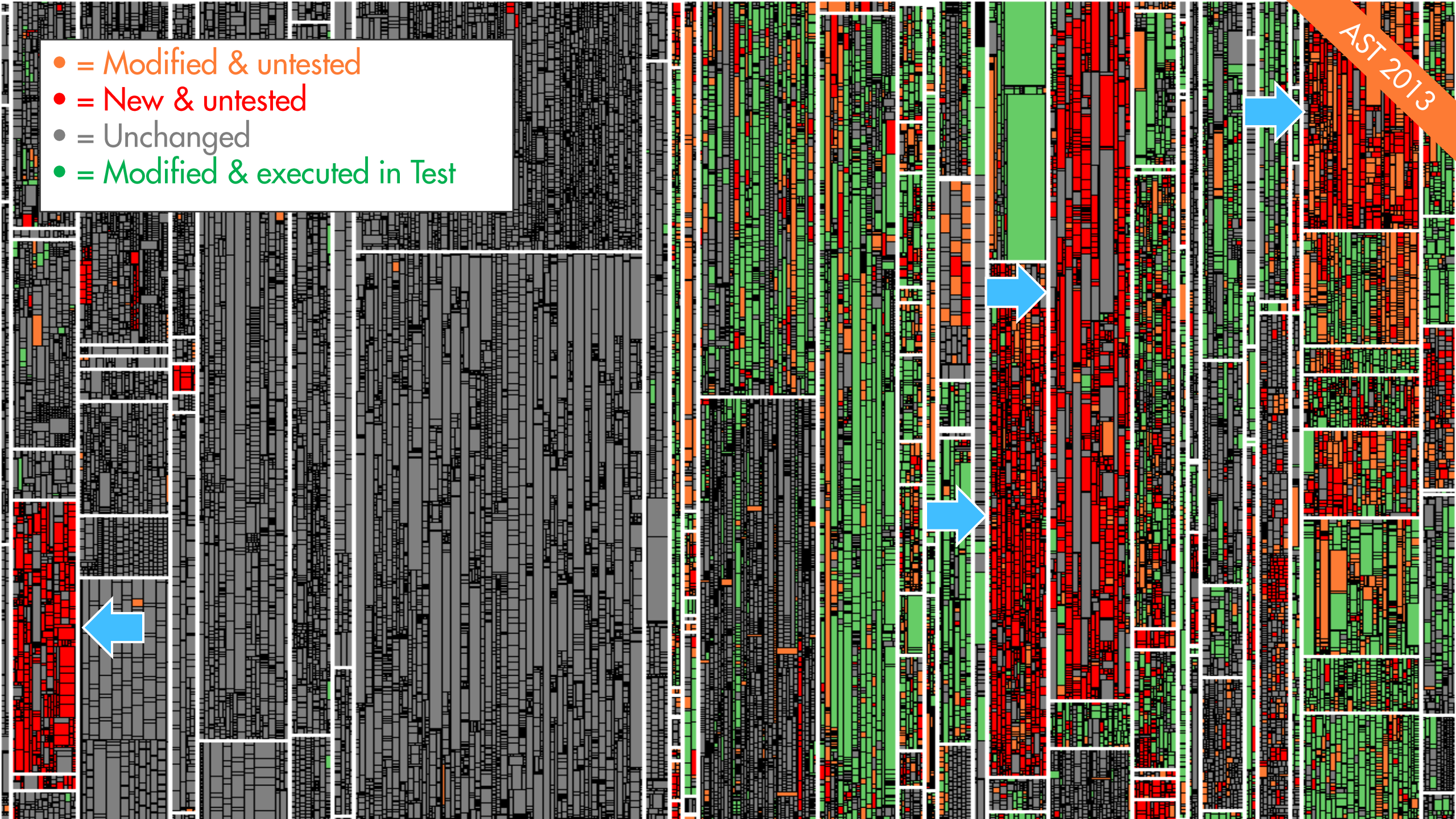

» …? «
» Which **changes** have **not** been **reviewed**? «
» Which **duplicated bugs** have only been fixed once? «
Where were **most bugs** located in the **past**?
» Where are **gaps** in my **tests**? «
» Are there **head monopolies**? «
» Which **tests** should I **execute next**? «
»Where is **technical debt**? «
» Which **code** is **unused**? «
Software Intelligence
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…

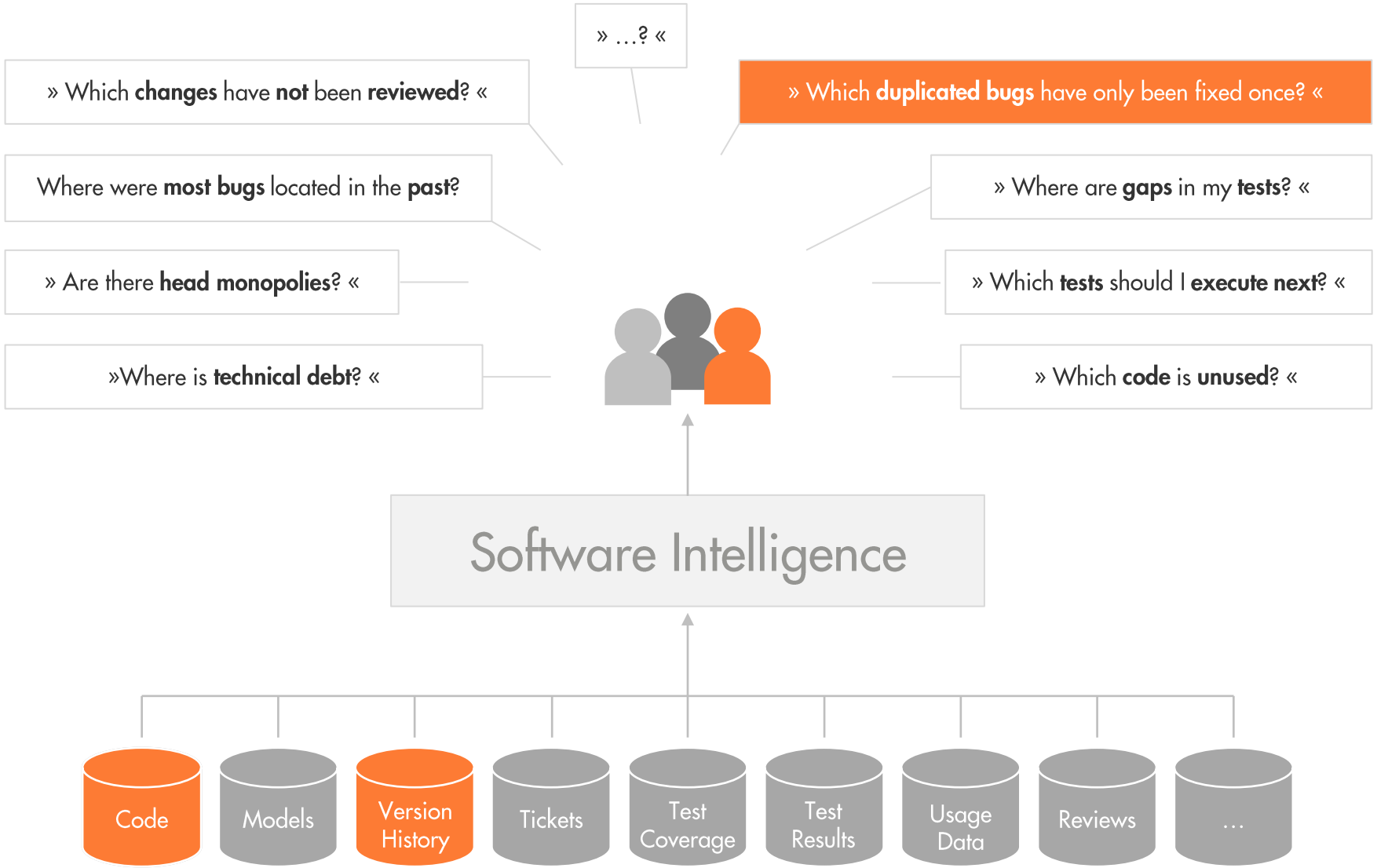

» …? «
» Which changes have not been reviewed? «
» Which duplicated bugs have only been fixed once? «
Where were most bugs located in the past?
» Where are gaps in my tests? «
» Are there head monopolies? «
» Which tests should I execute next? «
»Where is technical debt? «
» Which code is unused? «
Software Intelligence
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…

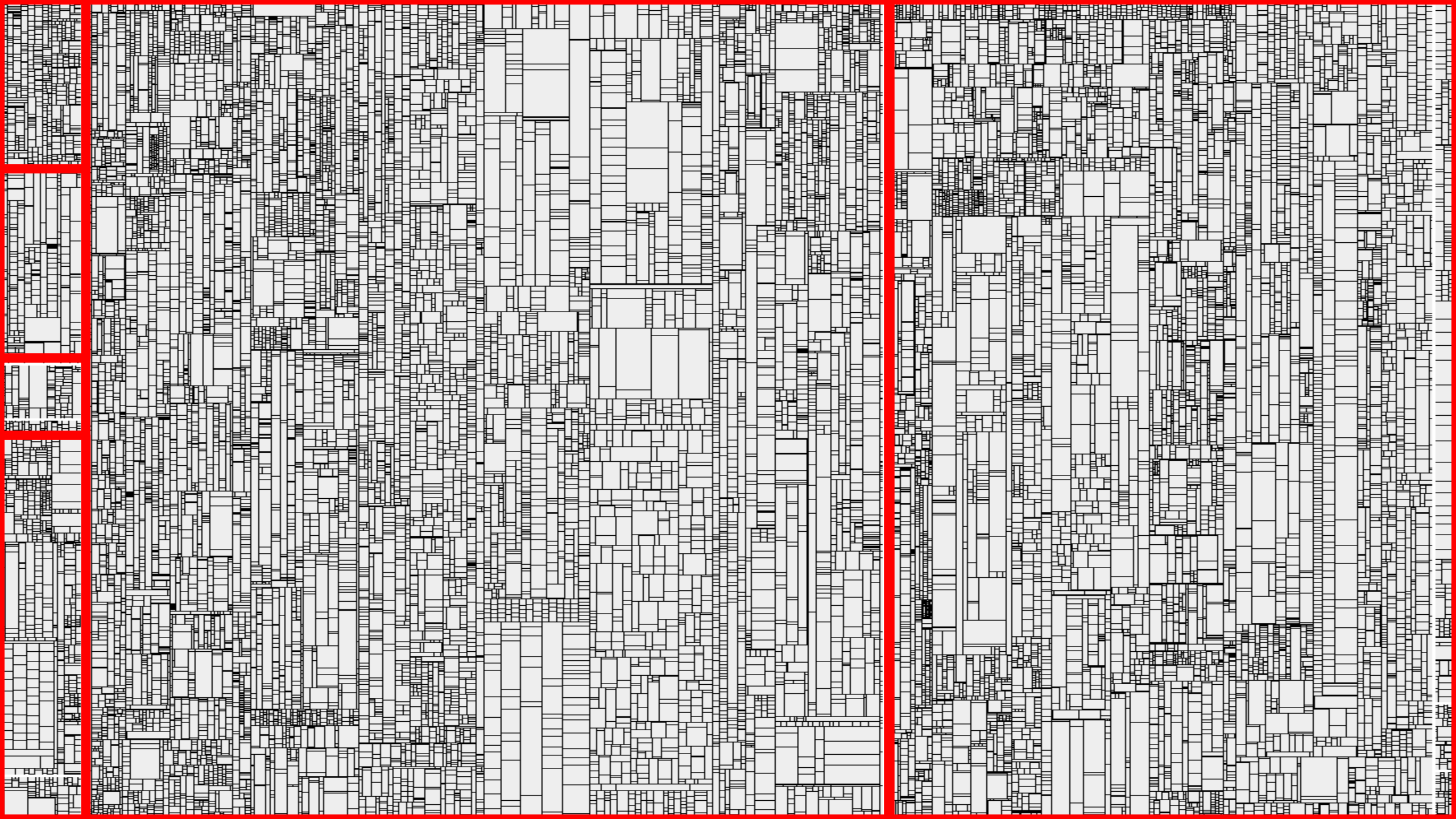


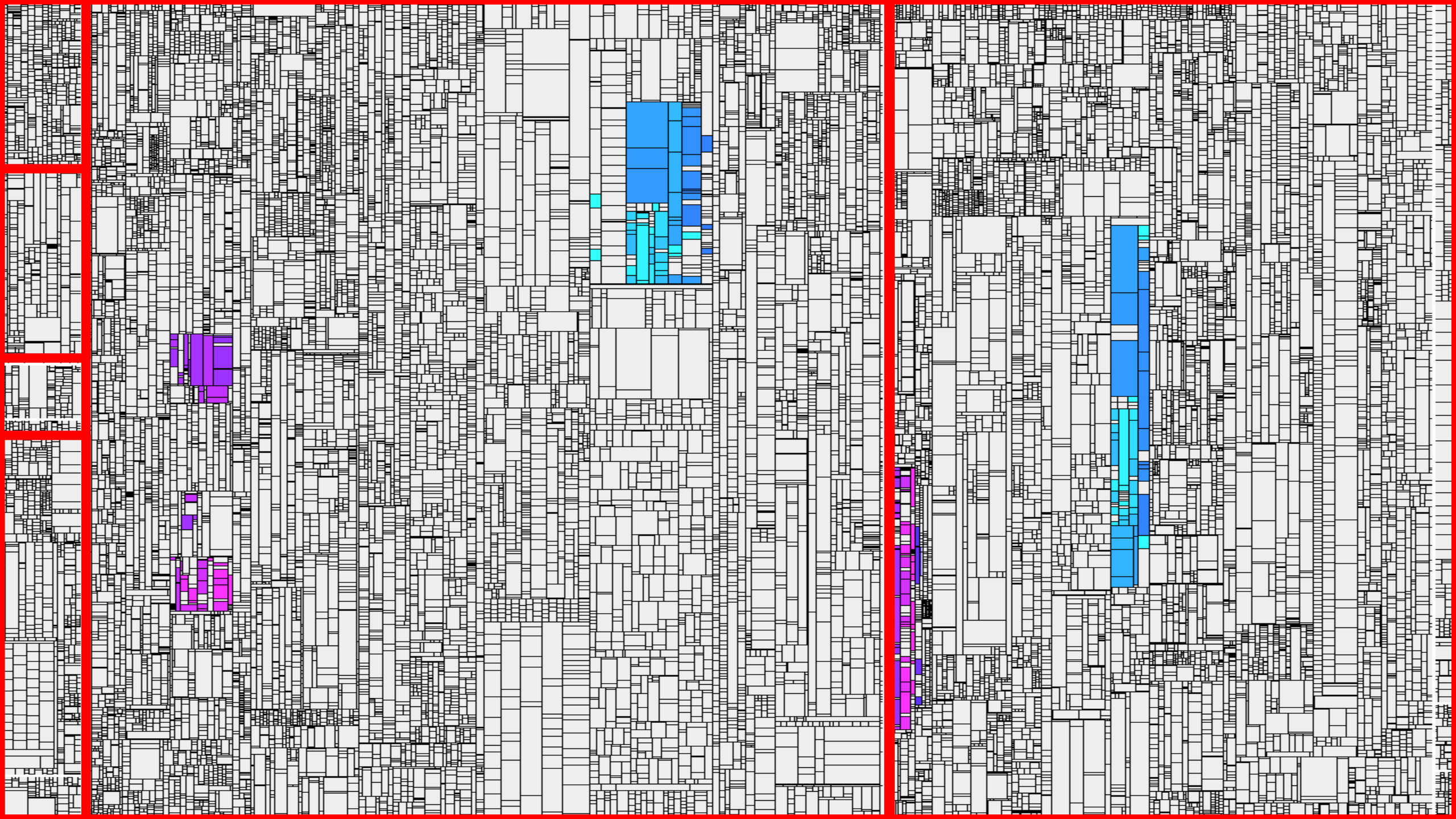

• = Changed
• = New

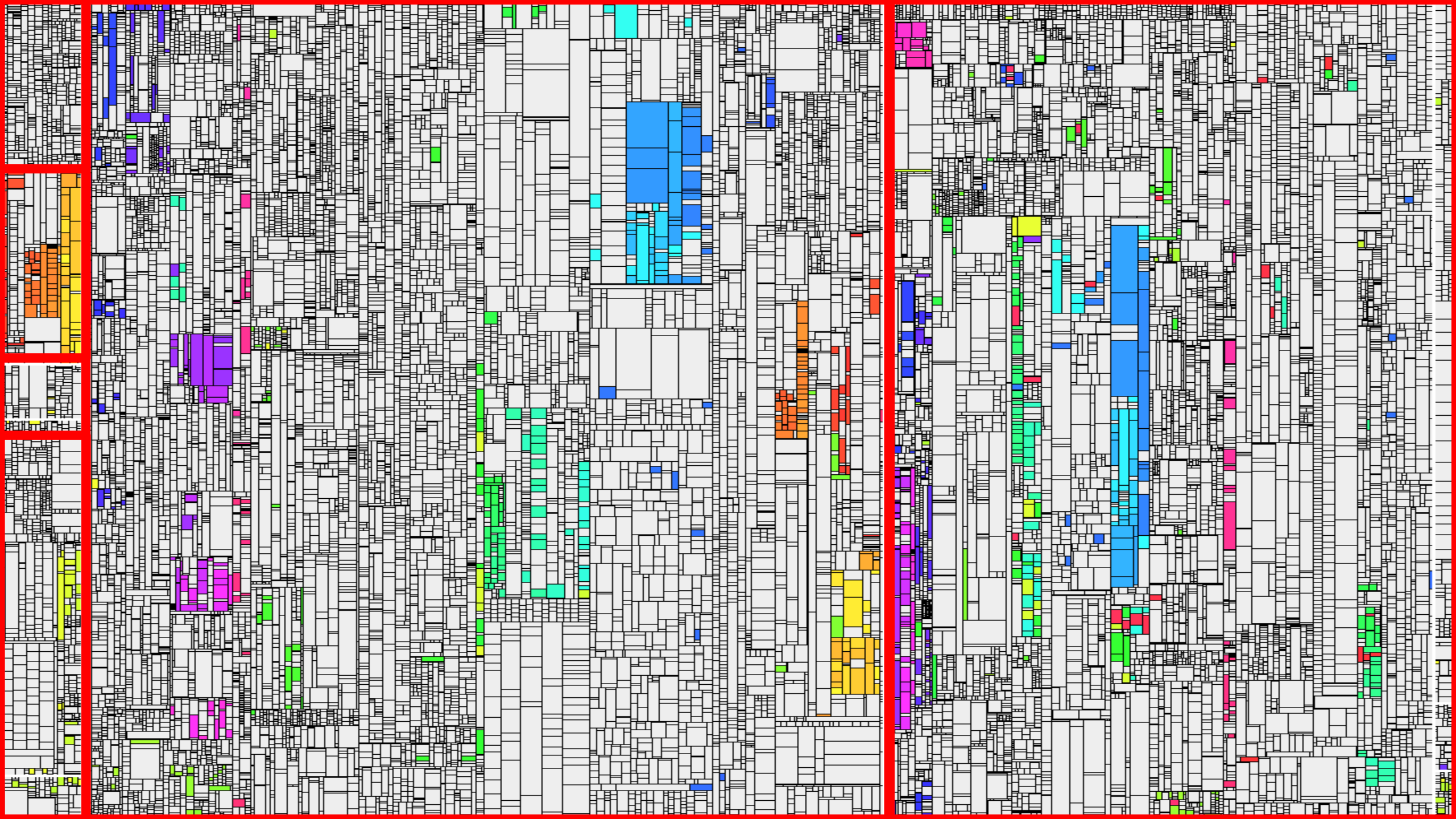

• = Executed in Test
Manual & automated tests


• = Modified & untested
• = New & untested
• = Unchanged
• = Modified & executed in Test
AST 2013


» …? «
» Which changes have not been reviewed? «
» Which duplicated bugs have only been fixed once? «
Where were most bugs located in the past?
» Where are gaps in my tests? «
» Are there head monopolies? «
» Which tests should I execute next? «
»Where is technical debt? «
» Which code is unused? «
Software Intelligence
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…







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

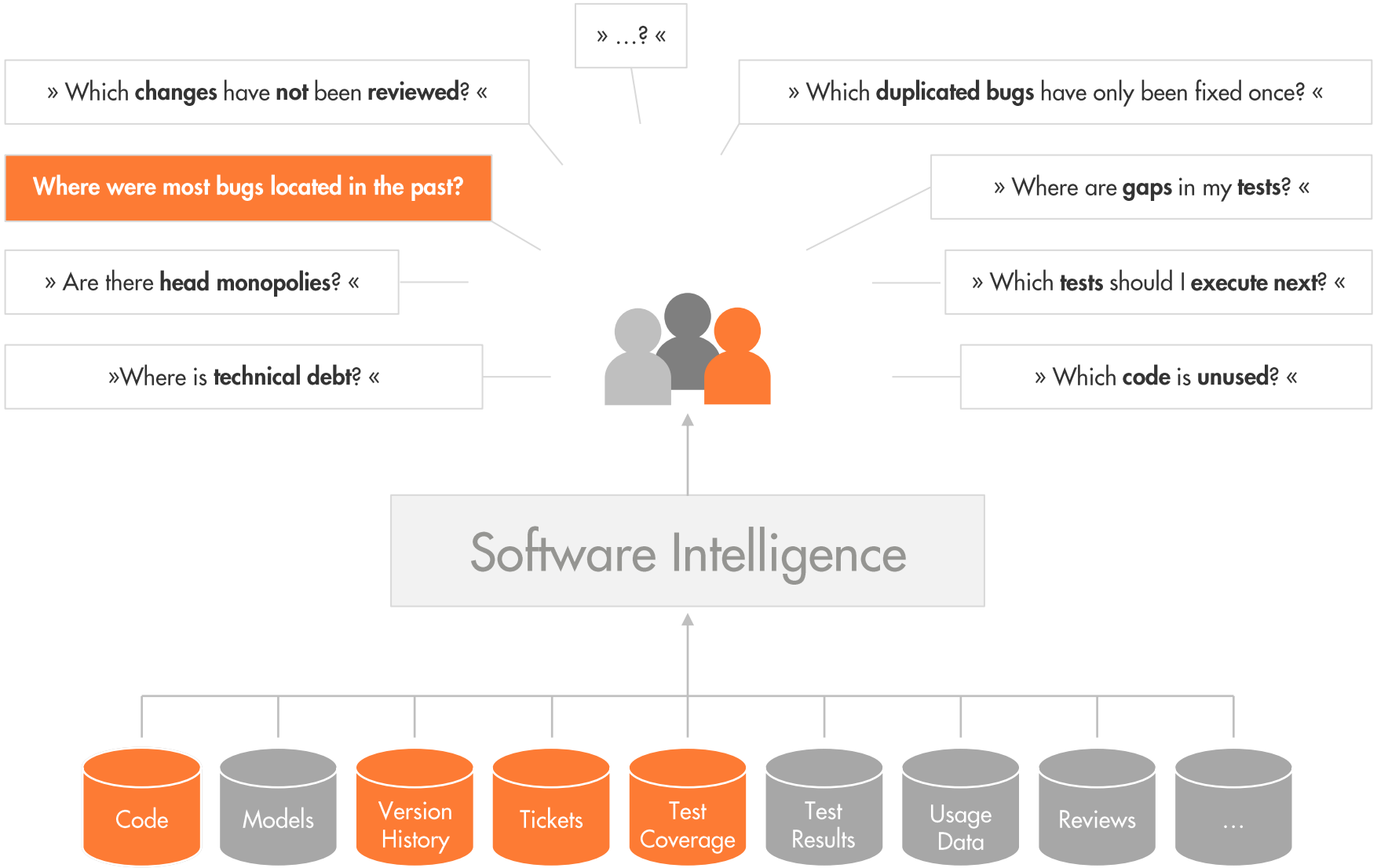
Collapse

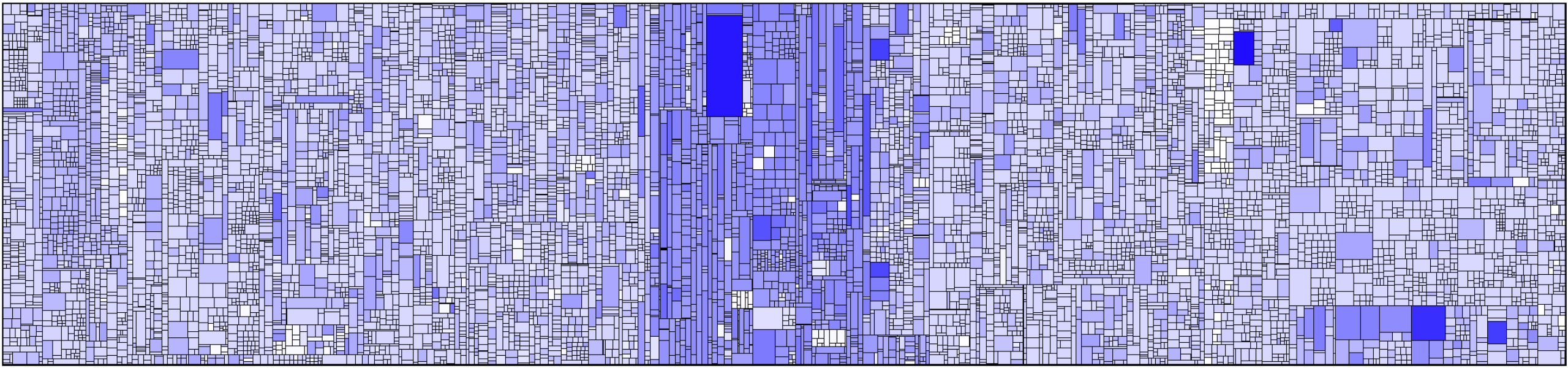
jenkins/test/src/main/java/org/jvnet/hudson/test/HudsonTestCase.java (revision 12d96a56...)

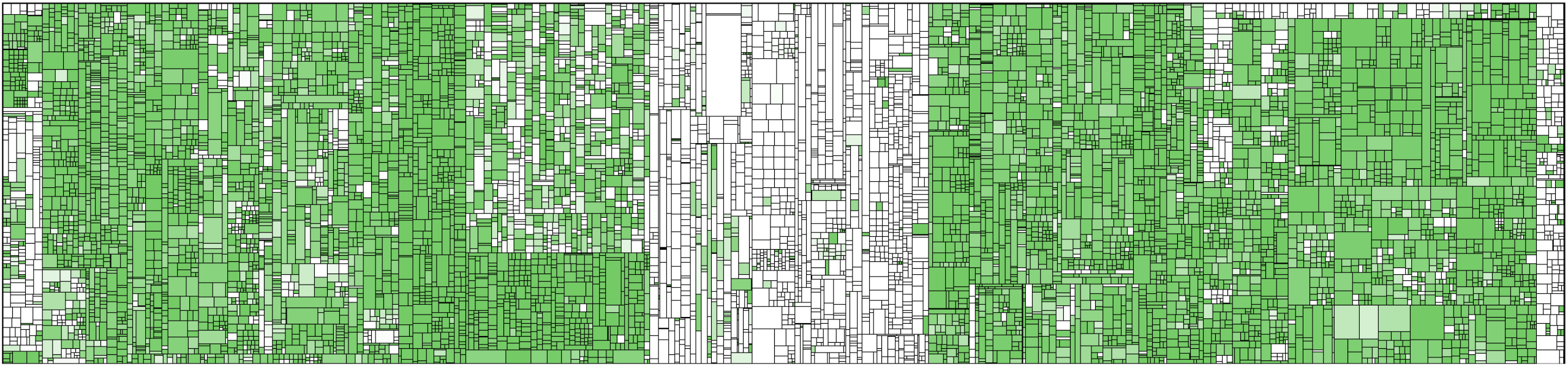
```
 if (lhs==null && rhs==null)     return;
 if (lhs==null)      fail("lhs is null while rhs="+rhs);
 if (rhs==null)      fail("rhs is null while lhs="+lhs);

 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());
```

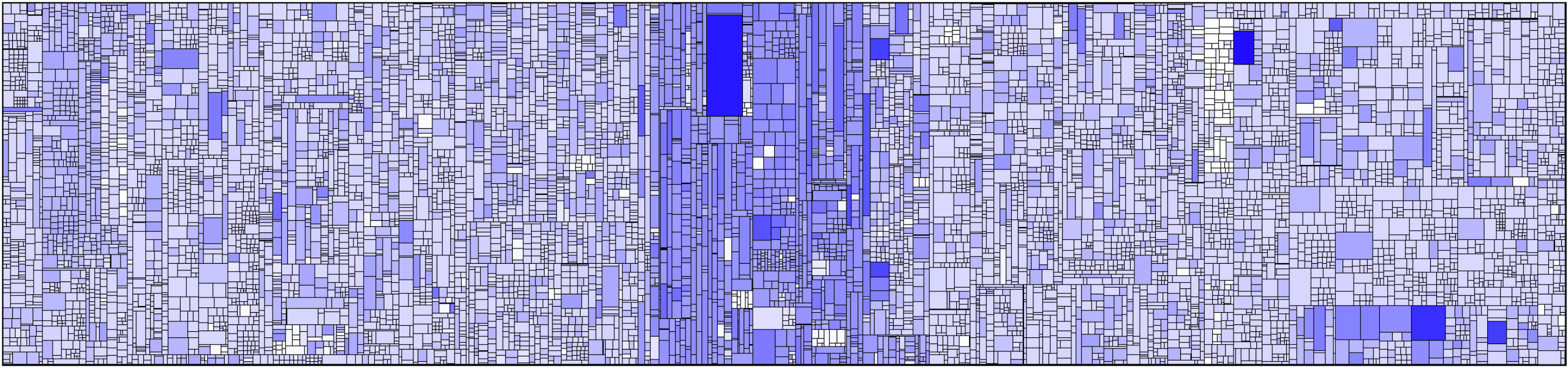
jenkins/test/src/main/java/org/jvnet/hudson/test/JenkinsRule.java (revision 3909f5ac...)

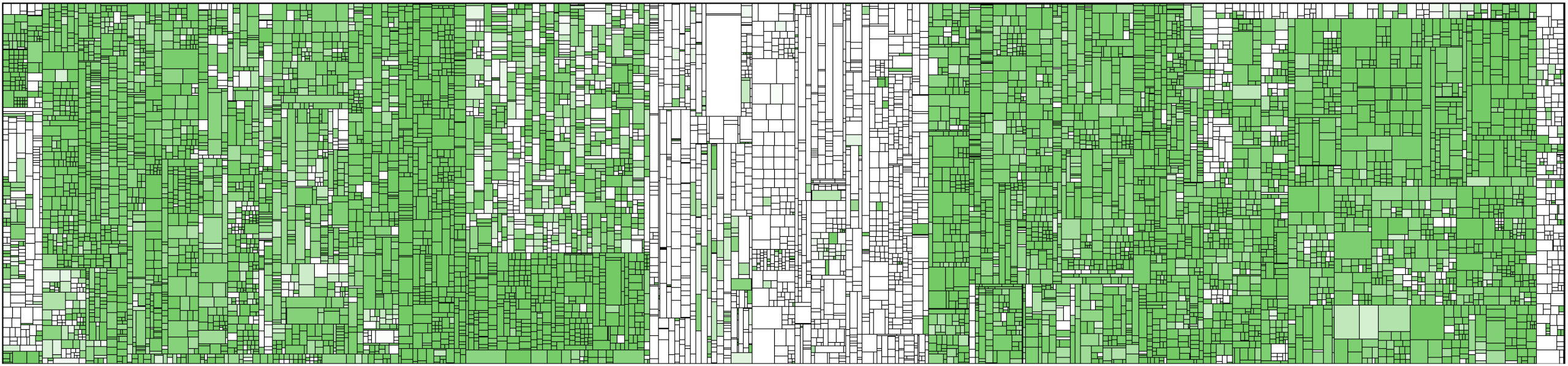
```
 if (lhs==null && rhs==null)     return;
 if (lhs==null)      fail("lhs is null while rhs="+rhs);
 if (rhs==null)      fail("rhs is null while lhs="+lhs);

 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 {
```


» …? «
» Which changes have not been reviewed? «
» Which duplicated bugs have only been fixed once? «
Where were most bugs located in the past?
» Where are gaps in my tests? «
» Are there head monopolies? «
» Which tests should I execute next? «
»Where is technical debt? «
» Which code is unused? «
Software Intelligence
Code
Models
Version History
Tickets
Test Coverage
Test Results
Usage Data
Reviews
…









Ich freue mich auf Fragen ☺

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

Link zu Folien

CQSE GmbH
Centa-Hafenbrädl-Str 59
81249 München
www.cqse.eu