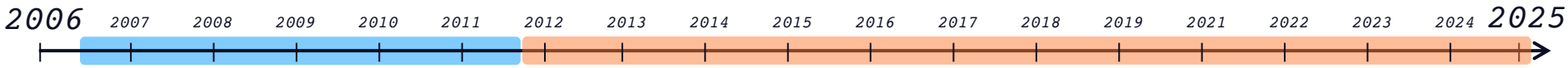
RBCN25

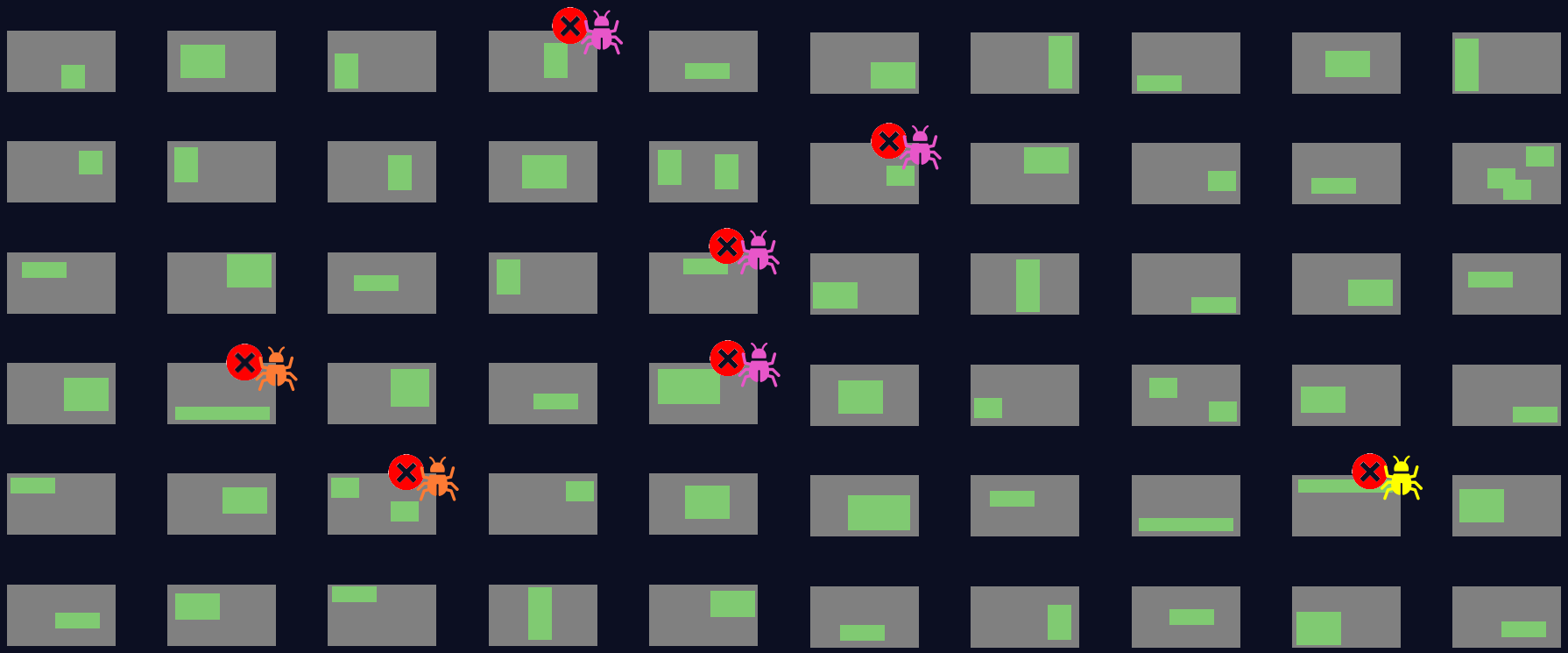
FROM Germany WITH LOVE

RBCN25

DEAR AI, WHICH TESTS SHOULD ROBOT FRAMEWORK EXECUTE NOW?

Dr. Elmar Juergens


2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2021
2022
2023
2024
2025



TEST SELECTION USE CASES

For a Quality Gate

- *before an expensive test execution that runs all tests*
- *Makes sure expensive run not wasted on broken SW*

During CI

- *Selected tests are executed more frequently (e.g. for each commit)*
- *Whole suite still executed infrequently (e.g. only on main branch, or over night)*

Test Selection for a QUALITY GATE

Picture1.png - Pixelitor 4.2.3

File Edit Layer Select Image Color Filter View Help

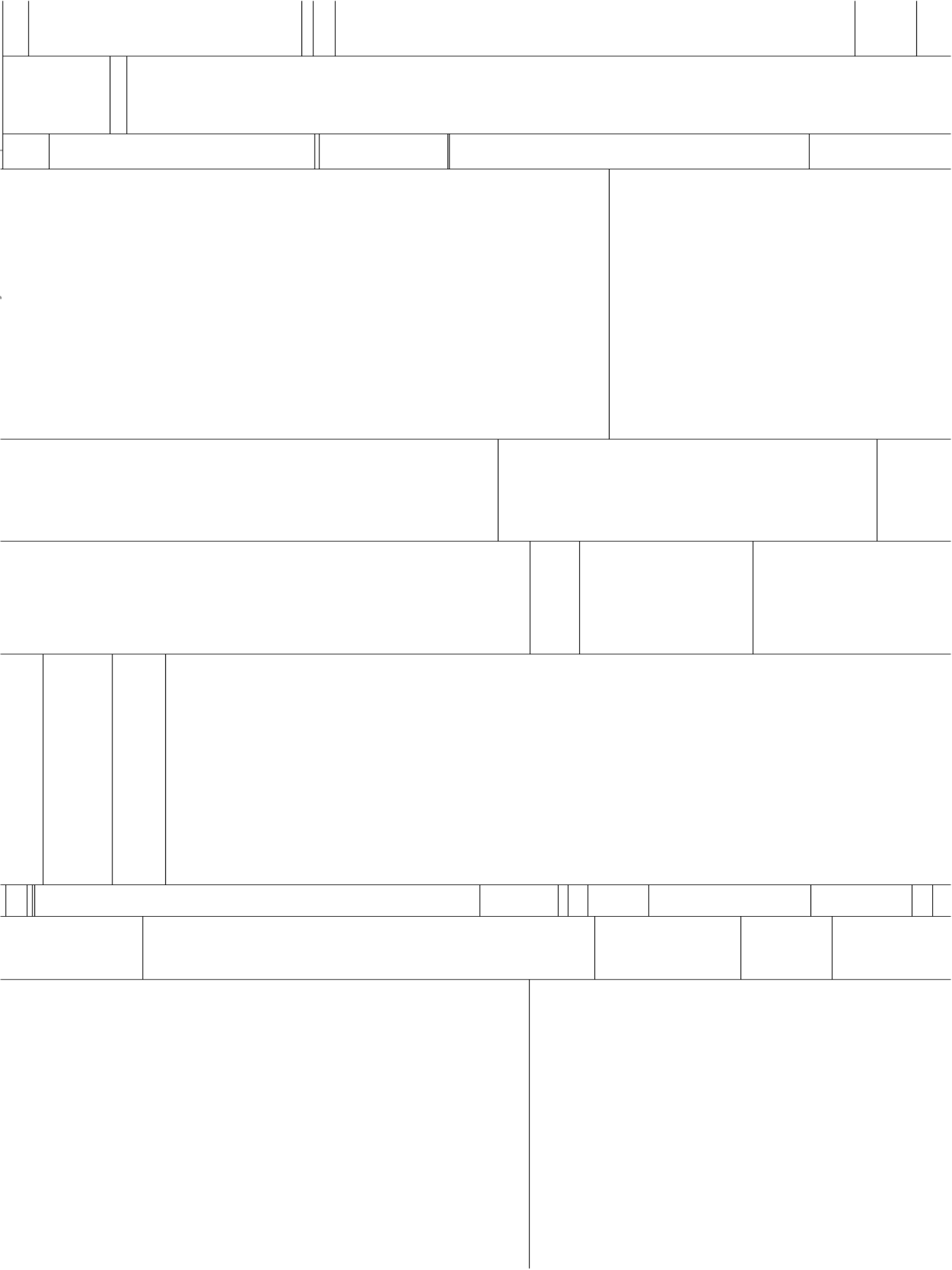
Sample Only the Active Layer/Mask

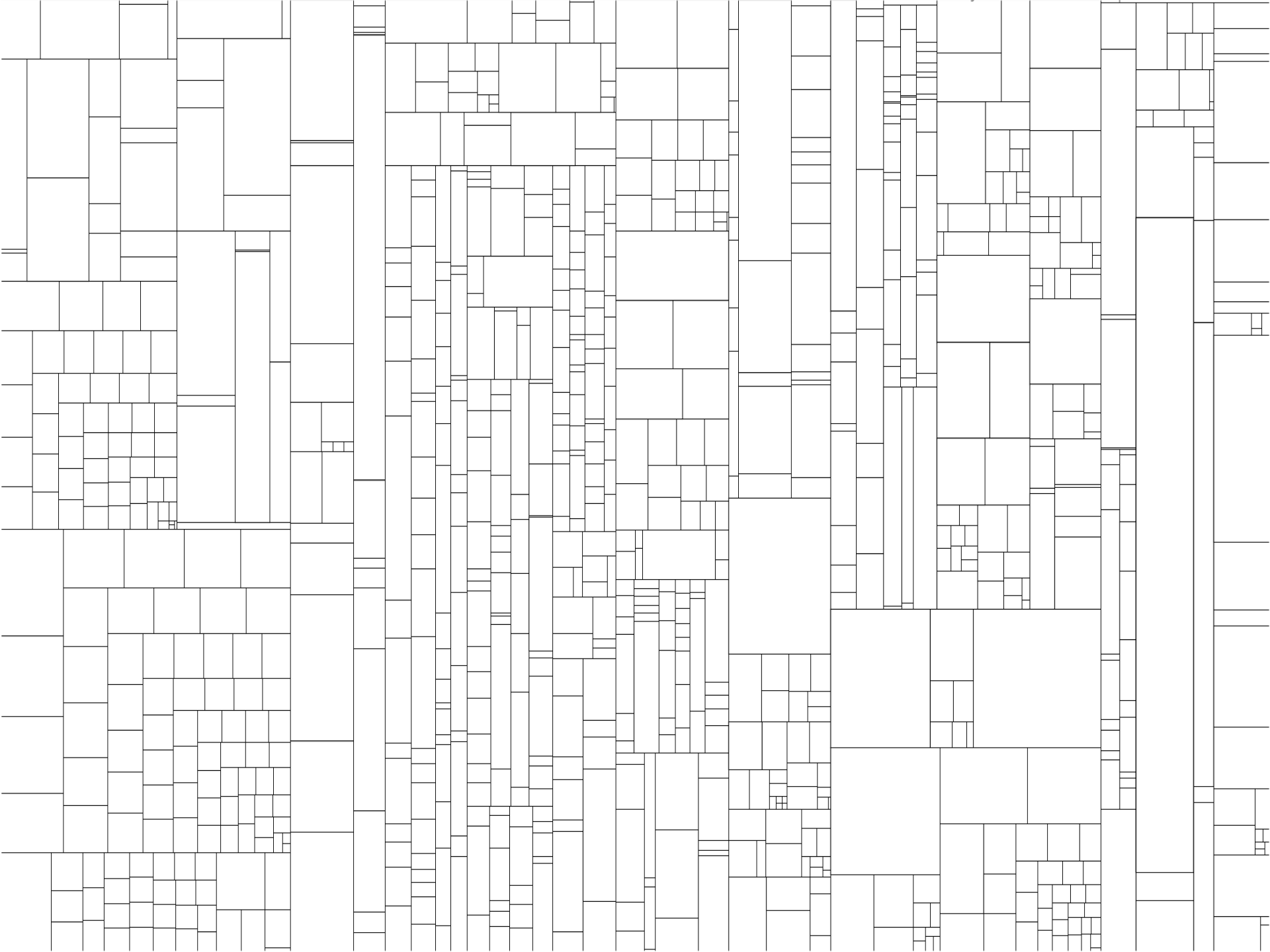
Untitled1 × Picture1.png ×

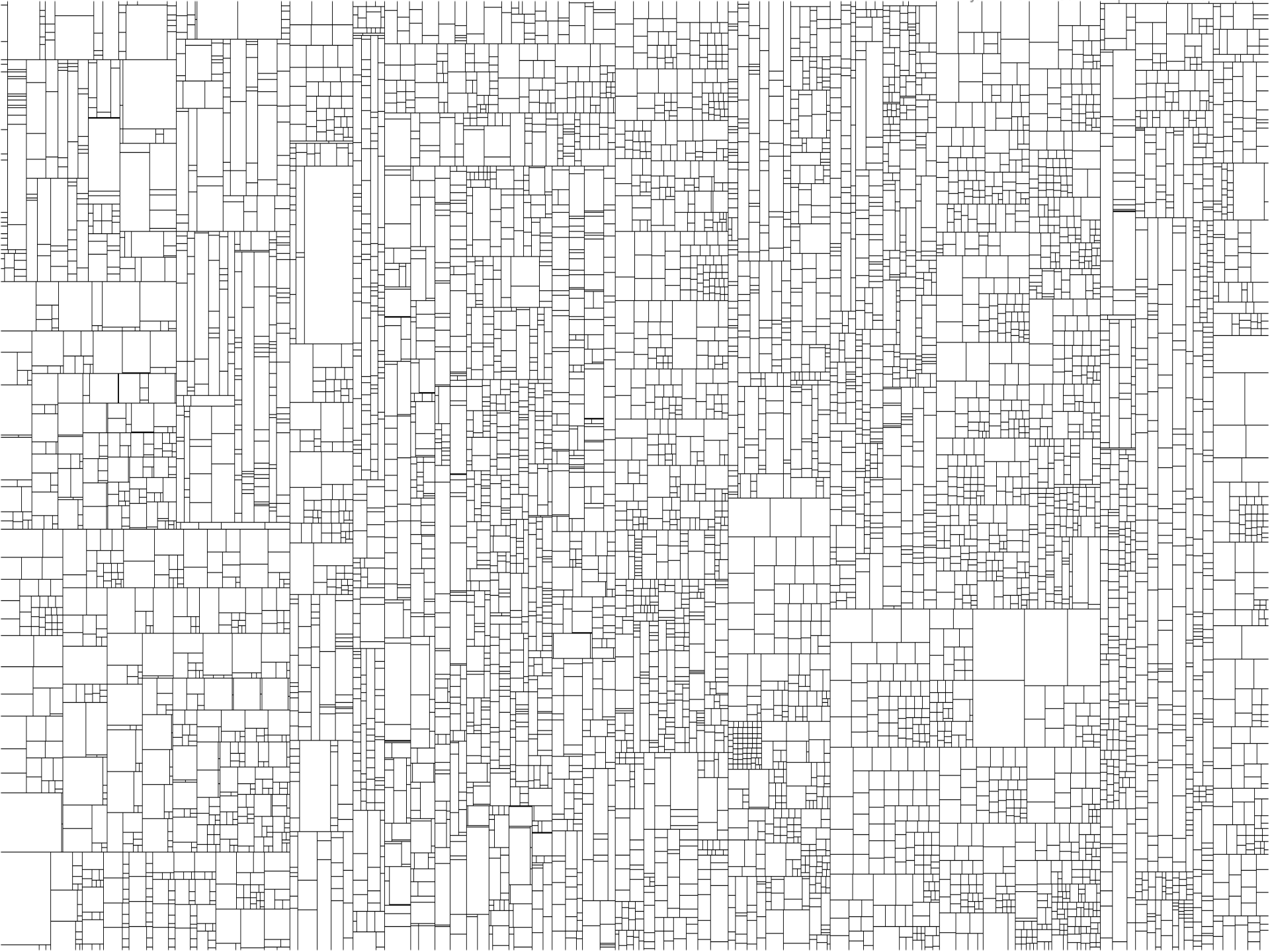
Picture1.png was opened.

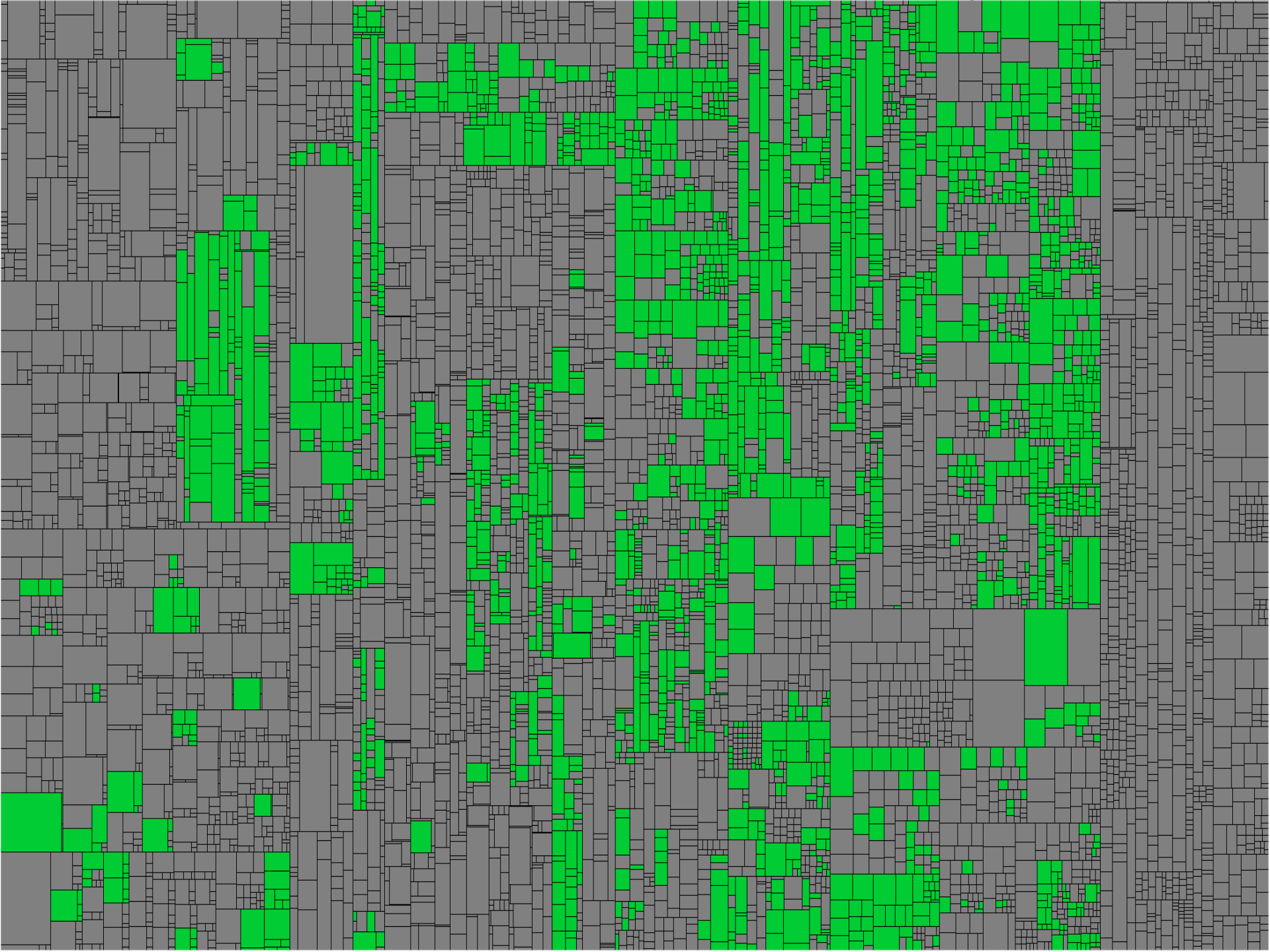
Zoom:

```
114
115     private static void createAndShowGUI(String[] args) {
116         assert calledOnEDT() : threadInfo();
117
118         Messages.setMsgHandler(new GUIMessageHandler());
119
120 //        GlobalKeyboardWatch.showEventsSlowerThan(100, TimeUnit.MILLISECONDS);
121
122         Theme theme = Themes.DEFAULT;
123         // if a LaF was set from the command line, then don't override it
124         if (System.getProperty("swing.defaultlaf") == null) {
125             theme = AppPreferences.loadTheme();
126             Themes.install(theme, false, true);
127         }
128
129         int uiFontSize = AppPreferences.loadUIFontSize();
130         String uiFontType = AppPreferences.loadUIFontType();
131
132         Font defaultFont = UIManager.getFont("defaultFont");
133         if (defaultFont != null) { // if null, we don't know how to set the font
134             if (uiFontSize != 0 || !uiFontType.isEmpty()) {
135                 Font newFont;
136                 if (!uiFontType.isEmpty()) {
137                     newFont = new Font(uiFontType, Font.PLAIN, uiFontSize);
138                 } else {
139                     newFont = defaultFont.deriveFont((float) uiFontSize);
140                 }
141
142                 FontUIResource fontUIResource = new FontUIResource(newFont);
143                 UIManager.put("defaultFont", fontUIResource);
144
145                 if (theme.isNimbus()) {
146                     UIManager.getLookAndFeel().getDefaults().put("defaultFont", fontUIResource);
147                 }
148             }
149         }
150
151         var pw = PixelitorWindow.get();
152         Dialogs.setMainWindowInitialized(true);
153
154         // Just to make 100% sure that at the end of GUI
155         // initialization the focus is not grabbed by
156         // a textfield and the keyboard shortcuts work properly
157         FgBgColors.getGUI().requestFocus();
158
159         TipsOfTheDay.showTips(pw, false);
160
161         MouseZoomMethod.load();
162         PanMethod.load();
163
164         // The IO-intensive preloading of fonts is scheduled
165         // to run after all the files have been opened,
166         // and on the same IO thread
167         openCLFilesAsync(args)
168             .exceptionally(throwable -> null) // recover
169             .thenAcceptAsync(v -> afterStartTestActions(), onEDT)
170             .thenRunAsync(Utils::preloadFontNames, onIOThread)
171             .exceptionally(Messages::showExceptionOnEDT);
172     }
173
```

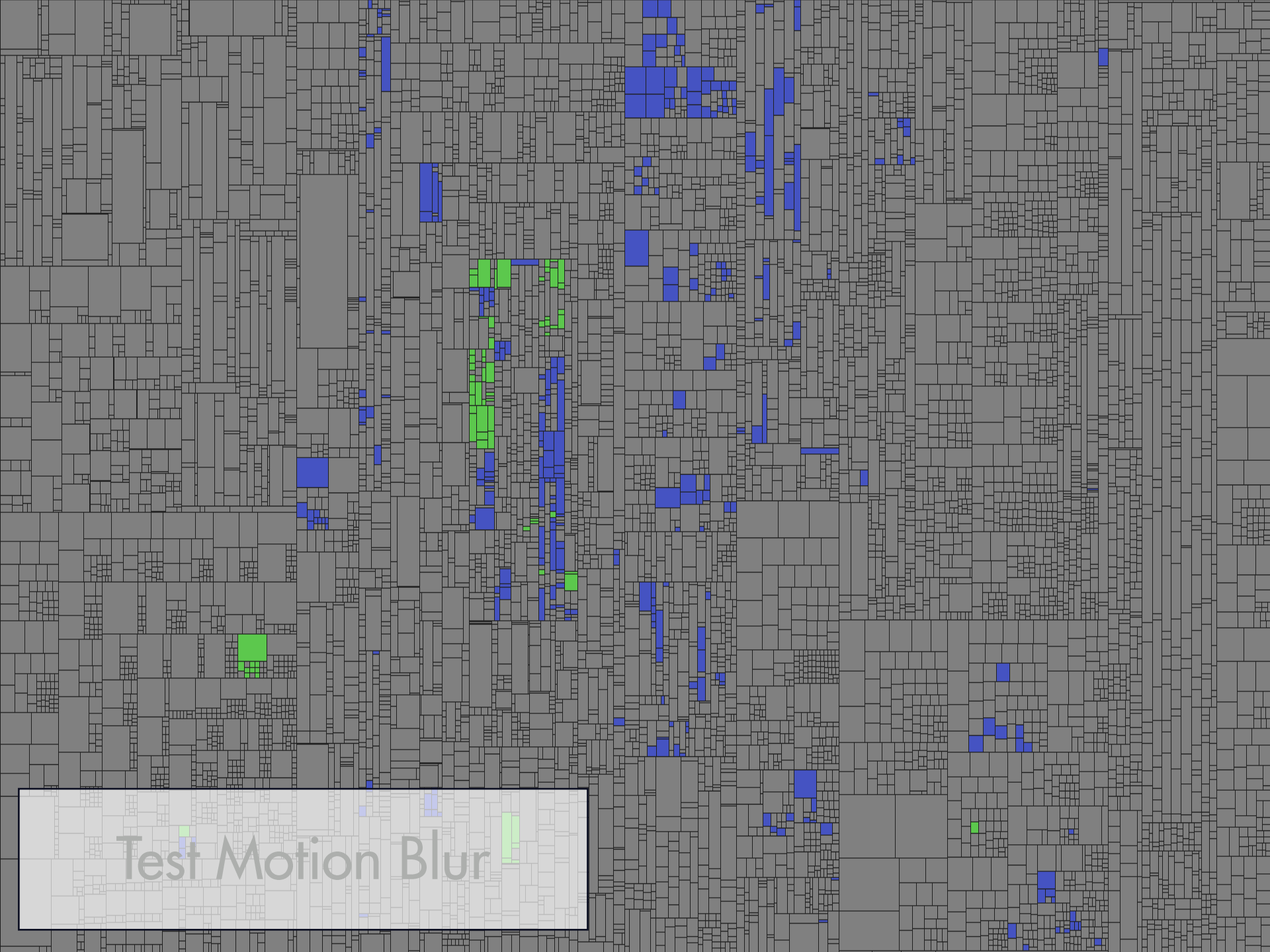


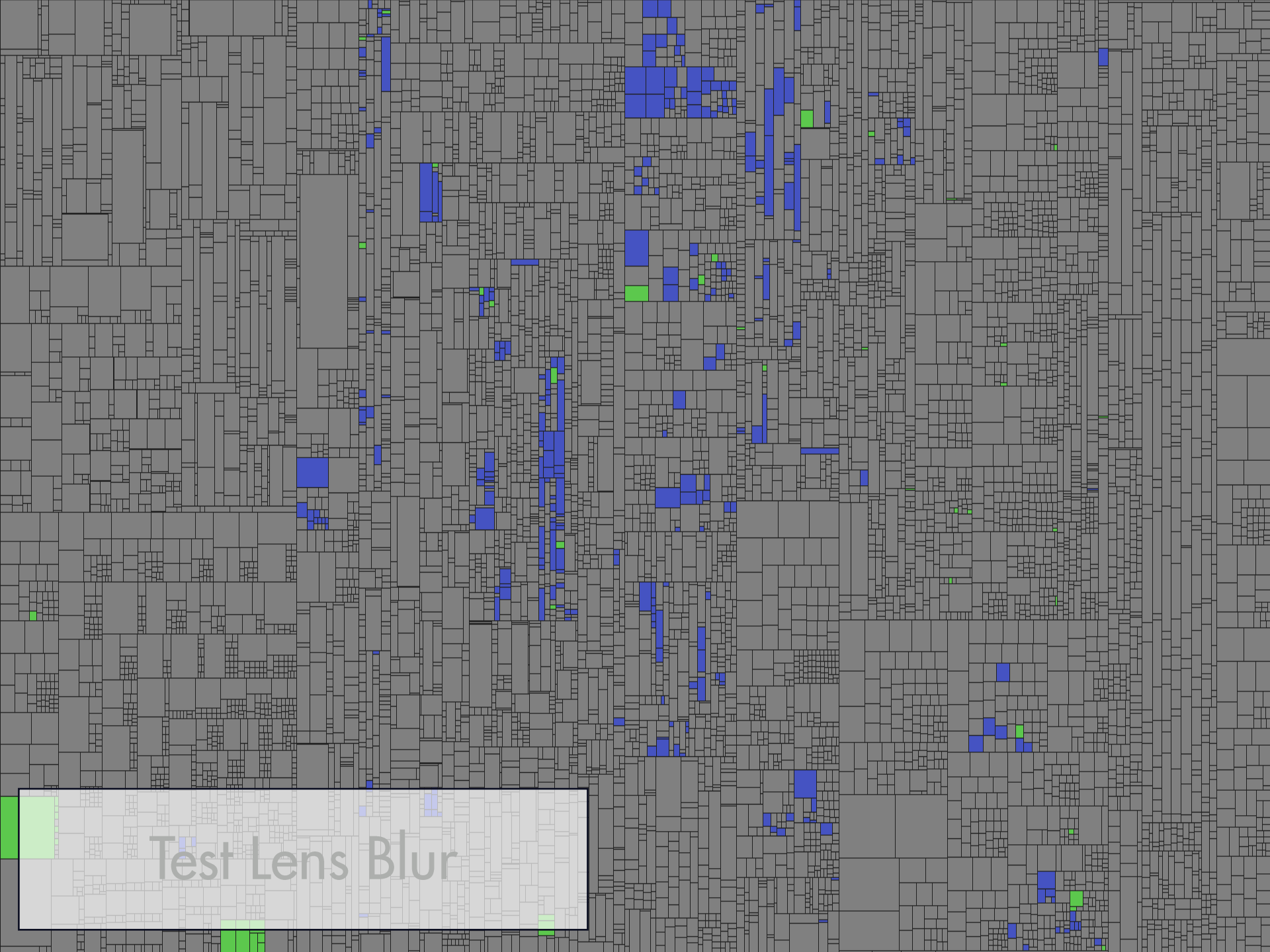


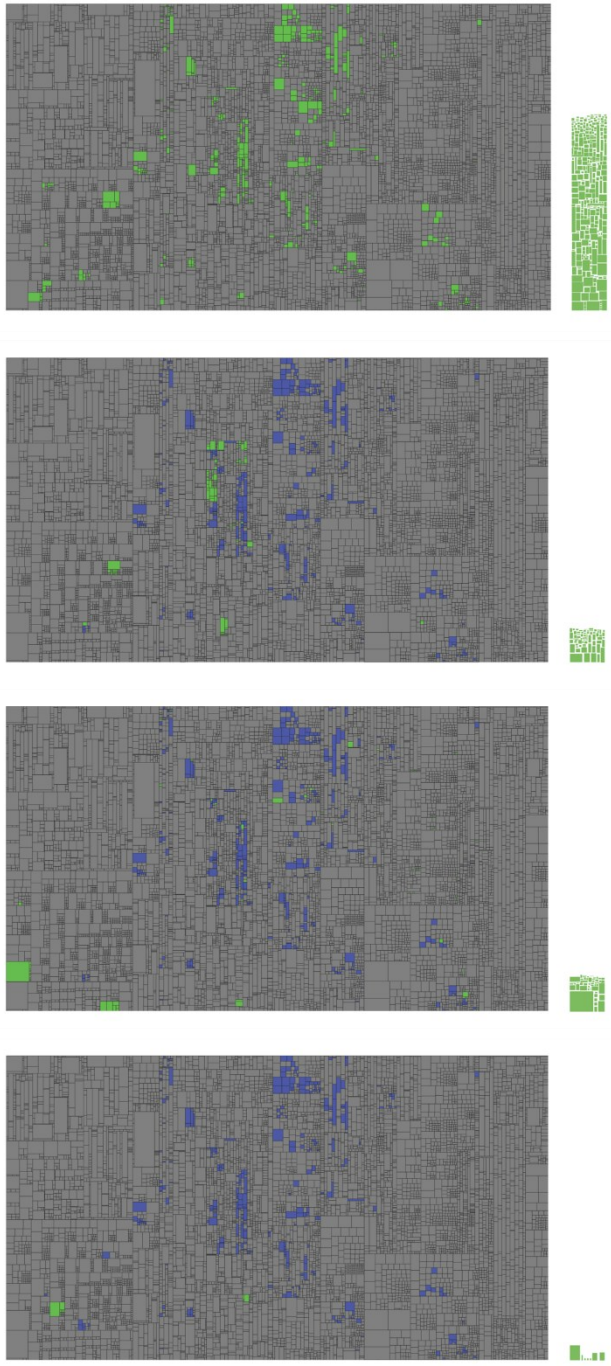




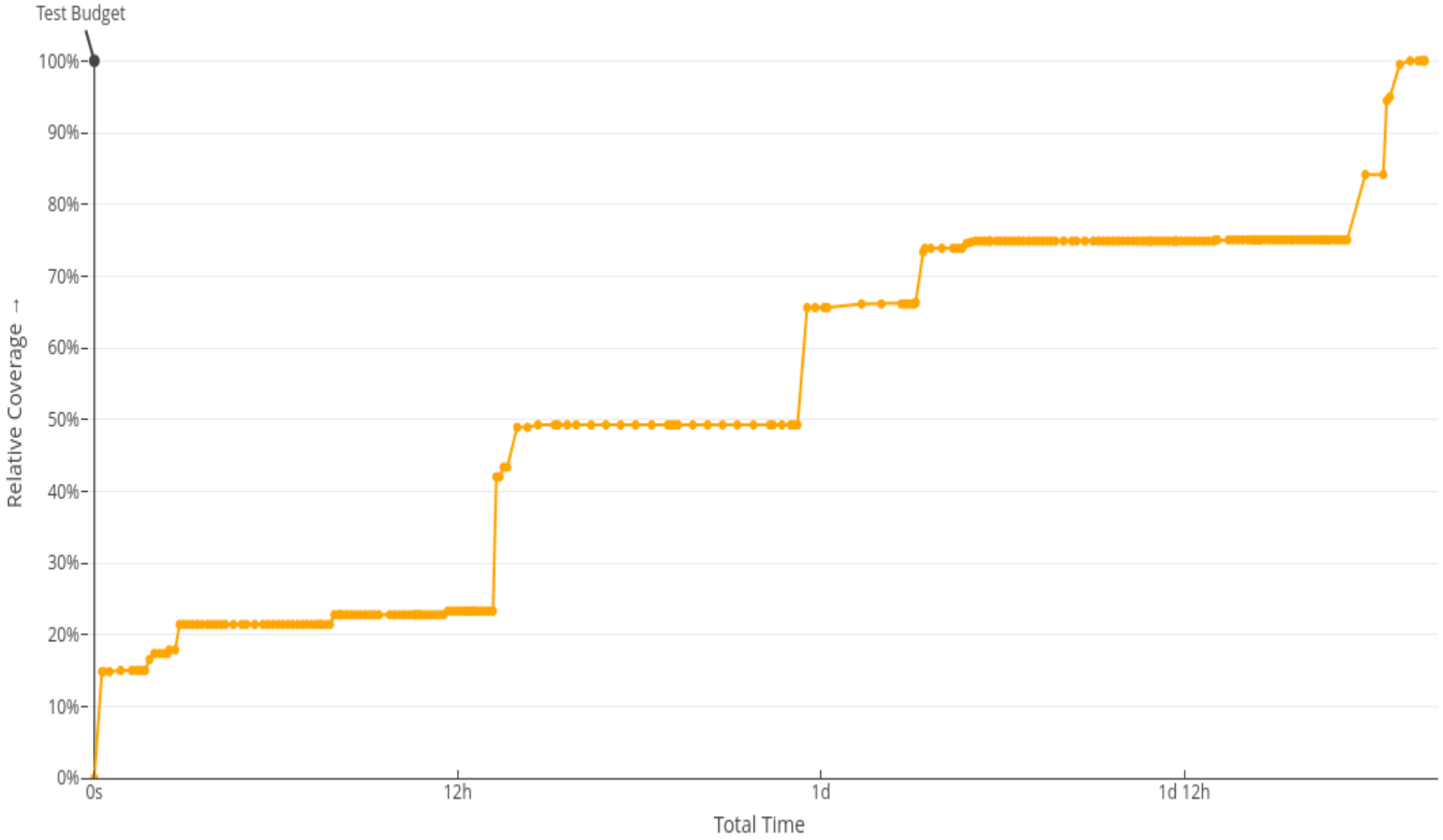

Test Gaussian Blur


Test Motion Blur


Test Lens Blur



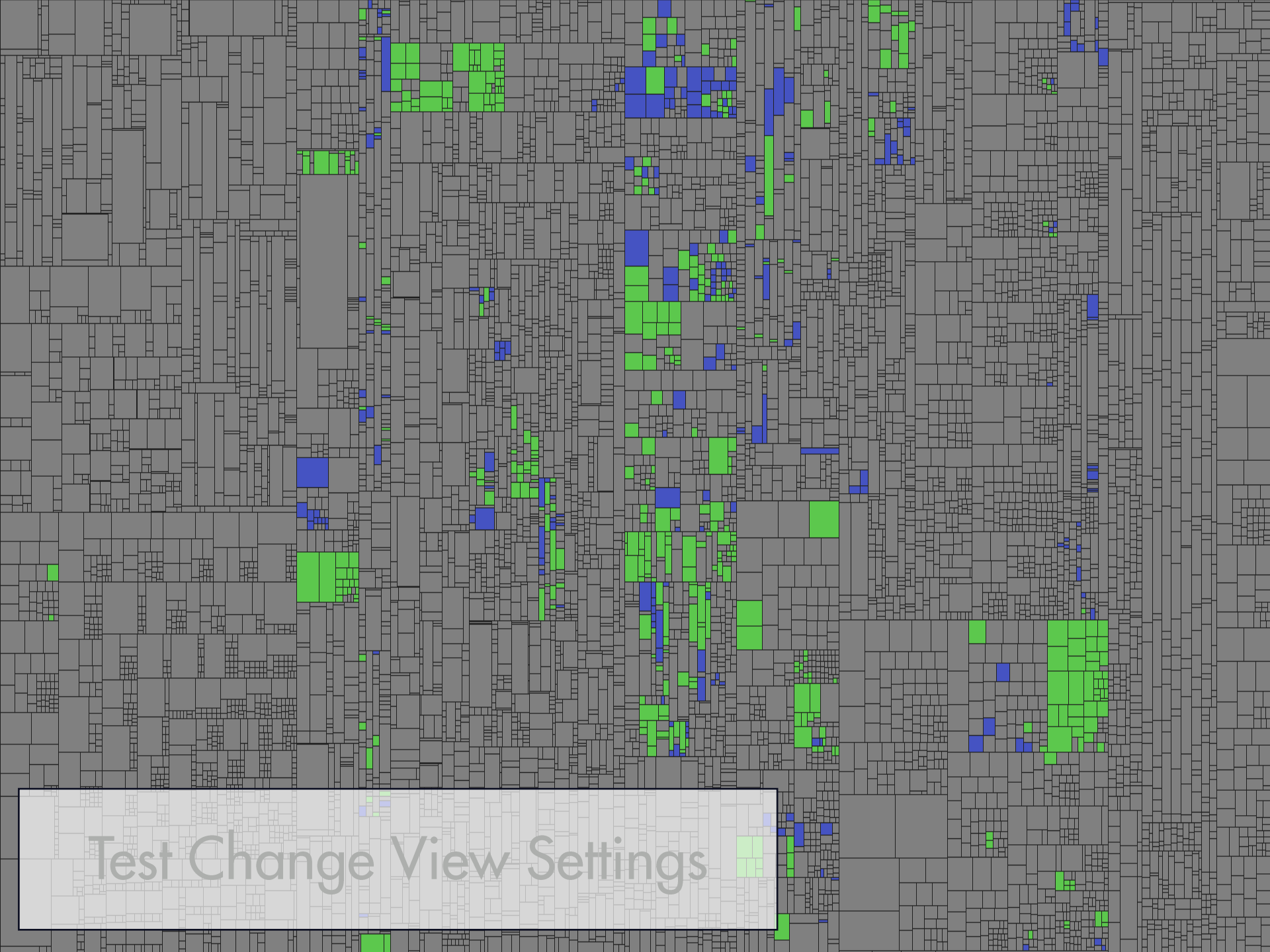
Coverage over Time

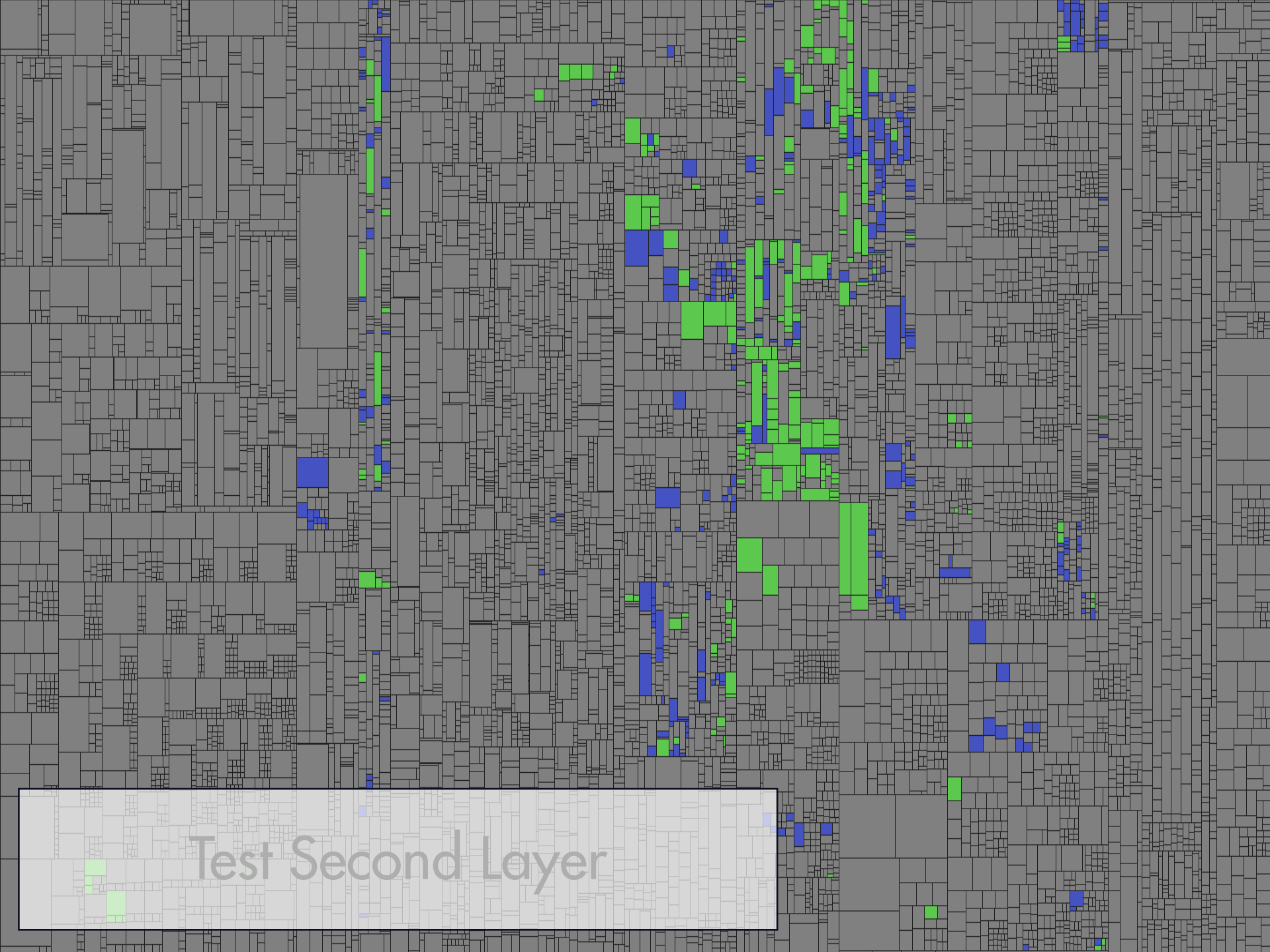


Results for Test Query & Budget Restriction

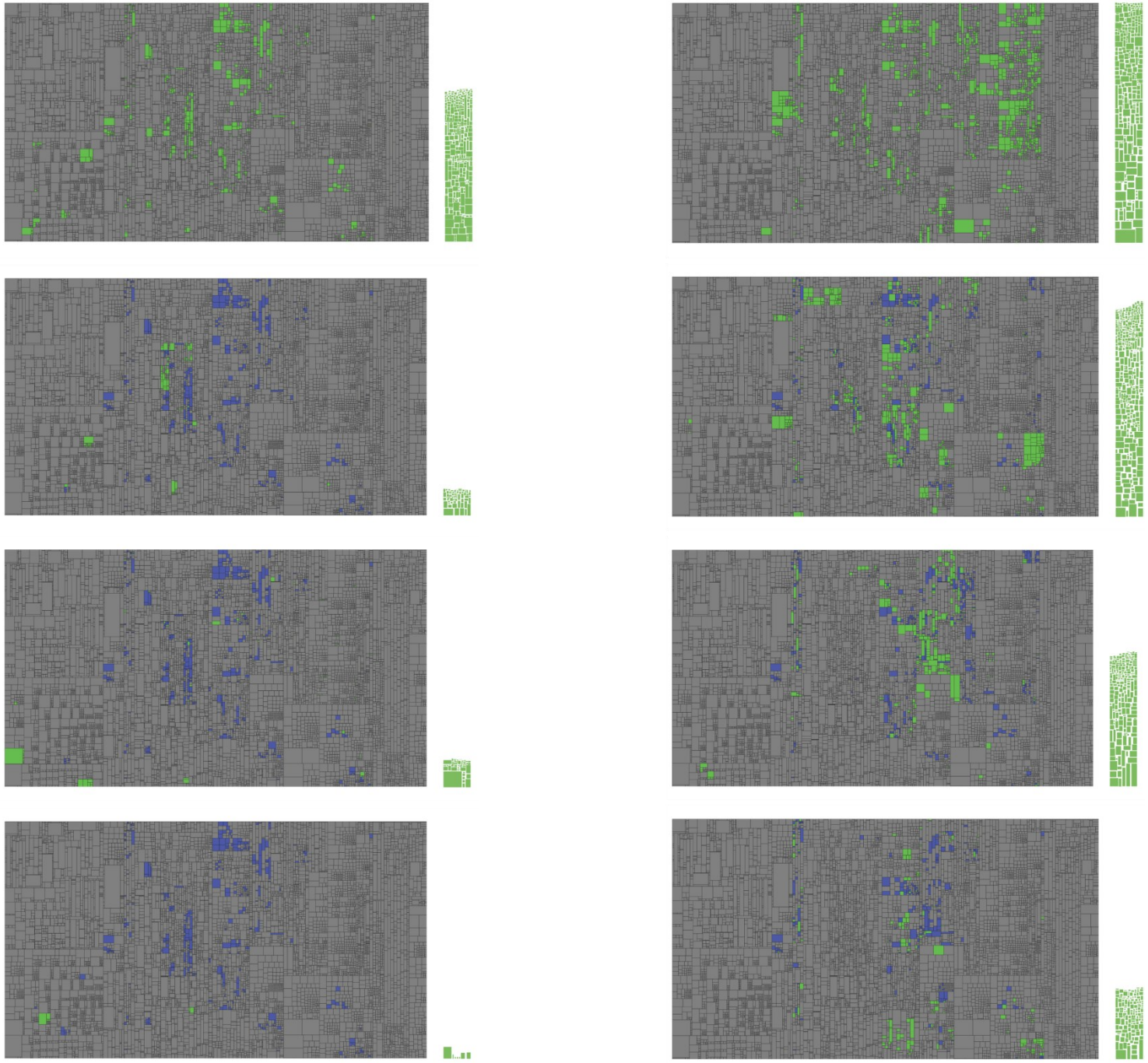
Relative Coverage: 0%, Selected Tests: 0 out of 236 (0%)


Test Create and Modify
Selection

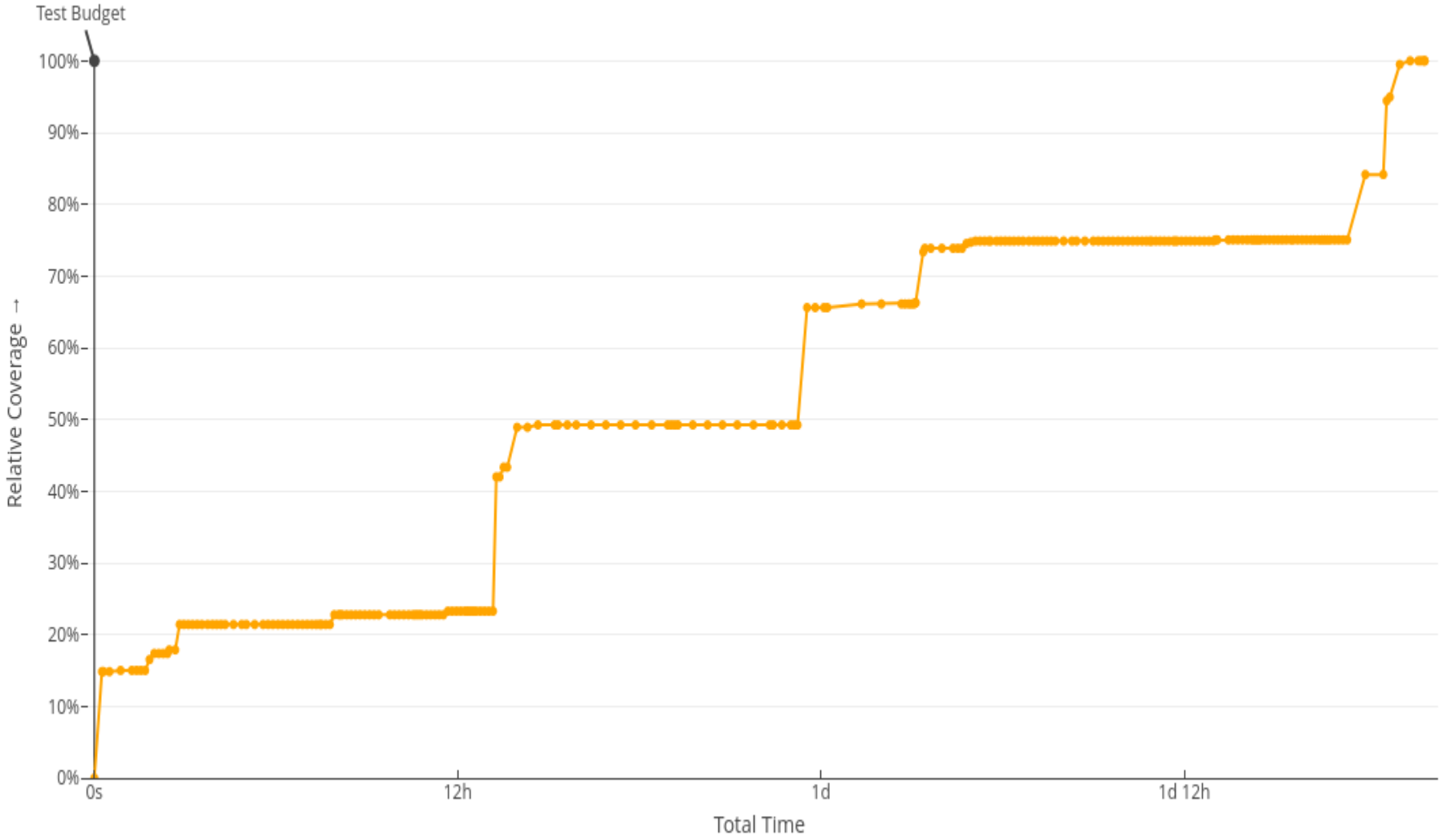

Test Change View Settings


Test Second Layer


Test Save Image



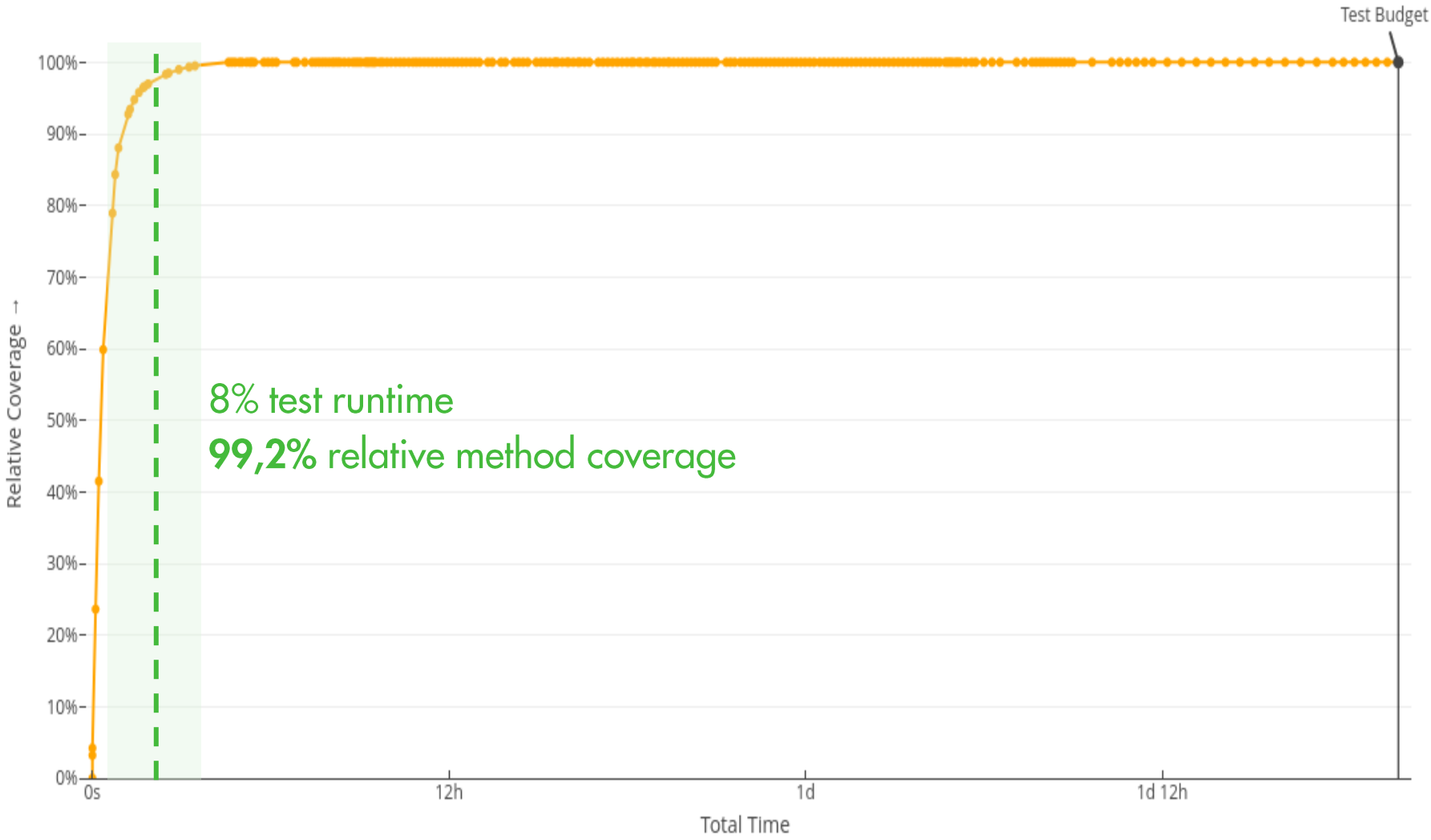
Coverage over Time



Results for Test Query & Budget Restriction

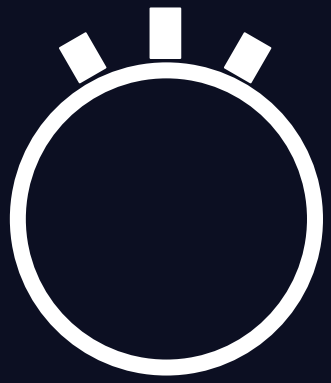
Relative Coverage: 0%, Selected Tests: 0 out of 236 (0%)

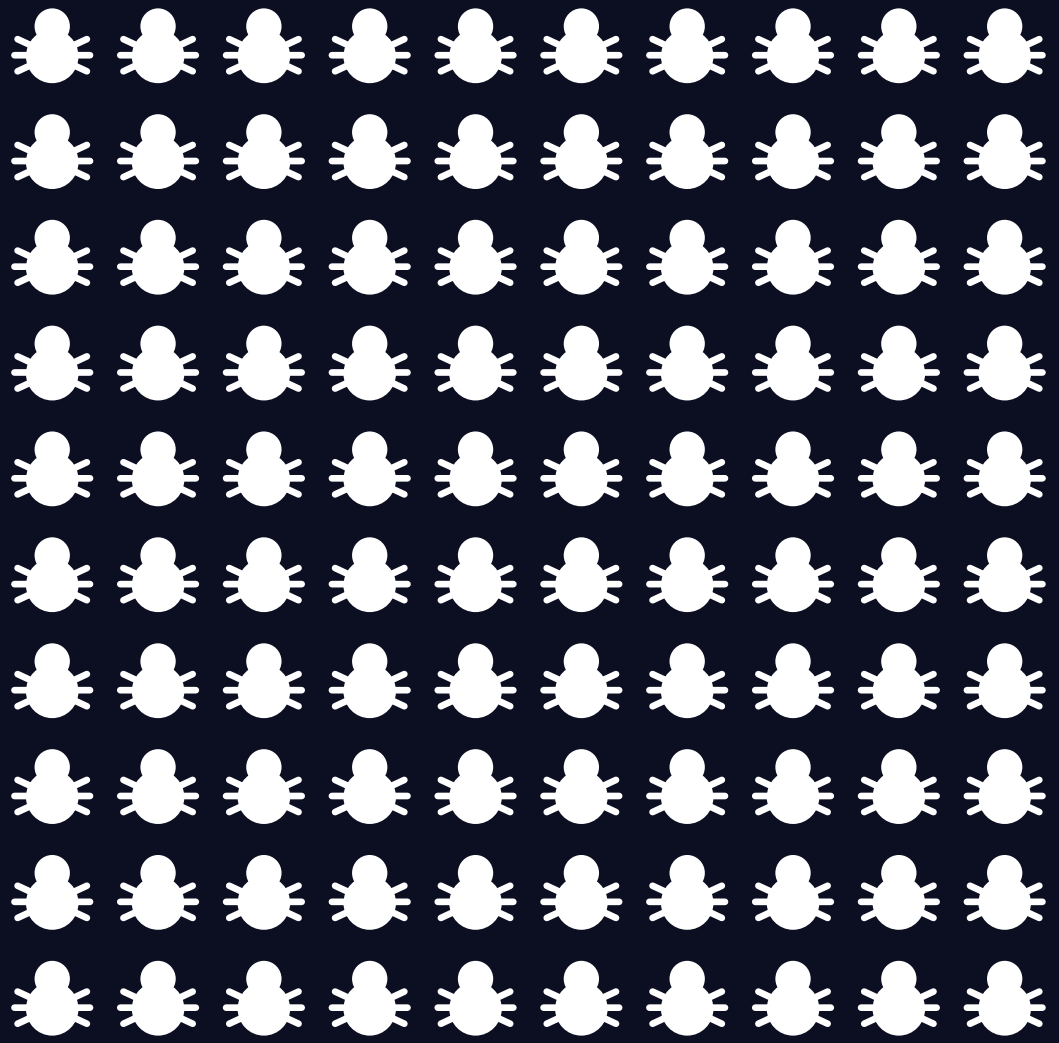
Coverage over Time



Results for Test Query & Budget Restriction

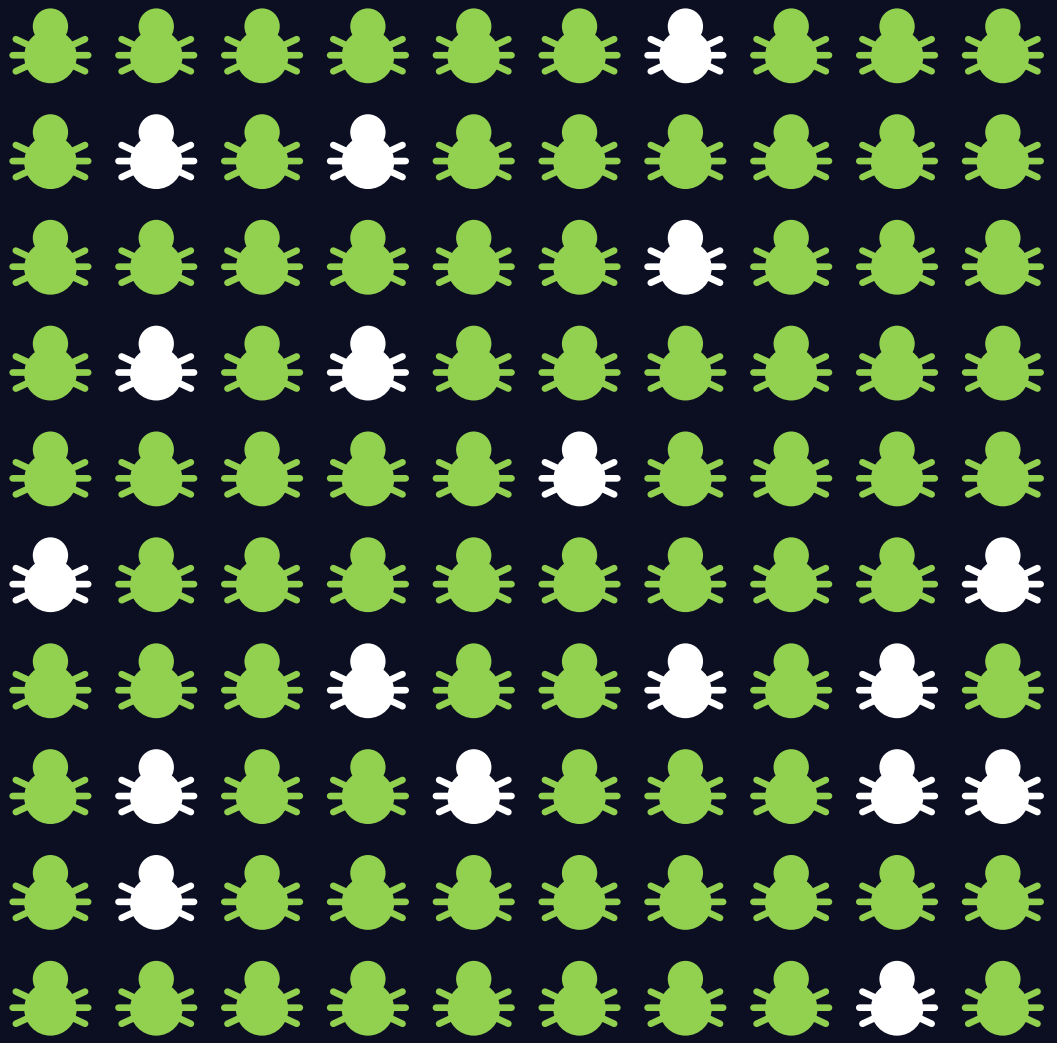
Relative Coverage: 100%, Selected Tests: 236 out of 236 (100%)





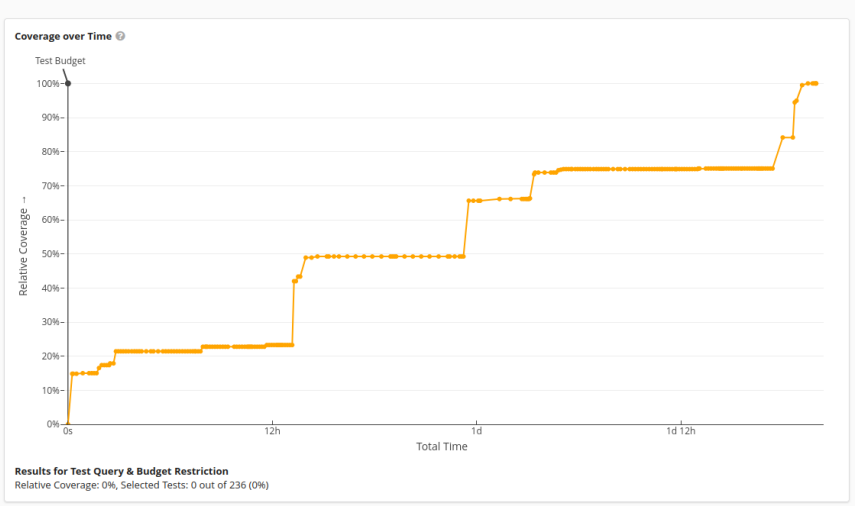
RBCN25

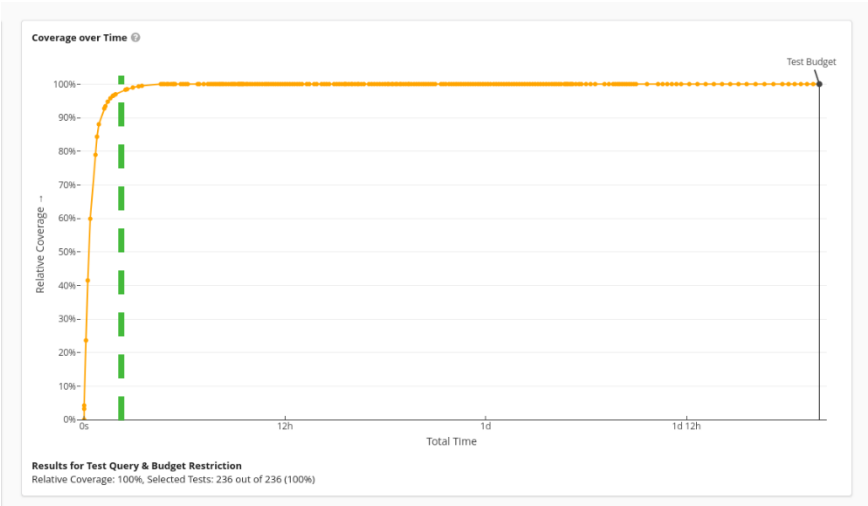
80%



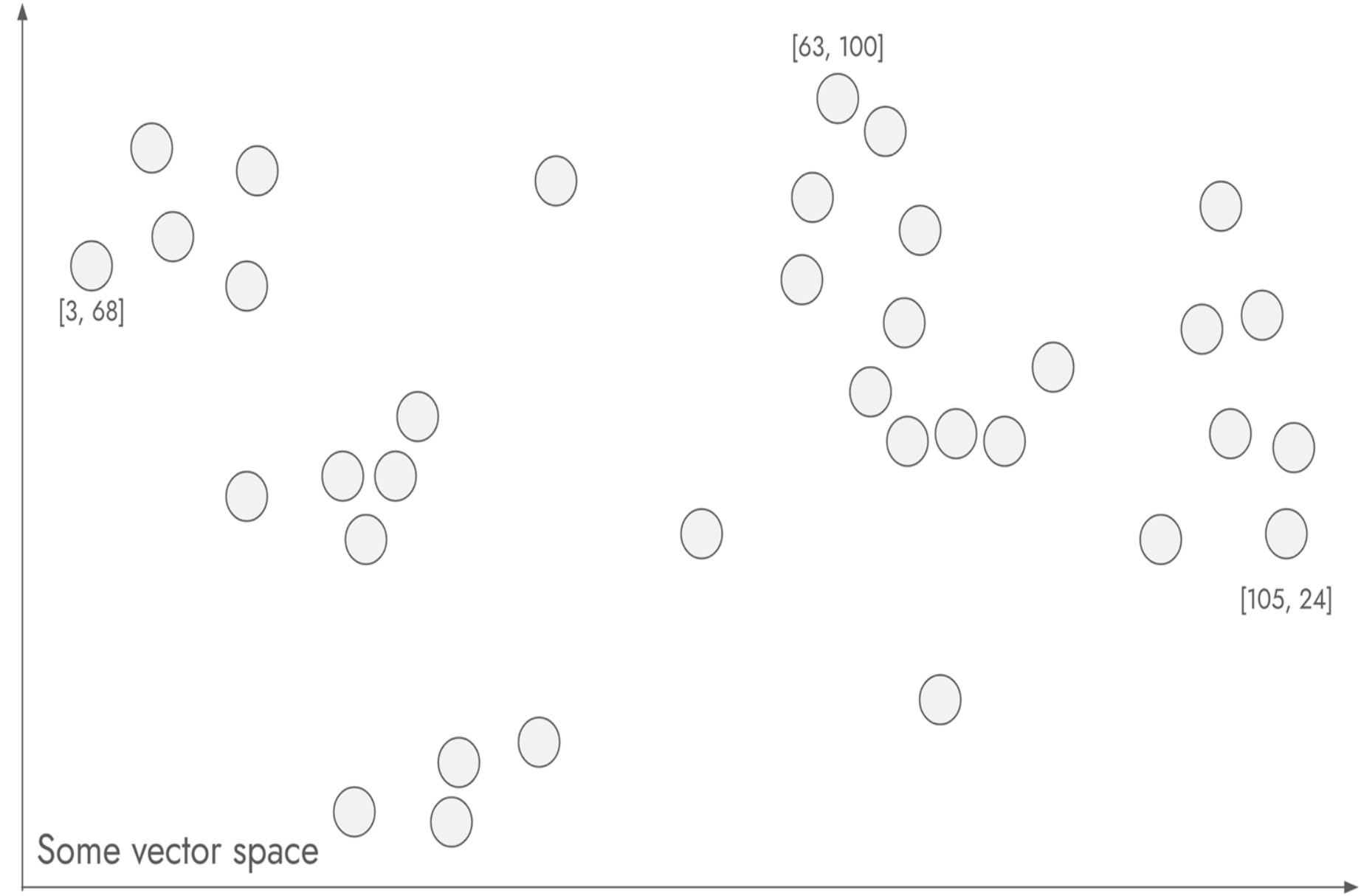
90%

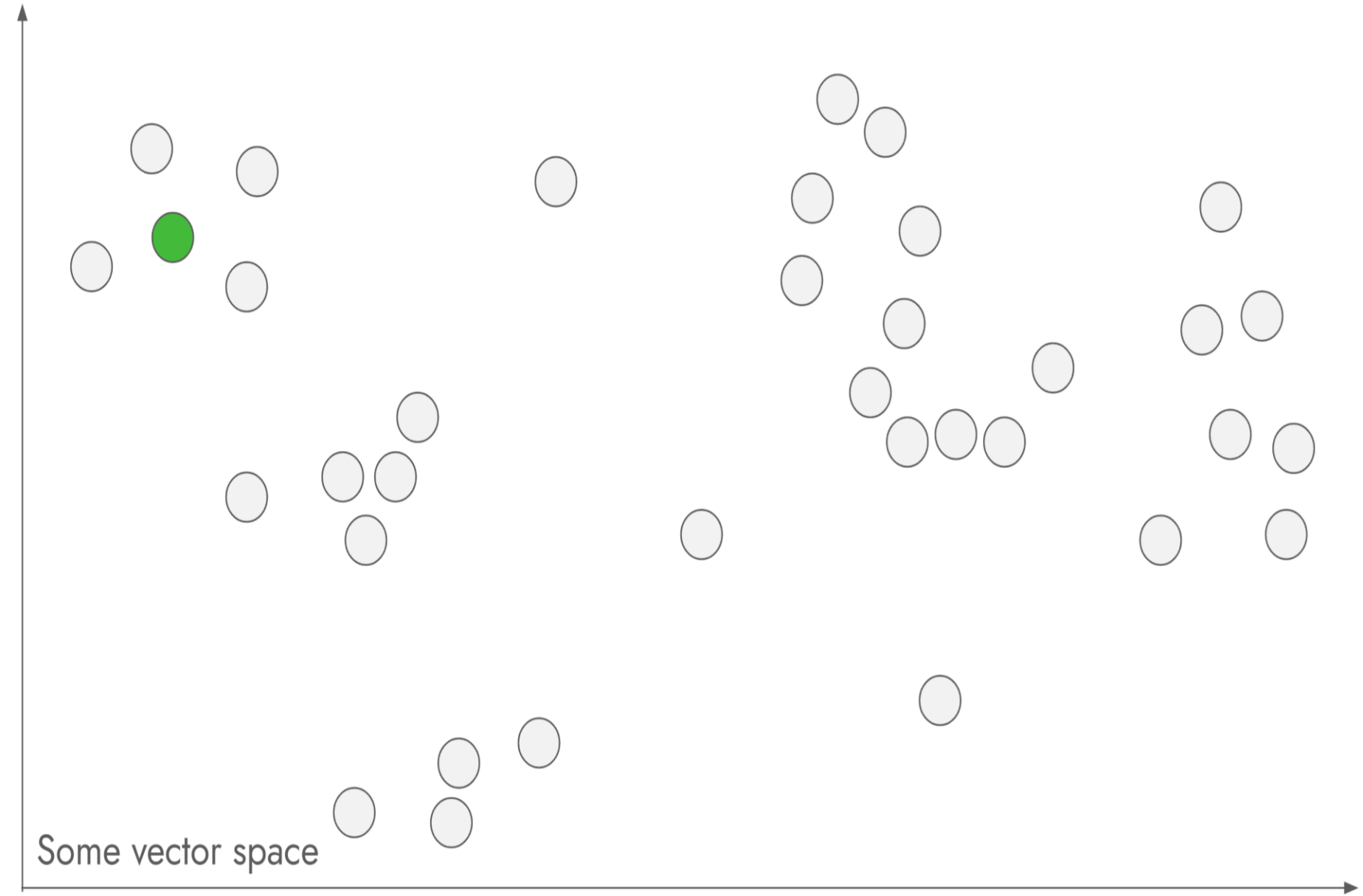
? Without Test-Case-Specific Code Coverage ?

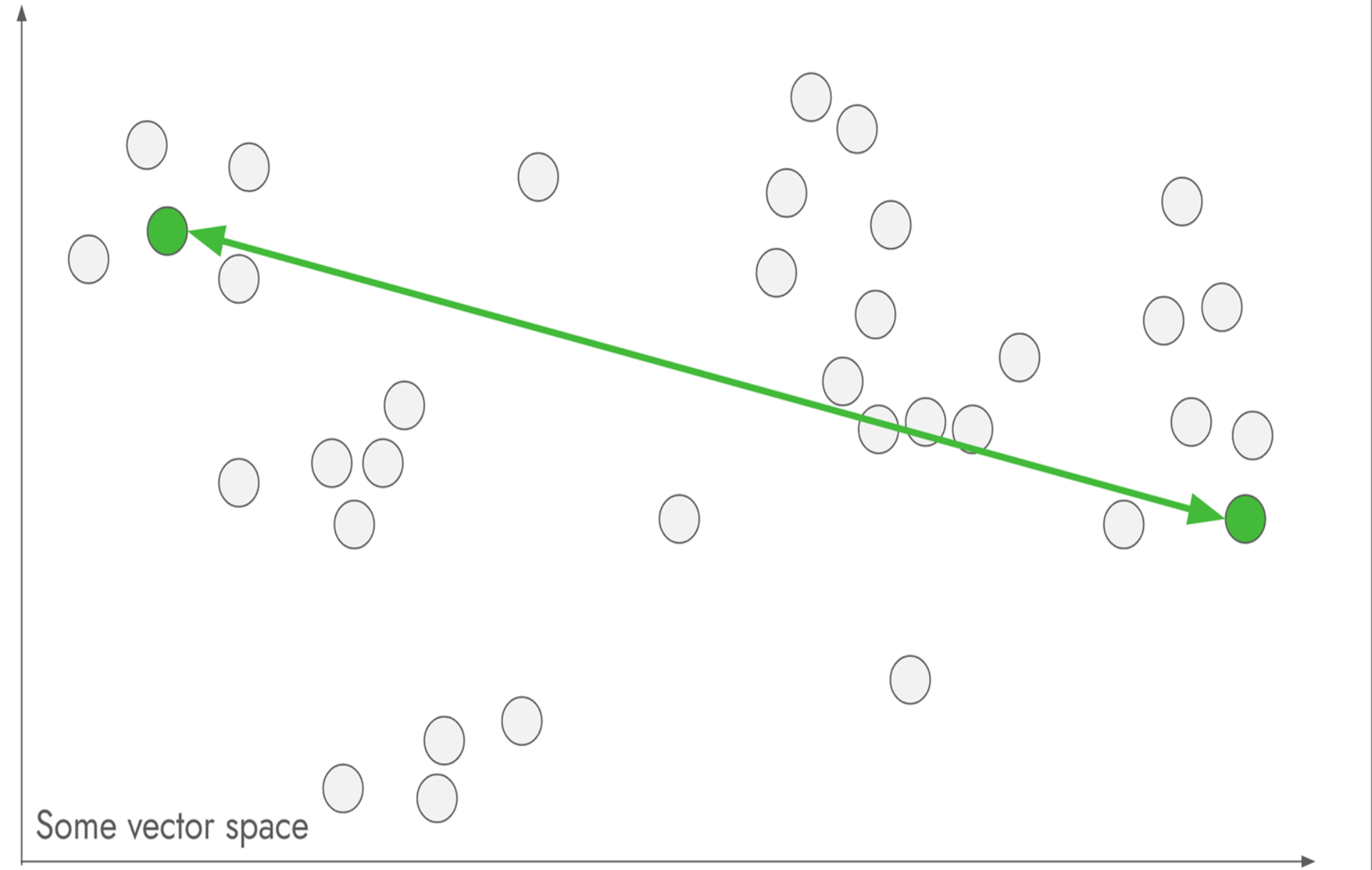


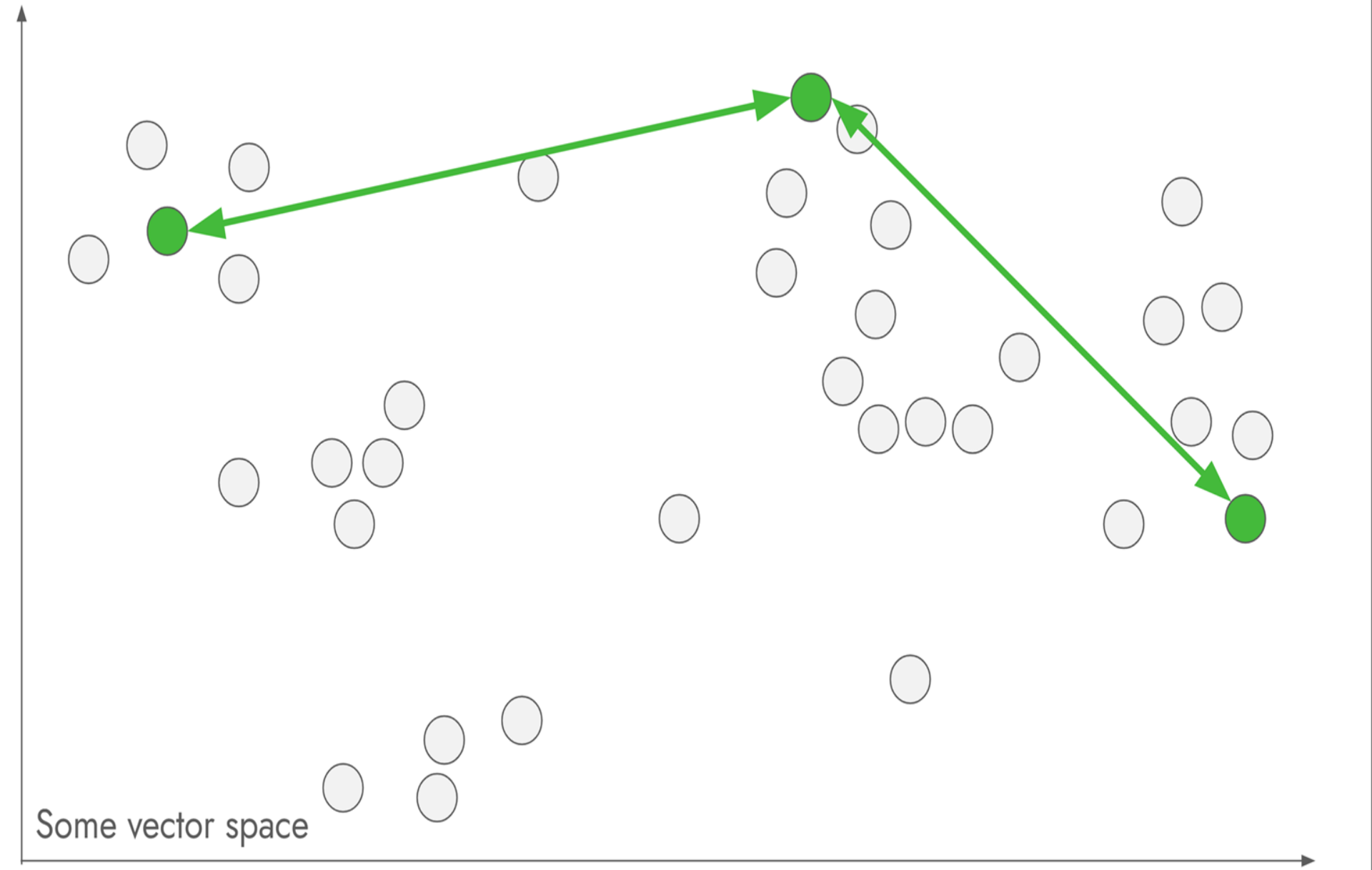


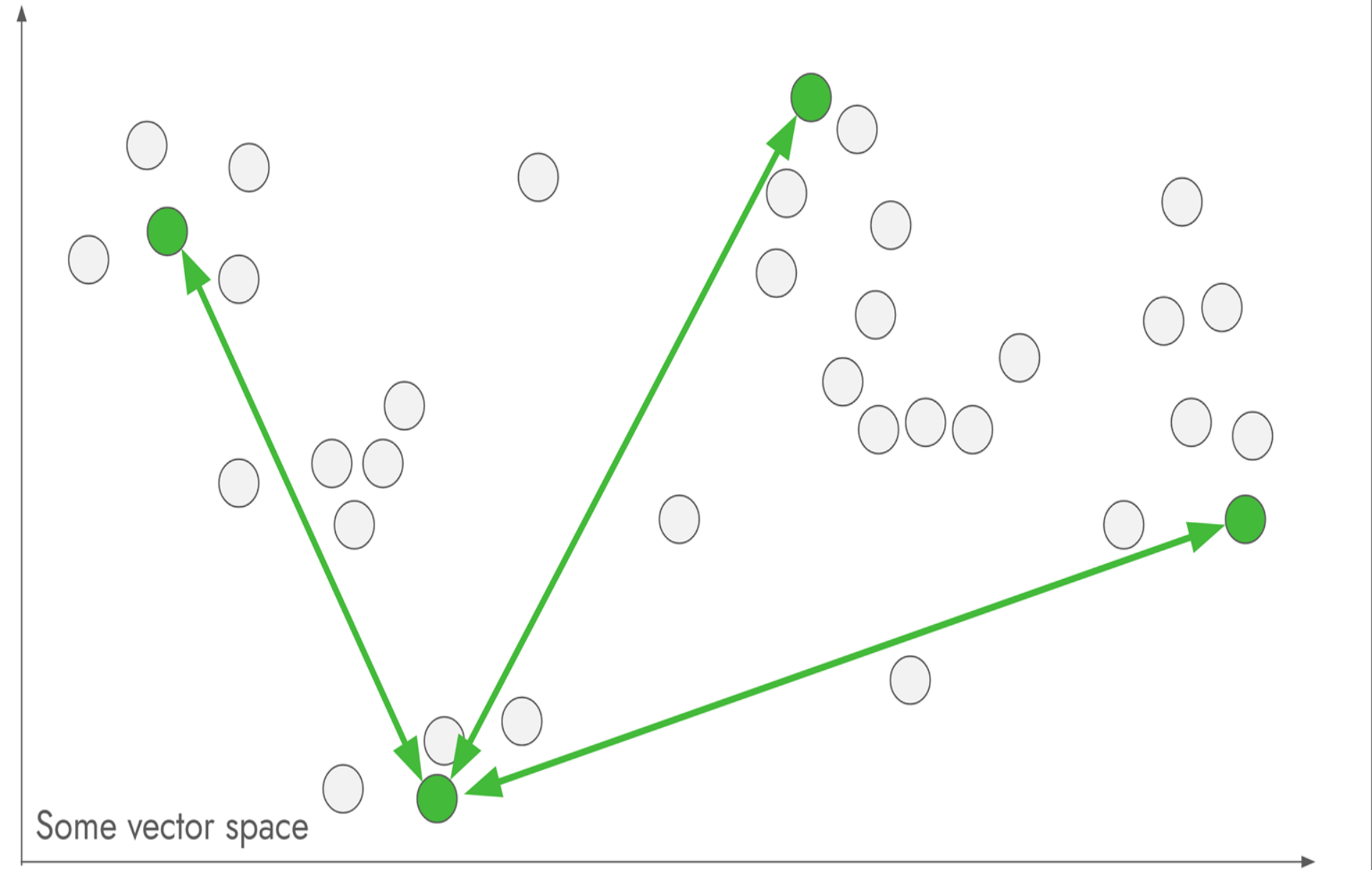
Sort by “Dissimilarity”


[63, 100]
[3, 68]
[105, 24]
Some vector space


Some vector space


Some vector space


Some vector space


Some vector space

An Evaluation of Distance Based Test Suite Reduction Techniques

Alessandro Escher
Technical University of Munich
Munich, Germany
alessandro.escher@tum.de

Raphael Nömmer
Technical University of Munich
Munich, Germany
noemmer@cqse.eu

***Abstract*—Efficient test suite selection is crucial in software testing due to the high cost of running extensive tests, particularly on large industry projects. Coverage-based techniques aim to maximize system execution within time constraints but often suffer from costly and complex coverage recording processes. This study explores alternative selection methods using test metadata and source code. Hierarchical Agglomerative Clustering (HAC) and a greedy approach were evaluated alongside distance measures based on package path distance and vector representations of test code.**

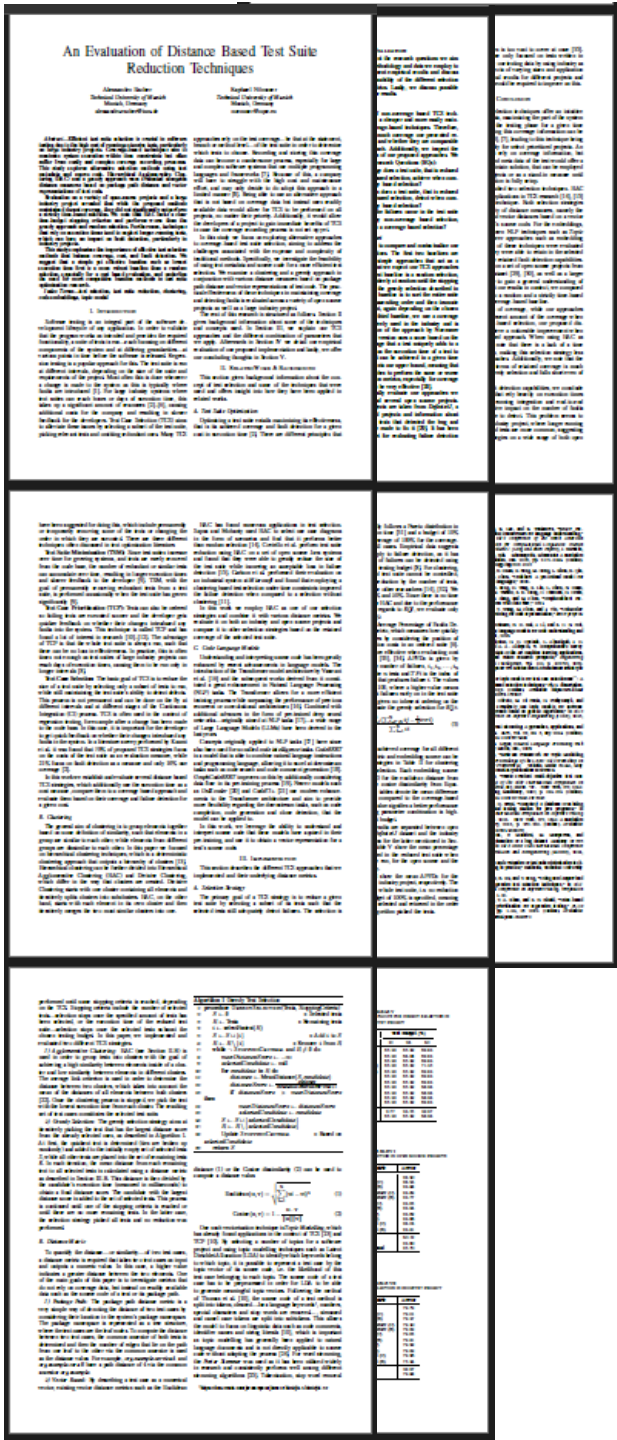
Evaluation on a variety of open-source projects and a large industry project revealed that while the proposed methods maintained decent coverage, they did not significantly outperform a strictly time-based selection. We note that HAC lacks a clear time-budget stopping criterion and performs worse than the greedy approach and random selection. Furthermore, techniques that rely on execution times tend to neglect longer-running tests, which can have an impact on fault detection, particularly in industry projects.

This study emphasizes the importance of effective test selection methods that balance coverage, cost, and fault detection. We suggest that a simple yet effective baseline such as lowest execution time first is a more robust baseline than a random selection, especially for a cost based evaluation, and underline the need for more competitive baseline methods in test suite optimization research.

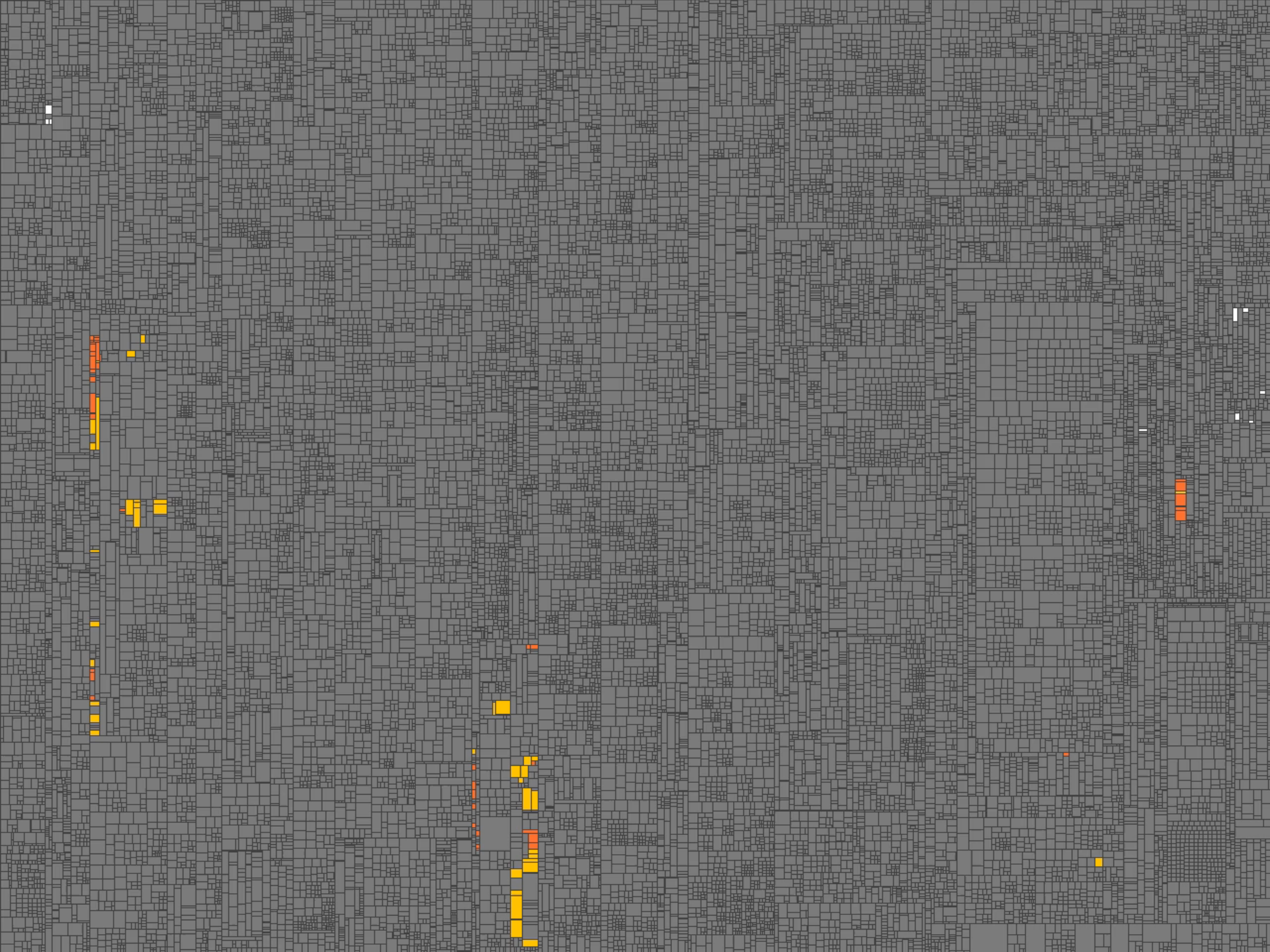
***Index Terms*—test selection, test suite reduction, clustering, code embeddings, topic model**

approaches rely on the test coverage—be that at the statement, branch or method level—of the test suite in order to determine which tests to choose. Recording and storing this coverage data can become a cumbersome process, especially for large and complex software systems that use multiple programming languages and frameworks [7]. Because of this, a company will have to struggle with the high cost and maintenance effort, and may only decide to do adopt this approach in a limited manner [8]. Being able to use an alternative approach that is not based on coverage data but instead uses readily available data would allow for TCS to be performed on all projects, no matter their priority. Additionally, it would allow the developers of a project to gain immediate benefits of TCS in case the coverage recording process is not set up yet.

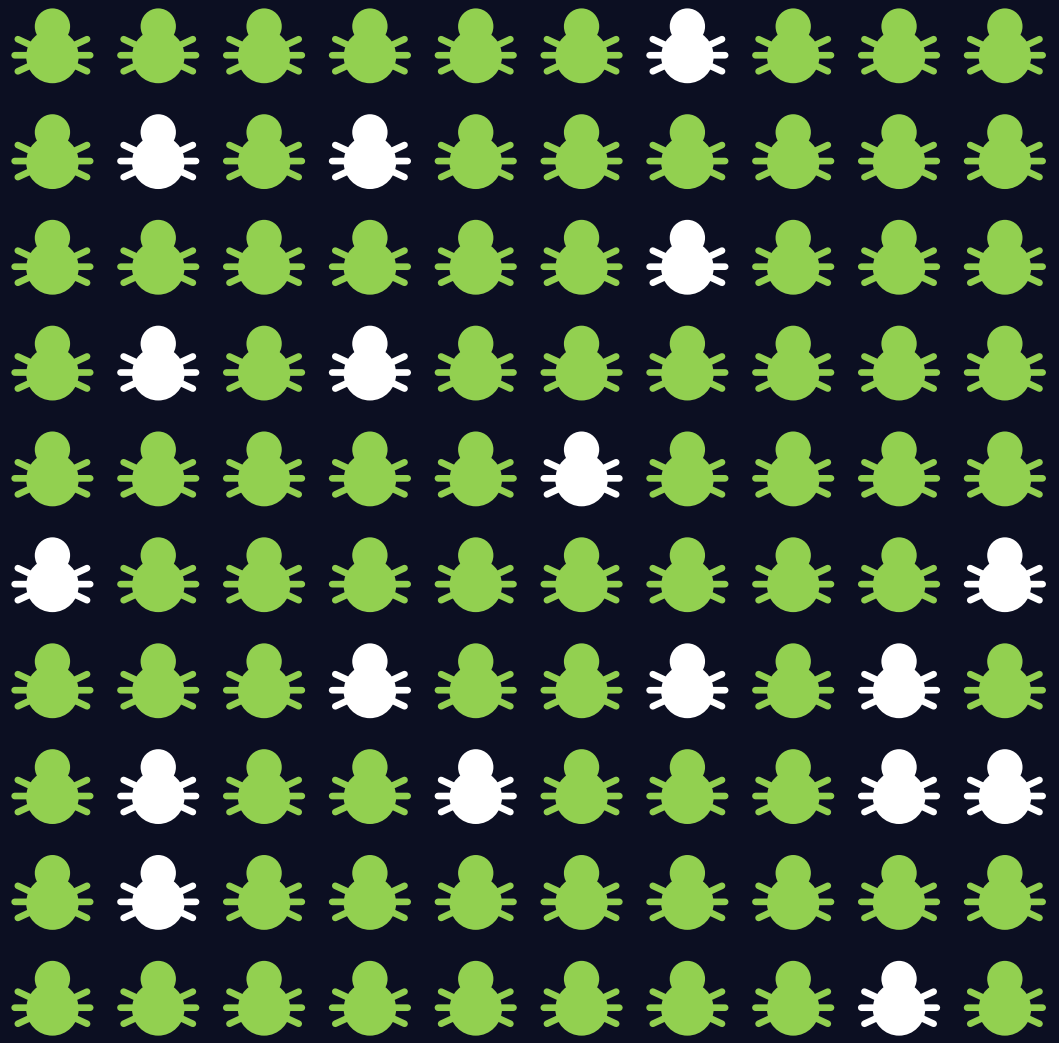
In this study we focus on exploring alternative approaches to coverage-based test suite selection, aiming to address the challenges associated with the expense and complexity of traditional methods. Specifically, we investigate the feasibility of using test metadata and source code for a more efficient test selection. We examine a clustering and a greedy approach in conjunction with various distance measures based on package path distance and vector representations of test code. The practical effectiveness of these techniques in maintaining coverage and detecting faults is evaluated across a variety of open source



Test Selection
for Continuous Integration

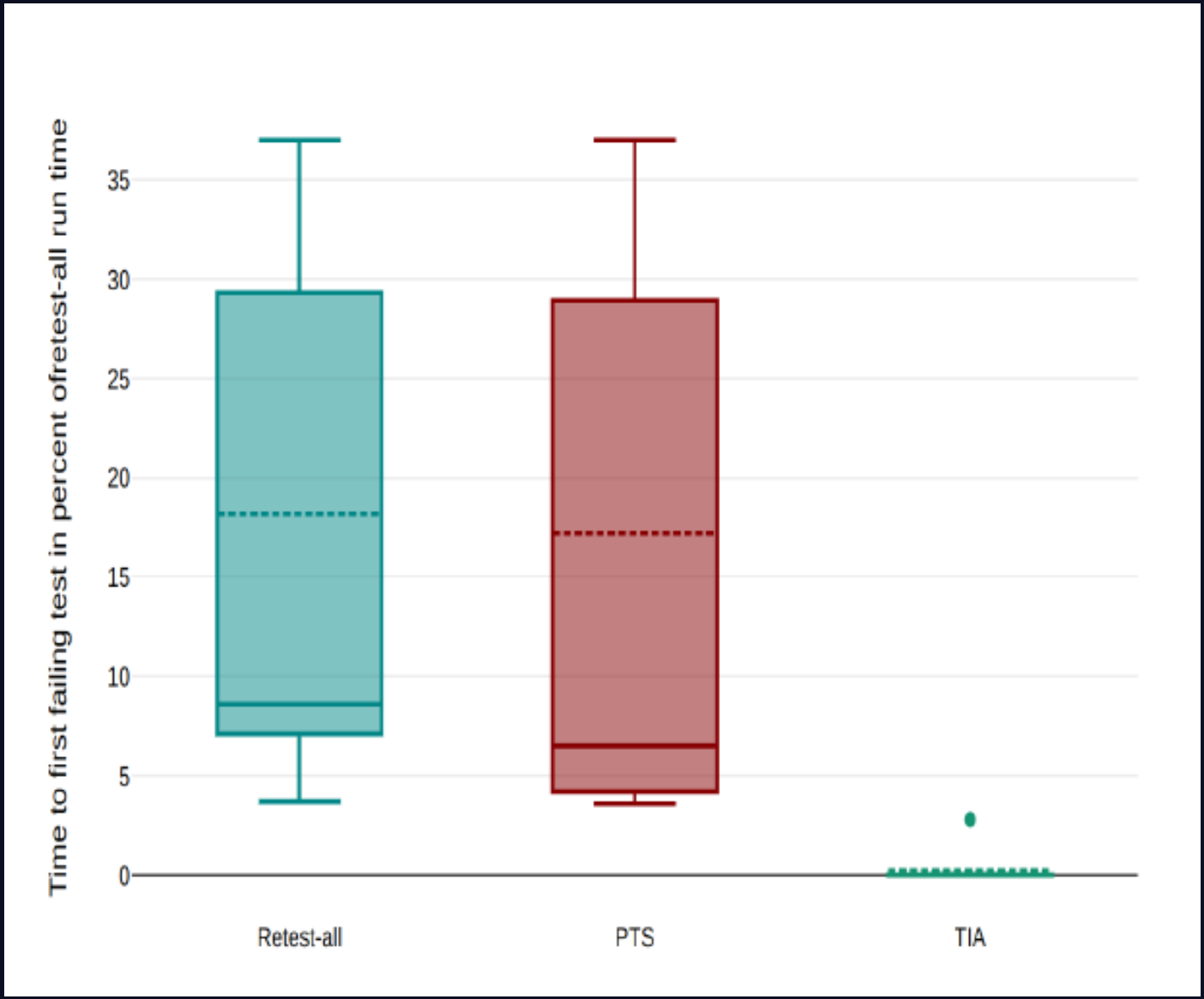


80%

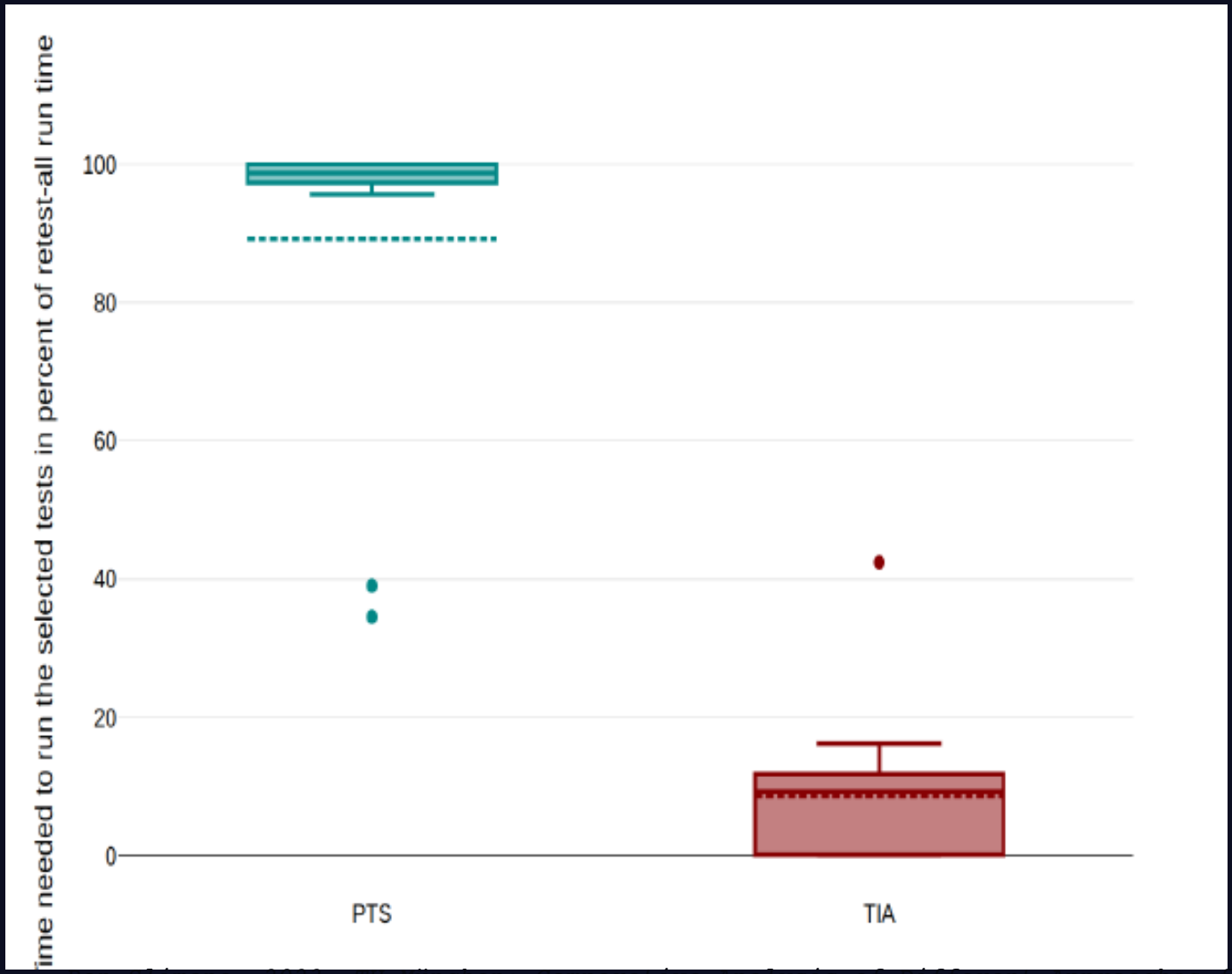


90%

? Without Test-Case-Specific Code Coverage ?



Ben Slimane, 2023, TU Munich: Comparative Analysis of Different Approaches for Test Impact Analyses for Real World Test Suites



Ben Slimane, 2023, TU München: Comparative Analysis of Different Approaches for Test Impact Analyses for Real World Test Suites

Ben Slimane, 2023, TU Munich: Comparative Analysis of Different Approaches for Test Impact Analyses for Real World Test Suites

Showing 4 changed files with 45 additions and 34 deletions

server/com.teamscale.index/src/main/java/com/teamscale/index/tracking/FindingsTracker.java

```
... ...  @@ -59,6 +59,7 @@ import com.teamscale.index.option.ProjectOptionIndex;
59  59   import com.teamscale.index.option.ProjectOptionRegistry;
60  60   import com.teamscale.index.simulink.tracing.DerivedFindingsIndexSynchronizerBase;
61  61   import com.teamscale.index.tracking.algorithm.FindingsTrackingAlgorithm;
    62 + import com.teamscale.index.tracking.algorithm.FindingsTrackingResult;
62  63   import com.teamscale.index.tracking.algorithm.TrackedElement;
63  64   import com.teamscale.index.tracking.algorithm.TrackedFindingWithContext;
64  65   import com.teamscale.index.tracking.index.FindingChurnCountIndex;
... ...  @@ -270,8 +271,10 @@ public class FindingsTracker extends AnalysisStepBase
270 271                FindingsTrackingAlgorithm trackingAlgorithm = createTrackingAlgorithm();
271 272
272 273                getProfilingMonitor().startProfiling("tracking");
273     -              PairList<TrackedFindingWithContext, TrackedFindingWithContext> trackedWithContext =
274     -                              .performTracking(baselineFindingsByBranch, changedFindings);
    274 +              FindingsTrackingResult trackingResult = trackingAlgorithm.performTracking(baseline
    275 +                              changedFindings);
    276 +              PairList<TrackedFindingWithContext, TrackedFindingWithContext> trackedWithContext =
    277 +                              .trackingResult();
275 278                getProfilingMonitor().stopProfiling("tracking");
276 279
277 280                PairList<TrackedFinding, TrackedFinding> tracked = trackedWithContext.map(TrackedF
... ...  @@ -290,7 +293,7 @@ public class FindingsTracker extends AnalysisStepBase {
290 293                              unchangedFindingsForUniformPath.getValues(), changedFindings, base
291 294
292 295                updateIdsAndPersistTrackedFindings(changedFindingsForUniformPath, unchangedFinding
293     -                              trackingAlgorithm, tracked, churnList);
    296 +                              trackingAlgorithm, tracked, churnList, trackingResult.changedEleme
294 297        }
295 298
296 299        /**
... ...  @@ -300,12 +303,11 @@ public class FindingsTracker extends AnalysisStepBase {
300 303        private void updateIdsAndPersistTrackedFindings(SetMap<UniformPath, TrackedFinding> change
301 304                        SetMap<UniformPath, TrackedFinding> unchangedFindingsForUniformPath,
302 305                        FindingsTrackingAlgorithm trackingAlgorithm, PairList<TrackedFinding, Trac
303     -                      FindingChurnList churnList) throws StorageException {
    306 +                      FindingChurnList churnList, Map<String, TrackedElement> changedElements) t
304 307
305 308                getProfilingMonitor().startProfiling("determine-ids");
306 309                Map<String, String> idMap = idManager.determineIds(churnList.getAddedFindings(),
307     -                              churnList.getFindingsAddedInBranch(), trackingAlgorithm.getChanged
308     -                              trackedFindingsByIdIndex);
```

```
47  import com.teamscale.index.repository.history.ElementHistoryIndex;
48  import com.teamscale.index.resource.TokenElementInfo;
49  import com.teamscale.index.simulink.tracing.DerivedFindingsIndexSynchronizerBase;
50  import com.teamscale.index.tracking.index.FindingChurnCountIndex;
51  import com.teamscale.index.tracking.index.FindingChurnListIndex;
52  import com.teamscale.index.tracking.index.FindingIdentificationIndex;
53  import com.teamscale.index.tracking.index.TrackedFindingsByIdIndex;
54  import com.teamscale.index.tracking.index.TrackedFindingsIndex;
55
56  import eu.cqse.check.framework.scanner.ELanguage;
57
58  /**
59   * Tests the {@link FindingsTracker}.
60   */
61  public class FindingsTrackerTest extends IndexTestCaseBase {
62
63          @Override
64          protected Collection<Class<? extends IProjectIndex>> getProjectIndexes() {
65                  return Arrays.asList(DerivedFindingsIndex.class, TrackedFindingsIndex.class, TrackedFindingsByIdIndex.class,
66                                  FindingChurnListIndex.class, FindingChurnCountIndex.class, CodeChangeIndex.class,
67                                  FindingBlacklistIndex.class, BranchAgnosticFindingBlacklistIndex.class,
68                                  FindingIdentificationIndex.class, ProjectOptionIndex.class);
69          }
70
71          @BeforeEach
72          void beforeEach() {
73                  StorageStringAbbreviator.clearCachesForTesting();
74          }
75
76          @Test
77          void testReverseSiblingLinking() throws Exception {
78                  String location = "foo/bar.cpp";
79                  String derivedLocation = "baz/model.mdl";
80                  storeTokenElement(location, ELanguage.CPP, "");
81                  storeTokenElement(derivedLocation, ELanguage.SIMULINK, "");
82
83                  IndexFinding finding = new IndexFinding("group", "category", "message", new ElementLocation(location));
84                  ArrayList<IndexFinding> findings = new ArrayList<>(Collections.singletonList(finding));
85                  openProjectIndex(FindingsIndex.class, TestBranchUtils.createReadHeadWriteTimestampAccessWithTestBranch(2))
86                                  .setFindings("partition", location, findings, null);
87
88                  QualifiedNameLocation modelLocation = new QualifiedNameLocation("qual/name", derivedLocation);
89                  IndexFinding derivedFinding = DerivedFindingsIndexSynchronizerBase.createDerivedFinding(finding,
90                                  Collections.singletonList(modelLocation), "partition");
91                  ArrayList<IndexFinding> derivedFindings = new ArrayList<>(Collections.singletonList(derivedFinding));
92                  openProjectIndex(DerivedFindingsIndex.class,
93                                  TestBranchUtils.createReadHeadWriteTimestampAccessWithTestBranch(2))
94                                  .setFindings("derived", derivedLocation, derivedFindings, null);
95
```

SUMMARY

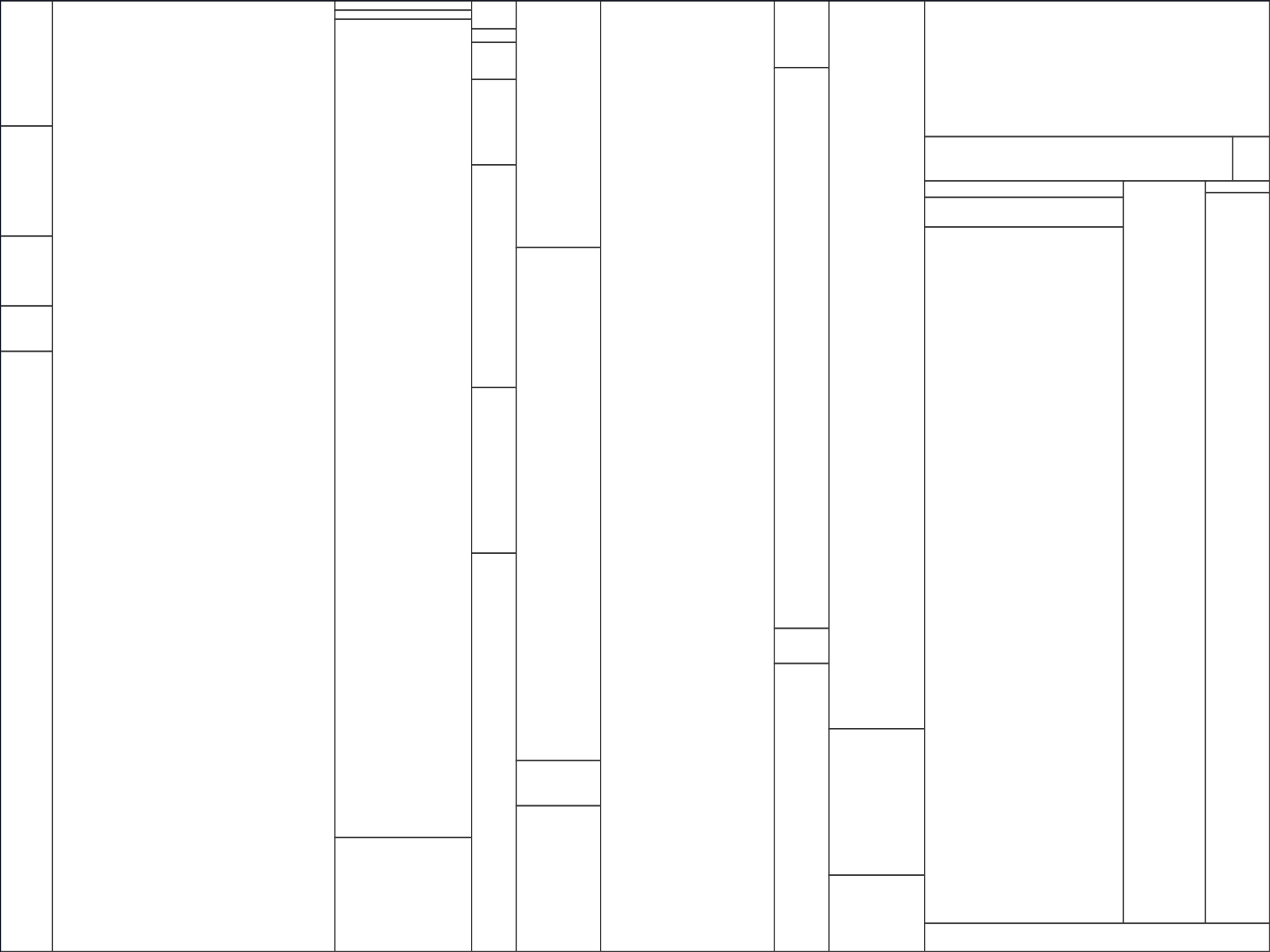
For Quality Gates *For Continuous Integration*

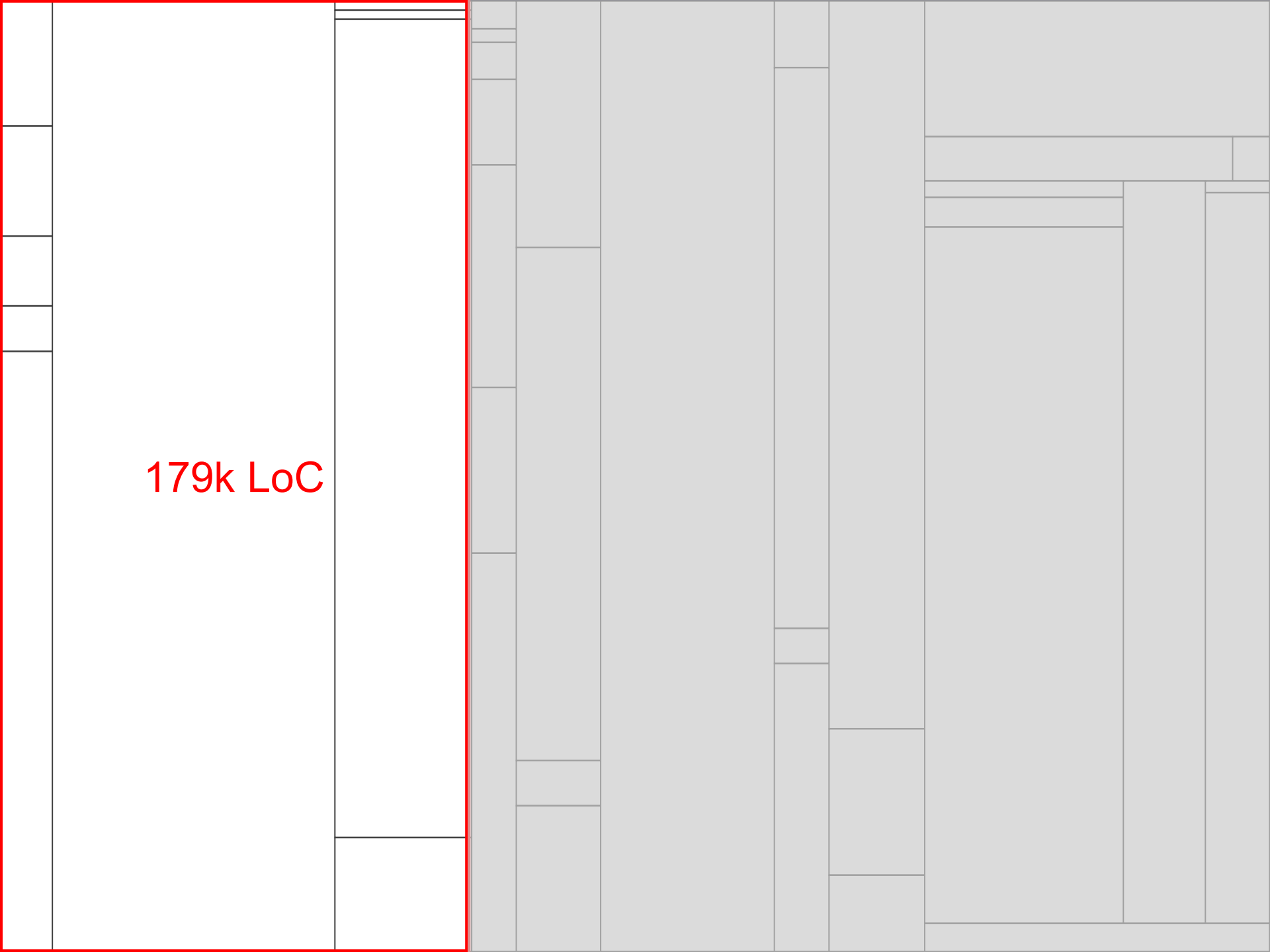
Pareto Optimization *Test Impact Analysis* *Test Coverage*

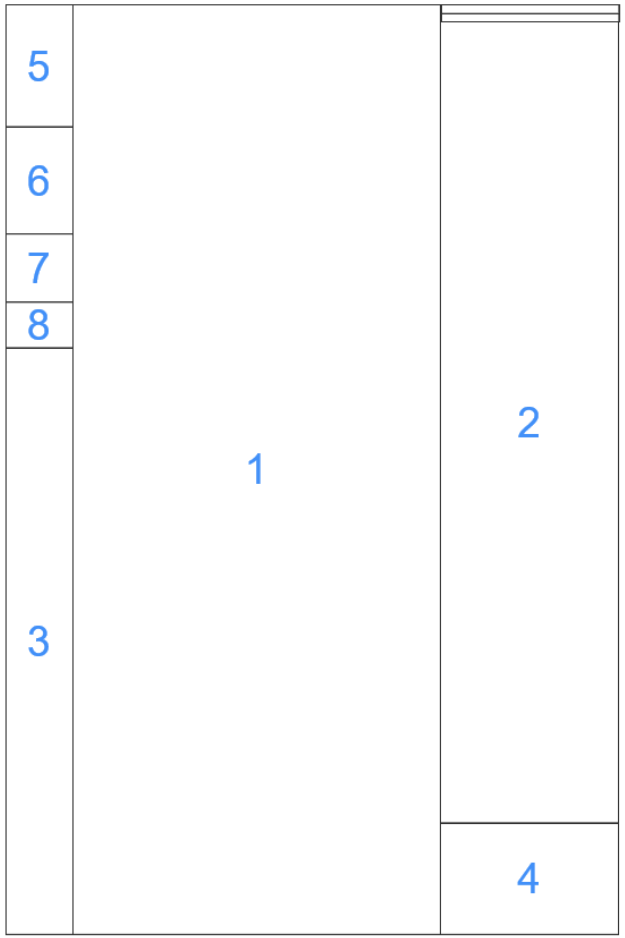
Precision & Effort

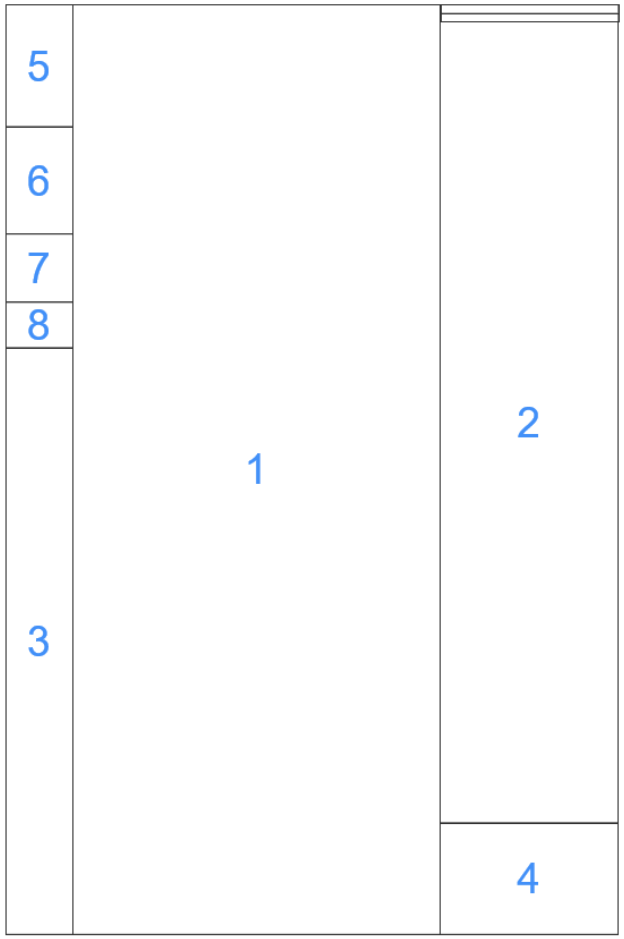
Distance Metrics *Information Retrieval* *Test Content*

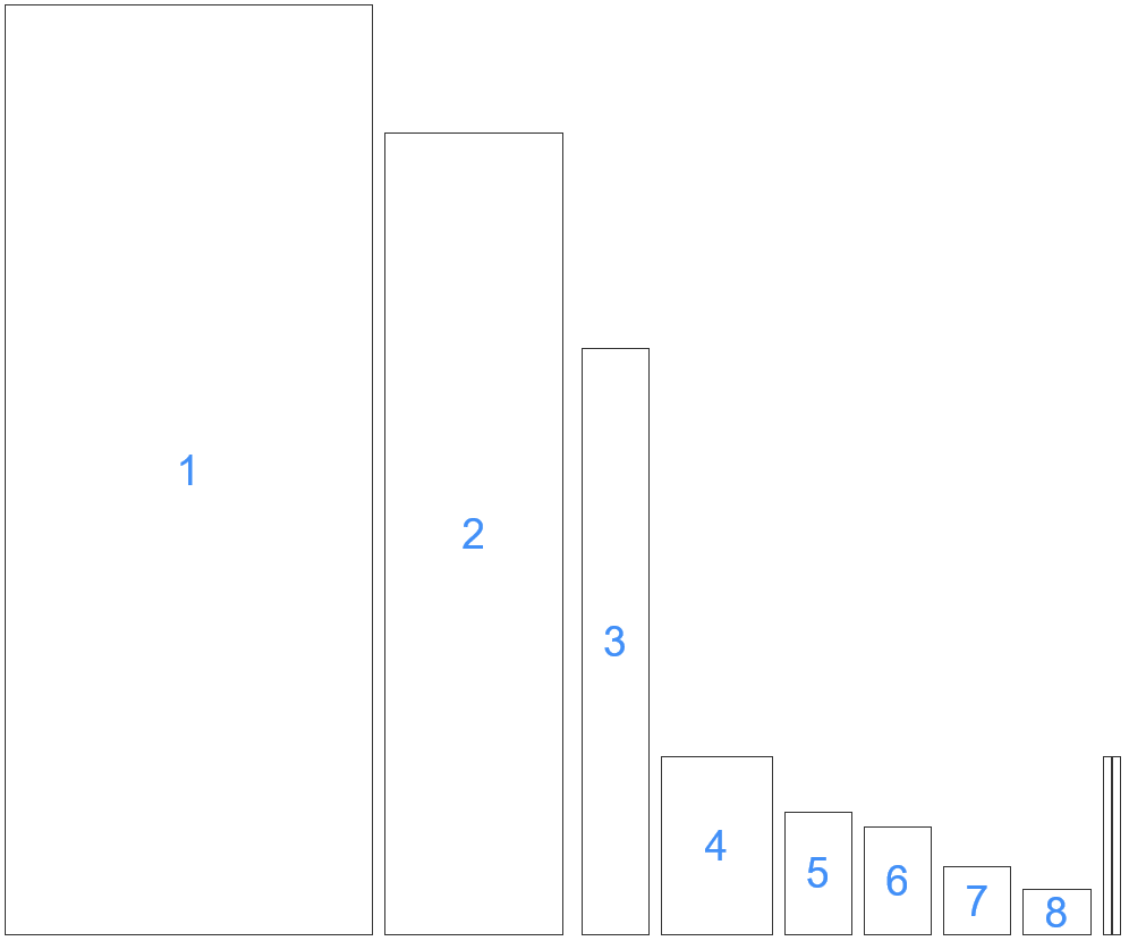
What about
Defect Prediction?

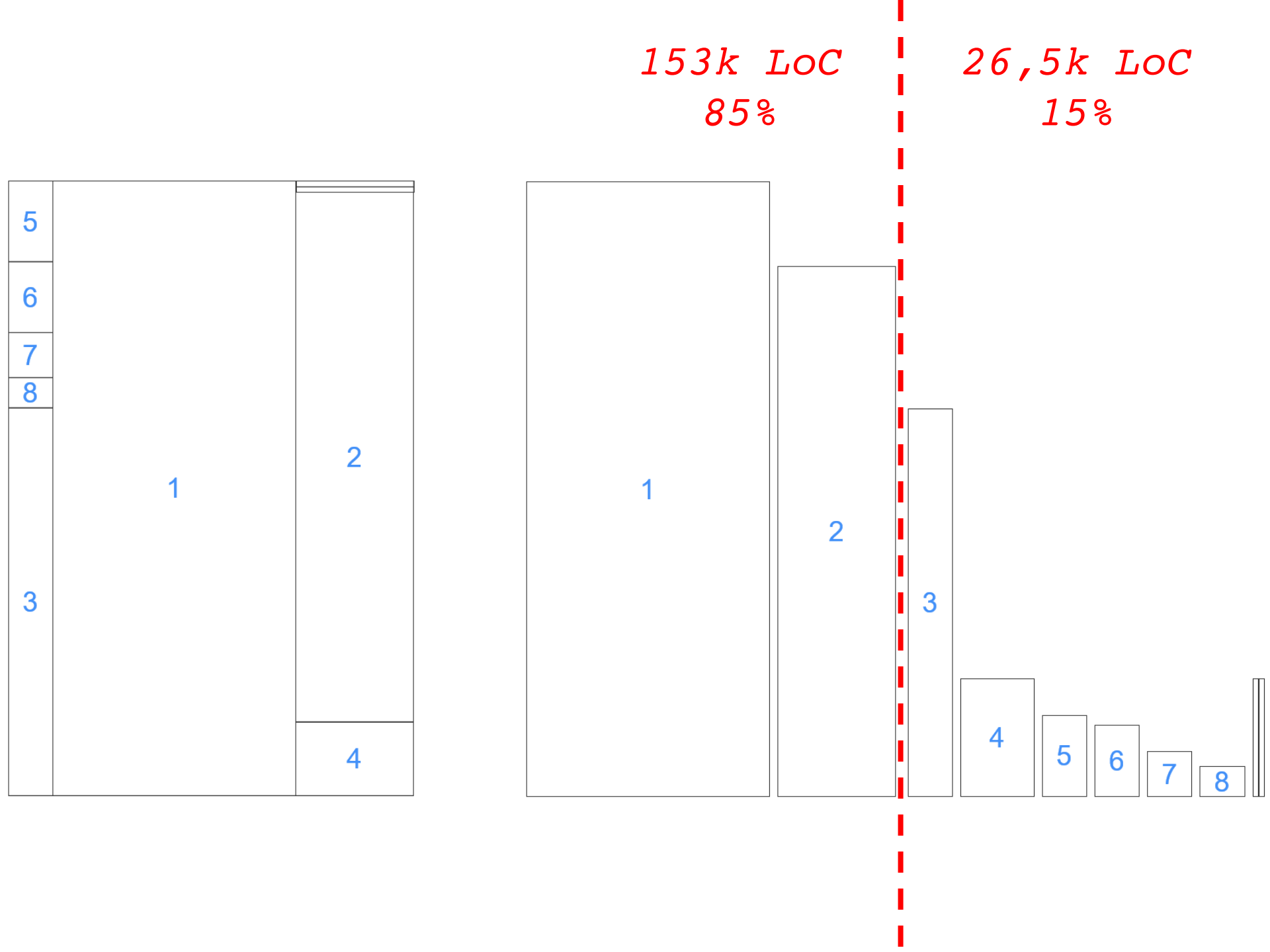


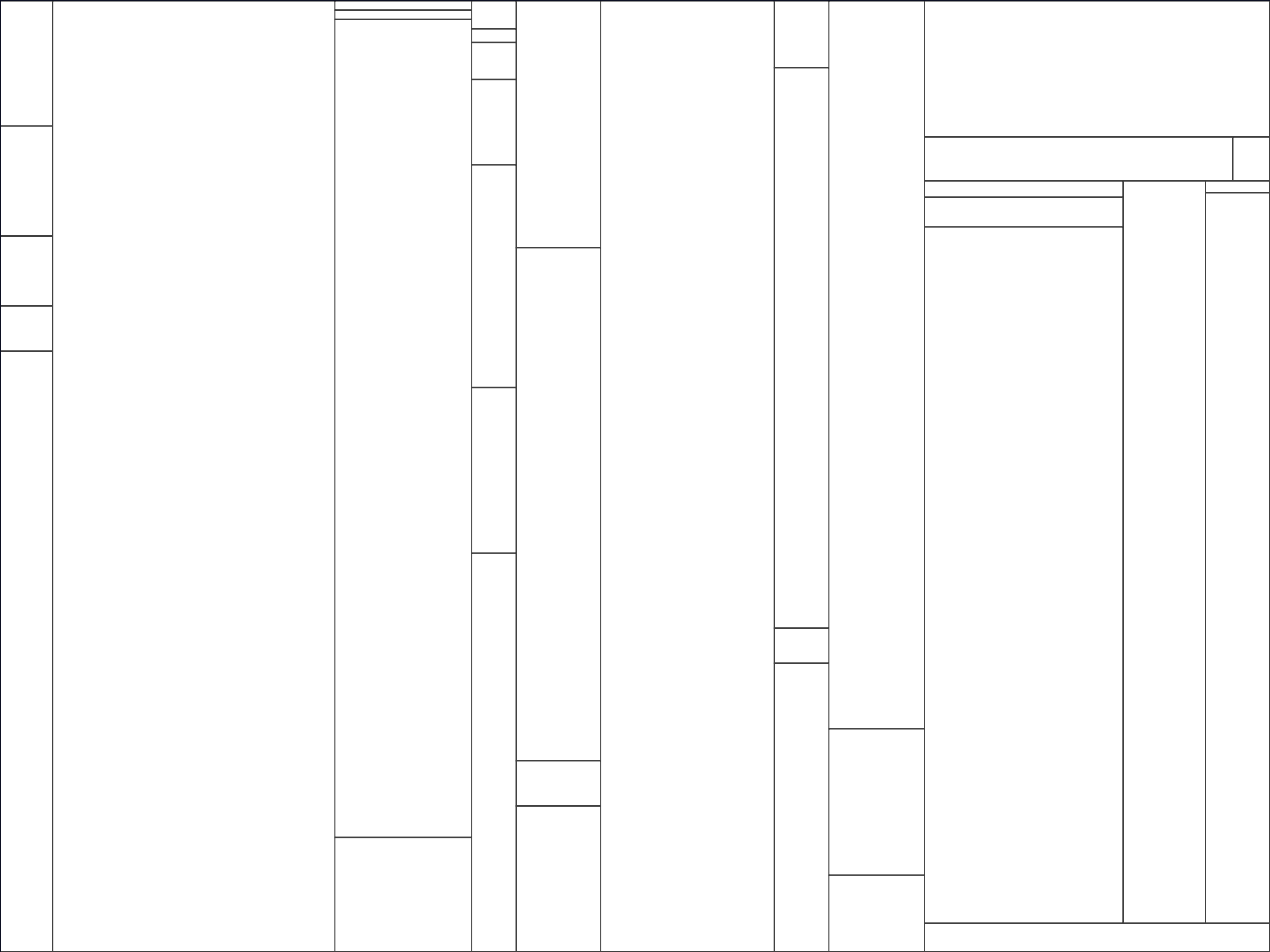

179k LoC

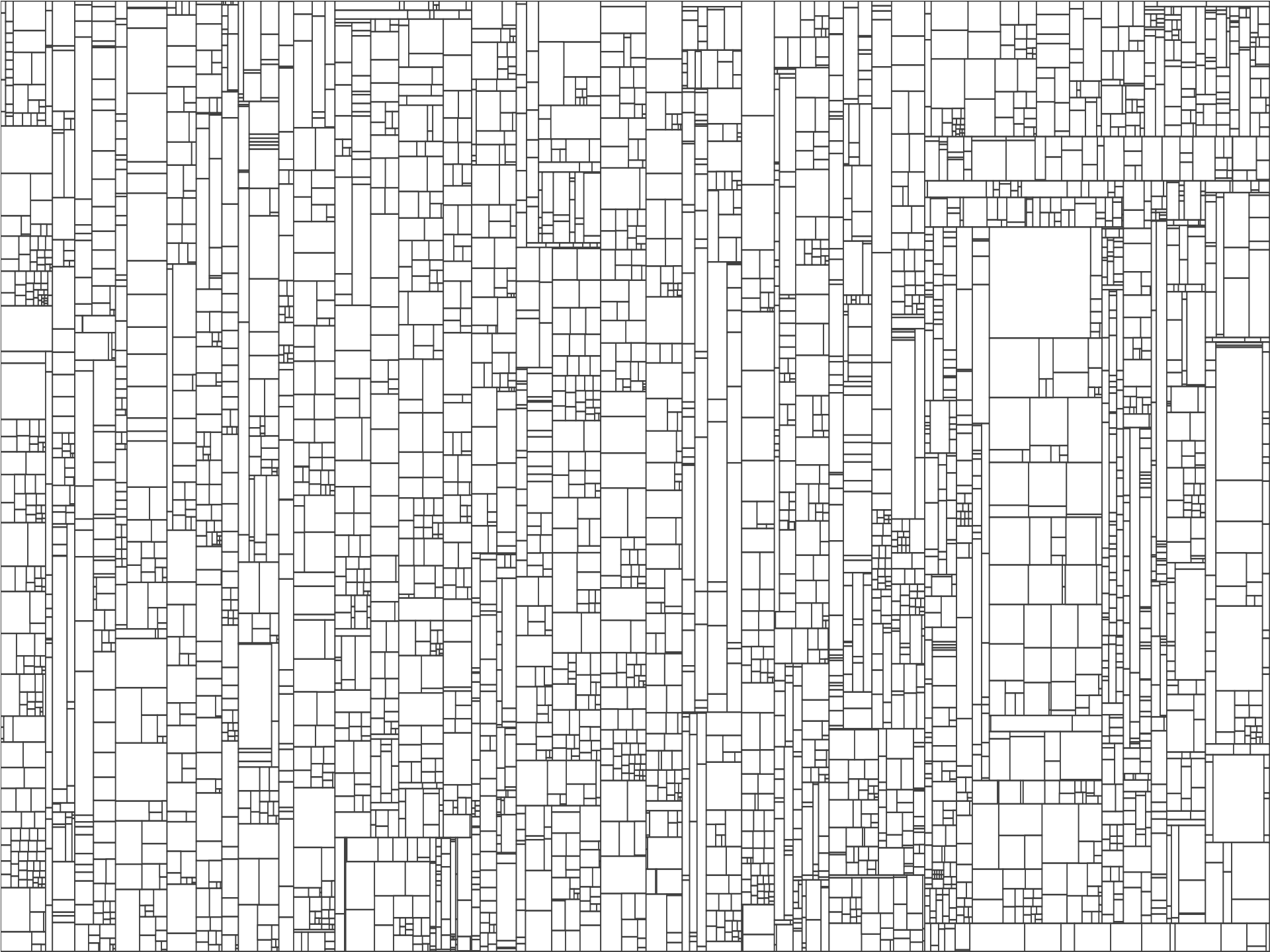

5
6
7
8
1
2
3
4

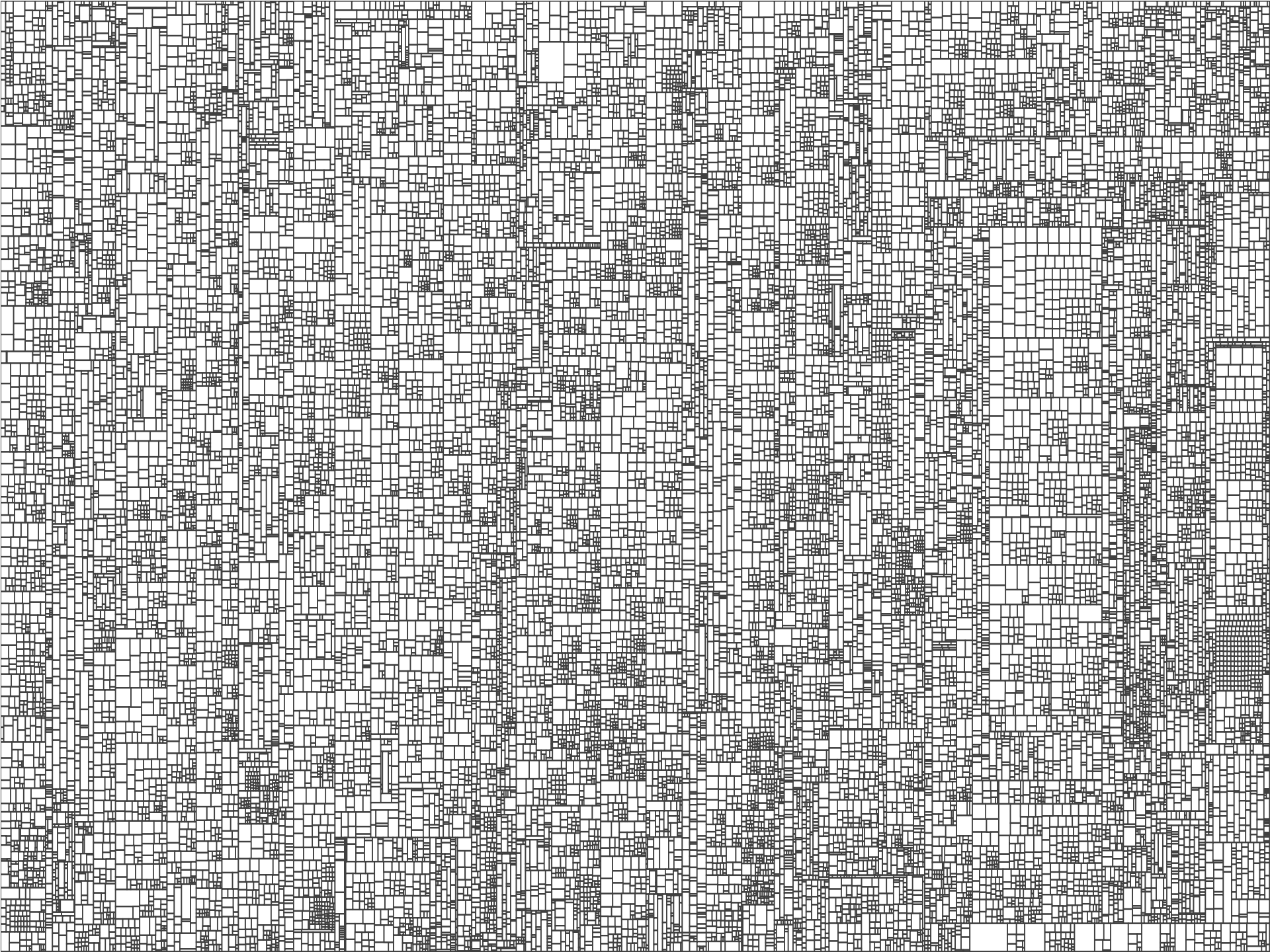

5
6
7
8
1
2
3
4

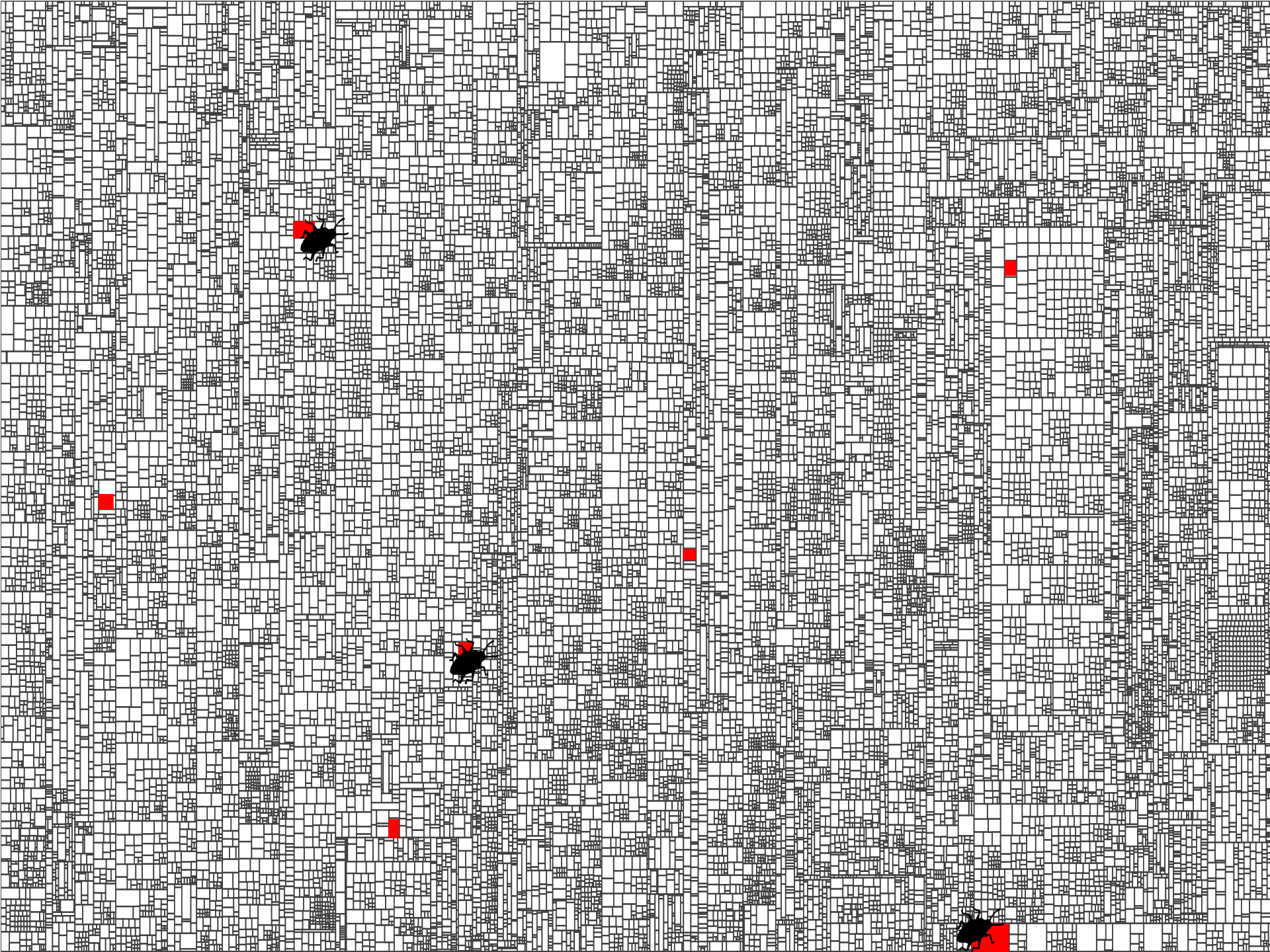

1
2
3
4
5
6
7
8

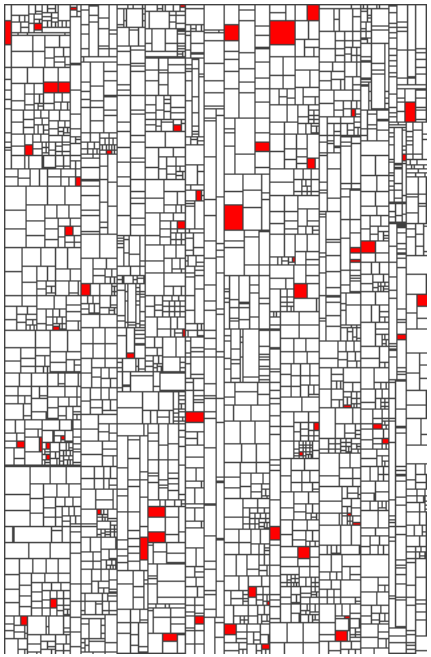

153k LoC
85%
26,5k LoC
15%
5
6
7
8
1
2
3
4
1
2
3
4
5
6
7
8





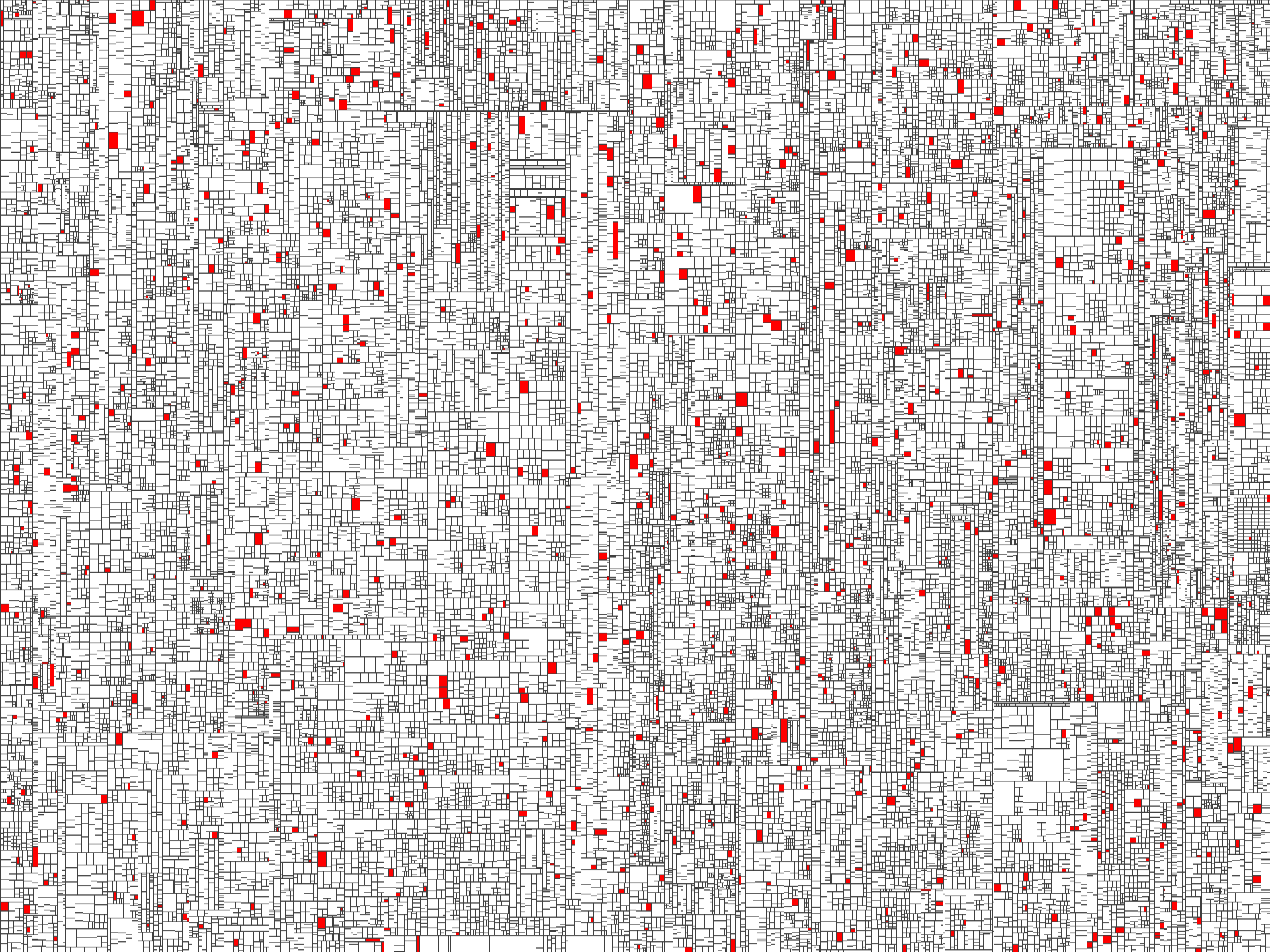


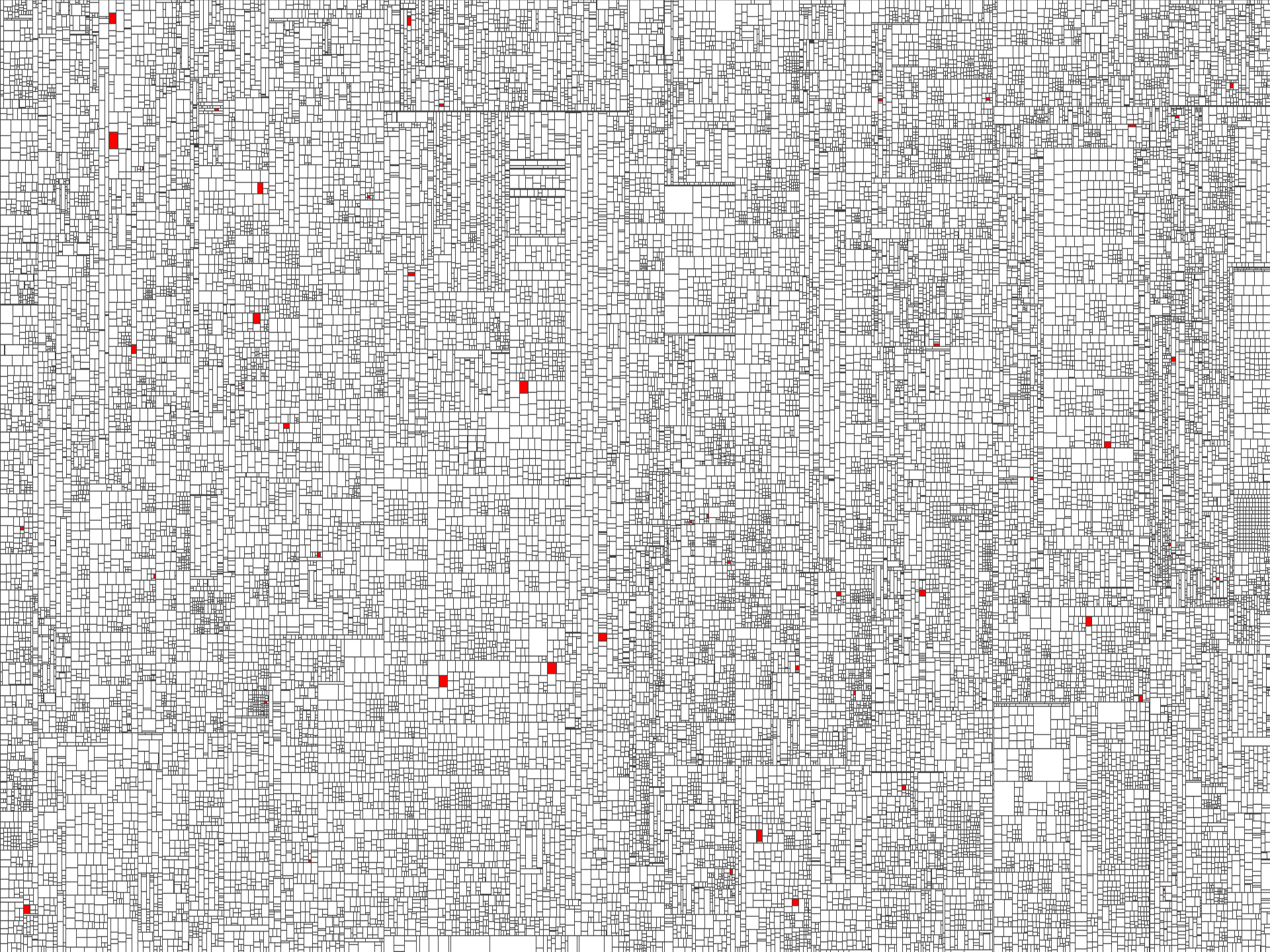




Release

Time of Study





EVALUATION

Release	# „defect prone“	Methods	# Bugs (Top 50)
1.4:	1127	0	
2.0:	1176	0	

Pascarella, Palomba, Bacchelli, Re-evaluating Method-Level Bug Prediction, 2018: Prediction not better than random classification.

Chowdhury, Uddin, Hemmati, Holmes, Method-Level- Bug Prediction: Problems and Promise, 2024:
Method-Level Bug Prediction performance „extremely poor“.

SUMMARY

Don’t always run all tests (if it takes too long).

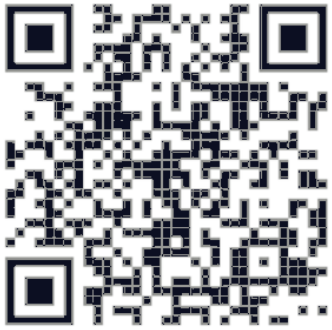
There are many test selection approaches that are fit for use in practice!

We are happy to discuss what works best in your context ☺

Test Gap Analysis

Reveal Untested Changes in Source-Code

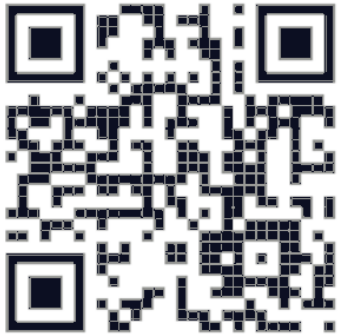
Watch recordings
tmscl.me/tga-ro25



Fast Feedback from Long-Running Tests

Test Selection for Ever-Growing Test Suites

Watch recordings
tmscl.me/ts-ro25



CONTACT

LOOKING FORWARD TO DISCUSSIONS ☺

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