

Ein Benchmark zum Formulieren der Erwartungshaltung an Codequalität

Und bei dir so?

CQSE

Dr. Nils Göde



Software-**Audits**



Kontinuierliche **Qualitäts-**
und **Testkontrolle**



 **Teamscale**

CQSE GmbH



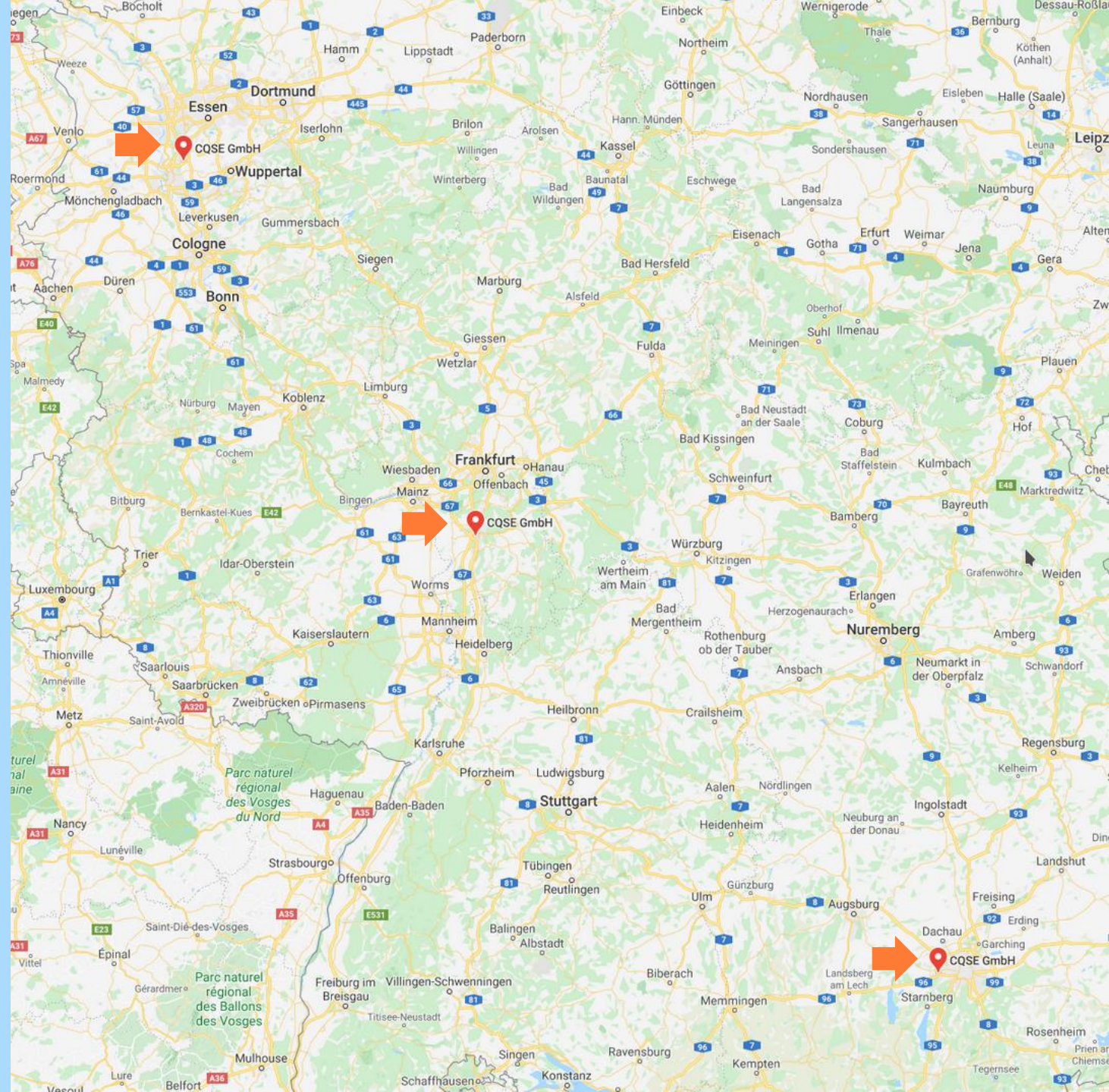
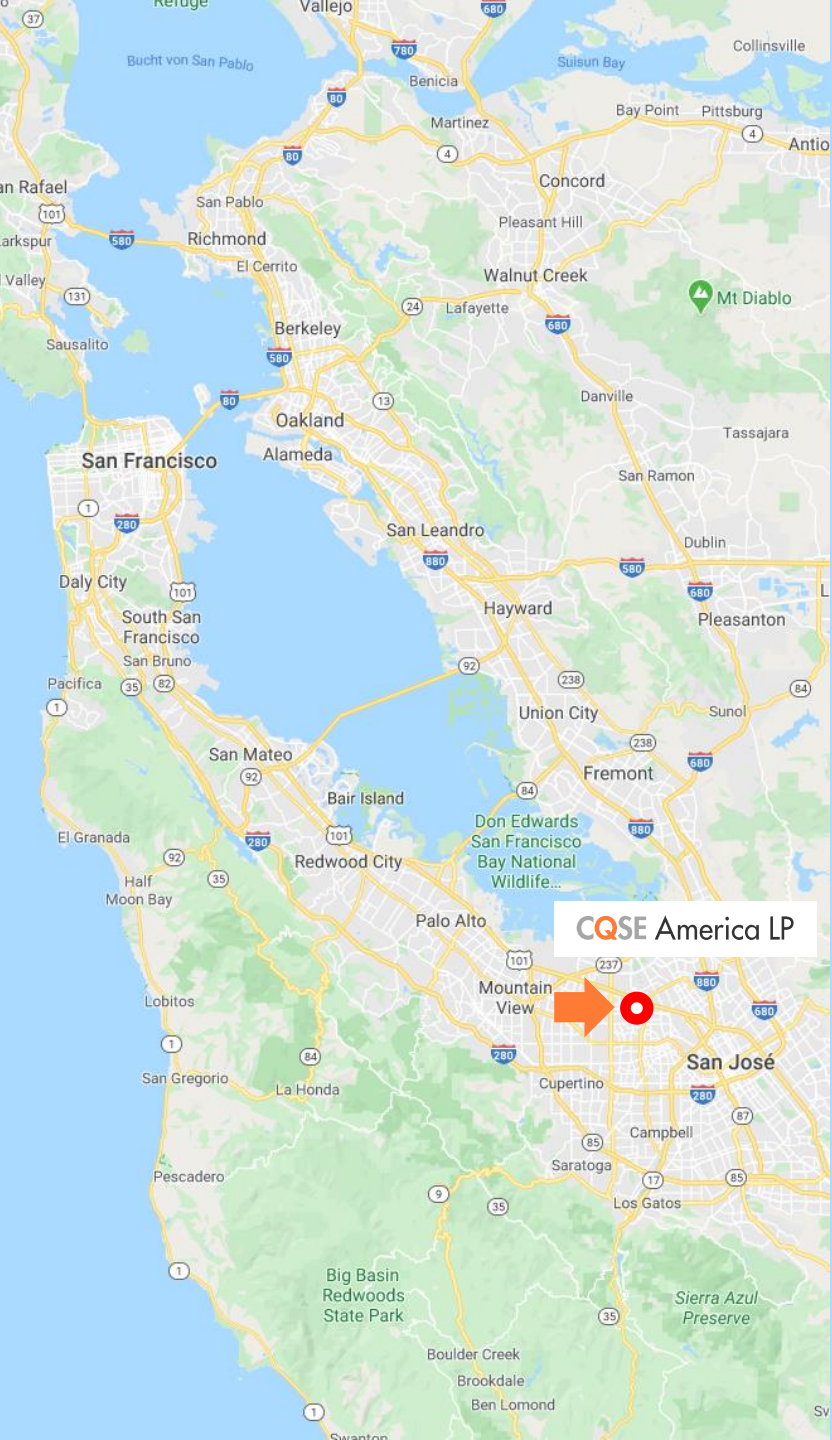
17+ **Promotionen** in
Software Engineering



Eigene **Forschung**



Enger Kontakt zu
Universitäten





```
28
29 import org.apache.commons.cli.ParseException;
30 import org.slf4j.Logger;
31 import org.slf4j.LoggerFactory;
32
33 /**
34  * JabRef's main class to process command line options and to start the UI
35  */
36 public class JabRefMain extends Application {
37
38     private static final Logger LOGGER = LoggerFactory.getLogger(JabRefMain.class);
39
40     private static String[] arguments;
41
42     public static void main(String[] args) {
43         arguments = args;
44         launch(arguments);
45     }
46
47     @Override
48     public void start(Stage mainStage) {
49         try {
50             // Fail on unsupported Java versions
51             ensureCorrectJavaVersion();
52             FallbackExceptionHandler.installExceptionHandler();
53
54             // Init preferences
55             final JabRefPreferences preferences = JabRefPreferences.getInstance();
56             Globals.prefs = preferences;
57             // Perform migrations
58             PreferencesMigrations.runMigrations();
59
60             configureProxy(preferences.getProxyPreferences());
61
62             Globals.startBackgroundTasks();
63
64             applyPreferences(preferences);
65
```



218.701 LOC

Qualität ?

Technische Schulden ?

Wartbarkeit ?

Zukunftssicherheit ?

Risiken ?

Handlungsbedarf ?



Statische Analyse

oder beliebiges anderes Werkzeug





- Dashboard
- Activity
- Findings
- Metrics
- Files**
- Components
- Issues

JobRef ▾

Current Metrics

Files

Path	Files	Lines of Code	Source Lines of Code	Method Length Assessment	Nesting Depth Assessment	Clone Coverage	Comment Completeness Assessment	Findings Density in Code Anomalies	Findings Density in Naming
src	1,821	218,701	140,878	<div style="width: 100%; height: 10px; background-color: green;"></div>	<div style="width: 100%; height: 10px; background-color: green;"></div>	9.6%	<div style="width: 100%; height: 10px; background-color: green;"></div>	5.6	3.1

	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef			9,6 %	22,8 %	5,6	3,1

	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef			9,6%	22,8%	5,6	3,1

```

169
170
171     1 2 3
172     4 for (int k = 0; k < tokens.length; k++) {
173         String token = tokens[k];
174         if (abbreviateThatIsSingleLetter) {
175             String[] dashes = token.split("-");
176
177             token = Arrays.asList(dashes).stream().map(BibtexNameFormatter::getFirstCharOfString)
178                 .collect(Collectors.joining("-"));
179         }
180
181         // Output token
182         sb.append(token);
183
184     5 if (k < (tokens.length - 1)) {
185         // Output Intertoken String
186     6 if (interToken == null) {
187         7 if (abbreviateThatIsSingleLetter) {
188         8 sb.append('.');
189         }
190         // No clue what this means (What the hell are tokens anyway???)
191         // if (lex_class[name_sep_char[cur_token]] = sep_char) then
192         //     append ex_buf_char and check (name_sep_char[cur_token])

```



Methodenlänge

Schachtelung

Klonüberdeckung

Kommentierung

Codeanomalien
pro 1.000 SLOCNaming-Verstöße
pro 1.000 SLOC

JabRef



9,6 %

22,8 %

5,6

3,1

```
125 }
126
127 private static void addResourceFile(String name, String resource, ZipOutput
128     ZipEntry zipEntry = new ZipEntry(name);
129     out.putNextEntry(zipEntry);
130     OpenDocumentSpreadsheetCreator.addFromResource(resource, out);
131     out.closeEntry();
132 }
133
134 private static void addFromResource(String resource, OutputStream out) {
135     URL url = OpenDocumentSpreadsheetCreator.class.getResource(resource);
136     try (InputStream in = url.openStream()) {
137         byte[] buffer = new byte[256];
138         synchronized (out) {
139             while (true) {
140                 int bytesRead = in.read(buffer);
141                 if (bytesRead == -1) {
142                     break;
143                 }
144                 out.write(buffer, 0, bytesRead);
145             }
146         }
147     } catch (IOException e) {
148         LOGGER.warn("Cannot get resource", e);
149     }
150 }
151 }
152
```

```
105 }
106
107 private static void addResourceFile(String name, String resource, ZipOutput
108     ZipEntry zipEntry = new ZipEntry(name);
109     out.putNextEntry(zipEntry);
110     OpenOfficeDocumentCreator.addFromResource(resource, out);
111     out.closeEntry();
112 }
113
114 private static void addFromResource(String resource, OutputStream out) {
115     URL url = OpenOfficeDocumentCreator.class.getResource(resource);
116     try (InputStream in = url.openStream()) {
117         byte[] buffer = new byte[256];
118         synchronized (out) {
119             while (true) {
120                 int bytesRead = in.read(buffer);
121                 if (bytesRead == -1) {
122                     break;
123                 }
124                 out.write(buffer, 0, bytesRead);
125             }
126         }
127     } catch (IOException e) {
128         LOGGER.warn("Cannot get resource", e);
129     }
130 }
131 }
132
```

This image displays a grid of 100 vertical panels, each containing a unique visualization or data representation. The panels are arranged in a 10x10 grid. The visualizations vary significantly in style and content, including:

- Bar charts and histograms with various color schemes (e.g., blue, green, red, purple, brown).
- Line graphs and scatter plots showing trends and data points.
- Tables and lists of text, some with highlighted rows or columns.
- Complex diagrams and flowcharts.
- Abstract patterns and textures.
- Small maps or geographical representations.
- Textual data, possibly code or lists.

The overall layout is highly structured and repetitive, suggesting a systematic exploration of different data visualization techniques or a comprehensive set of data analysis results. Each panel is separated by thin white lines, and the background of each panel is a solid color, often matching the primary colors used in the visualizations.

	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef			9,6%	22,8%	5,6	3,1

```

9  import org.jabref.logic.tion.Localization;
10 import org.jabref.model.search.rules.SentenceAnalyzer;
11
12 public class ContainsAndRegexBasedSearchRuleDescriber implements SearchDescriber {
13
14     private final boolean regExp;
15     private final boolean caseSensitive;
16     private final String query;
17
18     public ContainsAndRegexBasedSearchRuleDescriber(boolean caseSensitive, boolean regExp, String query) {
19         this.caseSensitive = caseSensitive;
20         this.regExp = regExp;
21         this.query = query;
22     }
23
24     @Override
25     public TextFlow getDescription() {
26         List<String> words = new SentenceAnalyzer(query).getWords();
27         String firstWord = words.isEmpty() ? "" : words.get(0);
28
29         String temp = regExp ? Localization.lang(

```

Methodenlänge

Schachtelung

Klonüberdeckung

Kommentierung

Codeanomalien
pro 1.000 SLOC

Naming-Verstöße
pro 1.000 SLOC

JabRef



9,6%

22,8%

5,6

3,1

```
437
438 List<String> names;
439 if (style.isSortByPosition()) {
440     // We need to sort the reference marks according to their
441     names = sortedReferenceMarks;
442 } else if (style.isNumberEntries()) {
443     // We need to sort the reference marks according to the s
444     // entries:
445     SortedMap<BibEntry, BibDatabase> newMap = new TreeMap<>(e
446     for (Map.Entry<BibEntry, BibDatabase> bibtexEntryBibtexDat
447         newMap.put(bibtexEntryBibtexDatabaseEntry.getKey(), b
448     }
449     entries = newMap;
450     // Rebuild the list of cited keys according to the sort o
451     cited.clear();
452     for (BibEntry entry : entries.keySet()) {
453         cited.add(entry.getCiteKeyOptional().orElse(null));
454     }
455     names = Arrays.asList(xReferenceMarks.getElementNames());
456 } else {
457     names = sortedReferenceMarks;
458 }
459
460 // Remove all reference marks that don't look like JabRef cit
```





```
323
324 * Sets the title of the main window.
325 */
326 public void setWindowTitle() {
327     BasePanel panel = getCurrentBasePanel();
328
329     // no database open
330     if (panel == null) {
331         // setTitle(FRAME_TITLE);
332         return;
333     }
334
335     String mode = panel.getBibDatabaseContext().getMode().getFormattedName();
336     String modeInfo = String.format(" (%s)", Localization.lang("%0 mode", mode));
337     boolean isAutosaveEnabled = Globals.prefs.getBoolean(JabRefPreferences.LOCAL_
338
339     if (panel.getBibDatabaseContext().getLocation() == DatabaseLocation.LOCAL) {
340         String changeFlag = panel.isModified() && !isAutosaveEnabled ? "*" : "";
341         String databaseFile = panel.getBibDatabaseContext()
342             .getDatabasePath()
343             .map(Path::toString)
344             .orElse(Localization.lang("untitled"));
345         // setTitle(FRAME_TITLE + " - " + databaseFile + changeFlag + modeInfo);
346     } else if (panel.getBibDatabaseContext().getLocation() == DatabaseLocation.SHA
347         // setTitle(FRAME_TITLE + " - " + panel.getBibDatabaseContext().getDBMSFu
```

	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef			9,6%	22,8%	5,6	3,1

```

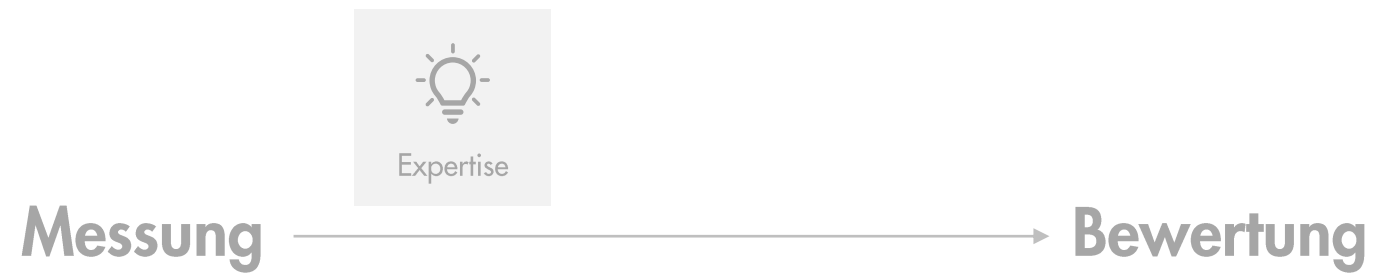
9 public class UndoableModifySubtree extends AbstractUndoableJabRefEdit {
10
11     /**
12      * A backup of the groups before the modification
13      */
14     private final GroupTreeNode m_groupRoot;
15
16     private final GroupTreeNode m_subtreeBackup;
17
18     /**
19      * The path to the global groups root node
20      */
21     private final List<Integer> m_subtreeRootPath;
22
23     /**
24      * This holds the new subtree (the root's modified children) to allow redo.
25      */
26     private final List<GroupTreeNode> m_modifiedSubtree = new ArrayList<>();

```


	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef			9,6 %	22,8 %	5,6	3,1

Handlungsbedarf ?

Messung ≠ Bewertung !



The first rule of functions is that they should be small. The second rule of functions is that *they should be smaller than that*. [...] **Functions should hardly ever be 20 lines long.**

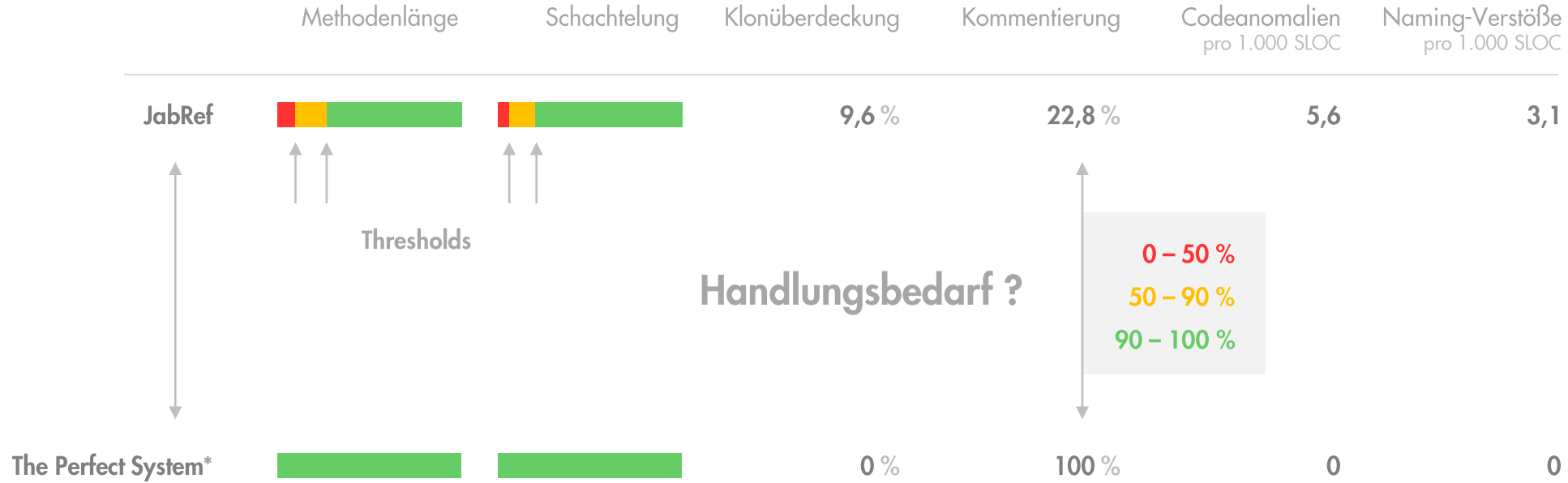
– Robert C. Martin, 2008

Number one in the stink parade is duplicated code. If you see the same code structure in more than one place, you can be sure that your program will be better if you find a way to unify them.

– Martin Fowler, 1999

Naming conventions make programs more understandable by making them easier to read. They can also give information about the function of the identifier [...]

– Java Code Conventions, 1997



* Existiert nicht

» Warum sind Methoden ab 30 Zeilen zu lang und nicht erst ab 50? «



Expertise



Thresholds

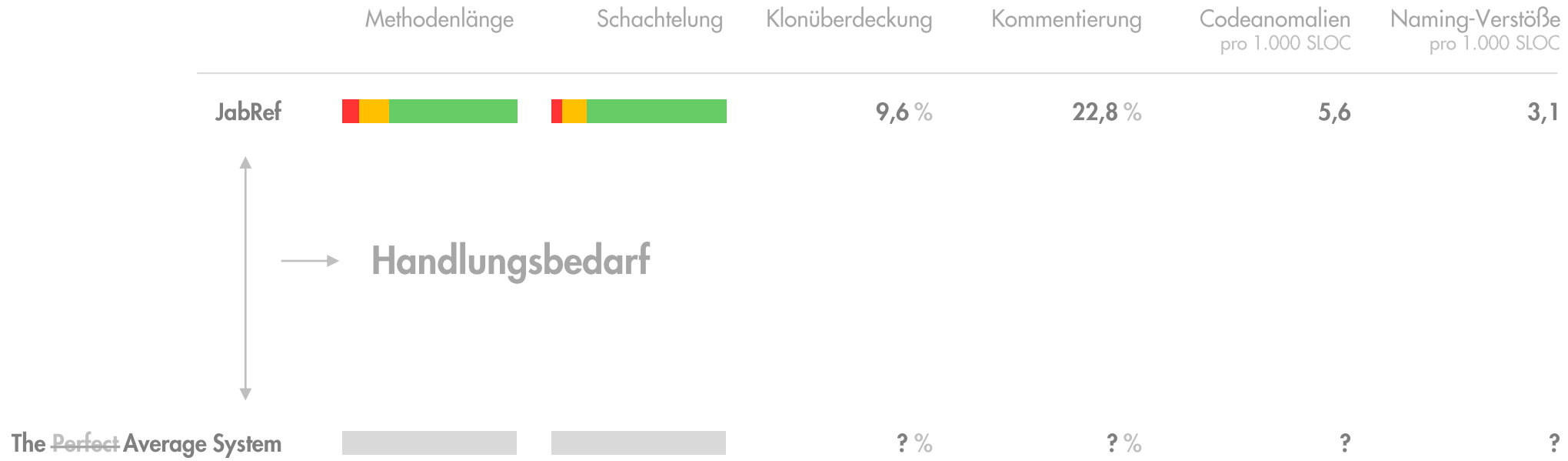


Benchmark

Messung



Bewertung

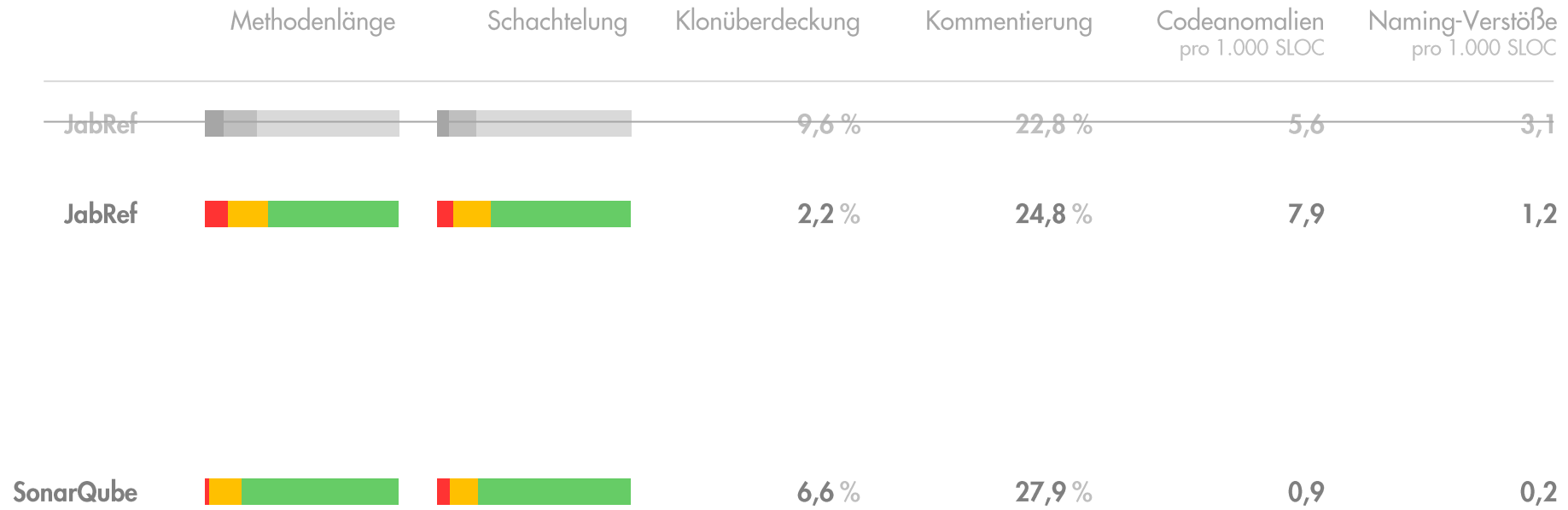


» Ja, aber ... «

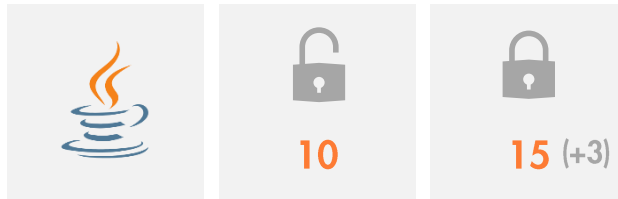
	Methodenlänge	Schachtelung	Klonüberdeckung	Kommentierung	Codeanomalien pro 1.000 SLOC	Naming-Verstöße pro 1.000 SLOC
JabRef		 ★	9,6 %	22,8 %	5,6	3,1
SonarQube	★ 		★ 6,6 %	★ 27,9 %	★ 0,9	★ 0,2

Klonüberdeckung pro Datei in JabRef

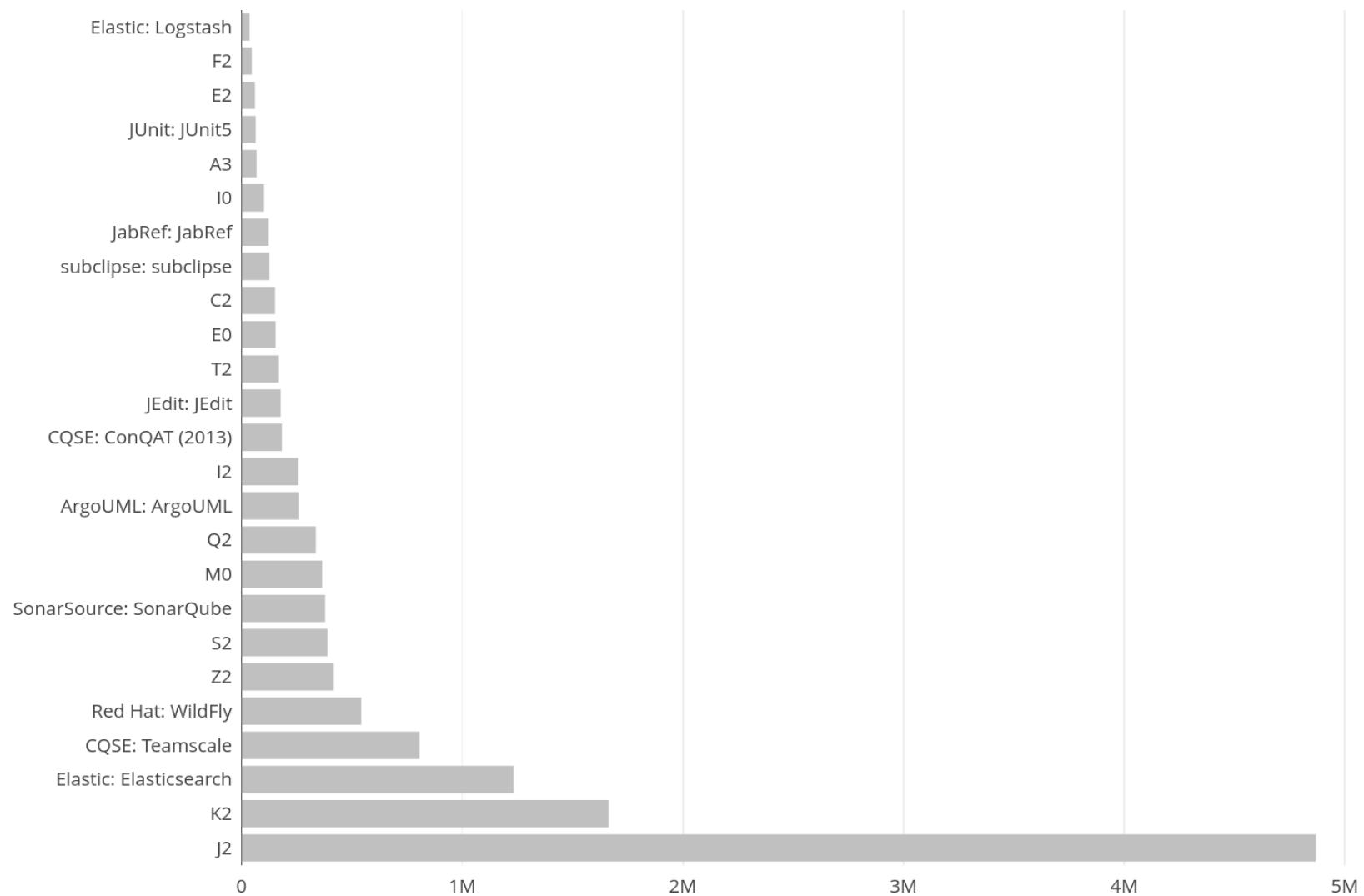




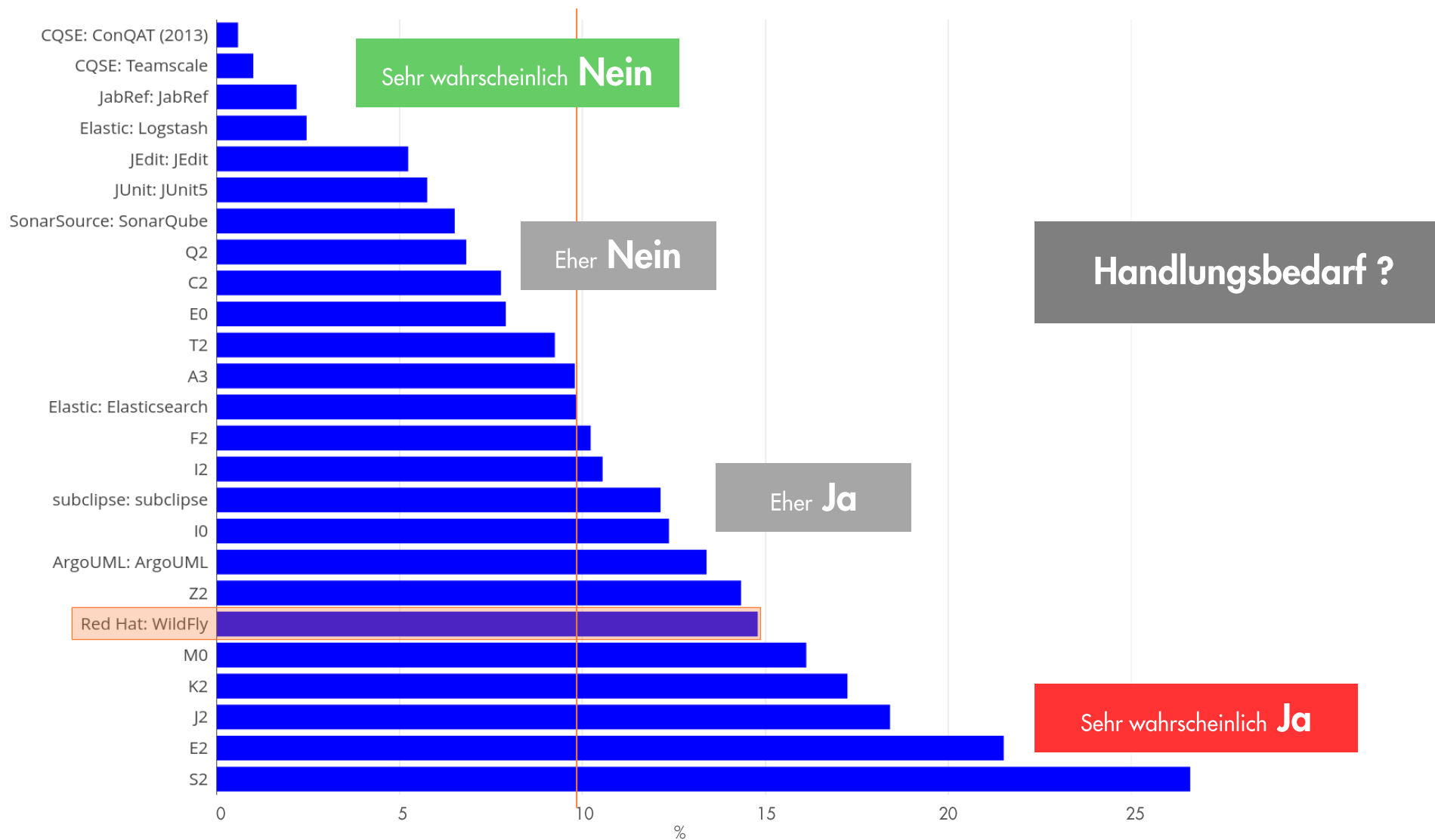
CQSE-Benchmark



Lines of Code



Klonüberdeckung



A treemap visualization showing the distribution of code files. The background is a dense grid of rectangles of various sizes and shades of blue. Several regions are highlighted with orange borders. Labels are placed over some of these highlighted regions.

src/clustering/ee/cache/.../scheduler/FastConcurrentDirectDeque.java

src/clustering/ee/cache/.../scheduler/PortableConcurrentDirectDeque.java

src/security/subsystem/.../lru/FastConcurrentDirectDeque.java

src/security/subsystem/.../lru/PortableConcurrentDirectDeque.java

src/jpa/hibernate4_1

src/jpa/hibernate5_3

src/jpa/hibernate4_3

src/jpa/hibernate5

src/jpa/hibernate4_1

```

73
74
75     private org.hibernate.stat.Statistics getBaseStatistics(EntityManagerFactory entityMa
76         if (entityManagerFactory == null) {
77             return null;
78         }
79         HibernateEntityManagerFactory entityManagerFactoryImpl = (HibernateEntityManagerF
80         SessionFactory sessionFactory = entityManagerFactoryImpl.getSessionFactory();
81         if (sessionFactory != null) {
82             return sessionFactory.getStatistics();
83         }
84         return null;
85     }
86
87     private org.hibernate.stat.EntityStatistics getStatistics(EntityManagerFactory entity
88     HibernateEntityManagerFactory entityManagerFactoryImpl = (HibernateEntityManagerF
89     SessionFactory sessionFactory = entityManagerFactoryImpl.getSessionFactory();
90     if (sessionFactory != null) {
91         return sessionFactory.getStatistics().getEntityStatistics(entityName);
92     }
93     return null;
94 }
95
96 private Operation entityDeleteCount = new Operation() {

```

src/jpa/hibernate4_3

```

73
74
75     private org.hibernate.stat.Statistics getBaseStatistics(EntityManagerFactory entityMa
76         if (entityManagerFactory == null) {
77             return null;
78         }
79         HibernateEntityManagerFactory entityManagerFactoryImpl = (HibernateEntityManagerF
80         SessionFactory sessionFactory = entityManagerFactoryImpl.getSessionFactory();
81         if (sessionFactory != null) {
82             return sessionFactory.getStatistics();
83         }
84         return null;
85     }
86
87     private org.hibernate.stat.EntityStatistics getStatistics(EntityManagerFactory entity
88     if (entityManagerFactory == null) {
89         return null;
90     }
91     HibernateEntityManagerFactory entityManagerFactoryImpl = (HibernateEntityManagerF
92     SessionFactory sessionFactory = entityManagerFactoryImpl.getSessionFactory();
93     if (sessionFactory != null) {
94         return sessionFactory.getStatistics().getEntityStatistics(entityName);
95     }
96     return null;
97 }
98
99 private Operation entityDeleteCount = new Operation() {

```

src/jpa/hibernate5

```

67
68
69     private org.hibernate.stat.Statistics getBaseStatistics(EntityManagerFactory entityMa
70         if (entityManagerFactory == null) {
71             return null;
72         }
73         SessionFactory sessionFactory = entityManagerFactory.unwrap(SessionFactory.class)
74         if (sessionFactory != null) {
75             return sessionFactory.getStatistics();
76         }
77         return null;
78     }
79
80     private org.hibernate.stat.EntityStatistics getStatistics(EntityManagerFactory entity
81     if (entityManagerFactory == null) {
82         return null;
83     }
84     SessionFactory sessionFactory = entityManagerFactory.unwrap(SessionFactory.class)
85     if (sessionFactory != null) {
86         return sessionFactory.getStatistics().getEntityStatistics(pathAddress.getValu
87     }
88     return null;
89 }
90
91 private Operation entityDeleteCount = new Operation() {

```

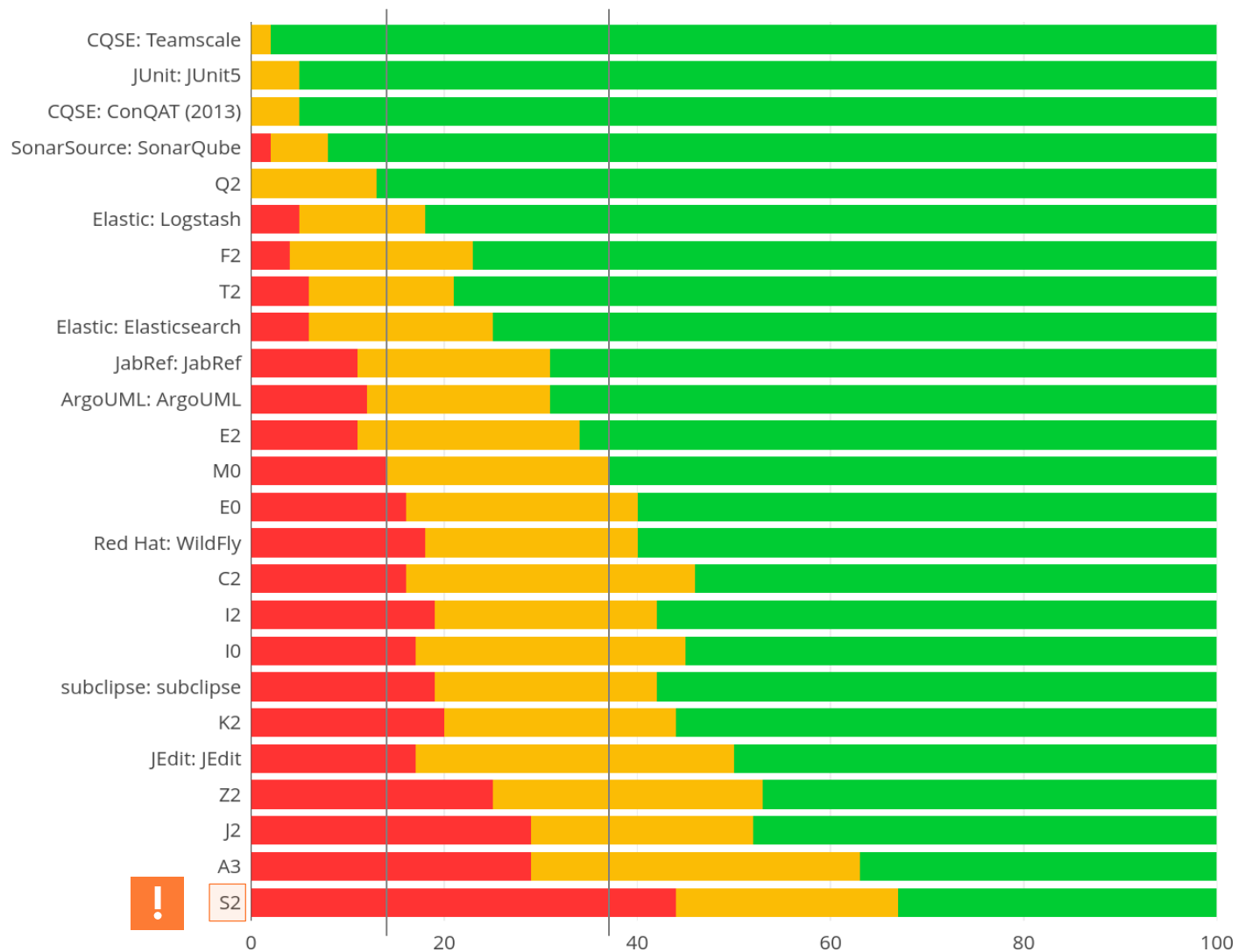
src/jpa/hibernate5_3

```

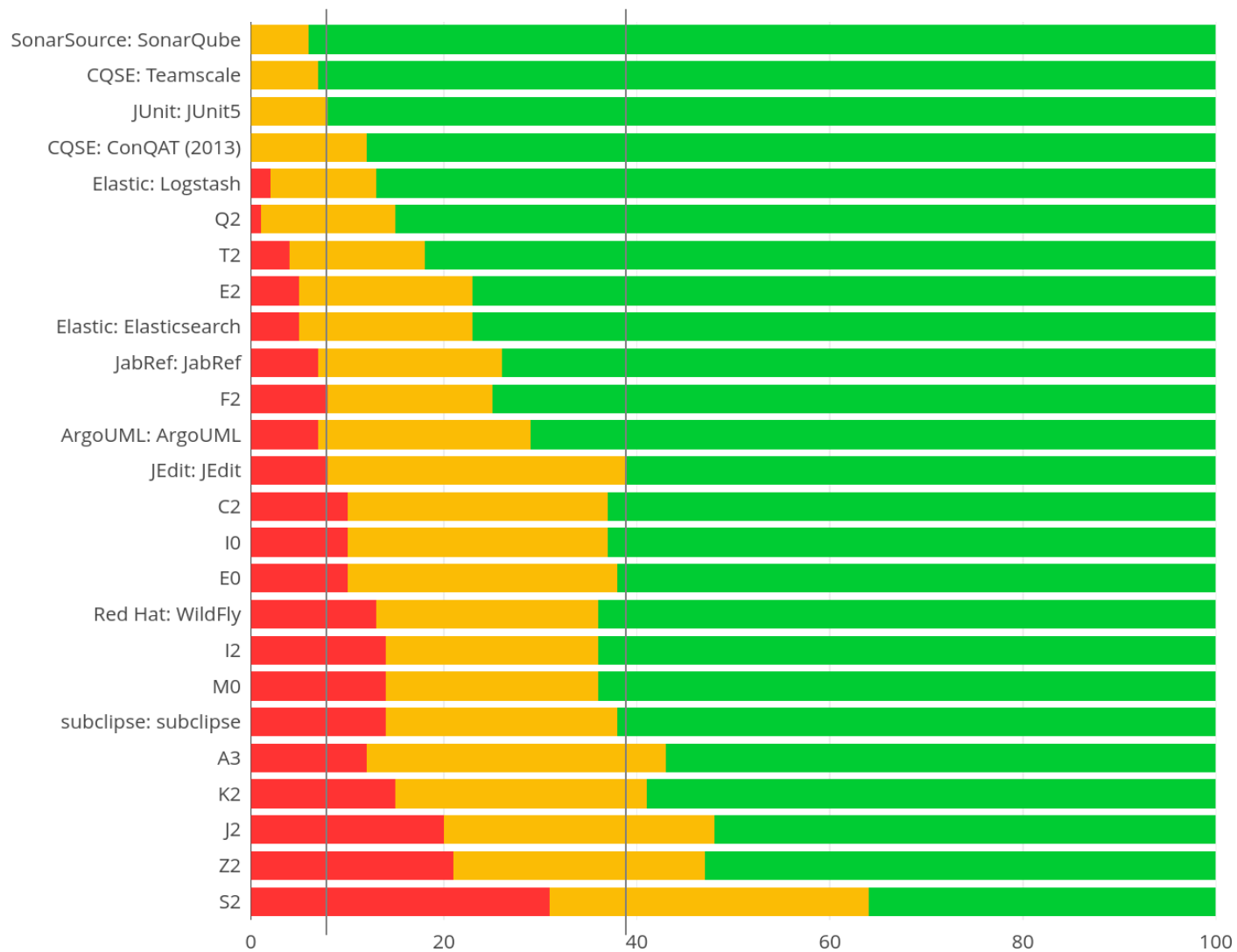
67
68
69     private org.hibernate.stat.Statistics getBaseStatistics(EntityManagerFactory entityMa
70         if (entityManagerFactory == null) {
71             return null;
72         }
73         SessionFactory sessionFactory = entityManagerFactory.unwrap(SessionFactory.class)
74         if (sessionFactory != null) {
75             return sessionFactory.getStatistics();
76         }
77         return null;
78     }
79
80     private org.hibernate.stat.EntityStatistics getStatistics(EntityManagerFactory entity
81     if (entityManagerFactory == null) {
82         return null;
83     }
84     SessionFactory sessionFactory = entityManagerFactory.unwrap(SessionFactory.class)
85     if (sessionFactory != null) {
86         return sessionFactory.getStatistics().getEntityStatistics(pathAddress.getValu
87     }
88     return null;
89 }
90
91 private Operation entityDeleteCount = new Operation() {

```

Methodenlänge

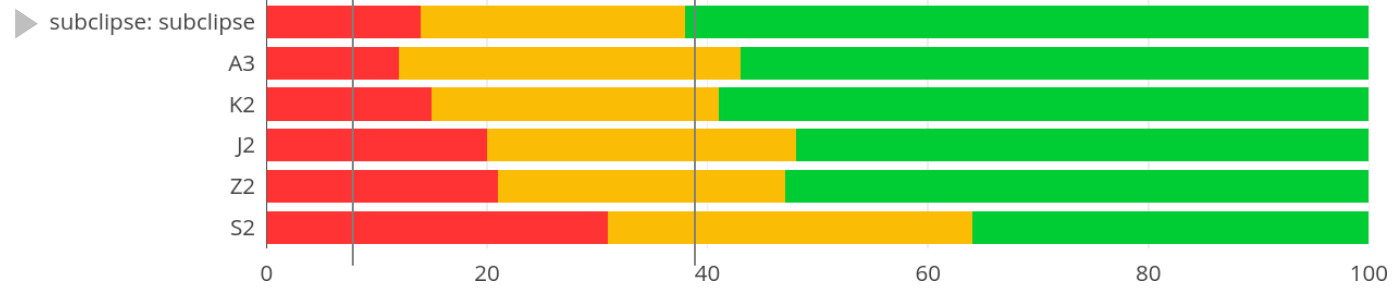


Schachtelungstiefe



So

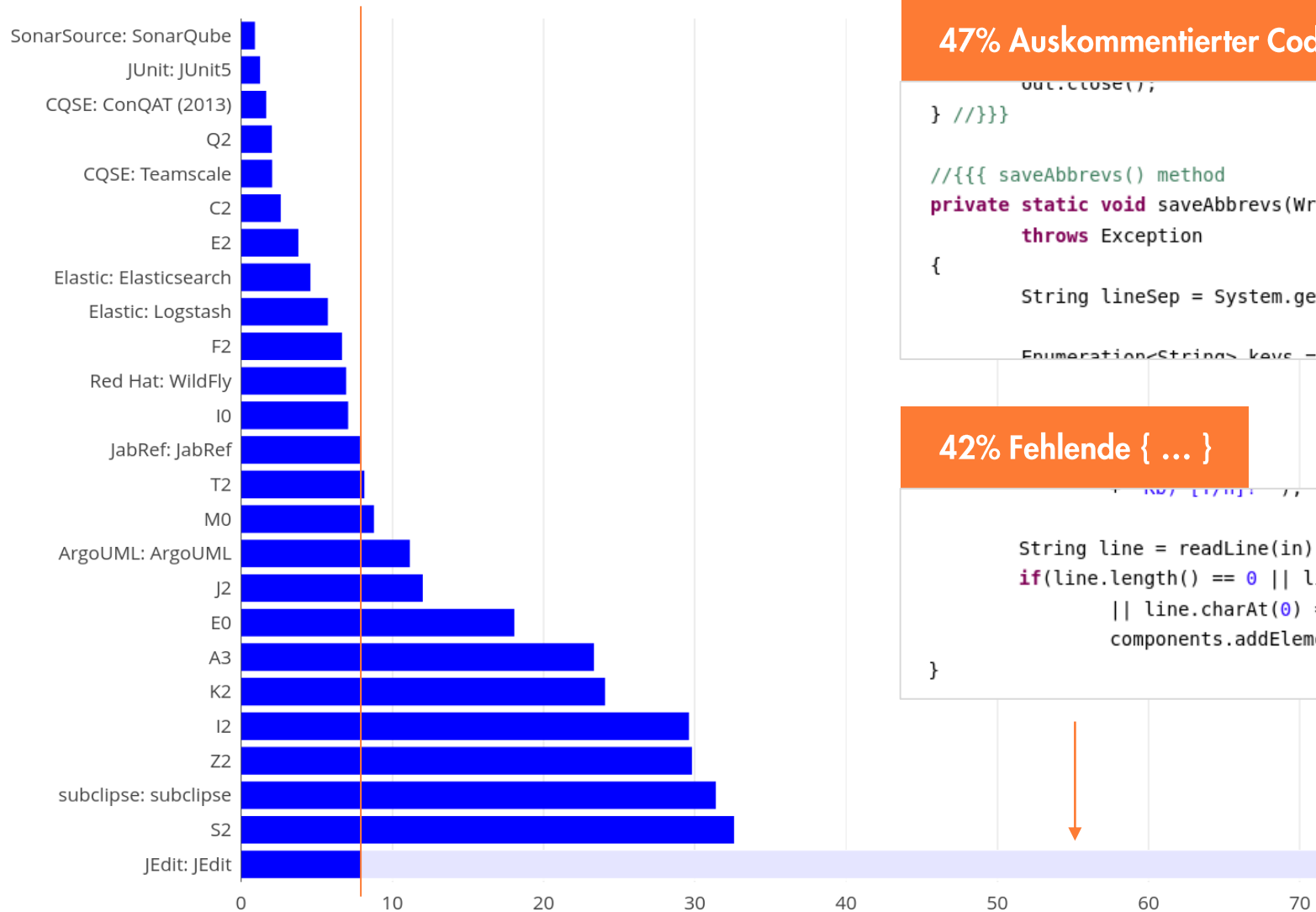
```
313 showCompareButton.setSelection(showCompare);
314 1 2 3 showCompareButton.addSelectionListener(
315     4 new SelectionListener(){
316     5 public void widgetSelected(SelectionEvent e) {
317     6 showComparePane(!showCompare);
318     7 if (showCompare) {
319     7 IStructuredSelection selection = (IStructuredSelection)resourceSelectionTree.getTreeView().getSelection();
320     8 if (!selection.isEmpty()) {
321     8 Object sel0 = selection.getFirstElement();
322     9 if (sel0 instanceof IFile) {
323     9 final ISVNLocalResource localResource= SVNWorkspaceRoot.getSVNResourceFor((IFile)sel0);
324     9 try {
325     10 // if any alternate compare actions are defined from the extension point
326     10 // then call those actions instead of showing the default compare dialog
327     11 if (alternateCompareActions.length > 0) {
328     11 StructuredSelection localResourceSelection = new StructuredSelection(localResource);
329     12 for (int i = 0; i < alternateCompareActions.length; i++) {
330     12 // make sure the selection is up to date
331     12 alternateCompareActions[i].selectionChanged(localResourceSelection);
332     12 alternateCompareActions[i].run();
333     12 }
334     12 } else {
335     12 setCompareInput(new SVNLocalCompareInput(localResource, SVNRevision.BASE, true));
336     12 }
337     12 } catch (Exception e1) {
338     12 }
339     12 }
340     12 }
341     12 }
342     12 }
```



Kommentierung



Codeanomalien / 1.000 SLOC



47% Auskommentierter Code

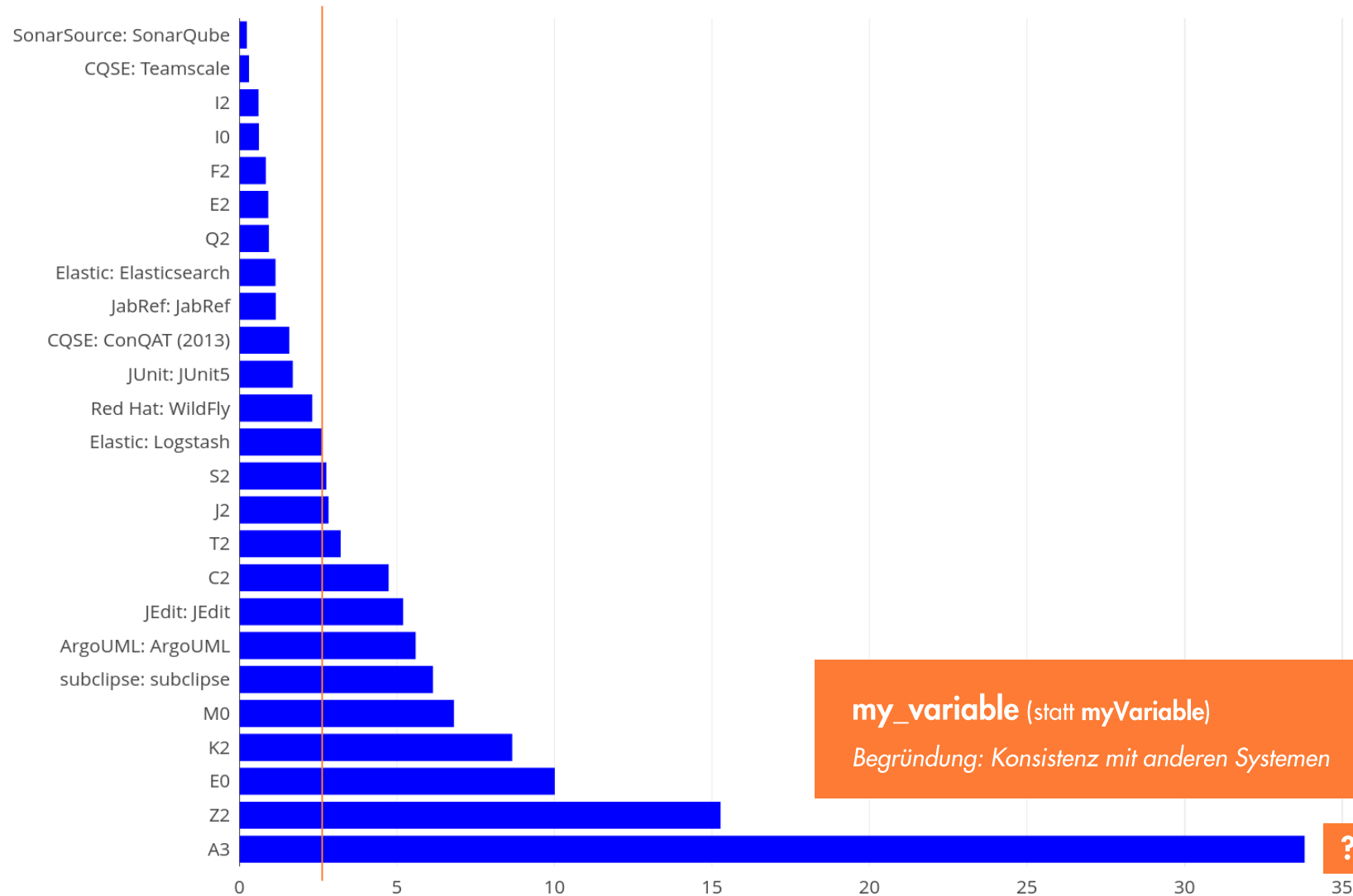
```
out.close();  
} //}}}  
  
//{{{ saveAbbrevs() method  
private static void saveAbbrevs(Writer out, Hashtable<String,Str  
    throws Exception  
{  
    String lineSep = System.getProperty("line.separator");  
  
    Enumeration<String> keys = abbrevs.keys();
```

42% Fehlende { ... }

```
String line = readLine(in);  
if(line.length() == 0 || line.charAt(0) == 'y'  
    || line.charAt(0) == 'Y')  
    components.addElement(fileset);  
}
```

?

Verletzungen der Namenskonventionen / 1.000 SLOC





Ein Benchmark ...



... ist ein wichtiger Baustein um von der **Messung zur Bewertung** zu kommen



... erlaubt von konkreten **Thresholds** und Check-Auswahl zu **abstrahieren**



... ist **keine automatische Bewertung** und muss interpretiert werden



... muss **gewissenhaft kuratiert** werden

Handlungsbedarf ?



Dr. Nils Göde · goede@cqse.eu · +49 176 10452662

CQSE GmbH
Centa-Hafenbrädl-Straße 59
81249 München



cqse.eu/SeaconDigital2021

CQSE
Continuous Quality in Software Engineering