COMP0104 Software Development Practice: Automated Code Review

Jens Krinke

Centre for Research on Evolution, Search & Testing
Software Systems Engineering Group
Department of Computer Science
University College London

Where are we now?

- Continuous Integration
- Continuous Inspection
  (Software Measurement, Technical Debt)
- Code Review

Continuous Integration

- Continuous Integration can be interruptive when it occurs post-merge.
- Solution:
  Move to pre-merge CI
  Integrate with Code Review

Software Measurement / Technical Debt

- Constant reminder off all issues in the project, including irrelevant issues.
- Introduction into new projects: Where to start to address issues?
- Developers ignore the results…
- Technical Debt increases…
- Managers are not happy…

How to do it right…

The Boy Scout Rule

“Always leave the campground cleaner than you found it.”

For software:

“Always check a module in cleaner than when you checked it out.”
– Robert C. Martin (Uncle Bob)

https://unsplash.com/photos/Ppz6b-YUDHw

Automated Code Review

- Focus on the change and its context instead of the whole project.
- Check for new issues
- Prevent new bugs
- Do not take on new debt

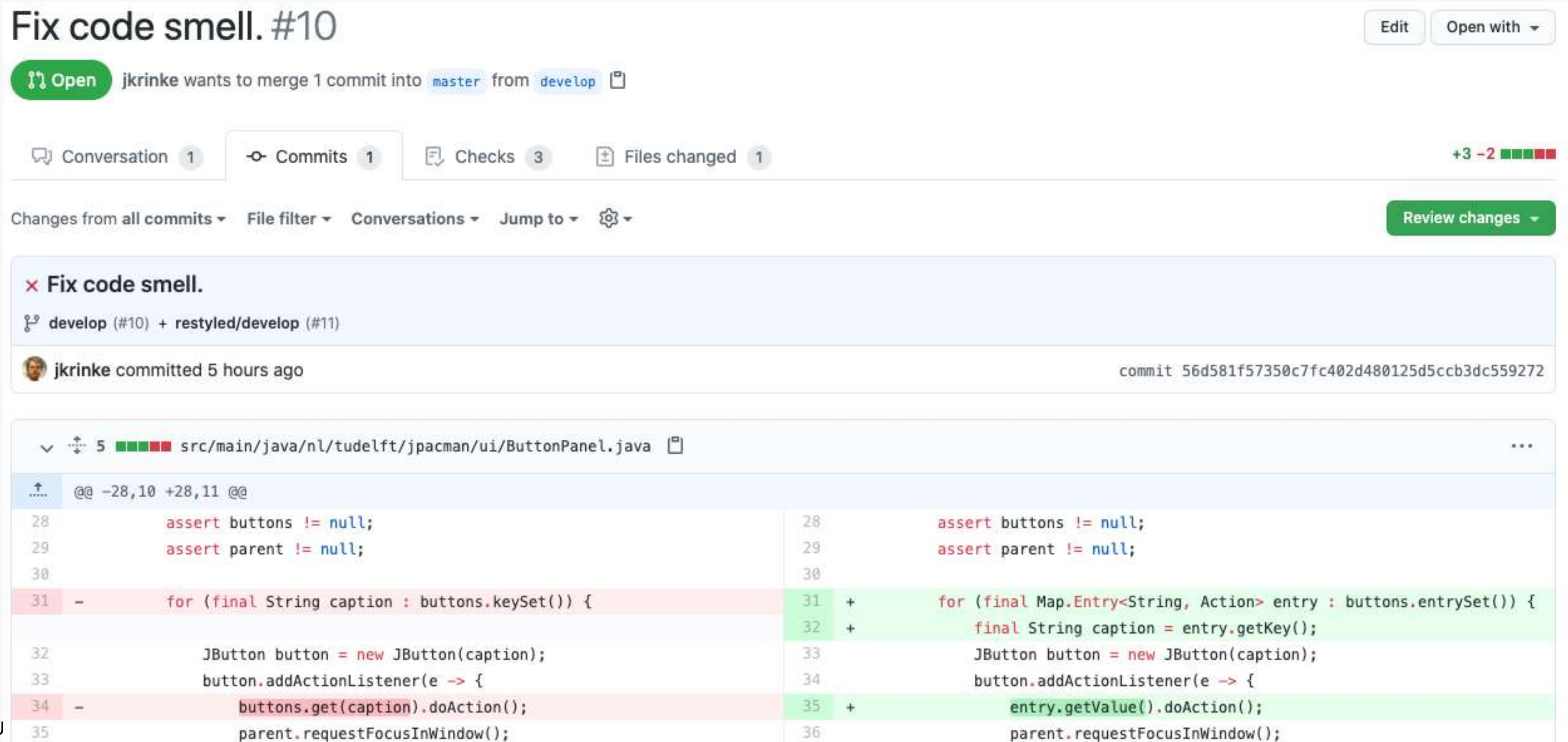
Automated Code Reviewers

Humans are not actually very good at

- Finding bugs and flaws
- Spotting code smells

Support human reviewers as much as possible by artificial reviewers!

SonarCloud as Review Bot



Fix code smell. #10

Open jkrinke wants to merge 1 commit into master from develop

Conversation 1 · Commits 1 · Checks 3 · Files changed 1 · +3 −2

Changes from all commits · File filter · Conversations · Jump to · Review changes

× Fix code smell.

develop (#10) + restyled/develop (#11)

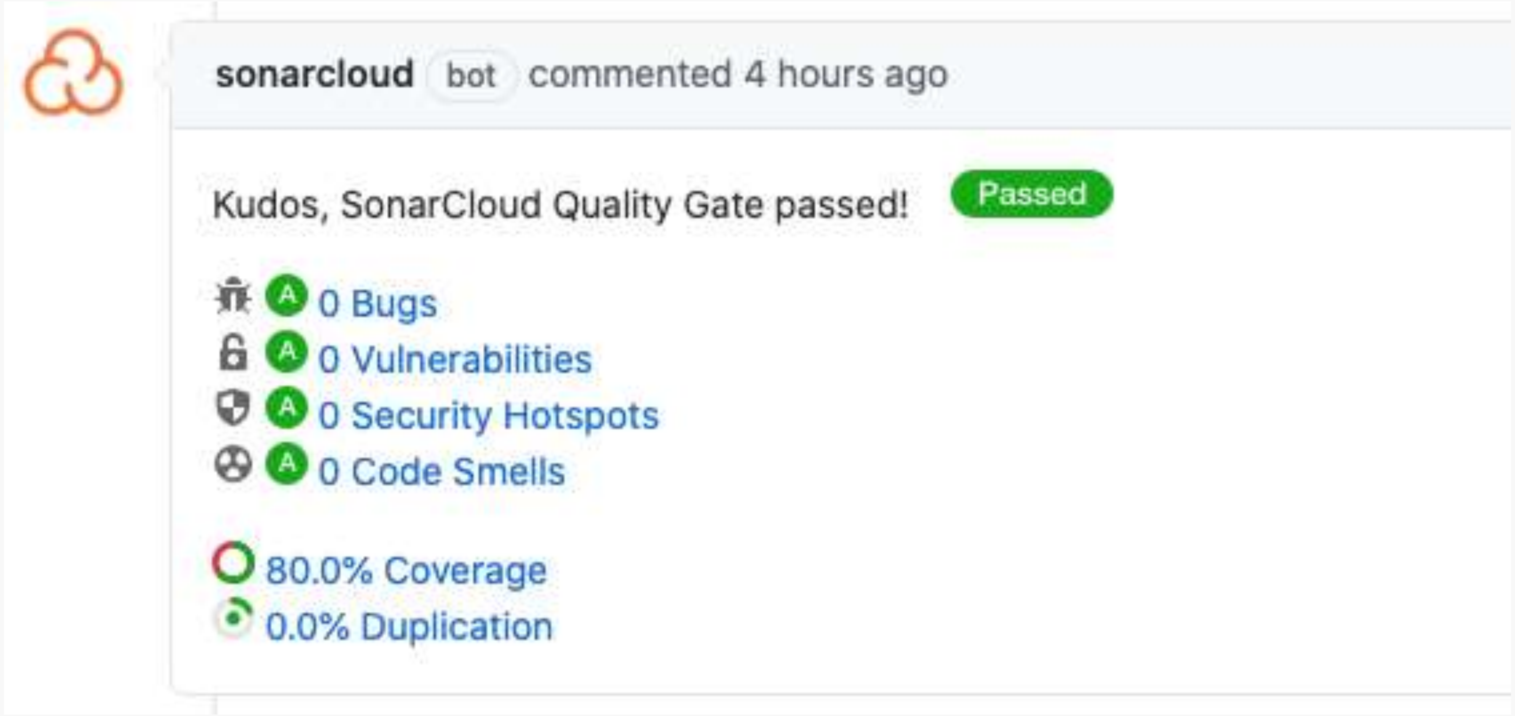
jkrinke committed 5 hours ago · commit 56d581f57350c7fc402d480125d5ccb3dc559272

5 src/main/java/nl/tudelft/jpacman/ui/ButtonPanel.java

```
@@ -28,10 +28,11 @@
         assert buttons != null;
         assert parent != null;

-        for (final String caption : buttons.keySet()) {
+        for (final Map.Entry<String, Action> entry : buttons.entrySet()) {
+            final String caption = entry.getKey();
             JButton button = new JButton(caption);
             button.addActionListener(e -> {
-                buttons.get(caption).doAction();
+                entry.getValue().doAction();
                 parent.requestFocusInWindow();
```

SonarCloud as Review Bot



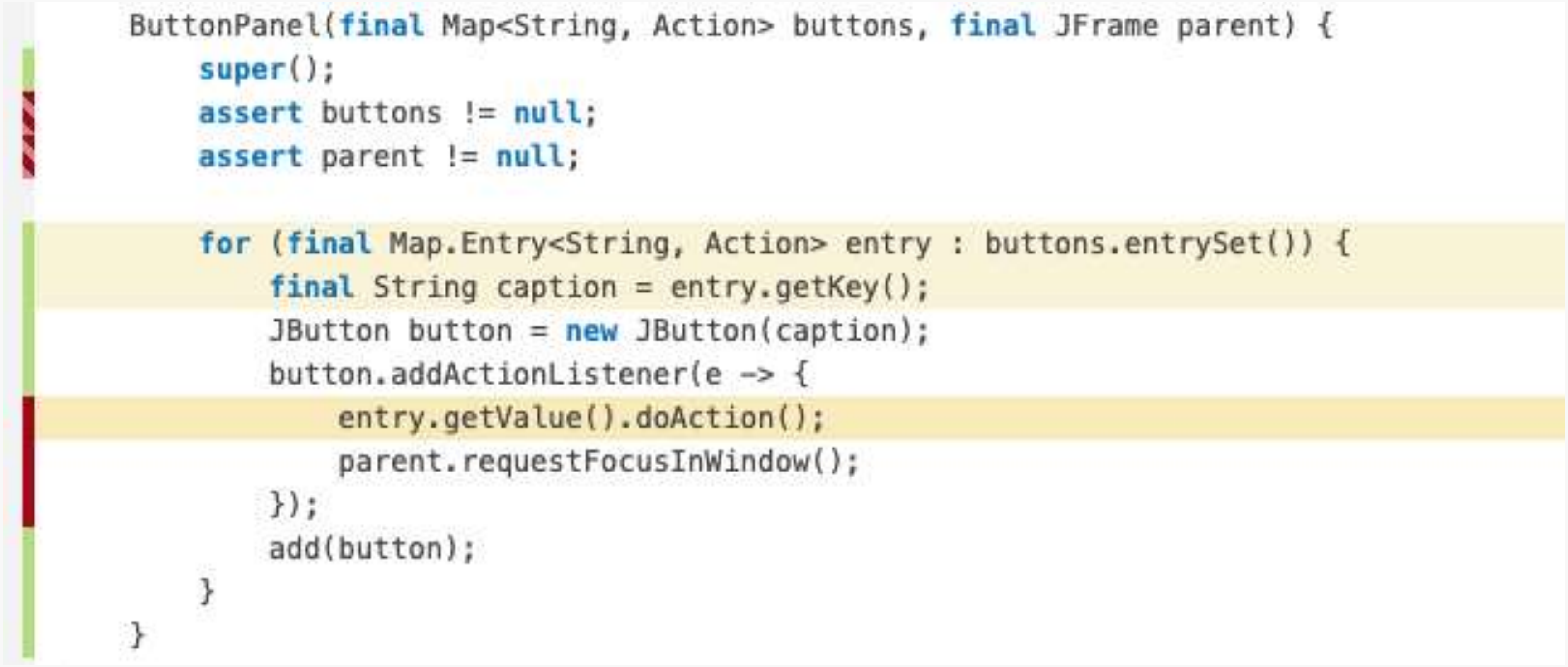
sonarcloud bot commented 4 hours ago

Kudos, SonarCloud Quality Gate passed! Passed

A 0 Bugs
A 0 Vulnerabilities
A 0 Security Hotspots
A 0 Code Smells

80.0% Coverage
0.0% Duplication

SonarCloud as Review Bot: Coverage

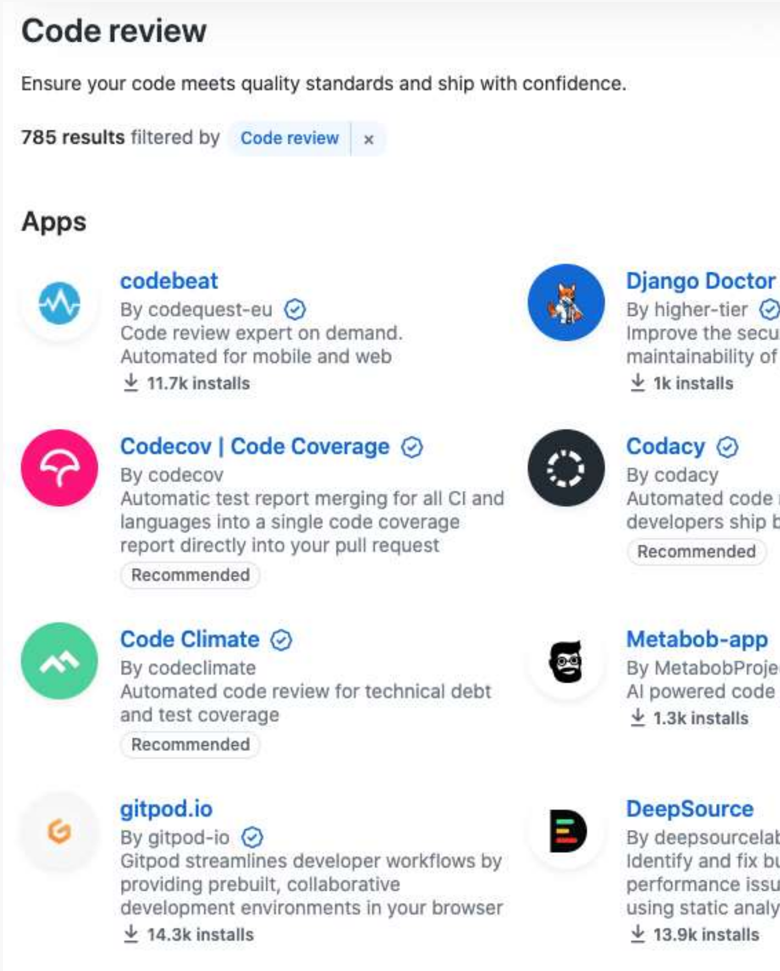


```
    ButtonPanel(final Map<String, Action> buttons, final JFrame parent) {
        super();
        assert buttons != null;
        assert parent != null;

        for (final Map.Entry<String, Action> entry : buttons.entrySet()) {
            final String caption = entry.getKey();
            JButton button = new JButton(caption);
            button.addActionListener(e -> {
                entry.getValue().doAction();
                parent.requestFocusInWindow();
            });
            add(button);
        }
    }
```

Review Bots are plentiful…

- Review Bots are services offered on GitHub.
- They cover a wide range of checks and languages.
- Which ones are useful?



Code review

Ensure your code meets quality standards and ship with confidence.

785 results filtered by Code review ×

Apps

codebeat
By codequest-eu
Code review expert on demand. Automated for mobile and web
11.7k installs

Django Doctor
By higher-tier
Improve the secu
maintainability of
1k installs

Codecov | Code Coverage
By codecov
Automatic test report merging for all CI and languages into a single code coverage report directly into your pull request
Recommended

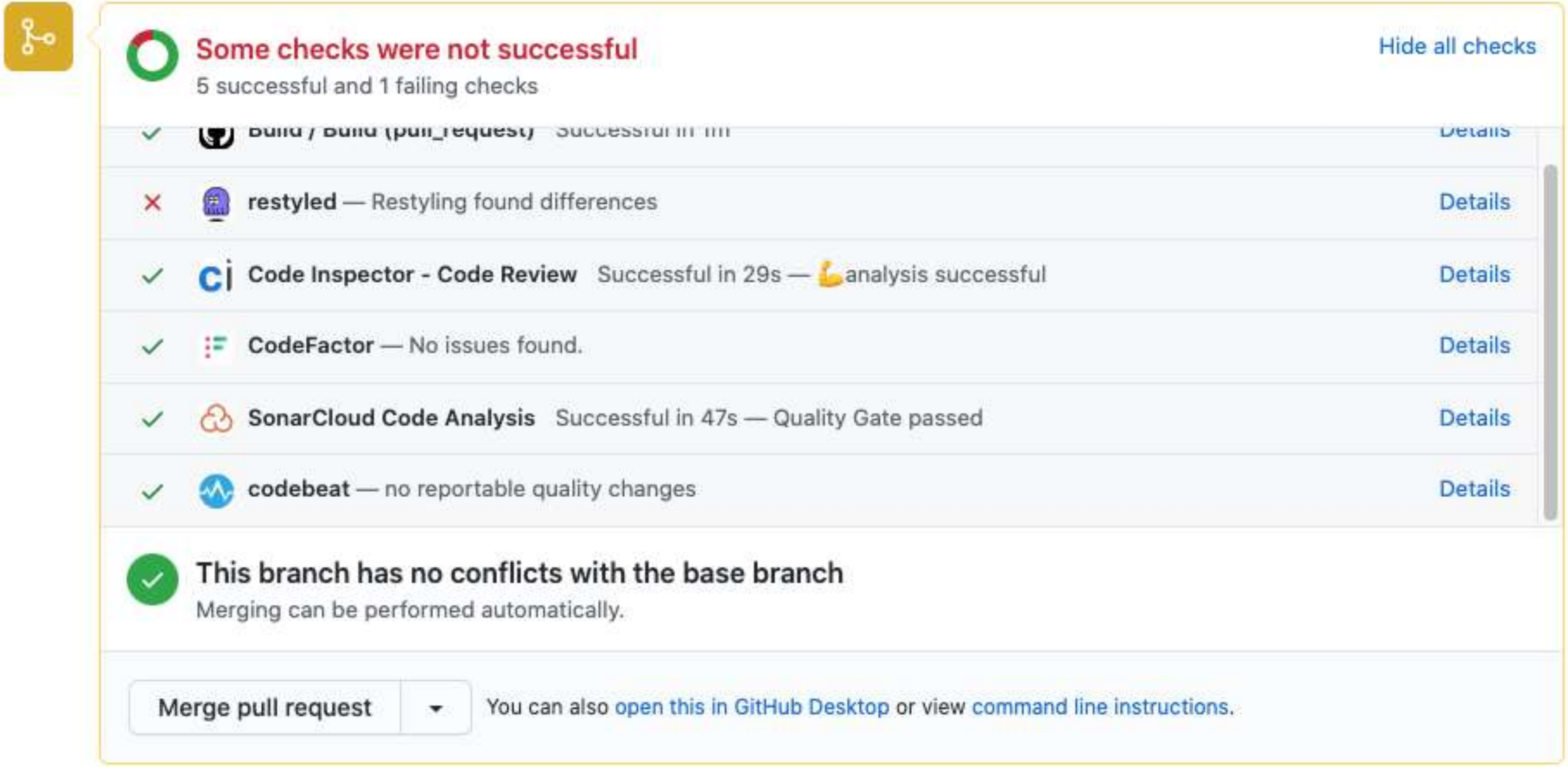
Codacy
By codacy
Automated code
developers ship b
Recommended

Code Climate
By codeclimate
Automated code review for technical debt and test coverage
Recommended

Metabob-app
By MetabobProje
AI powered code
1.3k installs

gitpod.io
By gitpod-io
Gitpod streamlines developer workflows by providing prebuilt, collaborative development environments in your browser
14.3k installs

DeepSource
By deepsourcelab
Identify and fix bu
performance issu
using static analy
13.9k installs



Some checks were not successful

5 successful and 1 failing checks

Hide all checks

✓ Build / Build (pull_request) Successful in 1m — Details

✗ restyled — Restyling found differences — Details

✓ Code Inspector - Code Review Successful in 29s — 💪analysis successful — Details

✓ CodeFactor — No issues found. — Details

✓ SonarCloud Code Analysis Successful in 47s — Quality Gate passed — Details

✓ codebeat — no reportable quality changes — Details

This branch has no conflicts with the base branch

Merging can be performed automatically.

Merge pull request

You can also open this in GitHub Desktop or view command line instructions.

The Boy Scout Rule

"Always leave the campground cleaner than you found it."

For software:

"Always check a module in cleaner than when you checked it out."
– Robert C. Martin (Uncle Bob)

https://unsplash.com/photos/Ppz6b-YUDHw

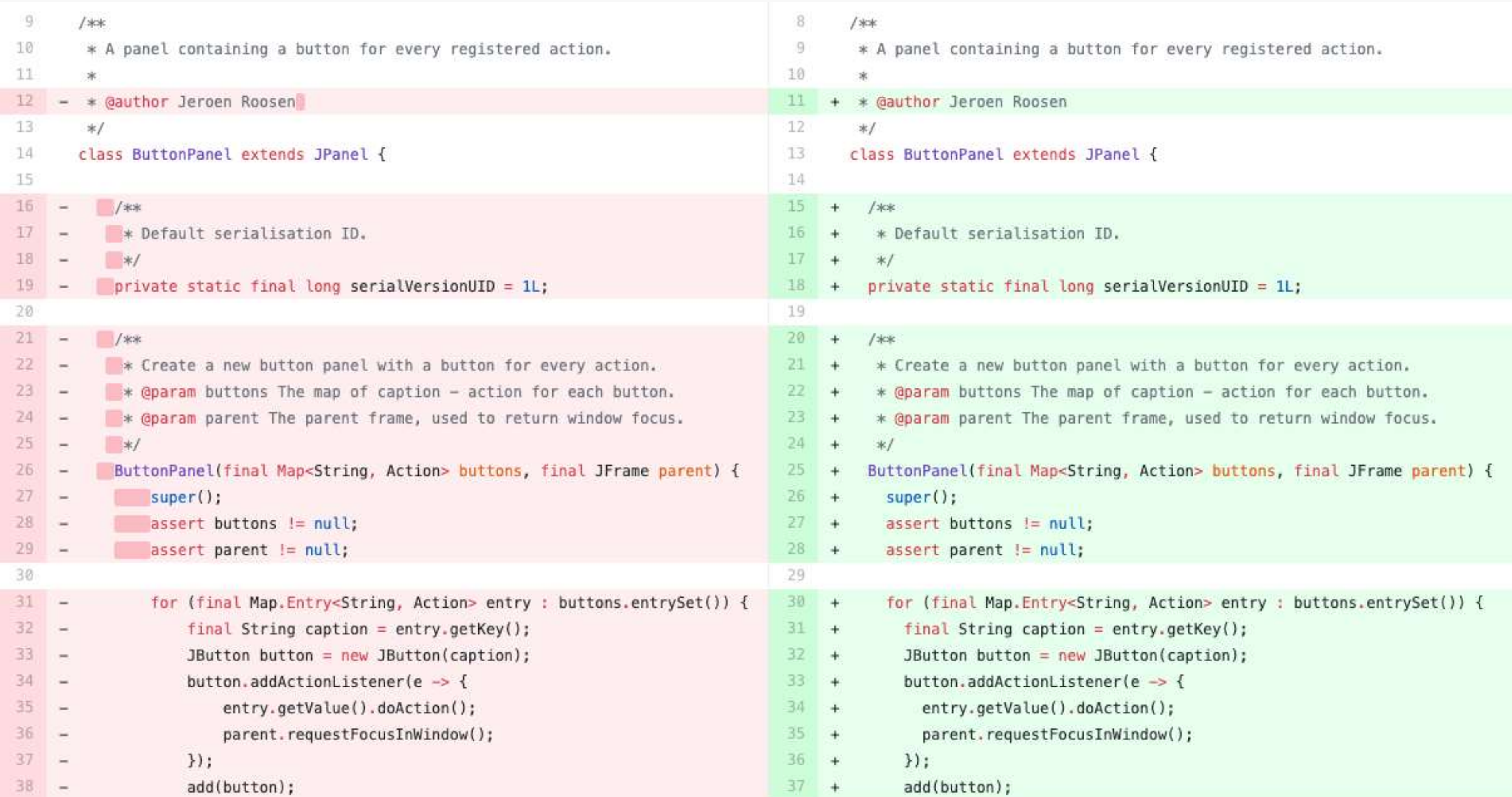
The Boy Scout Rule (amended)

“Always leave the campground cleaner than you found it, *but do not leave the campground*.”

For software:

“Always check a module in cleaner than when you checked it out, *but do not clean areas not affected by the change*.”

https://unsplash.com/photos/tRkBF9Ujqw4



```
 /**
  * A panel containing a button for every registered action.
  *
- * @author Jeroen Roosen
  */
 class ButtonPanel extends JPanel {

-	/**
-	 * Default serialisation ID.
-	 */
-	private static final long serialVersionUID = 1L;

-	/**
-	 * Create a new button panel with a button for every action.
-	 * @param buttons The map of caption - action for each button.
-	 * @param parent The parent frame, used to return window focus.
-	 */
-	ButtonPanel(final Map<String, Action> buttons, final JFrame parent) {
-		super();
-		assert buttons != null;
-		assert parent != null;

-		for (final Map.Entry<String, Action> entry : buttons.entrySet()) {
-			final String caption = entry.getKey();
-			JButton button = new JButton(caption);
-			button.addActionListener(e -> {
-				entry.getValue().doAction();
-				parent.requestFocusInWindow();
-			});
-			add(button);
```

```
 /**
  * A panel containing a button for every registered action.
  *
+ * @author Jeroen Roosen
  */
 class ButtonPanel extends JPanel {

+  /**
+   * Default serialisation ID.
+   */
+  private static final long serialVersionUID = 1L;

+  /**
+   * Create a new button panel with a button for every action.
+   * @param buttons The map of caption - action for each button.
+   * @param parent The parent frame, used to return window focus.
+   */
+  ButtonPanel(final Map<String, Action> buttons, final JFrame parent) {
+    super();
+    assert buttons != null;
+    assert parent != null;

+    for (final Map.Entry<String, Action> entry : buttons.entrySet()) {
+      final String caption = entry.getKey();
+      JButton button = new JButton(caption);
+      button.addActionListener(e -> {
+        entry.getValue().doAction();
+        parent.requestFocusInWindow();
+      });
+      add(button);
```

Why not to touch unchanged code?

- Any change affects the change history and the metadata.
- git blame will no longer identify
  the last change of the functionality.
- Unnecessary changes cause the review to be larger,
  take longer and cost more.
- Even style changes need to be tested and covered.

Concepts

- Human reviewer are not very good at finding bugs, they should be supported by artificial reviewers.
- Instead of measuring and observing the overall project state, it is better to prevent that a change introduces new issues.
- It is usually better to only fix code smells or style violations when and where a functionality change is necessary.