Preliminary exploratory analysis: PMD Alerts + Commits

Bruno Crotman

20/10/2019

Contents

1 Introduction **1**

2 PMD Alerts **1**

3 Preliminary analysis **2**
3.1 Collecting Data 2
3.2 Evolution of the code size 2
3.3 Types of alert 5
3.4 Priorities of the Alerts 5

4 Conclusions and next steps **5**

5 References **7**

1 Introduction

This document shows the how to collect PMD alerts related to each commit of a project and introduces an analysis about this information.

In the Section 2, this document explains briefly how PMD Source Code Analyzer works and which kind of alerts it generates. In the Section 3 the data collection and a brief analysis using the project Twitter4J is shown. The section 4 discusses the next steps for the work.

2 PMD Alerts

PMD is a source code analyzer that finds common flaws, like unused variables, empty catch blocks, unnecessary object creation and so forth (Dangel 2019).

In Java, the programing language in which Twittter4J is written, there are 8 groups of rules that generate alerts:

- Best Practices: enforce generally accepted best practices, e.g. the usage of foreach instead of for when applicable, avoidance of hard coded IP addresses, avoidance of the reassignment of loop variables.
- Code Style: enforce specific ccode style, e.g, the non static classes must declare at leas one constructor, control statement must have braces.
- Design: e.g. abstract classses must have methods, classes must not have too many fields or methods
- Documentation: e.g. the comments' size must be within certain limits
- Error prone: detect constructs that are either broken, extremely confusing or prone to runtime errors, e.g. assigment to operands, if statements whose conditionals are always true or always false.

PMD lets you choose which types of alerts to use and even create new rules. In this work a “quick start” ruleset, that comes as a basic configuration, is used.

3 Preliminary analysis

In this preliminary analysis, the GitHub repository related to Twitter4J project (Twitter4J 2019) is cloned and each commited code is analyzed by PMD (). The information is

3.1 Collecting Data

The following code collects data from every commit made to Twitter4J project.

Each commit is checked out and the PMD alerts are generated.

In the end, all the PMD alerts are stored.

Data can be loaded from the files that serve as cache

3.2 Evolution of the code size

After an instability, it seems that the project evolves in a steady pace from the commit 300 until aproximately commit 1450, until the release of version 3.0.0. After that, the number of lines of code ceases to increase, and so do the number of alerts and the number of alerts per line of code.

Figure 1 shows this evolution, in terms of PMD alerts per line of code, total PMD alerts and total lines of code. The smaller red dots are releases of minor versions. The bigger and labeled ones are major version releases.

Figure 2 shows the number of commits made in each quarter.

Evolution along the commits

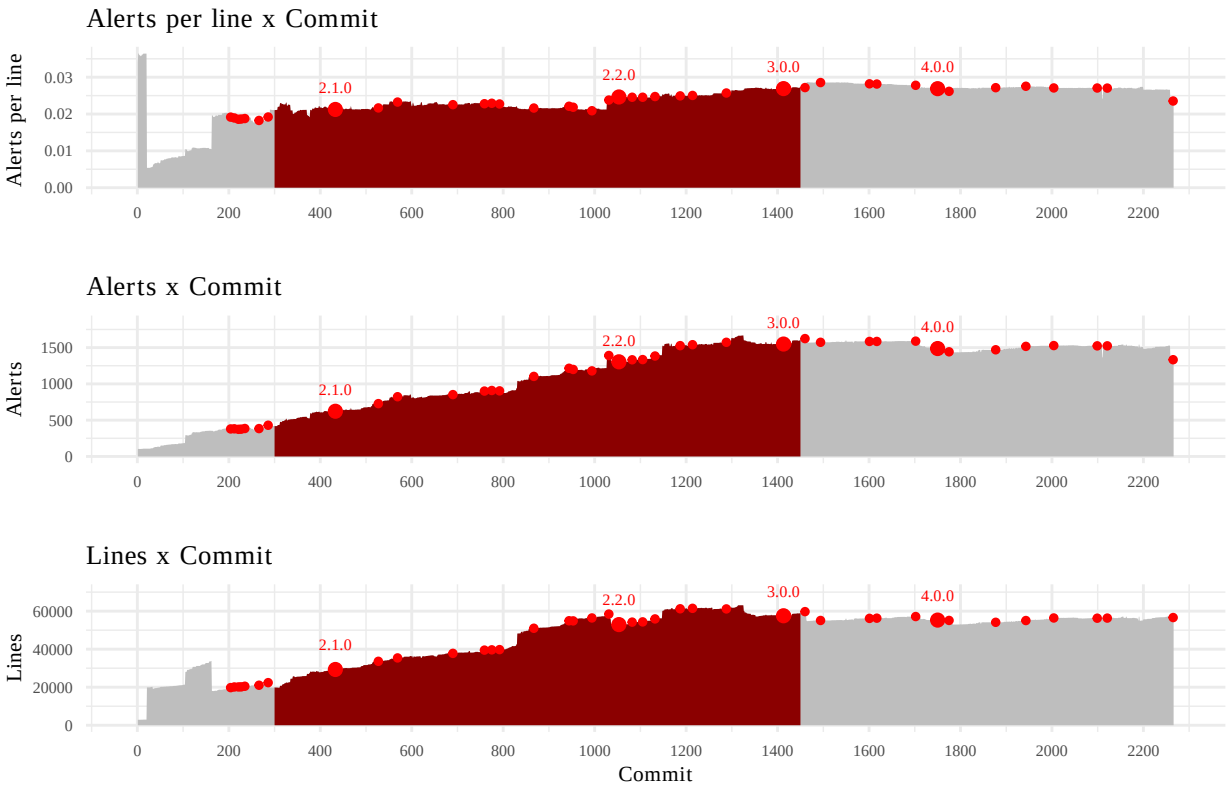

Figure 1: Project evolution by commit

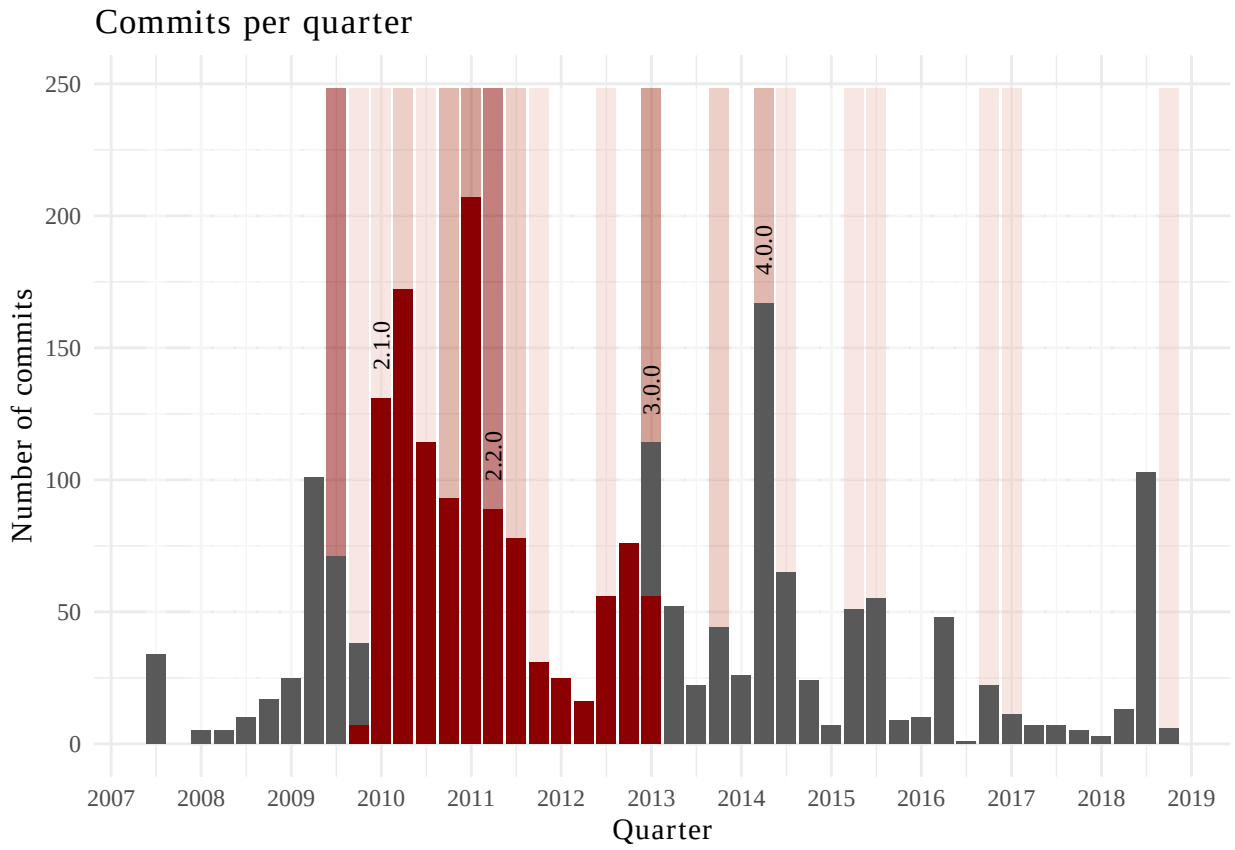

Figure 2: Project evolution by quarter

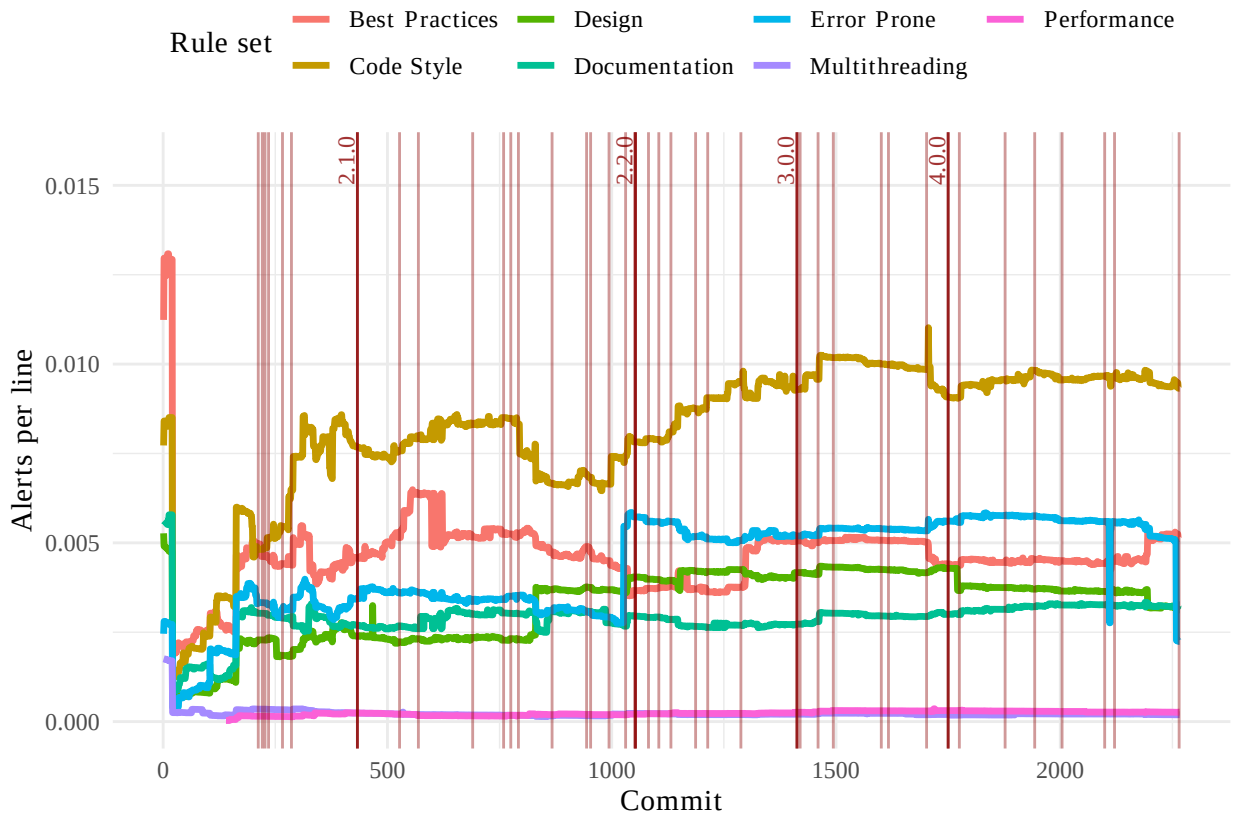

Figure 3: Evolution of the types of alerts

3.3 Types of alert

It's possible to analyze the amount of each type of alert in the code and how they evolve along the commits.

Figure 3 shows the evolution of the types of alerts by commit and releases. The light vertical lines refer to the release of a patch. The labeled vertical lines correspond to the minor and major releases.

Code Style alerts are most common, and Error Prone alerts are the second most common at the end of the period. There is a fast growth in Error Prone alerts just before the release of the version 2.2.0.

3.4 Priorities of the Alerts

PMD assigns priorities to the alerts. Priority 1 is the worst.

Figure 4 show the alerts by priority.

There is a big increase just before the release of version 2.2.0. The origin of this increase must linked with the increase in the Error Prone alerts refered in Section 3.3.

4 Conclusions and next steps

In the next weeks we will dive into this data. Now that we have the tools we need to grab the data linking alerts and commits of the projects, we can go through some analysis:

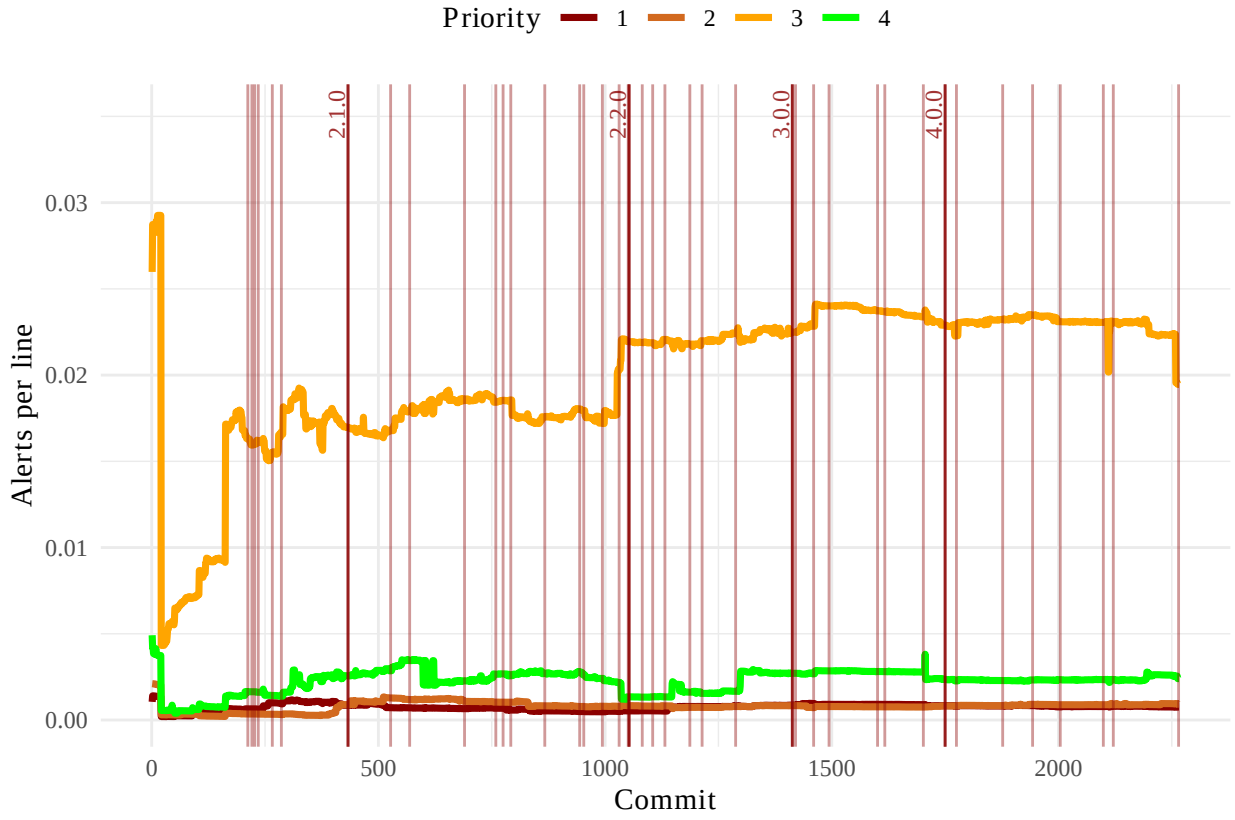

Figure 4: Evolution of the alerts by priority

- Can we find developers that consistently commit with too much alerts per line?
- Is “Alerts per line” a good proxy for kludges ?
- Does the behavior of the developers change when they are aproaching a release or when the frequency of the commits is higher?
- Does the team change its behavior when a member starts to kludge?
- Does a change in the team behavior degrade the quality of the code?

5 References

Dangel, Andreas et. al. 2019. “PMD Source Code Analyzer.” 2019. https://pmd.github.io/.

Twitter4J. 2019. “Twitter4J.” 2019. http://twitter4j.org/en/.