Effect of code bad smells on modularity

**Section1:**

Code smells are signs of poor design and/or style that can be found in source code. They go against best practices and design principles, which affects comprehension and maintenance and encourages a detrimental effect on the quality of software and ease of evolution.

Software with code smells won't necessarily stop working; it will still provide an output, but it may process more slowly, run at a higher risk of failure and mistakes, and be more susceptible to faults in the future. Code smell is a factor in poor code quality, which raises the technical debt.

Modularity is Easier to develop, Easier to maintain (extend, modify, fix, …), Easier to test.

Goal:

Here we are going to analyze the effect of code smells on modularity of the project.

Question:

Does really code bad smell has an impact on the project modularity?

Metrics:

We will be using ck metrics tool to generate different metrics and based on the metrics we will calculating the software quality.

**Section2:**

|  |  |  |
| --- | --- | --- |
| Sno | ProjectName | Description |
| Project1 | quickstart-android | A collection of quickstart samples demonstrating the Firebase APIs on Android |
| Project2 | dropwizard | Dropwizard is a sneaky way of making fast Java web applications. It's a little bit of opinionated glue code which bangs together a set of libraries which have historically not sucked |
| Project3 | CoreNLP | Stanford CoreNLP Provides a set of natural language analysis tools written in Java. |
| Project4 | testing-samples | A collection of samples demonstrating different frameworks and techniques for automated testing. |
| Project5 | Activiti | Activiti is a light-weight workflow and Business Process Management (BPM) Platform targeted at business people, developers and system admins. |
| Project6 | questdb | QuestDB is an open-source time-series database for high throughput ingestion and fast SQL queries with operational simplicity. |

***Section3:***

**CK metrics:**

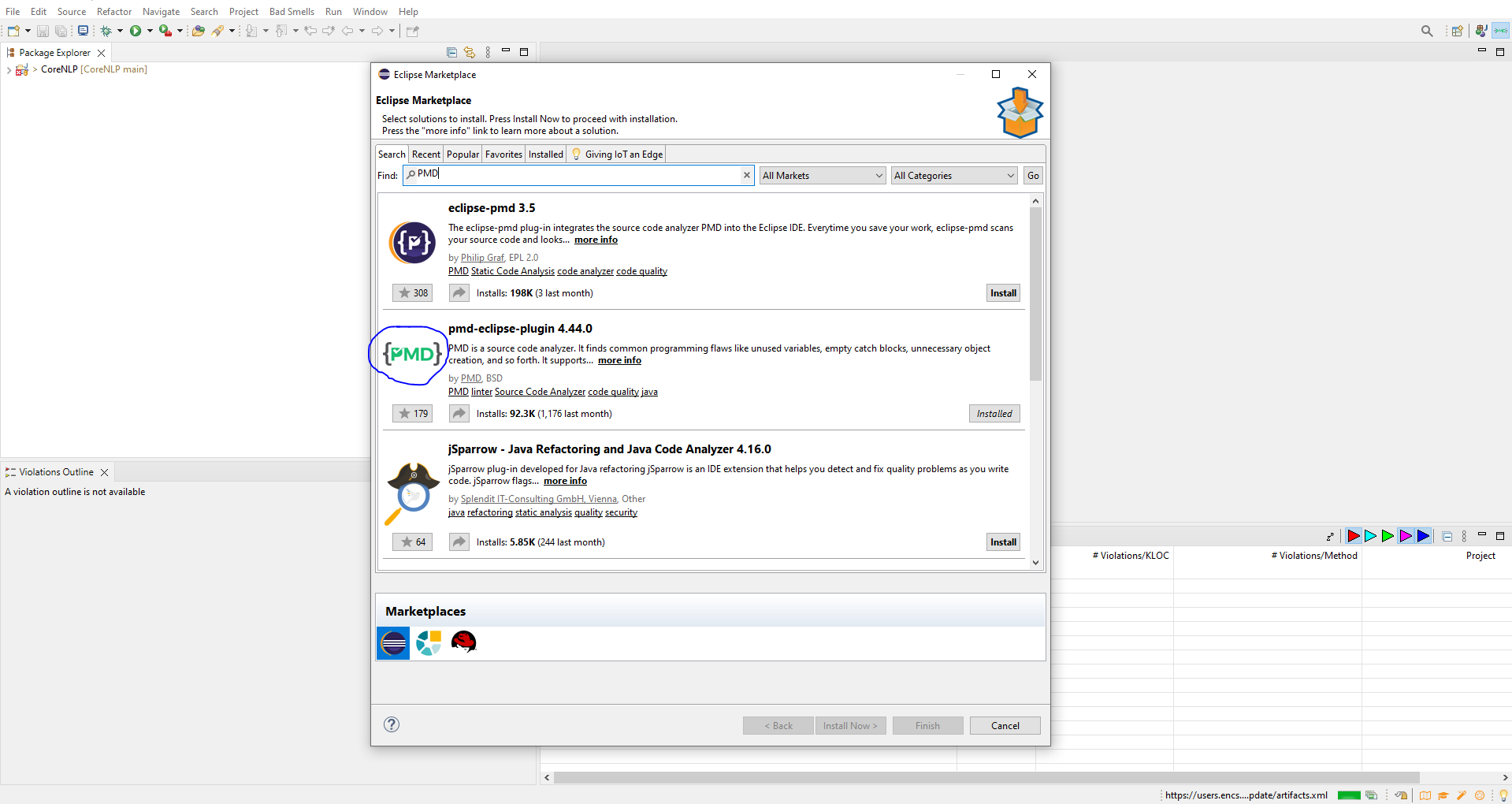
For calculation of metrics here we are using CK metrics. The Chidamber&Kemerer metrics suite originally consists of 6 metrics calculated for each class: WMC, DIT, NOC, CBO, RFC and LCOM1. The original suite has later been amended by RFC´, LCOM2, LCOM3 and LCOM4 by otherauthors.Through the use of static analysis, CK determines class-level and method-level code metrics in Java projects.

For estimating modularity, we are especially taking CBO, LCOM values into consideration.

**PMD:**

There are many tools to detect code bad smells available in the market. Here we are using PMD. We have used this eclipse plug in to detect the code smell.

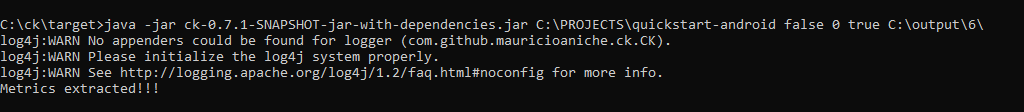
PMD3 is an open source tool for Java and an Eclipse plugin that detects many problems in Java code, including two of the code smells of our interest: God Class and God Method. The detection techniques are based on metrics.

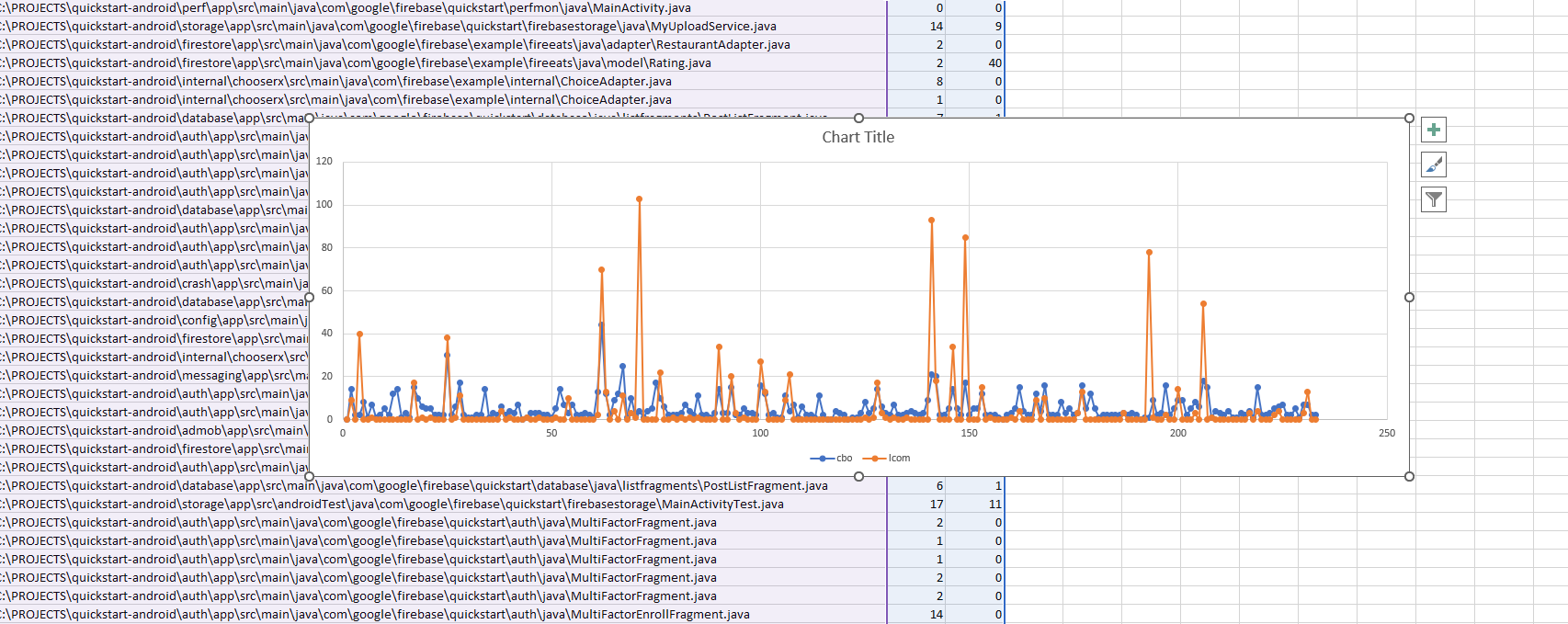


***Section4:***

Project1: quickstart-android

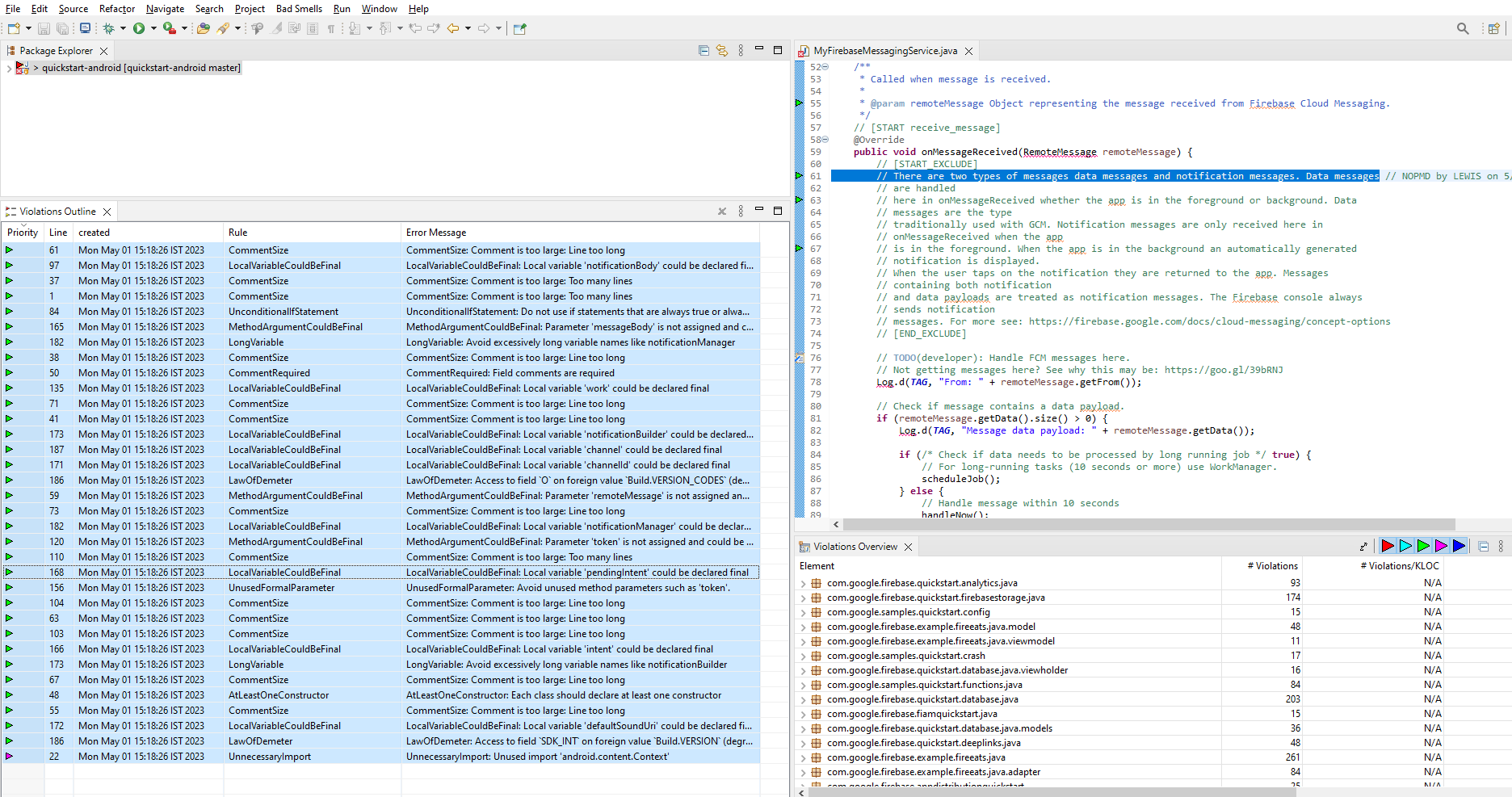
I am executing the ck metrics for the project:



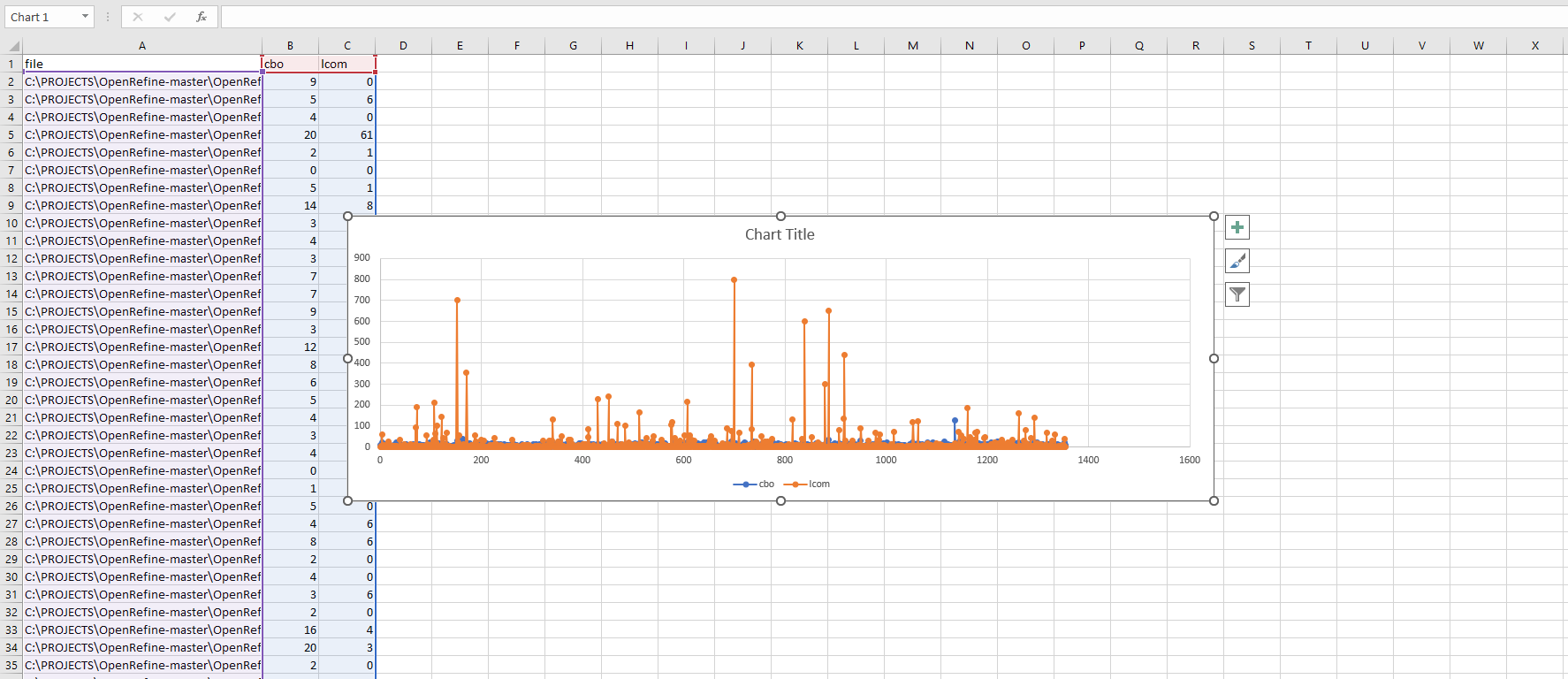


For all the charts , x axis (horizontal) is number of classes, y axis vertical is the lcom, cbo values.

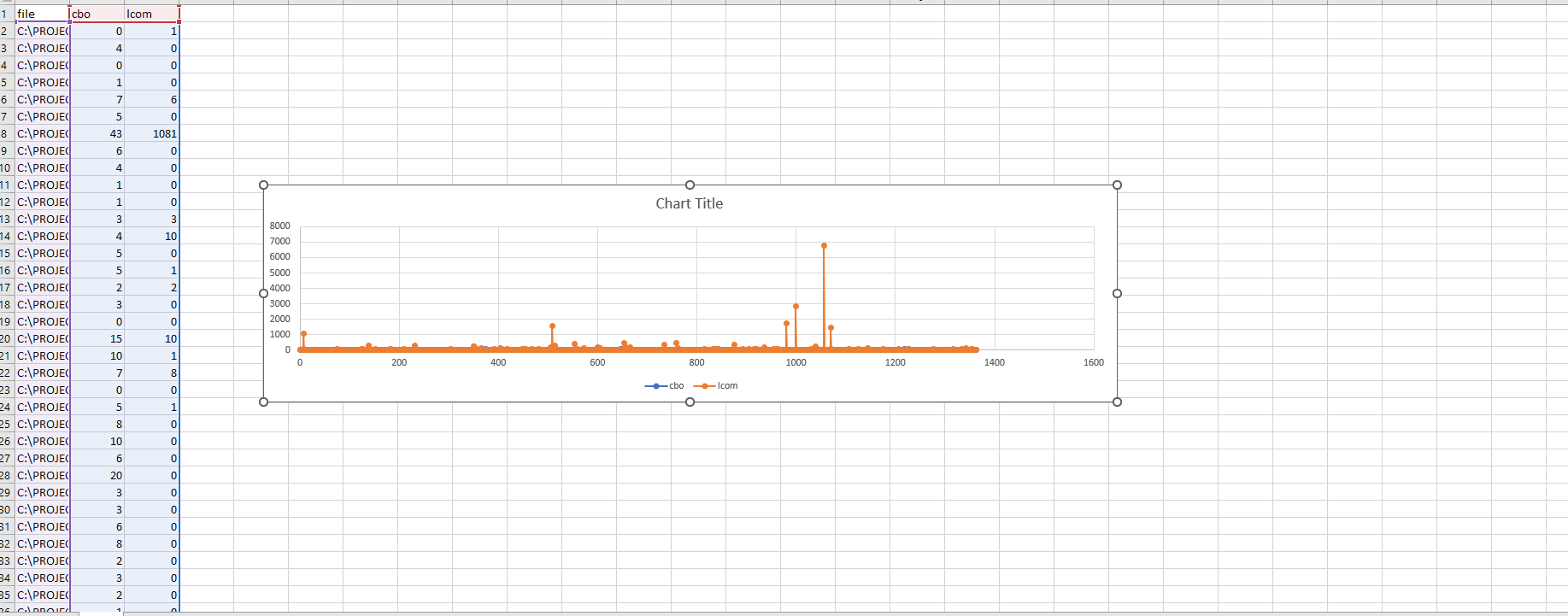
Here is the bad smells detected for the project ‘quickstart-android’ using PMD tool.



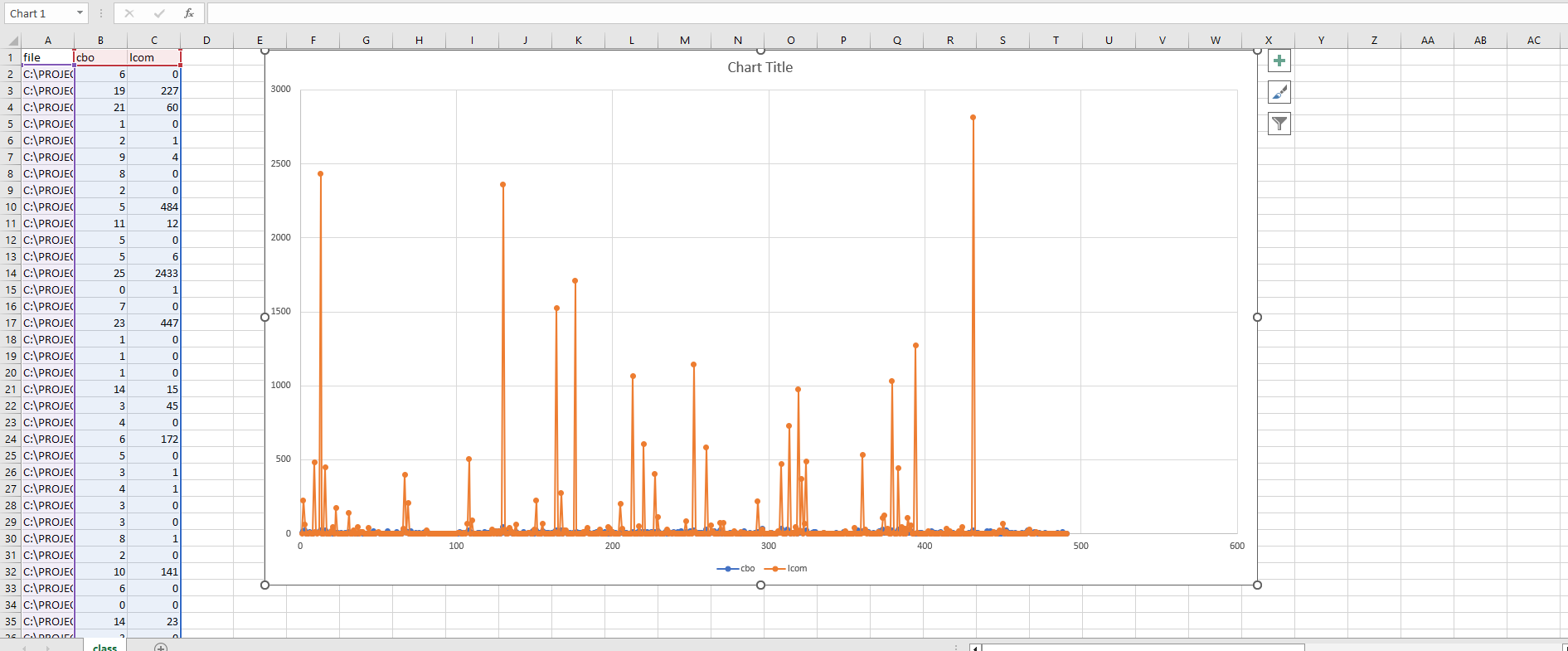
Project2:OpenRefine



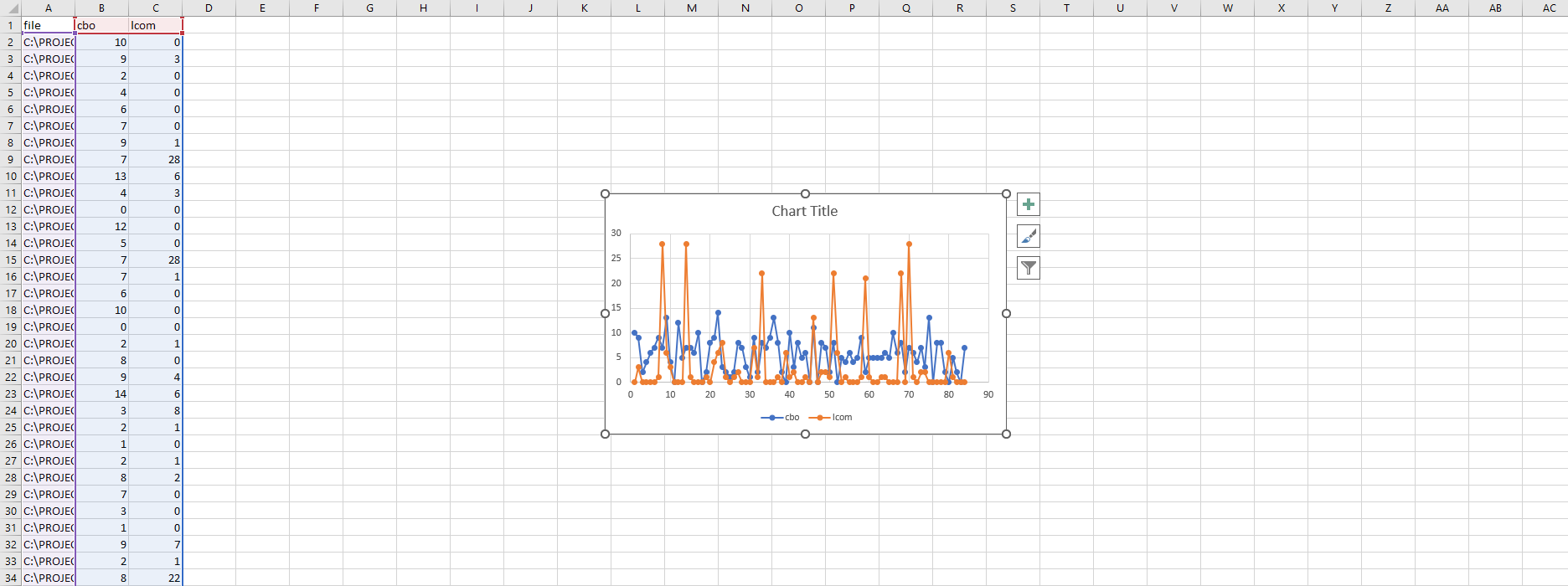
Project3: dropwizard



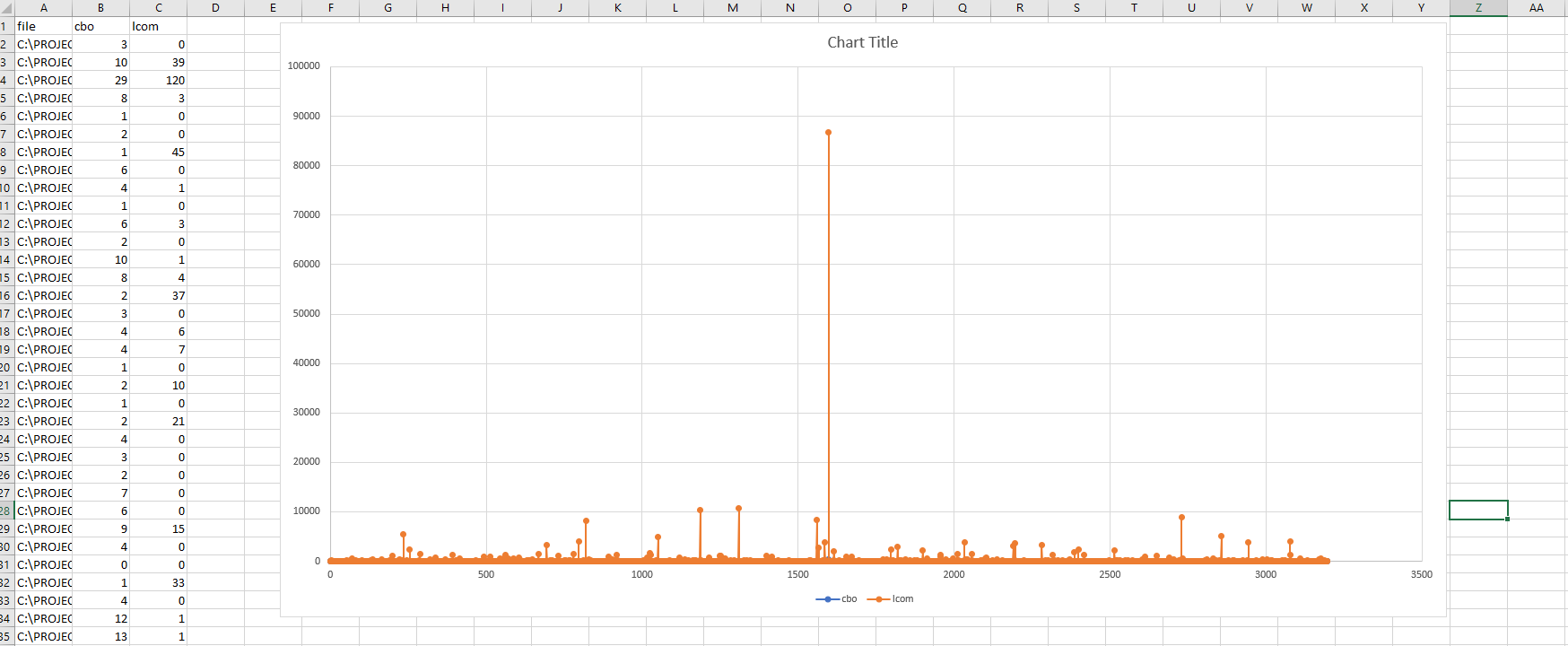
Project4: CoreNLP



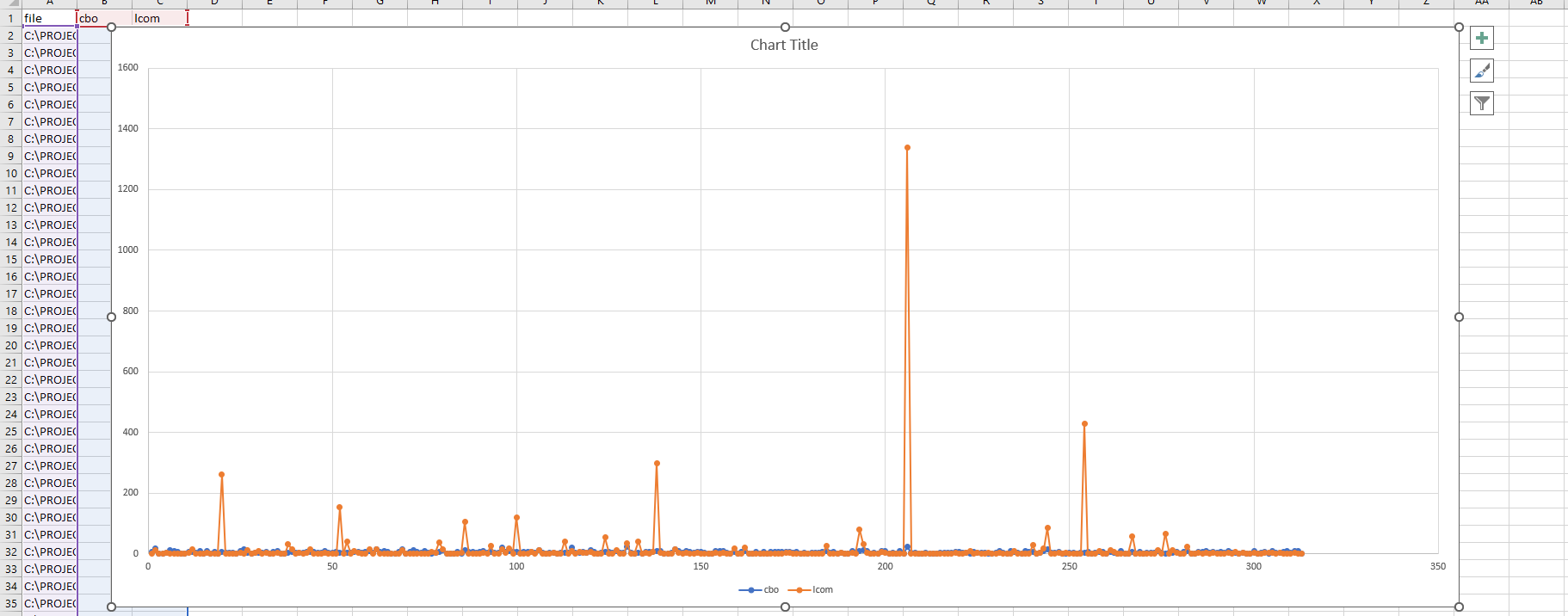
Project5: testing-samples-main



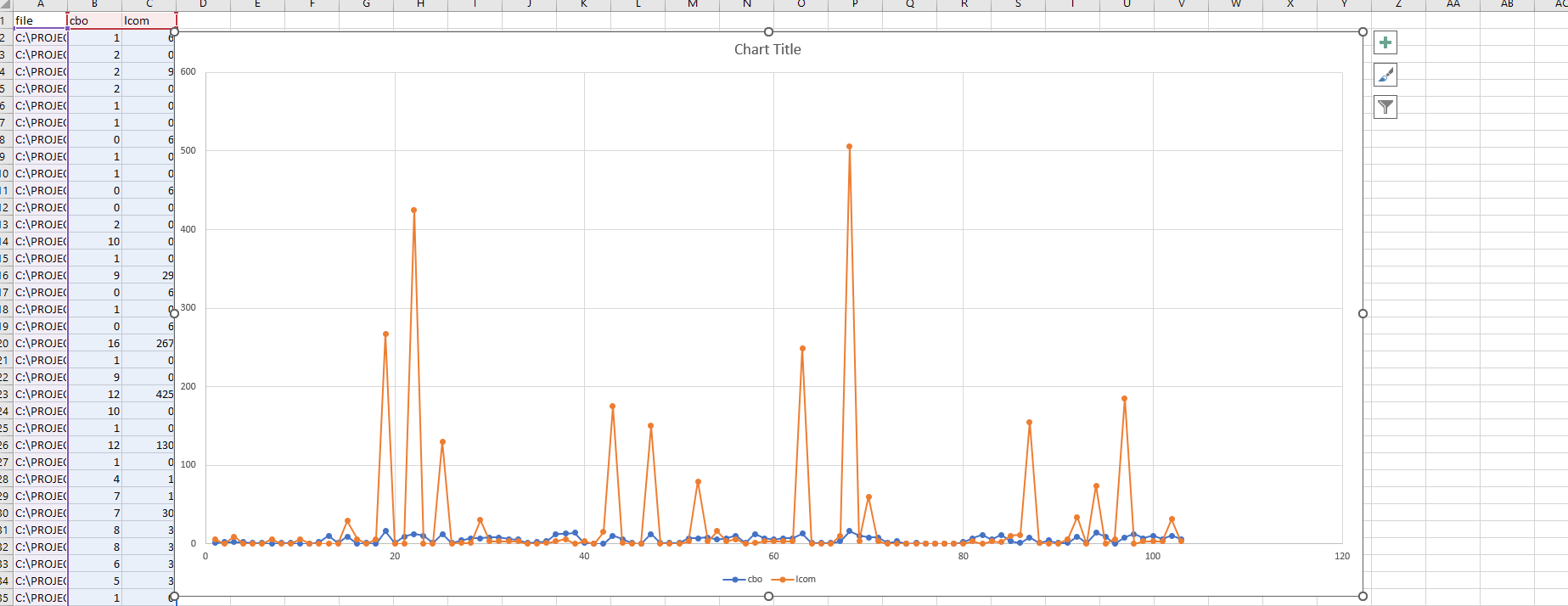
Project6: Activiti-develop



Project7:webmagic



Project 8: MagicIndicator-main



Conclusion:

So, Bad smells attract more defects and makes the project less modular.

High volumes of lcom,cboindictae less modularity.