OBJECT ORIENTED DEVELOPMENT
WEEK 6 - GROUP ASSIGNMENT 2

Submitted By

Uday Vurrinkala

Srinivas Komali

Harshavardhan Pullakandam

Section 1:

Objective

The main objective of this study is to find out the effect of bad smell of code on the modularity of projects and these projects will be based on Java programs.

GQM approach

Goal

The goal of the study is to increase the quality of code in Java projects.

Questions

1. How the modules of Java programs are interacting with the interface correctly?
2. What are the negative impacts of the bad smell of coding which is indicated by various C&K metrics?
3. Mention the properties of class that represent the code's bad smells in a program?

Metrics An evaluation of above question will be conducted as per the following given conditions:

- **Number of inactive class<0**: A program needs to be chosen for searching a single class which cannot inherit values from the global variable of the parent class.
- **Size>10000:** Selection of the program must happen in such a manner that a large and complex program gets selected.
- **Range of bad Smell between 100 and 150:** The range of flaws or bad smell code should be ranged from 100 to 150.

In order to conduct a development activity, the above criterias have been set up so that evaluations can take place with few complexities providing a real time initiative for a programmer.

Section 2:

Ten projects based on Java programs have been chosen from GitHub that can meet the requirement of our evaluation. The characteristics of each program is given in the table below:

Name of Program	Description	Number of Inactive Class	Size	Range of Bad Smell
apollo	Apollo is a reliable configuration management system suitable for microservice configuration management scenarios	10	83037	110
ghidra	Ghidra is a software reverse engineering (SRE) framework	7	20049	105
java-design-patterns	Design patterns implemented in Java	12	19714	115
mall	The project is an e-commerce	8	24740	126

	system consisting of a front-end mall system and a back-end management system, implemented using SpringBoot+My Batis and deployed with Docker containers.			
spring-framework	Spring Framework	9	47428	120

Section 3: Evaluation of the selected tool:

First Tool:

In order to conduct empirical study a second time, we have applied the CK metric tool of the Java program. It has been observed that static analysis is used by the tool so that various metrics of software application can be computed. We have downloaded these CK code metrics from GitHub and we used it by following the instructions given by the author of the ReadMe file. It uses Command Line Interface (CLI) which figures out the status of all the classes used in the Java program. The following command lines have been written below:

java -jar ck-x.x.x-SNAPSHOT-jar-with-dependencies.jar
<project dir>
<use jars:true|false>
<max files per partition, 0=automatic selection>

<variables and fields metrics? True|False>

<output dir>

Second Tool:

We have applied a PMD tool to conduct static code evaluation on our source code of Java program. It is a kind of tool that can be easily downloaded in the local drive and it helps the programmer to recognize the flaws in the Java programming such as bugs or dead code.

We have selected the PMD tool for static code evaluation because it is the most preferred tool in the IT industry and possesses a great reputation for exploring the error as well as providing instruction to mitigate it. The command lines are written as follows:

pmd.bat check -d

<Project Directory>

-f <filetype>

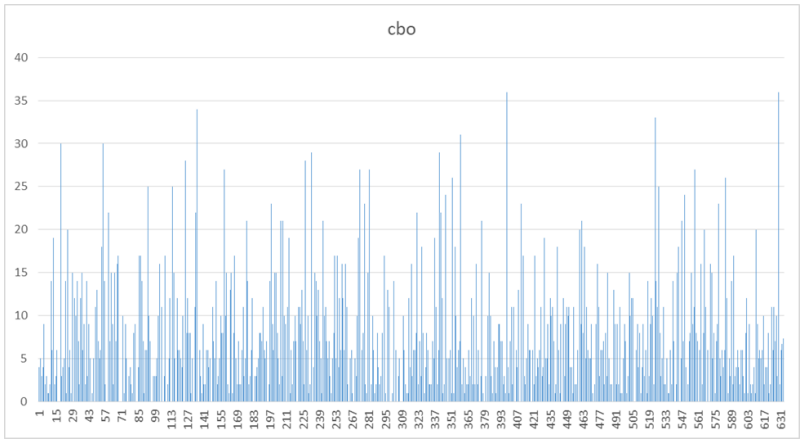
-R <ruleset.xml>

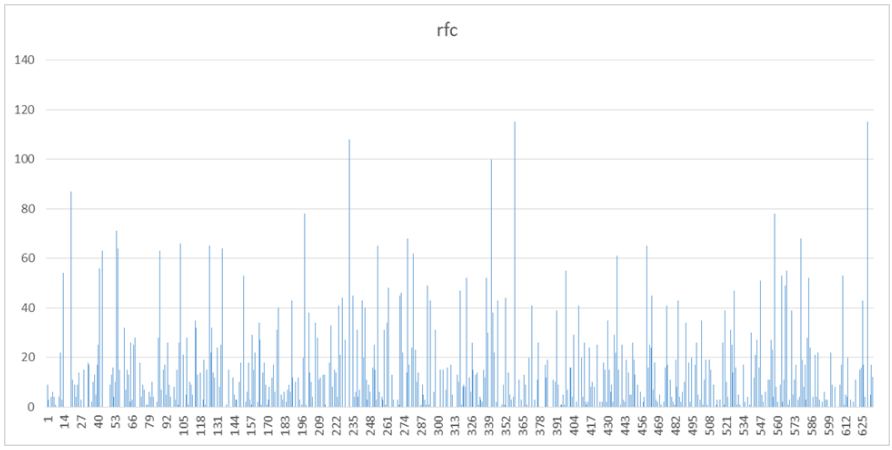
-r <fileName>

Section 4:

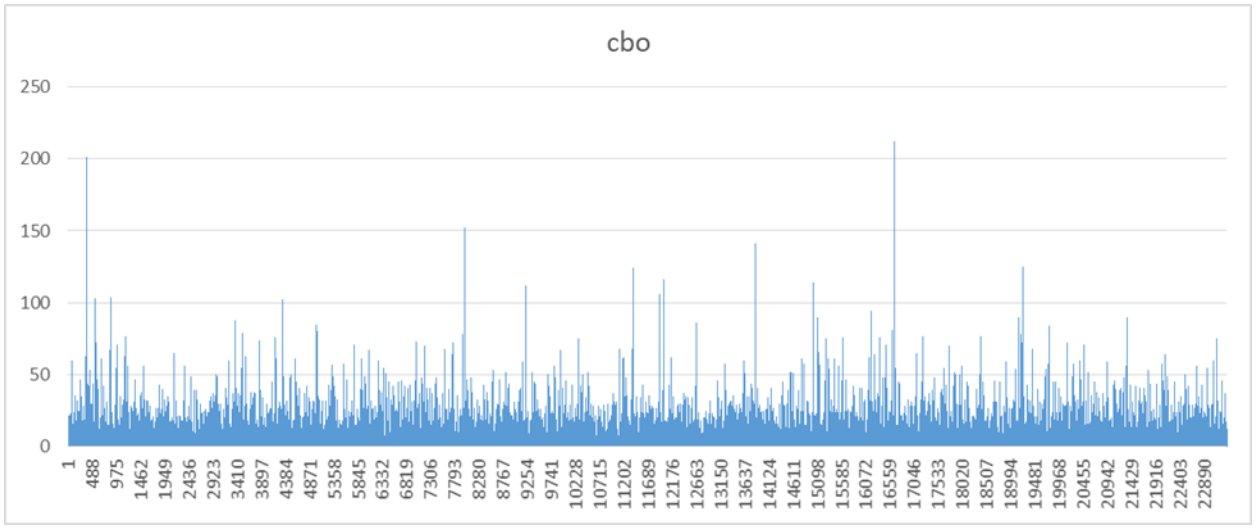
Results:

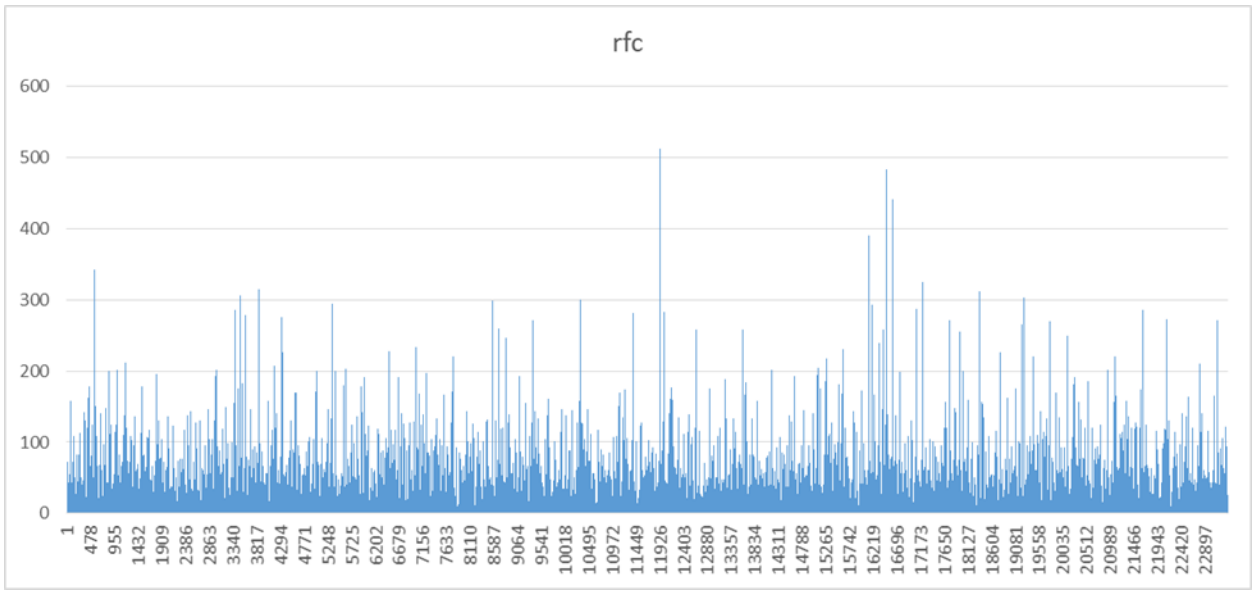
Project 1: apollo



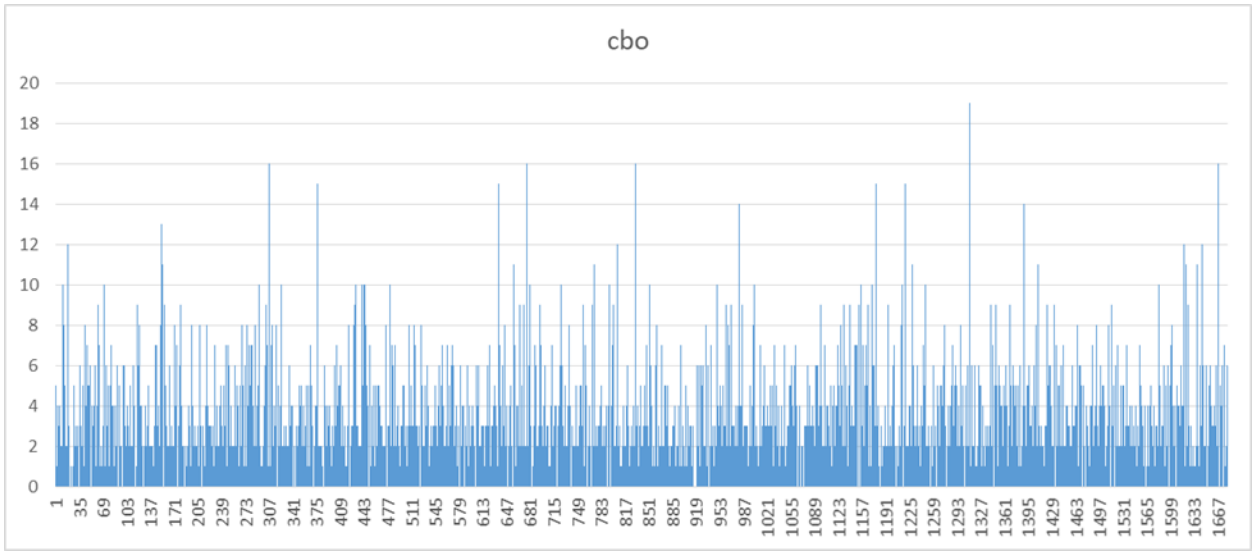


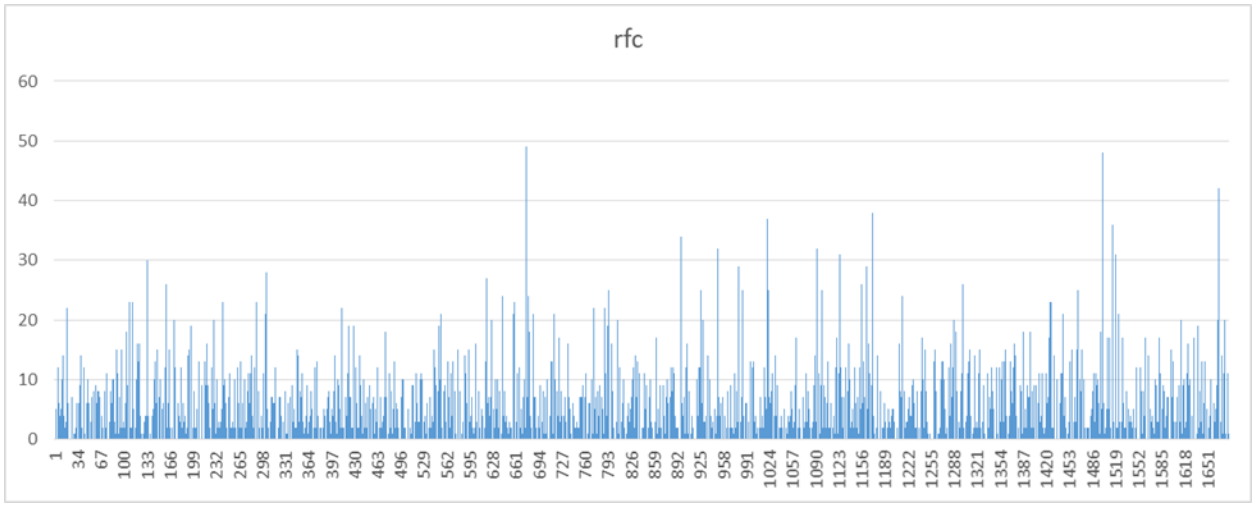
Project 2: ghidra



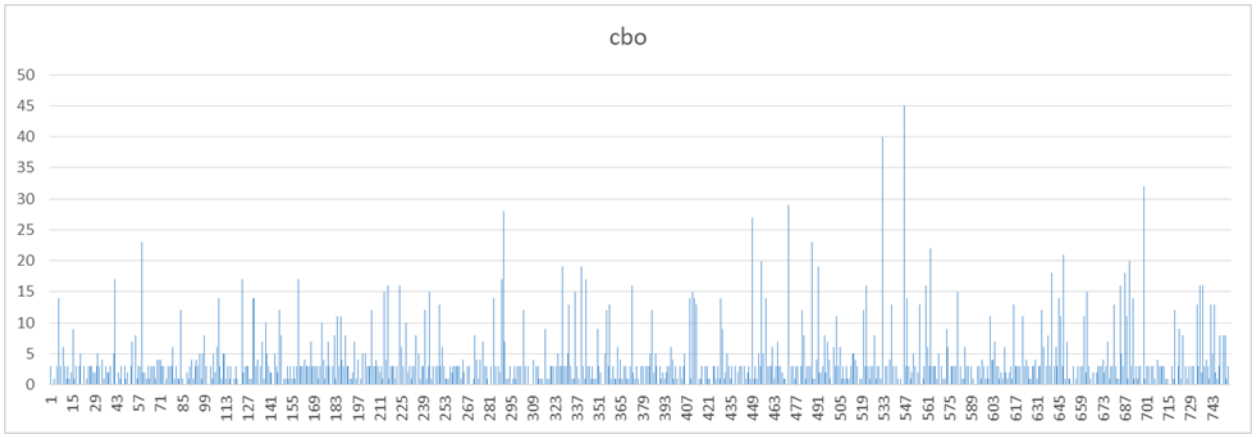


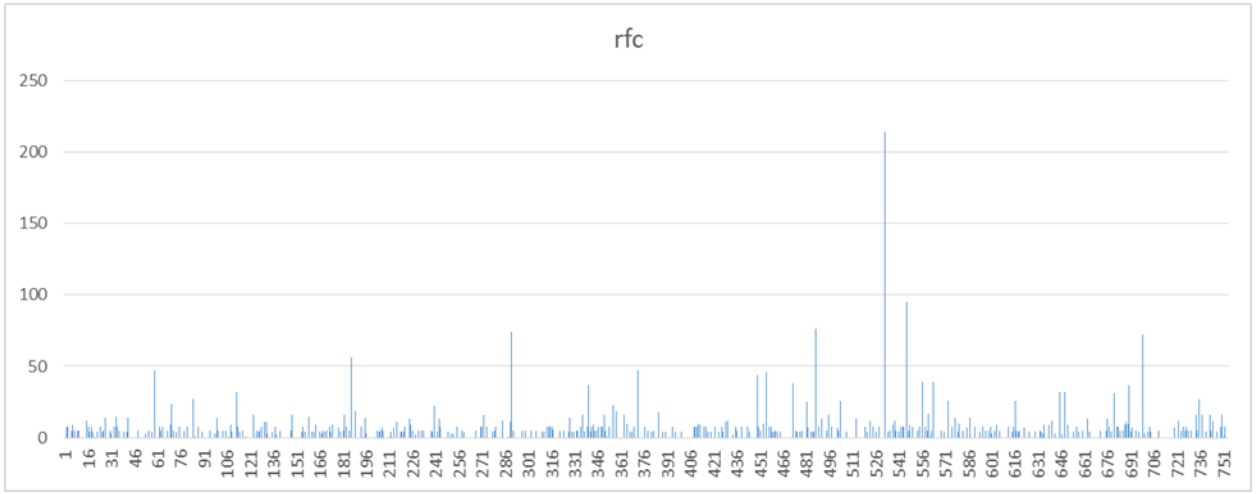
Project 3 : java-design-patterns:-



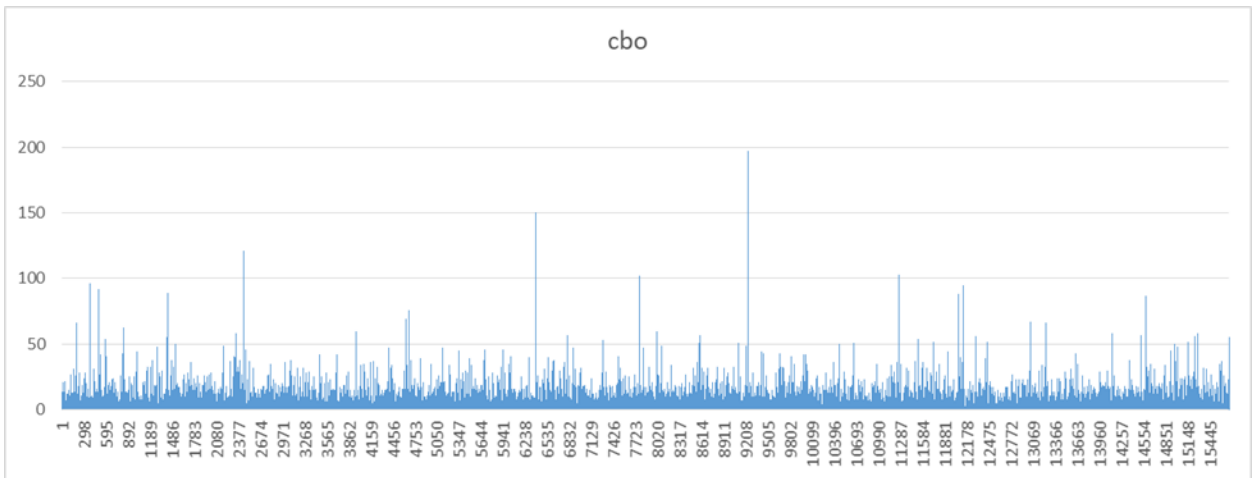


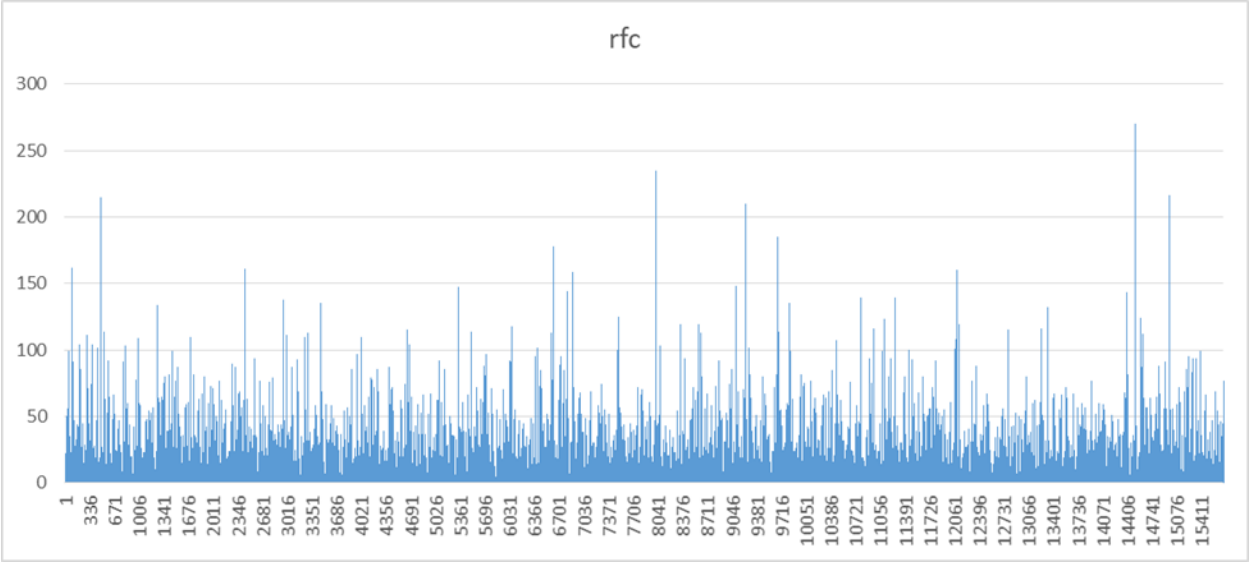
Project 4: mall





Project 5: spring-framework





Section 5:

Conclusions

From the evaluation of the chosen Java program, it could be depicted that the study of the project had been changed frequently in terms of its foul order level. Few projects like UETool have a comparatively lower percentage of classes along with its bad orders. Likewise, other projects like Aisenweibo are being found to be higher in percentage. Collectively, a routine monitoring and rectification of bad smells is emphasized in the data for enhancing the quality of software projects.

Reference

Kurmangali, A., Rana, M. E., & Ab Rahman, W. N. W. (2022, March). Impact of Abstract Factory and Decorator Design Patterns on Software Maintainability: Empirical Evaluation using CK Metrics. In *2022 International Conference on Decision Aid Sciences and Applications (DASA)* (pp. 517-522). IEEE.

Rathee, A., & Chhabra, J. K. (2022). Metrics for reusability of java language components. *Journal of King Saud University-Computer and Information Sciences*, *34*(8), 5533-5551.

Komolov, S., Dlamini, G., Megha, S., & Mazzara, M. (2022). Towards Predicting Architectural Design Patterns: A Machine Learning Approach. *Computers*, *11*(10), 151.