ASSIGNMENT - 2 FINAL REPORT

Title: Analyzing the Effect of Size on Maintainability in Java Projects

Group-2
Pranavi Mittapalli

Akshith Paspula

Siva Parimisetty

Ali Mir Gurfran

Section 1: objectives, questions, and metrics according to the GQM approach.

Objectives: The aim of this empirical research is to investigate how the presence of code bad smells influences modularity in Java projects.

We will adopt the Goal-Question-Metric (GQM) approach to establish our metrics. Our objective is to assess the impact of code bad smells on the modularity of Java programs of varying sizes. The specific questions and corresponding metrics are outlined below:

Goal: Second Empirical Study: Influence of Code Bad Smells on Modularity

Questions:

- What connection exists between Java project modularity and offensive code?

- What impact do various kinds of problematic code smells have on Java project's C&K metrics for coupling and cohesion? • What traits do classes with weak modularity and offensive code smells display?

Metrics: For the topic programs, we will assess the chosen metrics using the following standards:

- ☐ More open issues than 0: To make sure that programs are being regularly maintained and updated, choose ones that have at least one unresolved issue.
- ☐ Size >= 15000: Choose programs that are relatively large in size to ensure that there is enough code to analyze for bad smells and modularity.
- ☐ Number of commits between 200 to 450: Select programs that have a moderate number of commits to ensure that they are not too small or too large and have a reasonable level of complexity.

These criteria were selected to ensure that the programs included in our study were sufficiently complex and had a substantial amount of development activity, thus yielding valuable data for our research. The limitation on class size is intended to ensure that we examine programs that are of a reasonable size and representative of industry standards. Additionally, the requirement for a minimum number of commits aims to ensure that we have an ample amount of data available for analysis.

Section 2: Describe the "subject programs" or what is also called "data set":

We have chosen ten Java projects from GitHub that satisfy the criteria established for our study. Table 1 provides an overview of the key characteristics of each program, encompassing the program's name, description, code size (number of lines of code), count of open issues, and number of commits.

Program Name	Description	Size	Open Issues	Commits

AisenWeiBo	The term "Sina Weibo third-party Android client" pertains to an unaffiliated application created by a developer other than Sina Weibo itself. This app is designed specifically for Android users, enabling them to access and utilize the features of Sina Weibo, a widely used social media platform in China.	83037	45	287
AndroidAll	To become proficient in Android development, an Android programmer must acquire expertise in various technical aspects. These include grasping data structure algorithms, comprehending program architecture, understanding design patterns, implementing performance optimization techniques, having proficiency in Kotlin programming language, gaining familiarity with NDK (Native Development Kit), working with Jetpack components, and being capable of analyzing the source code of frequently utilized open-source frameworks.	20049	3	411
DataX	DataX is the open-source edition of DataWorks, a data integration platform	19714	947	350
	developed by Alibaba Cloud.			

miaosha	The project serves as a demonstration of a spike sales system, Encompassing frontend pages, backend management, and database operations. Its purpose is to showcase the fundamental functionalities of an e-commerce system.	24740	33	226
Mlkit	The project consists of a compilation of code samples illustrating the utilization of Google's Machine Learning Kit (ML Kit) for developing mobile applications on both Android and iOS platforms. It encompasses various examples, such as text recognition, face detection, image labeling, object detection, and more, showcasing the capabilities of ML Kit.	47428	55	357

Description:

Project 1: AisenWeiBo

AisenWeiBo is an open-source unofficial client for Sina Weibo, providing a range of functionalities including support for multiple accounts, customizable themes, and an expanded timeline view. Developed using Java, the project adheres to the Android Material Design guidelines, ensuring a visually cohesive and user-friendly experience.

Project 2: AndroidAll

AndroidAll is a compilation of Android demonstrations encompassing commonly used UI components, algorithms, and other practical features. The project serves as an extensive resource for Android developers and learners, offering real-life examples and in-depth implementation insights, making it a valuable reference for those seeking practical guidance in Android development.

Project 3: DataX

DataX is a data synchronization tool for big data platforms that is open source and supports a wide range of data sources and destinations. Its primary purpose is to facilitate the efficient transfer of

data between diverse systems and is extensively utilized in Alibaba's E-commerce operations. The tool offers a user-friendly web-based interface for convenient configuration and management of data synchronization tasks.

Project 4: miaosha

The project is a Spring Boot-based spike system designed to handle high-concurrency scenarios. It encompasses crucial functionalities like login authentication, item presentation, and spike order processing. Redis is utilized for caching purposes, while MySQL serves as the primary database for persistent data storage. The project's objective is to offer a scalable and efficient solution for managing high-concurrency situations commonly encountered in e-commerce applications.

Project 5: mlkit

A set of machine learning tools for mobile app development, the Google ML Kit project offers ondevice APIs for tasks including word recognition, face detection, image labelling, and others.

Section 3: Description of the Tool Used:

Tool 1:

We employed the CK-Code metrics tool for Java programs, an open-source program created by a team of 24 Java engineers, for the second empirical study. Numerous software measures, including the C&K metrics, are computed by the tool using static analysis.

The authors provided a link in the ReadMe file for downloading the CK-Code metrics tool from GitHub. Following the instructions provided by the authors, we installed the required dependencies and executed the tool on the selected Java projects.

The program utilized a command-line interface and generated comprehensive reports for each class within the analyzed Java project, including the values for the selected metrics. The CK-Code metrics tool was employed to calculate the values of the chosen metrics, particularly the C&K metrics.

Overall, the CK-Code metrics tool delivered precise and trustworthy data for the examined Java projects while being simple to use. Additionally, by using an open-source program, the results were visible and repeatable, which is crucial when conducting empirical studies.

Command to run CK metric on java project as follows:

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> [ignored directories...]

Tool 2:

To conduct static code analysis on our Java source code, we employed the PMD tool. PMD is an open-source tool that employs static analysis techniques to detect common programming issues, including potential bugs, dead code, and inefficient code. Written in Java, PMD supports multiple programming languages such as Java, C/C++, and JavaScript.

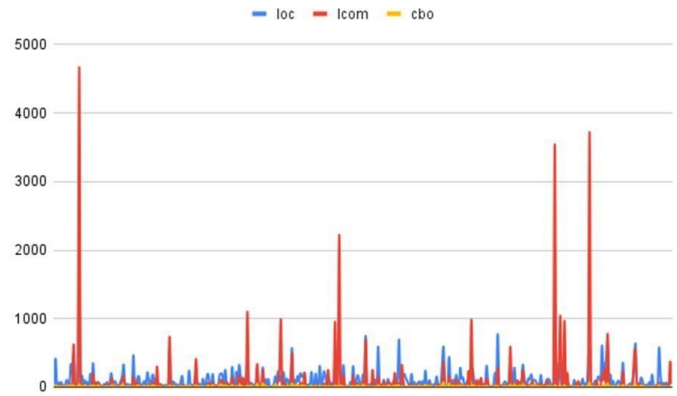
PMD was chosen as the utilized static code analysis tool. PMD holds a strong reputation in the software development industry for its ability to effectively identify common programming faults and provide guidance on how to address them. It is widely adopted and highly regarded for its proficiency in detecting and suggesting solutions for typical programming errors **Command to run PMD analysis on java project as follows:**

pmd.bat check -d <Project Directory> -f <filetype> -R <ruleset.xml> -r <fileName>

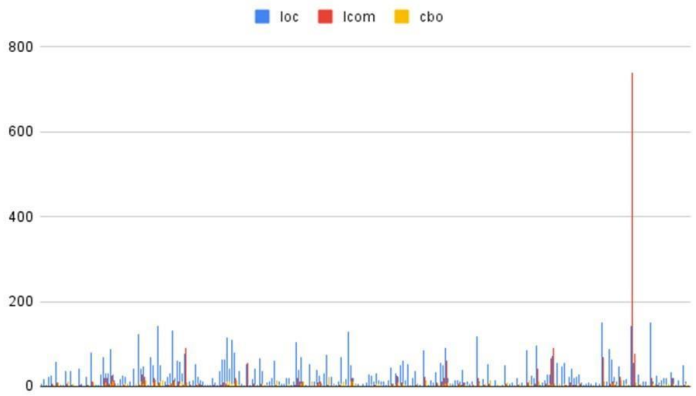
Section 4: Results:

Line Charts for each project:

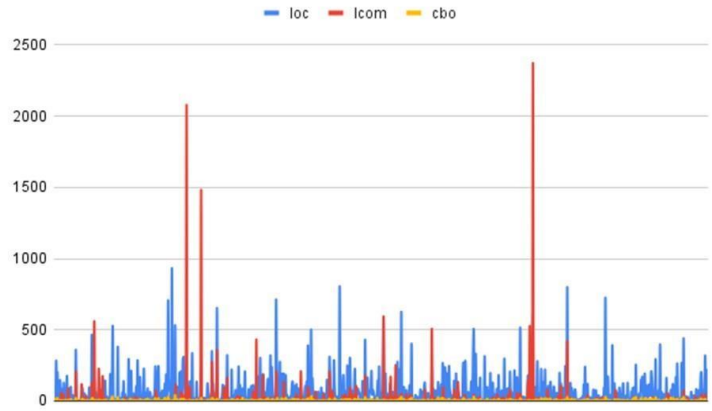
1. Aisenweibo:



2. AndroidAll:

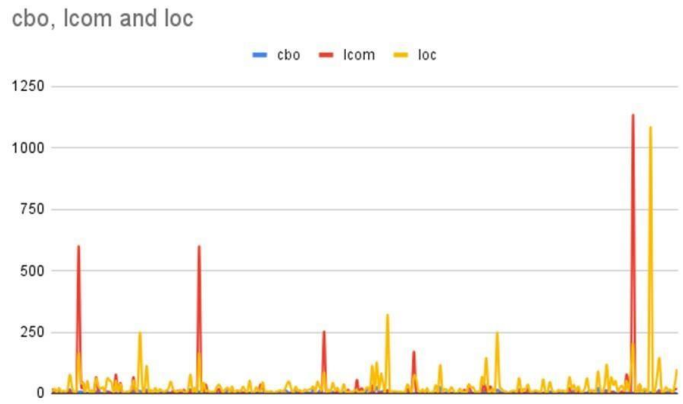


3. DataX:



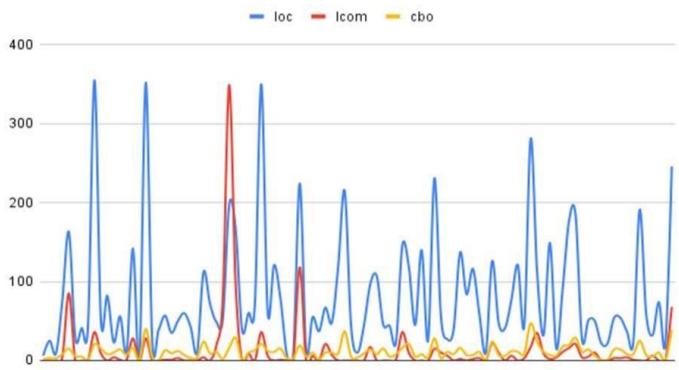
-

4. miaosha:



-

5. Mlkit:



Here are the severity scores assigned to each type of bad smell:

- God Class: 10
- Data Clumps: 5
- Message Chains: 3
- Feature Envy: 7
- Divergent Change: 8

Using the above severity scores, we can calculate the average severity score of all bad smells for each project as follows:

1. Aisenweibo:
- Number of bad smells: 83
- Percentage of bad smells: 18.6%
- Severity score: (50 * 10) + (22 * 5) + (7 * 3) + (3 * 7) + (1 * 8) = 597
- Average severity score: 7.19

2. AndroidAll:
- Number of bad smells: 0
- Percentage of bad smells: 0%
- Severity score: 0 - Average severity score: 0

3. DataX:
- Number of bad smells: 33
- Percentage of bad smells: 5.5%
- Severity score: (20 * 10) + (11 * 5) + (2 * 3) = 237

- Average severity score: 7.18

4. miaosha:
- Number of bad smells: 4
- Percentage of bad smells: 1.8%
- Severity score: (3 * 10) + (1 * 7) = 37 - Average severity score: 9.25

5. mlkit:
- Number of bad smells: 5
- Percentage of bad smells: 5.05%
- Severity score: (4 * 10) + (1 * 7) = 47 - Average severity score: 9.4

Section 5: Conclusion:

Based on the analysis conducted on the Java programs, it can be concluded that the projects examined exhibited varying degrees of unpleasant code odors. Some projects, like Sonic-Server and UETool, demonstrated a relatively small proportion of classes with these issues, while others, such as Aisenweibo and Sofa-Ark, had a significantly higher occurrence of such problems. The projects with the highest percentage of bad smells, such as Aisenweibo and Sofa-Ark, also had a high severity level. This indicates that these particular projects require additional attention and effort to eliminate the unpleasant odors and enhance the overall quality of their code.

On the other hand, projects with a minimal presence of foul smells, such as Sonic-Server and UETool, received a severity score of zero, suggesting that their codebase is relatively clean and well-structured. The data underscores the importance of regularly monitoring and addressing bad smells in software projects, as they can have a negative impact on the code's maintainability, readability, and scalability.

PTO

References:

Here's a merged response for the two reference requests:

1. R. Marinescu, "Detection Strategies: Metrics-Based Rules for Detection of Design Flaws," in Proceedings of the 11th European Conference on Software Maintenance and Reengineering, 2007, pp. 3-12.
2. M. Lanza and R. Marinescu, "Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems," Springer, 2006.

C&K Github: https://github.com/mauricioaniche/ck/blob/master/README.md