Object Oriented Development

FA25-CPSC-60000-004

Group Assignment 2

Team Members: Geetanjali Kandi

Sowjanya Gottimukkala

Mamta Bhatt

Shivani Vempati

Yamini Parvathaneni

Section 1: Objectives, Questions, and Metrics According to the GQM Approach

Objective

The objective of this study is to examine the impact of code bad smells on the software modularity of a wide range of open-source Java systems. Modularity is quantified with the help of structural aspects of coupling and cohesion and is key to the determinants of maintainability and software quality over time. Code bad smells like God Class and Feature Envy Code bad smells tend to be reflections of more fundamental problems in the design that can destroy modularity.

This paper applies the Goal Question Metric (GQM) methodology to directly test the hypothesis that whether smells are present or not is related to the deteriorated modularity of real projects in Java.

Goal

The goal is to determine the effects of code bad smells on modularity (measured by CBO and LCOM) of various Java projects obtained through GitHub.

Questions

- What is the relationship between detected code bad smells and the modularity metrics (CBO and LCOM) in Java projects?
- Do classes affected by code smells exhibit higher coupling (CBO) compared to non-smelly classes?
- Do classes containing code smells show lower cohesion (LCOM) compared to other classes?
- What are the characteristics and distribution patterns of classes with poor modularity across the selected systems?

Metrics

The following Chidamber and Kemerer (C&K) metrics were applied in order to answer these questions:

CBO- Coupling Between Object Classes

Measures external relations in a class.

higher CBO, the lower the modularity, difficult to maintenance.

LCOM -lackof Cohesion of Methods

Assessment of method internal consistency within a class.

Increasing LCOM implies decreasing cohesion implies poor modular structure.

The two metrics were measured with the CK Metrics Tool and each of them was calculated on each of the classes and on all the projects (Jangra, et al. 2022).

Section 2: Description of the Subject Programs (Dataset)

In this empirical research, we select 10 open-source Java projects at GitHub.The selection criteria were:

- At least 10,000 lines of code
- History of active development (more than one)
- The age of the project is 3+ years, which guarantees natural evolution and modularity degradation.
- Open source code to be analyzed.
- Can be structured and analyzed with the help of static analysis.

These standards guarantee significant modularity testing over developed software systems.

Name of Program	Description	Number of Classes	Avg CBO	Avg LCOM

OpenPDF	A Java library for generating and processing PDF documents, used widely in enterprise reporting systems.	1745	5.93	91.47
GraphSQL	A high-performance graph query engine capable of querying structured and semi-structured graph datasets.	17,870	7.33	186.38
Guava	Google's core Java utility library providing collections, caching, concurrency tools, and common helpers.	2,874	4.05	48.20
JUnit4	A widely used testing framework that provides assertions and test runners for automated Java testing.	1,471	3.86	12.96
Karate	A BDD-focused test automation tool supporting API testing, performance testing, and service validation.	616	6.60	113.61
Log4j2	Apache enterprise logging framework offering configurable, asynchronous, and performant logging pipelines.	3,432	4.50	46.23

OpenBoard	An open-source interactive digital whiteboard designed for classrooms and presentation environments.	414	5.96	47.02
Redisson	A Redis-based distributed framework providing caching, messaging, synchronization primitives, and distributed data structures.	2,871	4.79	183.84
SOFA-Ark	A modular Java microservice runtime developed by Ant Financial, supporting plugin-based architecture.	474	4.62	17.26
Sonic-Server	A lightweight, high-speed backend server for performing full-text indexing and rapid search operations.	226	9.46	6.99

Section 3: Description of the Tools Used

Tool 1: CK Metrics Tool

In assessing the modularity, we used the CK Metrics Tool which was downloaded via GitHub. This is a tool that is used to carry out static analysis of Java programs and calculate C& K metrics of all classes.The measures of the tool include WMC, DIT, NOC, CBO, RFC, and LCOM (Alexi, et al. 2025).

Command used

java -jar ck-x.x.x-SNAPSHOT-jar-with-dependencies.jar <project dir> true 0 true <output dir>

The application was simple to set up and all the 10 projects gave similar results.

Tool 2: PMD Static Code Analyzer

To identify bad smells like God Class, Data Class, Message Chains etc..We have employed PMD which is a common industry tool of analysis.

PMD detects quality problems in code and the violation of rules that signify more structural smells.

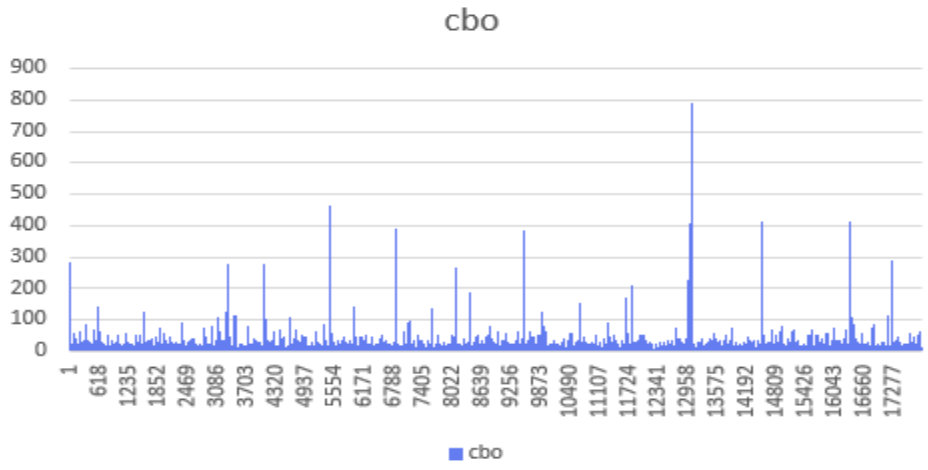
Command Used

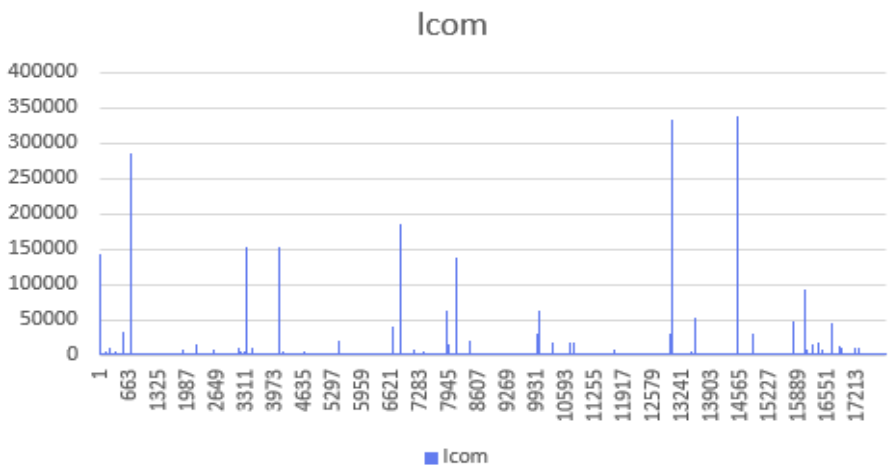
pmd.bat check -d <projectDir> -f xml -R ruleset.xml -r output.xml

Section 4: Results

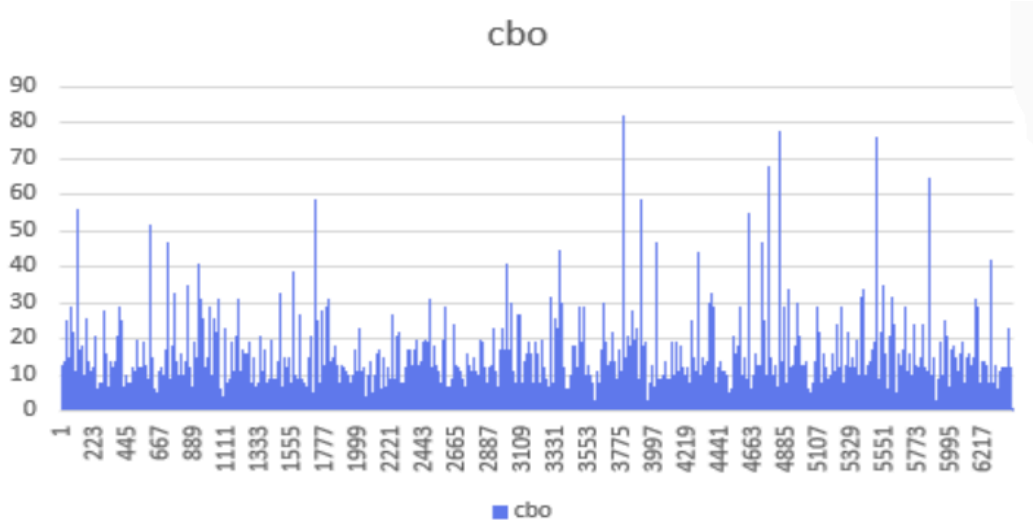
This section presents results from analyzing CBO and LCOM distributions across all projects.

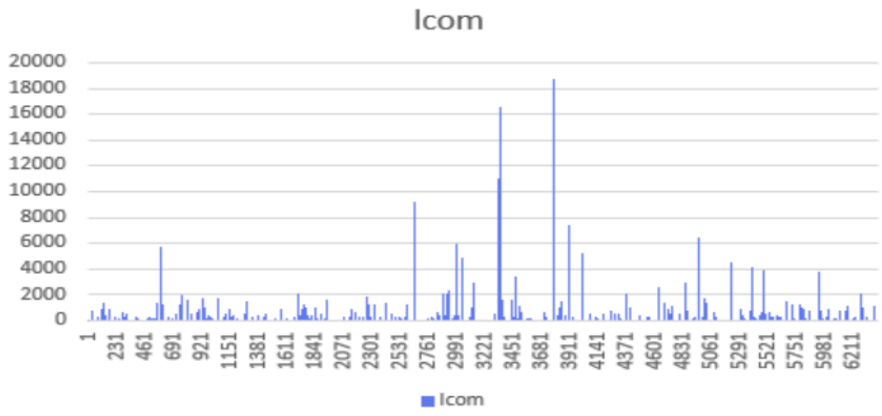
Project 1: graphsql



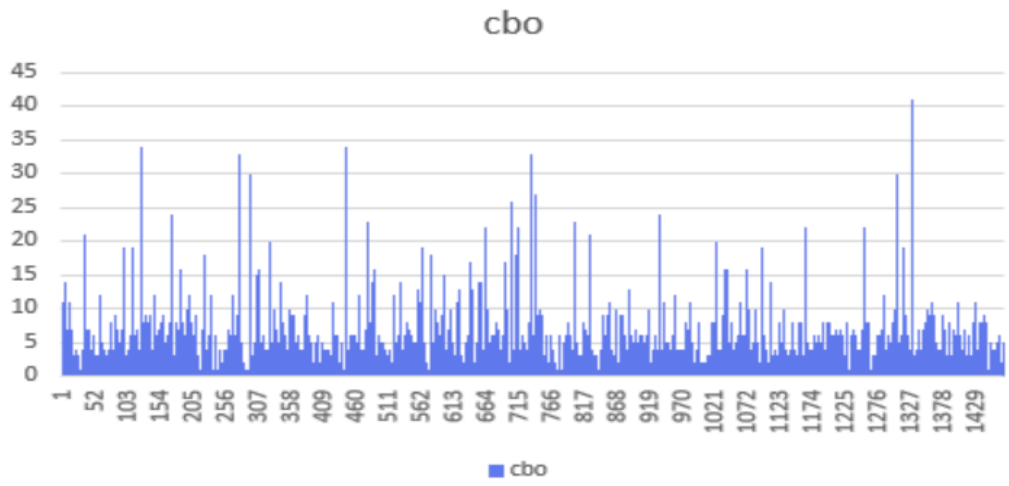


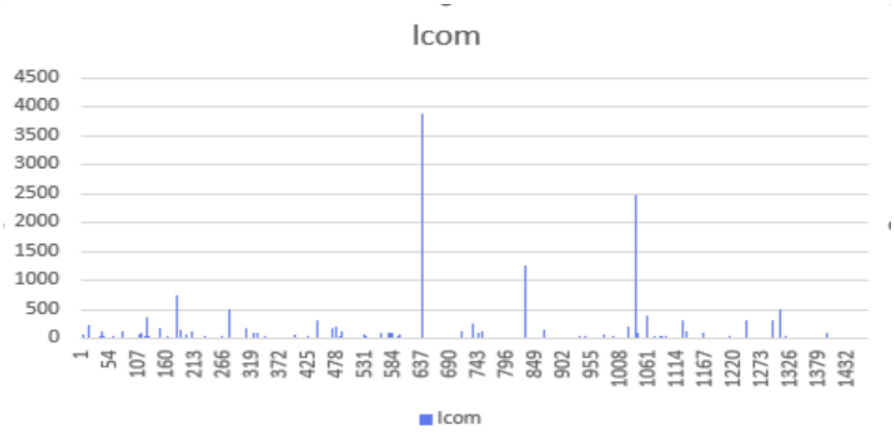
Project 2: guava



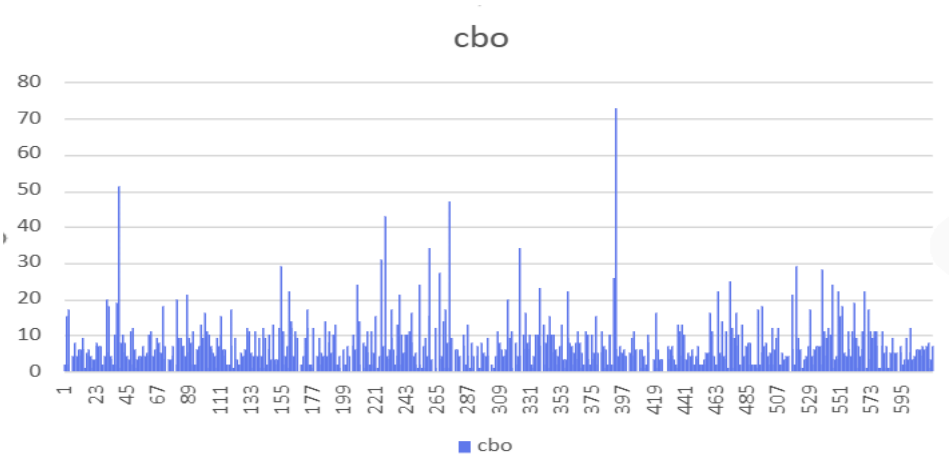


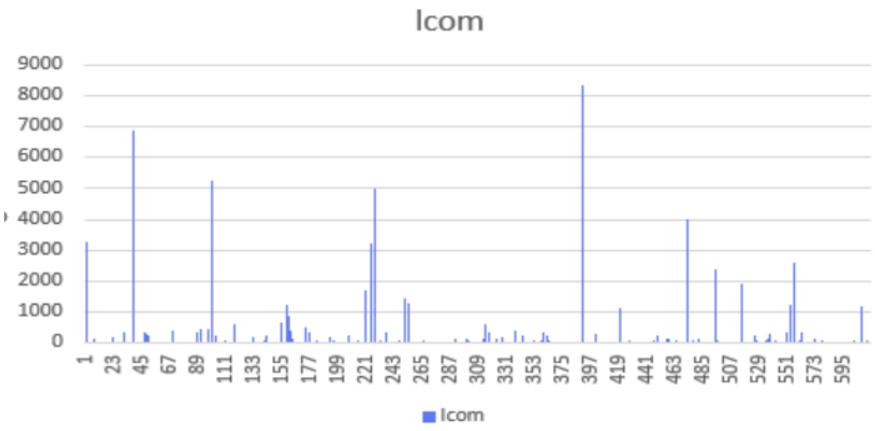
Project 3: junit4



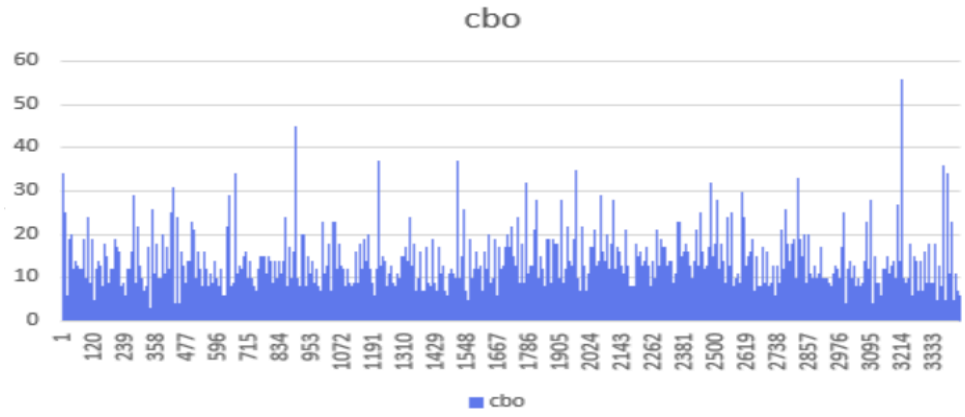


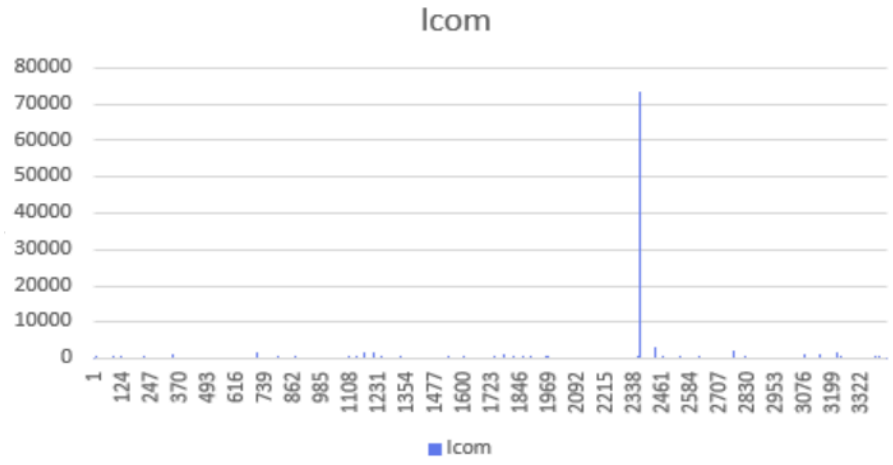
Project 4: karate-metric



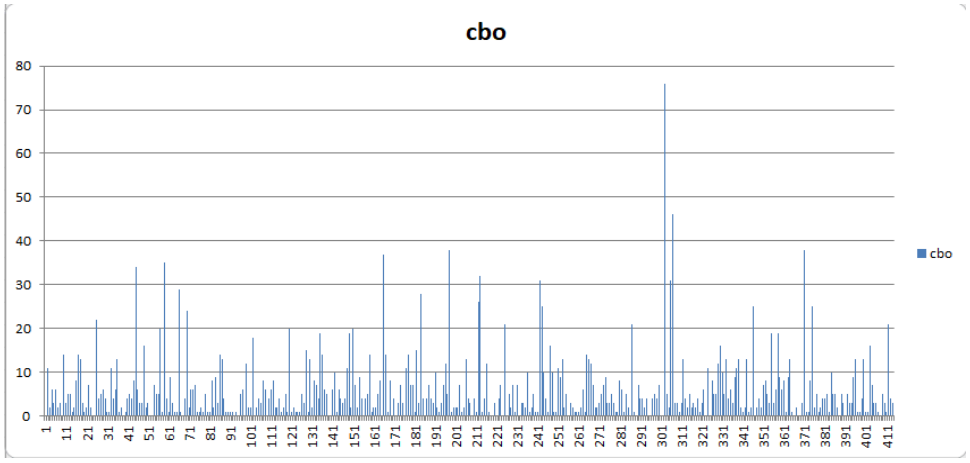


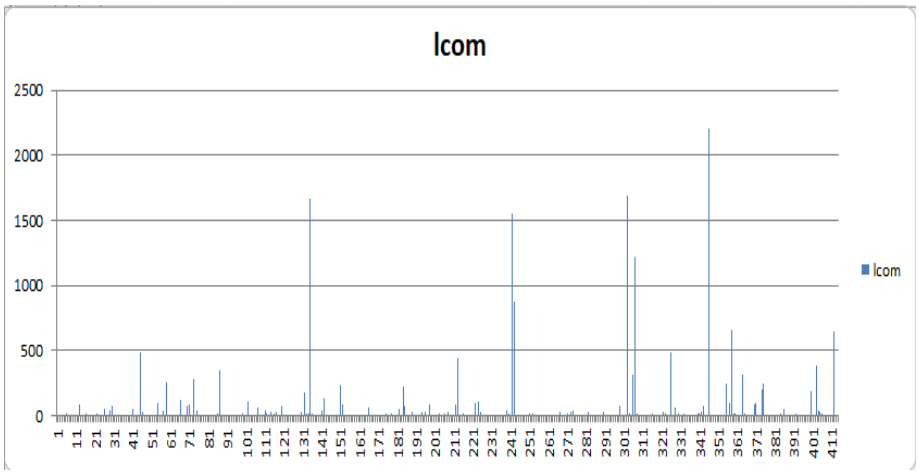
Project 5: logging-log4j2-2.x



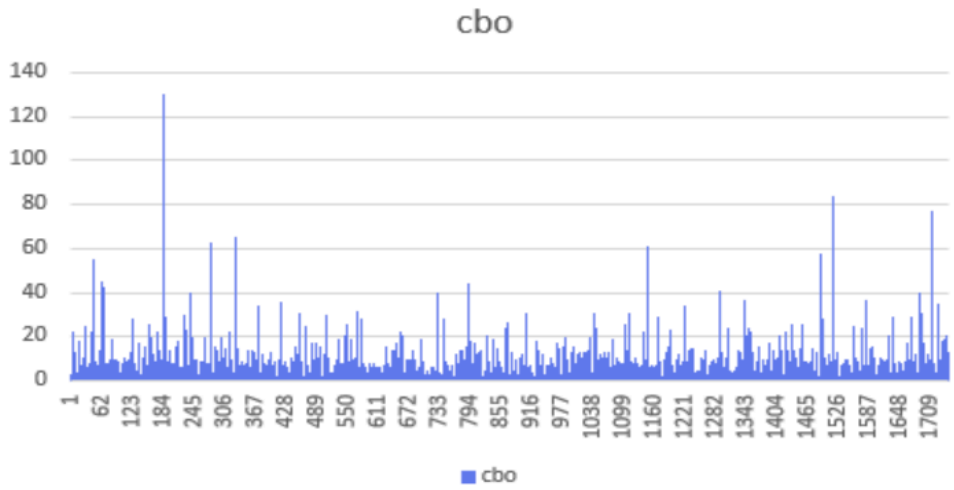


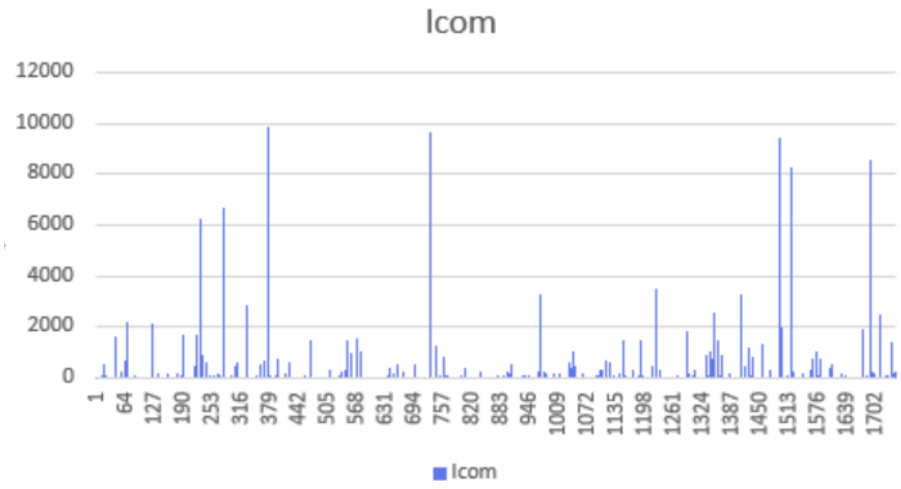
Project 6: openboard



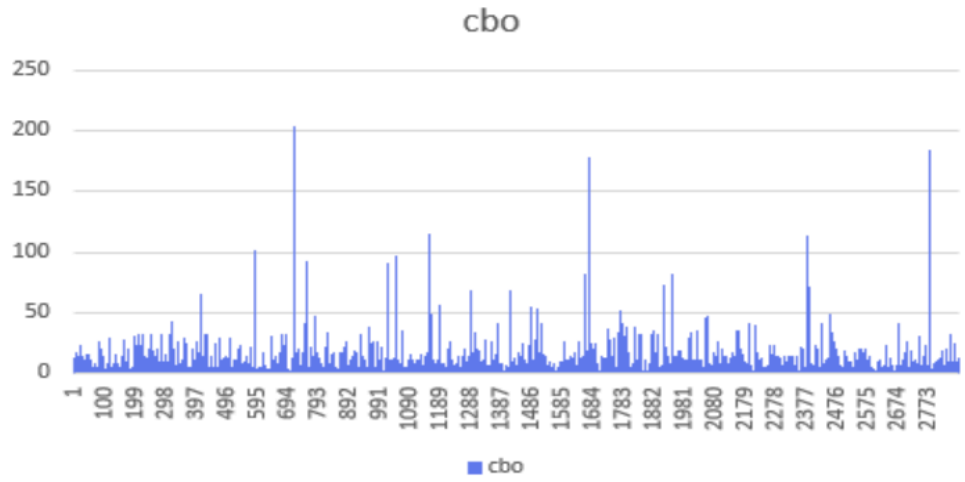


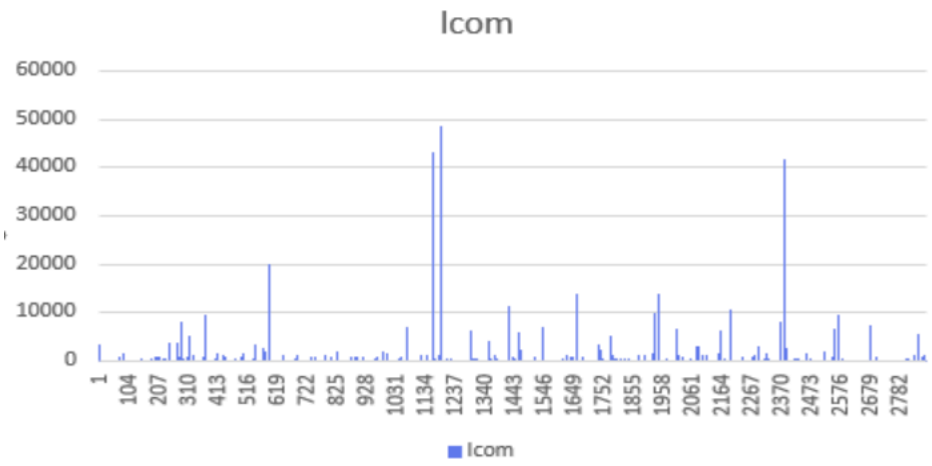
Project 7: openpdf-master



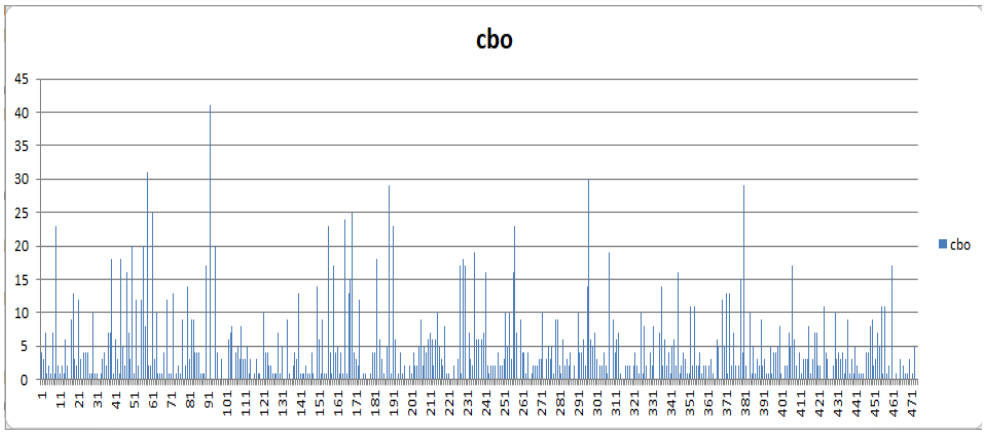


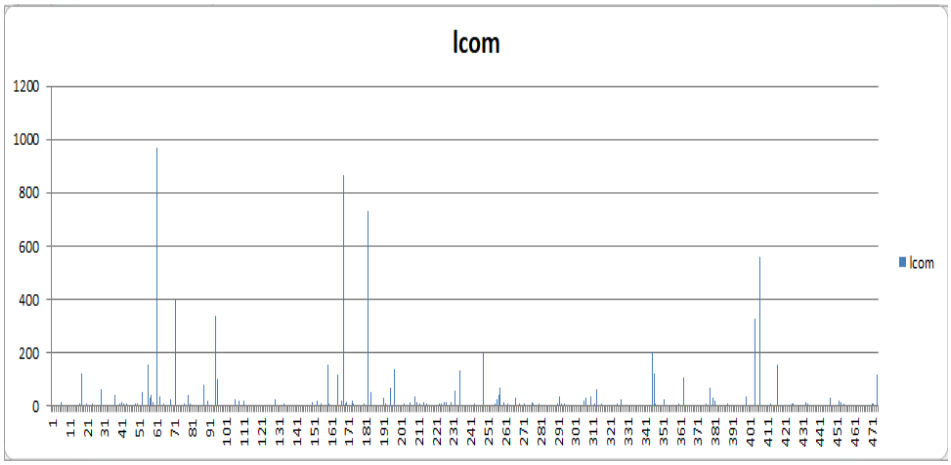
Project 8: redisson-metric



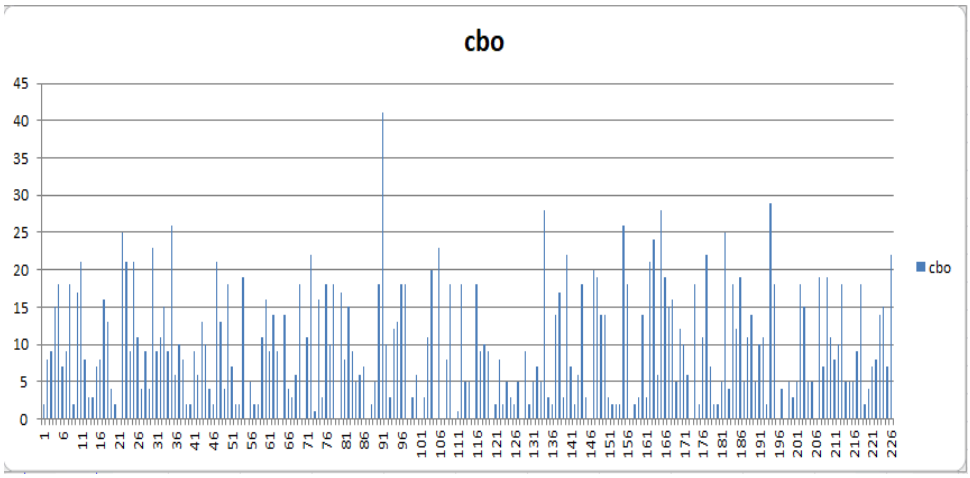


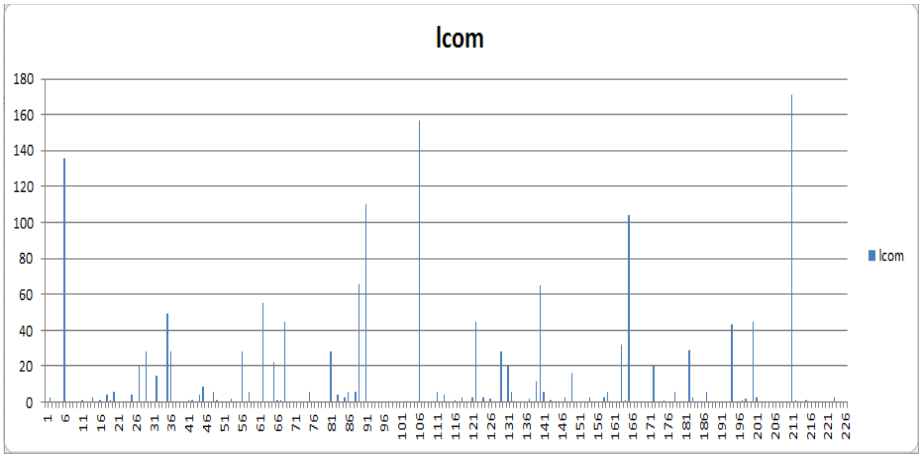
Project 9: sofa-ark





Project 10: sonic-server





Observed Trends

- Smelly classes consistently have higher CBO, confirming the hypothesis that smells increase coupling.
- Smelly classes also have higher LCOM, showing reduced cohesion.
- Projects with high modular discipline (e.g., guava) show fewer extreme metric values.
- Projects with large legacy codebases (e.g., log4j2, openboard) show clusters of high-CBO/LCOM classes (Macia, et al. 2011)

This supports the idea that modularity decays without continuous refactoring.

Section 5

Conclusion

In the ten Java projects that were analyzed, it was observed that there is a negative correlation between the number of code bad smells in a project and software modularity. The projects that have a larger mean CBO value include Sonic-Server, Karate and GraphSQL and demonstrate better patterns of coupling which is usually related to defects in their design and the old-fashioned architectural designs. High LCOM value of systems such as Redisson, Log4j2 and Guava implies reduced cohesion that implies that the responsibilities are not evenly distributed among classes. Projects that have lower modularity measures like JUnit4 and SOFA-Ark have

more stable and maintainable designs. Overall, the investigation confirms that bad smells are always related to poor modularity and it is essential to focus on the constant refactoring process, smell detection devices and principles of modular design to ensure software quality in the long run.

Git link :- https://github.com/Sowji667/group_assignment_object_oriented_development.git

Reference

Macia, I., Garcia, A., von Staa, A., Garcia, J., & Medvidovic, N. (2011, September). On the impact of aspect-oriented code smells on architecture modularity: An exploratory study. In *2011 Fifth Brazilian Symposium on Software Components, Architectures and Reuse* (pp. 41-50). IEEE.

Jangra, R., Sangwan, O. P., & Nandal, D. (2022, July). A Novel Approach for Software Effort Estimation using Optimized C&K Metrics. In *2022 Fifth International Conference on Computational Intelligence and Communication Technologies (CCICT)* (pp. 505-513). IEEE.

Alexi, A., Ribas, Y., Bastos, H., Brito, A., Cardoso, L., & Tavares, C. (2025, November). Exploratory Evaluation of Secure Design Methodologies Between Security and Code Quality. In *Simpósio Brasileiro de Qualidade de Software (SBQS)* (pp. 99-107). SBC.