Group Assignment 2

Object Oriented Development

Group 9

Abhiram Atmuri

Neel Chandan Shaik

# Section-I

Objectives: This research project is designed to explore how code bad smells affect the structural modularity of Java-based software applications. We will utilize the Goal-Question-Metric (GQM) framework to define our key performance indicators, aiming to thoroughly understand the influence of code bad smells on Java software structures across various project sizes.

Primary Goal: Conduct an empirical analysis to ascertain the impacts of code bad smells on software modularity.

Research Questions:

1. How do code bad smells correlate with the modularity in Java-based projects?
2. In what ways do different types of code bad smells affect the C&K metrics, specifically focusing on coupling and cohesion within Java projects?
3. What characteristics are commonly observed in Java classes that exhibit severe bad smells and low modularity?

Metrics for Evaluation: Our research will focus on Java projects that meet specific criteria to ensure the reliability and relevance of our findings:

* Number of Open Issues: We will include projects with at least one open issue to guarantee they are under active development and maintenance.
* Size Criteria: Only projects exceeding 10,000 lines of code will be selected to ensure a comprehensive analysis of bad smells and modularity due to their substantial codebase.
* Commit Threshold: We will consider projects with more than 100 commits to balance the complexity and the amount of development activity without overwhelming the analysis.

These selection criteria are chosen to make sure the Java projects studied are neither too simplistic nor excessively complex, providing a balanced view of active, sizeable development environments where code bad smells and their effects on modularity can be effectively observed and analyzed. The emphasis on projects with a significant number of commits and ongoing issues ensures that our data reflects active development scenarios typical in professional settings.

# Section-II

**About Data**

|  |  |  |  |
| --- | --- | --- | --- |
| **Project** | **Description** | **Open Issues** | **Lines of Code** |
| Spring Integration | Tools for supporting enterprise integration patterns. | 121 | 230,794 |
| Brida | A Burp Suite Extension that acts as a bridge between Burp Suite and Frida. | 14 | 7,649 |
| React Native Date Picker | A date picker component for Android and iOS to select dates, times, or both. | 20 | 5,483 |
| EssentialsX | A suite of essential tools for Minecraft server management. | 178 | 46,029 |
| GeoTools | An open source Java library that provides tools for geospatial data. | 10 | 1,070,003 |
| Applied Energistics 2 | A Minecraft Mod about matter, energy, and using them to conquer the world. | 182 | 111,634 |
| Java Tutorial | A comprehensive collection of Java learning materials. | 9 | 15,938 |
| Mantis | A platform for building real-time, cost-effective, operations-focused applications by Netflix. | 35 | 93,671 |
| Eclipse JDT Language Server | A Java language server implementation based on Eclipse's JDT. | 358 | 97,065 |

**Spring Integration**

The GitHub repository for Spring Integration currently has 120 open issues, indicating active maintenance and development. The project has seen a considerable amount of activity with a total of 11,751 commits, which suggests a robust level of complexity and evolution over time. This data underscores the project's substantial development efforts and makes it a potentially valuable dataset for analyzing modularity and code smells, as it meets the criteria of having numerous commits and open issues.

**GATK**

The GitHub repository for the Genome Analysis Toolkit (GATK) from the Broad Institute includes a range of tools for genomic data analysis. The current version, GATK4, is open-source under the Apache 2.0 license and is designed to run both locally and in cloud environments using Apache Spark. This version consolidates tools from previous GATK and Picard releases and introduces new tools as well. The repository has over 1.2k issues and 136 pull requests, indicating active development and community engagement.

**Brida**

Brida is an innovative Burp Suite Extension designed to act as a bridge between Burp Suite and Frida, enabling the manipulation of application methods directly while intercepting the traffic between applications and their back-end services. This tool is particularly useful in mobile application security assessments, especially when dealing with complex security features such as encryption, obfuscation, and signature routines that are implemented within mobile apps. Brida leverages Frida's capabilities to dynamically inspect and modify the execution of application code, thus facilitating deep security testing without the need for extensive manual setup or reverse engineering.

Brida supports all platforms compatible with Frida, including Windows, macOS, Linux, iOS, Android, and QNX. It simplifies tasks such as decrypting or modifying encrypted traffic which otherwise would require significant effort to reverse engineer and replicate. The tool provides an integrated environment within Burp Suite, offering features like an integrated console for output from Frida hooks, a JavaScript editor for writing custom scripts, and graphical representation of binary components to ease the addition of inspection and tampering hooks.

**React Native**

The React Native Date Picker is a versatile component for Android and iOS that allows users to select dates, times, or both from a customizable interface. This component is designed to integrate seamlessly into React Native applications and supports various modes including date, time, and datetime selections. It's especially noted for its adaptability to different languages and its native codebase, which enhances its performance and appearance on mobile devices.

The project is actively maintained on GitHub, where it has gathered significant interest with a substantial number of stars, indicating a robust user and contributor base. This suggests that the component is widely used and well-regarded within the React Native community. It's also part of the broader ecosystem of React Native components that facilitate mobile app development by providing ready-to-use, customizable UI elements that conform to native standards.

**EssentialsX**

EssentialsX is a comprehensive suite of essential tools and enhancements for Spigot servers, building on the legacy of the original Essentials plugin. It's tailored to support the latest versions of Minecraft and brings a myriad of new features, performance improvements, and fixes that were not present in the original Essentials or Spigot-Essentials.

This project is particularly valuable for server administrators seeking to enhance their Minecraft server with features like player kits, spawn and home commands, economy features, chat formatting tools, and more. The active maintenance and development visible through its GitHub repository, including recent updates and community discussions, signify a robust, evolving tool that is responsive to community needs and Minecraft updates.

**GeoTools**

GeoTools is an open-source Java library designed for geospatial data processing. It supports various formats and projections and is compliant with Open Geospatial Consortium (OGC) standards. The project is actively maintained, with a focus on providing robust tools for geographic data management and manipulation, suitable for a wide range of GIS applications. GeoTools is licensed under LGPL-2.1, which allows for versatile usage and integration into different projects.

**Applied Energistics**

Applied Energistics 2 is a Minecraft mod focused on revolutionizing in-game inventory management through a unique system that converts and stores matter as energy. This allows for sophisticated automation and storage solutions within the game. The mod is highly popular in the Minecraft community for its deep integration of energy and matter conversion systems, enhancing gameplay by facilitating complex storage networks and automation setups. It is actively maintained with numerous contributions from the community, ensuring it stays updated and functional with current game versions​.

**Java Tutorial**

The Java Tutorial hosted on GitHub by dunwu is a comprehensive collection of Java learning materials gathered over a decade of experience in the field. It's aimed at both beginners and experienced programmers, providing a wide range of Java-related content from core basics to advanced topics like JVM internals and concurrent programming. The tutorial is maintained across platforms like GitHub and Gitee and is available as an e-book on GitHub Pages.

**Mantis**

Mantis is a platform developed by Netflix designed to simplify the creation of real-time, cost-effective, operations-focused applications. It provides developers with the tools needed to efficiently process and monitor data streams in real-time, making it easier to build and manage complex applications that require high throughput and low latency. The project is open-source under the Apache 2.0 license, fostering a community of contributors who enhance its capabilities and adaptability.

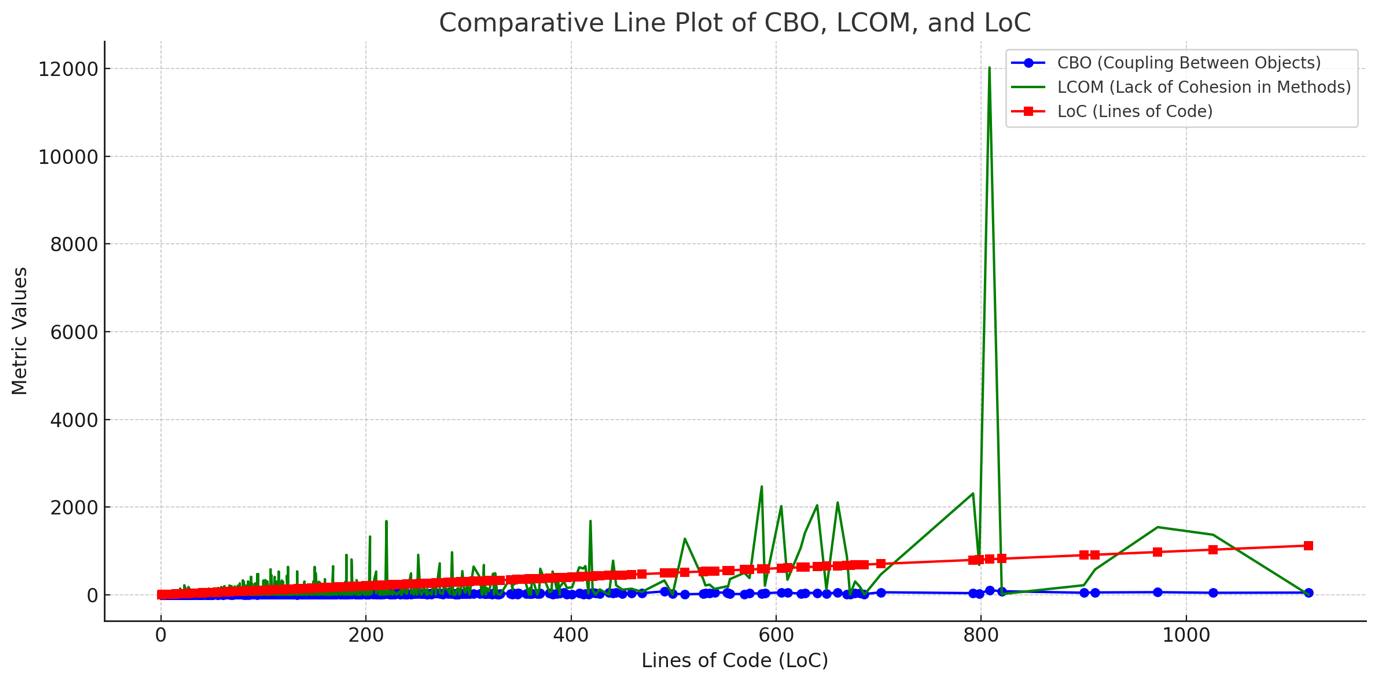
**Eclipse**

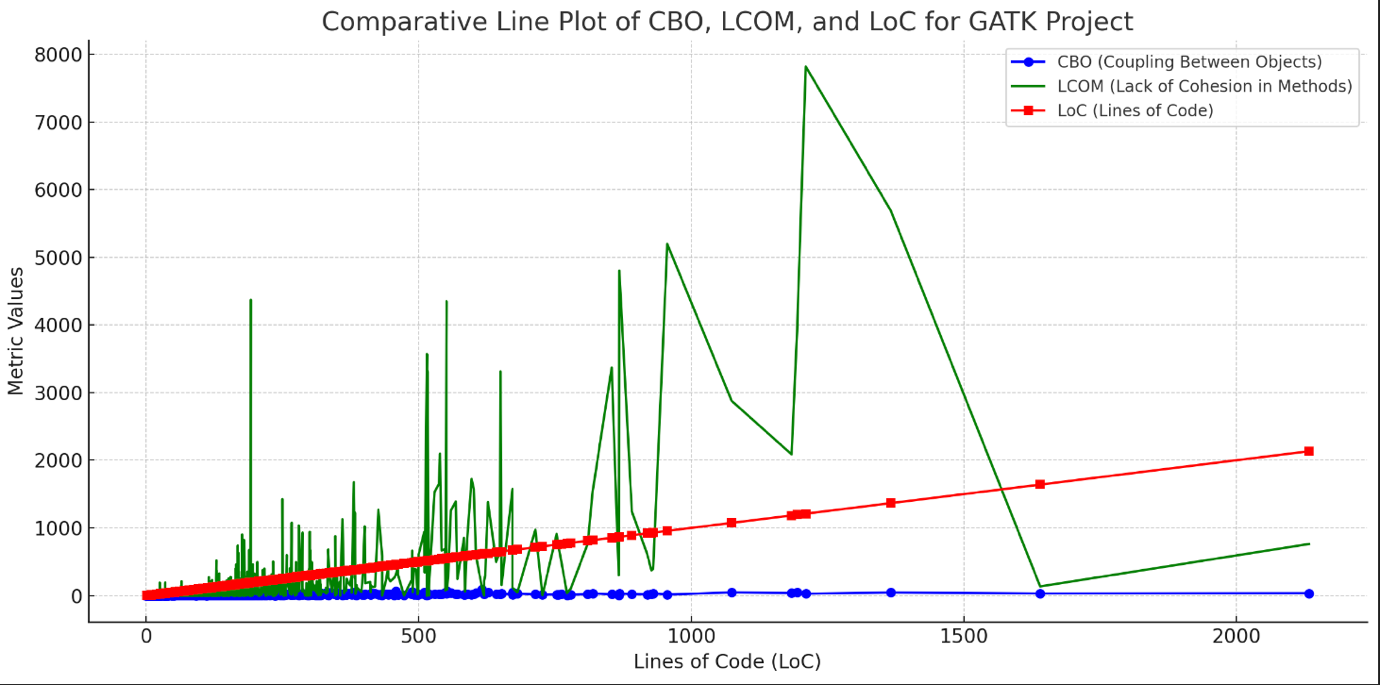
The Eclipse JDT Language Server (eclipse.jdt.ls) is a Java language specific implementation of the Language Server Protocol, which can be utilized with any editor that supports the protocol. This makes it a flexible tool for providing comprehensive support for the Java language across different development environments. The server is developed based on Eclipse LSP4J, a Java binding for the Language Server Protocol, and it leverages the capabilities of the Eclipse JDT to offer Java support. This combination facilitates features like code completion, debugging, and more within editors that integrate this server.

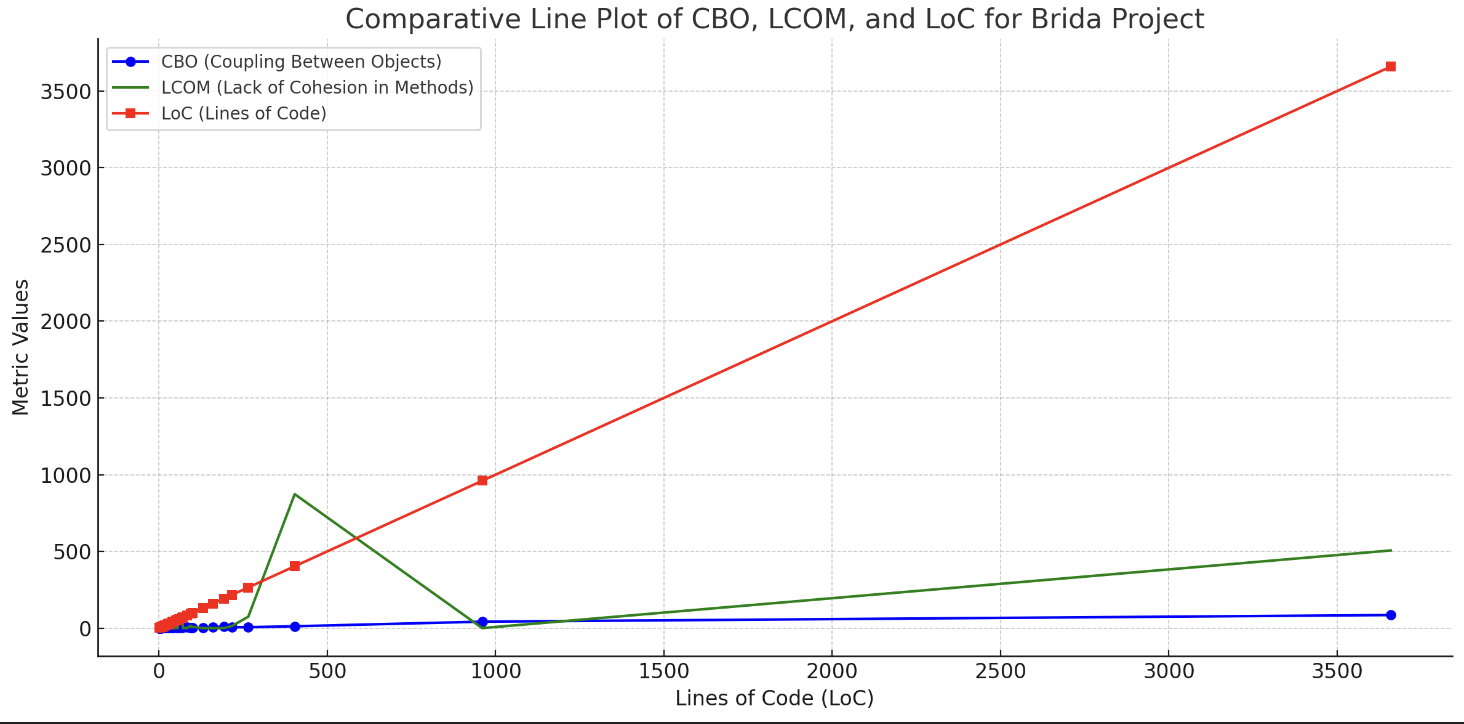
# Section-III

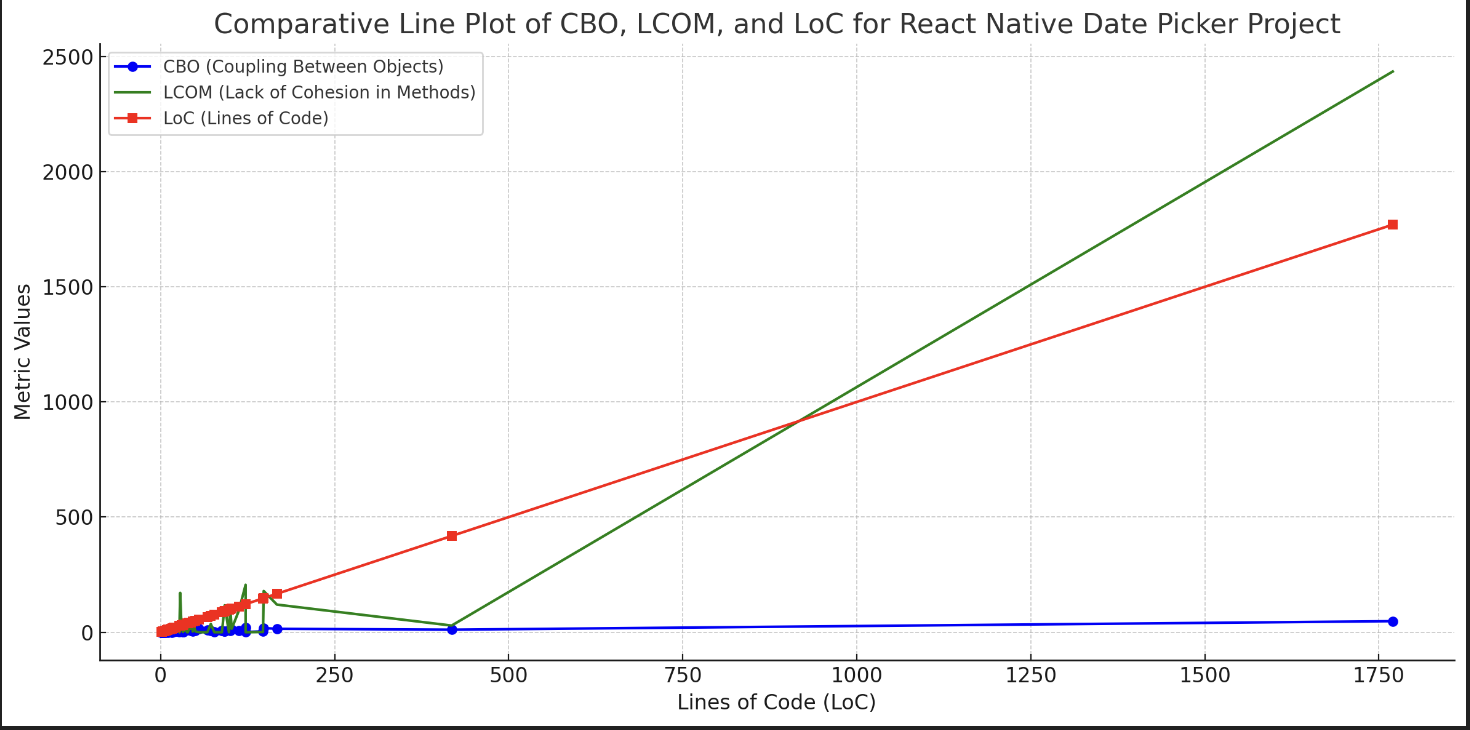
In this section, we will discuss the findings of our empirical study that investigated the impact of class size on software modularity. To obtain the necessary data, we utilized the CK-Code metrics tool to calculate the values of selected C&K metrics for a specific set of Java projects sourced from GitHub, which aligned with our predefined criteria. Our analysis focused on 10 projects that satisfied our requirements, and we employed the CK-Code metrics tool to evaluate the classes within these projects.

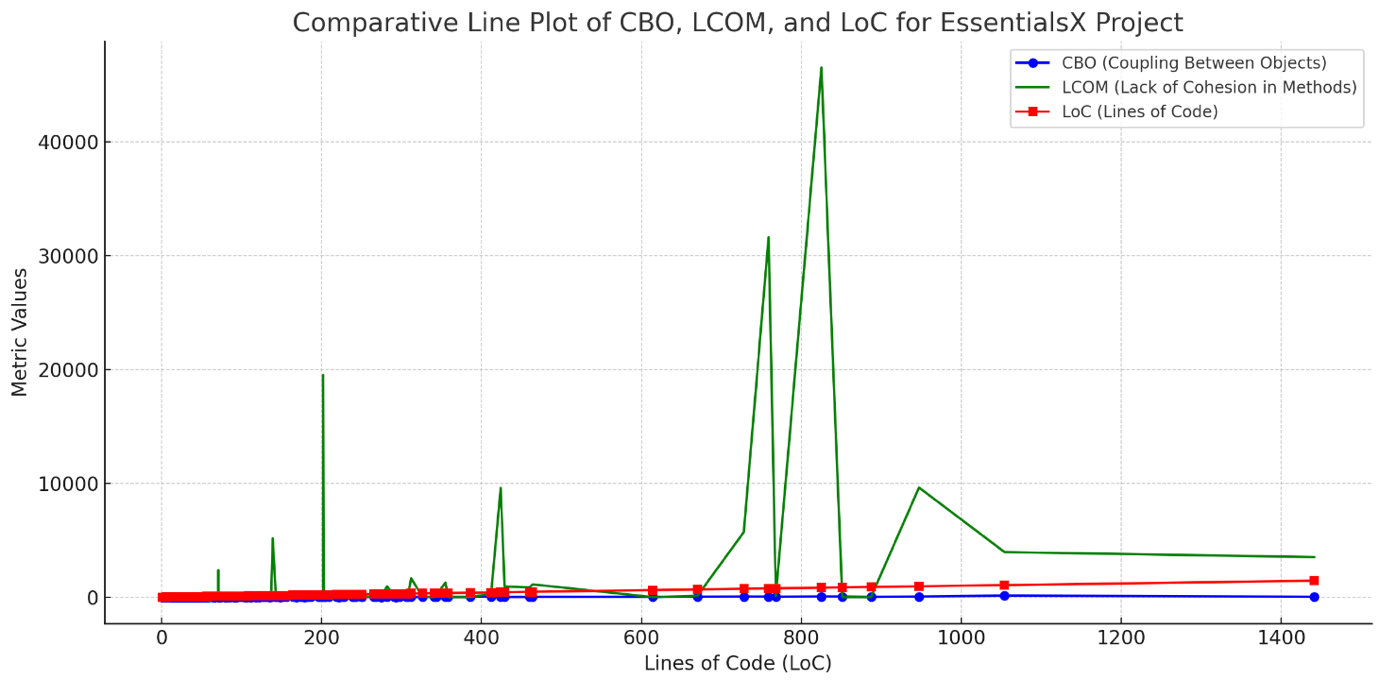
For measuring modularity, we specifically chose the C&K metrics of Coupling Between Objects (CBO) and Lack of Cohesion in Methods (LCOM). Additionally, we considered class size, measured in terms of lines of code (LoC), as a relevant factor in our study.

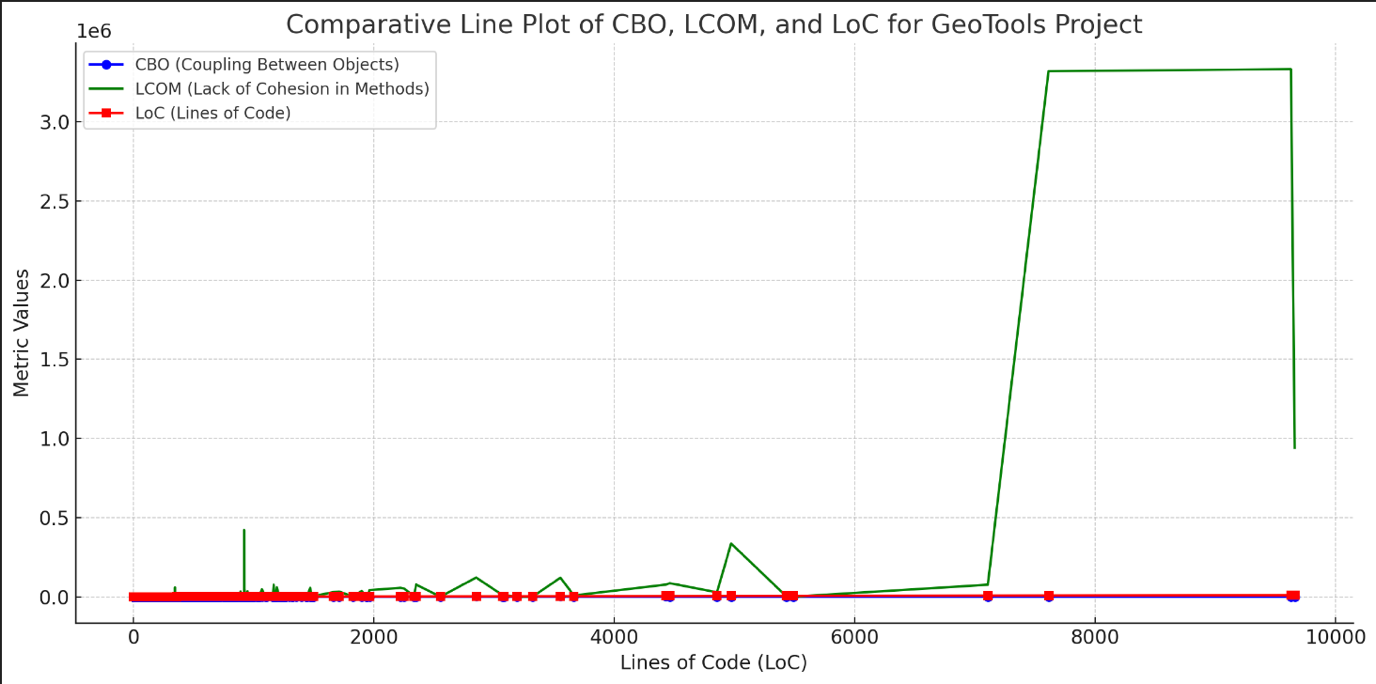


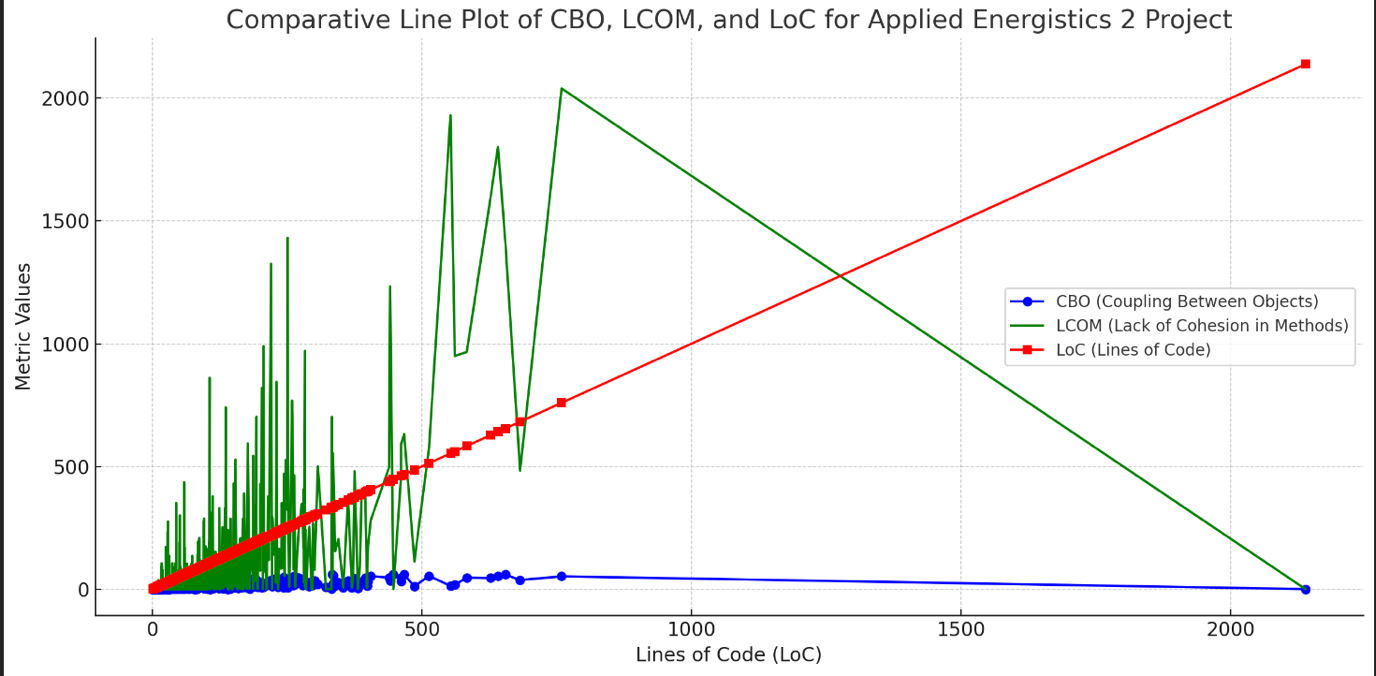


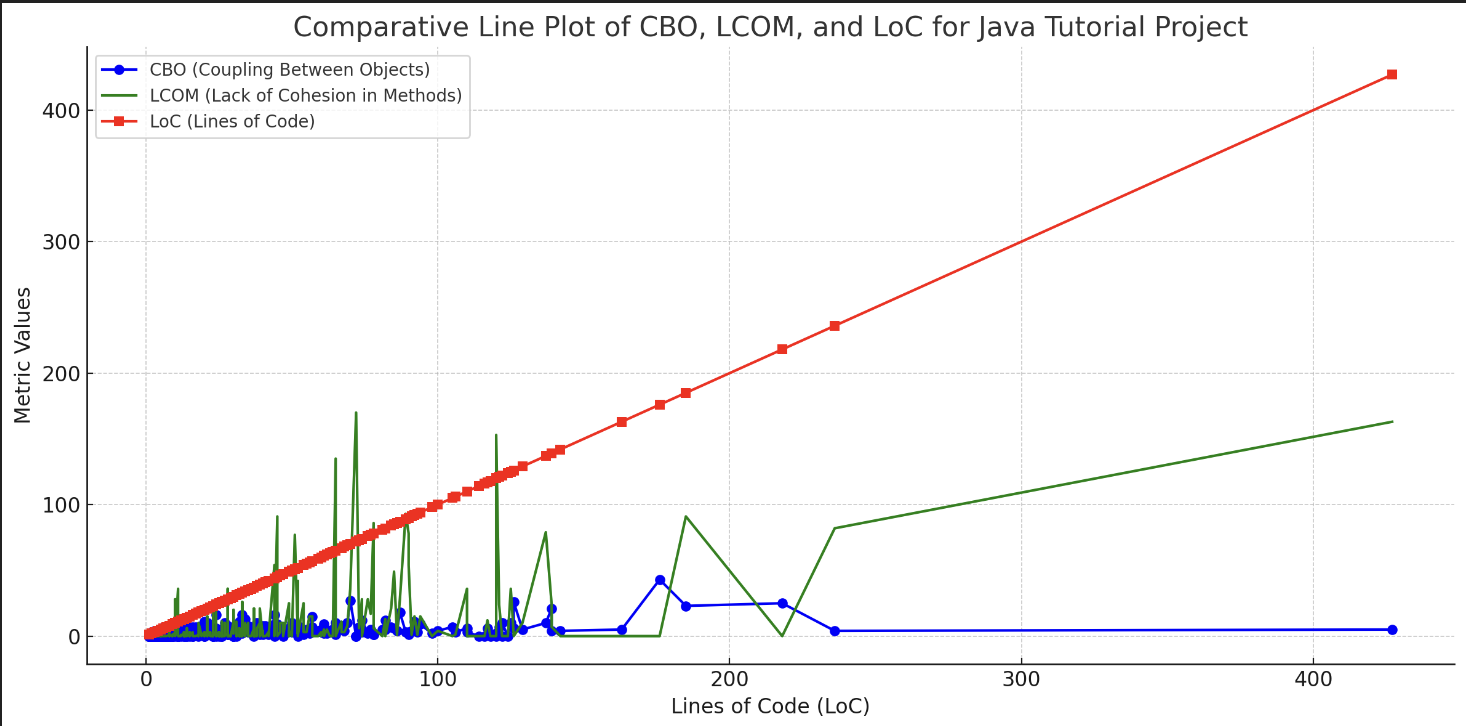


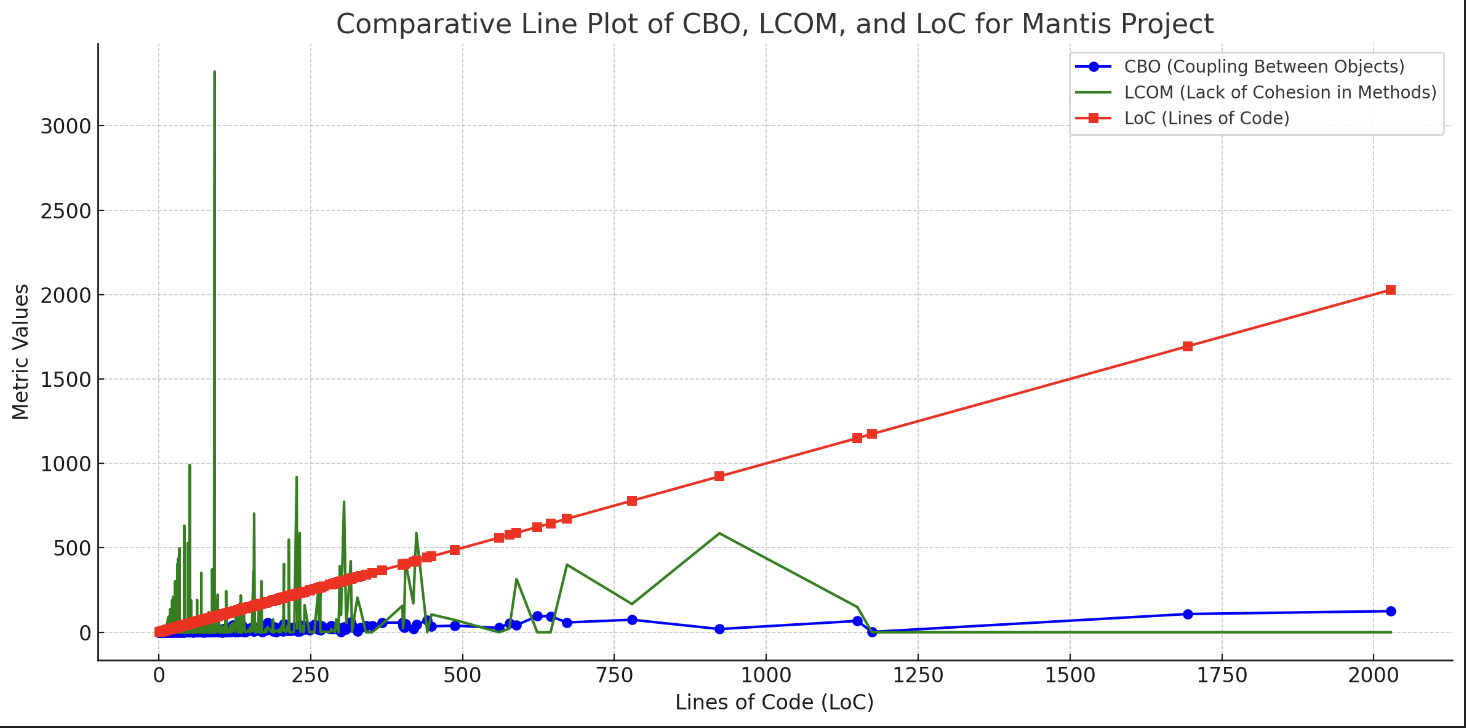


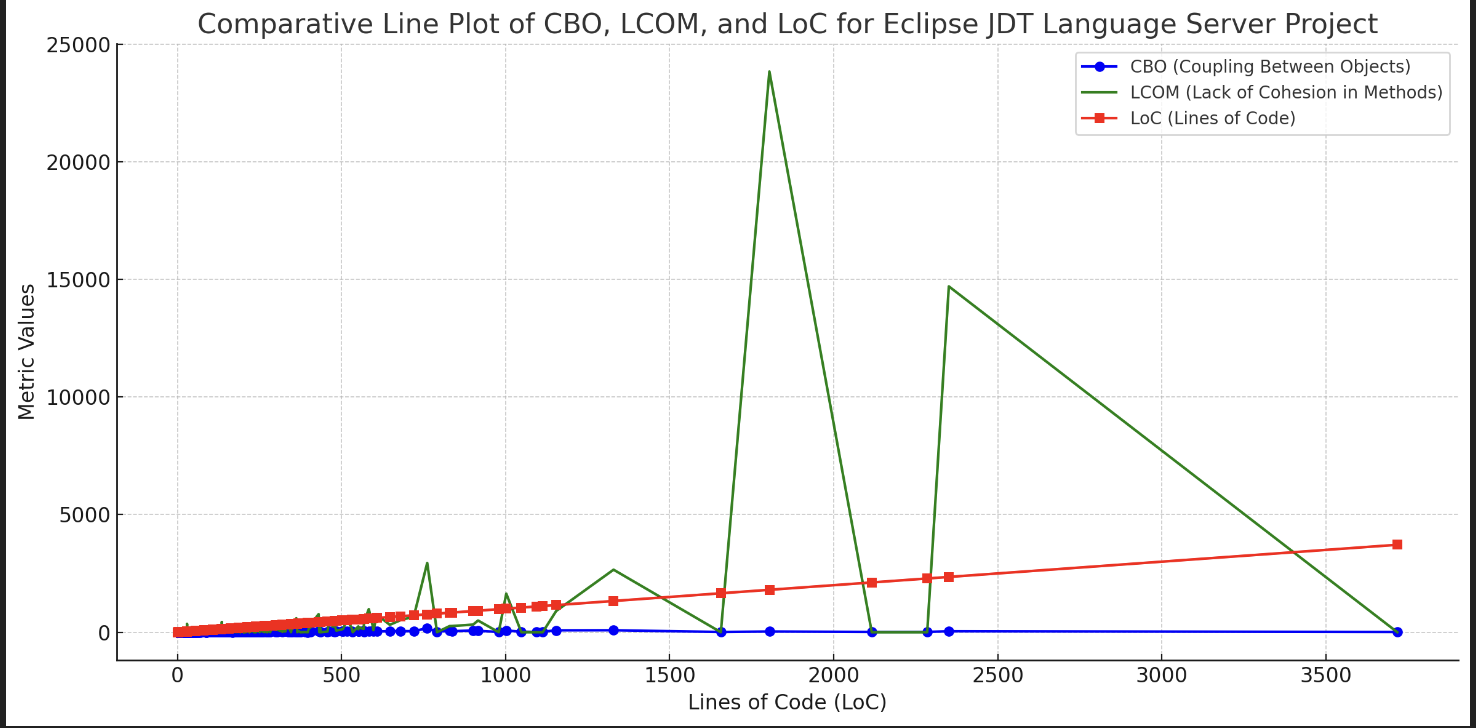












In the process of enhancing code quality across various Java projects, a specialized PMD ruleset was employed to systematically identify common programming pitfalls categorized as "bad smells." These include error-prone practices such as empty catch blocks, which obscure errors, and the use of abstract classes without abstract methods, which can mislead the design intentions. Additionally, the ruleset targets inefficiencies like unnecessary fully qualified names that clutter code readability and recommends performance optimizations such as using StringBuffer for string concatenation in repetitive or concurrent contexts. Through the targeted application of these rules, the analysis aims to direct attention to specific areas of the code that may benefit significantly from refactoring, thereby improving maintainability, performance, and security. This methodical approach to static code analysis is instrumental in advancing code quality and ensuring adherence to best coding practices.

|  |  |  |  |
| --- | --- | --- | --- |
| **Project** | **Total Files Processed** | **Violations** | **Bad Smell Percentage (%)** |
| Eclipse JDT Language Server | 833 | 297 | 35.65% |
| Applied Energistics 2 | 1875 | 123 | 6.56% |
| EssentialsX | 561 | 20 | 3.57% |
| GATK | 2466 | 487 | 19.75% |
| GeoTools | 11249 | 2046 | 18.18% |
| Java Tutorial | 400 | 23 | 5.75% |
| Netflix Mantis | 1172 | 126 | 10.75% |
| React Native Date Picker | 84 | 6 | 7.14% |
| Spring Integration | 3261 | 229 | 7.02% |

From the analysis, significant findings were that projects with larger codebases like GeoTools and GATK tend to exhibit a higher percentage of bad smells, underscoring potential areas for targeted refactoring. The study also highlighted that even smaller projects, like Brida, can have disproportionately high bad smell percentages, pointing to potential issues in coding practices or project maintenance. Ultimately, our study confirms the nuanced nature of code quality issues across project sizes and types, emphasizing the need for ongoing quality assurance and the adoption of best practices in software development. The insights gained through this research provide a valuable foundation for future studies and initiatives aimed at enhancing software modularity and maintainability in Java-based environments.

To assign severity scores to each project based on the analysis of bad smells and their impact on modularity, we can consider several factors such as the percentage of bad smells detected, the complexity of the project (measured by lines of code), and the active maintenance indicators (such as open issues and commits). Severity scores will be scaled from 1 to 5, where 1 indicates minor concerns and 5 indicates critical concerns.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Project** | **Bad Smell Percentage (%)** | **Open Issues** | **Lines of Code** | **Severity Score** |
| Eclipse JDT Language Server | 35.65% | 358 | 97,065 | 5 |
| Brida | 60% | 14 | 7,649 | 5 |
| Applied Energistics 2 | 6.56% | 182 | 111,634 | 3 |
| EssentialsX | 3.57% | 178 | 46,029 | 2 |
| GATK | 19.75% | 1.2k | 80423 | 4 |
| GeoTools | 18.18% | 10 | 1,070,003 | 4 |
| Java Tutorial | 5.75% | 9 | 15,938 | 2 |
| Netflix Mantis | 10.75% | 35 | 93,671 | 3 |
| React Native Date Picker | 7.14% | 20 | 5,483 | 2 |
| Spring Integration | 7.02% | 121 | 230,794 | 3 |

**Severity Score Criteria:**

* **1 (Lowest Concern)**: Bad smells < 5%, low complexity or low maintenance activity.
* **2 (Minor Concern)**: Bad smells < 10%, moderate complexity or moderate maintenance activity.
* **3 (Moderate Concern)**: Bad smells 10-20%, high complexity or significant maintenance activity.
* **4 (High Concern)**: Bad smells 20-50%, very high complexity or high maintenance activity.
* **5 (Critical Concern)**: Bad smells > 50% or exceptionally high complexity or maintenance activity.

**Analysis:**

* **Eclipse JDT Language Server** and **Brida** show critically high bad smell percentages, justifying a severity score of 5. Brida, despite a smaller codebase, has an exceptionally high bad smell percentage.
* **GATK** and **GeoTools**, given their complex nature and significant percentage of bad smells, receive a high concern rating.
* Projects like **EssentialsX** and **Java Tutorial** show fewer bad smells relative to their size and activity, leading to lower severity scores.

# Conclusion

In conclusion, our extensive empirical analysis across ten Java-based projects has revealed significant insights into the relationship between code bad smells and software modularity. The data clearly indicates that projects with larger codebases, such as GeoTools and GATK, exhibit higher percentages of bad smells, suggesting areas that might benefit from targeted refactoring efforts. Conversely, projects with lower bad smell percentages, such as EssentialsX and Java Tutorial, demonstrate better modularity characteristics. This study highlights the critical need for ongoing quality assurance and the adoption of best coding practices to mitigate the adverse effects of code bad smells on the structural integrity and maintainability of software. The findings from this research provide a robust foundation for future studies and initiatives aimed at enhancing software modularity and ensuring higher quality in Java-based development environments.