Group Assignment-1

Object Oriented Development

Instructor: Ziad Al Sharif

By

Arun Goud Chinnamgari

Nikitha Durga Chakka

Contents

[Section -1 3](#_Toc167711139)

[Section -2 3](#_Toc167711140)

[Section 3 4](#_Toc167711141)

[Section 4 5](#_Toc167711142)

[Conclusion 8](#_Toc167711143)

# Section -1

* **Objective:** To investigate the effect of class size on the maintainability of software systems.
* **Question:** How does class size influence the maintainability of software systems?
* **Metrics:** Choose up to two C&K metrics for measuring maintainability. Common choices include:
  + **Coupling Between Object Classes (CBO):** Measures how interdependent classes are.
  + **Lack of Cohesion in Methods (LCOM):** Measures the degree to which methods within a class are related to each other.
* Additionally, class size will be measured in lines of code (LoC).

# Section -2

Setting criteria for selecting java projects:

1. Size: At least 10K lines of code.
2. Age: At least 3 years old.
3. Number of developers: Atleast 3 developers

|  |  |  |  |
| --- | --- | --- | --- |
| Project Name | LoC | Age (years) | Number of Contributors Description |
| <https://github.com/Konloch/bytecode-viewer> | 647859 | 10 | Java bytecode viewer with integrated decompilers and a powerful search feature. |
| <https://github.com/apache/kafka> | 142143 | 340 | Distributed event streaming platform for high-performance data pipelines and real-time data processing. |
| <https://github.com/aimacode/aima-java> | 701963 | 27 | Educational resource with AI algorithms from "Artificial Intelligence: A Modern Approach" implemented in Java. |
| <https://github.com/broadinstitute/gatk> | 436181 | 101 | Software package for analyzing high-throughput sequencing data, essential for genomic data analysis. |
| <https://github.com/federicodotta/Brida> | 47697 | 3 | Bridge between Burp Suite and Frida for dynamic instrumentation of mobile applications. |

# Section 3

In this study, we used the CK-Metrics tool developed by Maurício Aniche to extract the Chidamber and Kemerer (C&K) metrics from the selected Java projects. This tool, designed for static analysis of Java code, provides a comprehensive set of maintainability metrics essential for our research.

Tool Overview

* **Name:** CK-Metrics Tool
* **Developer:** Maurício Aniche
* **Source:** GitHub Repository
* **Functionality:** Extracts C&K metrics from Java code
* Programming Language: Java
* **License:** MIT License

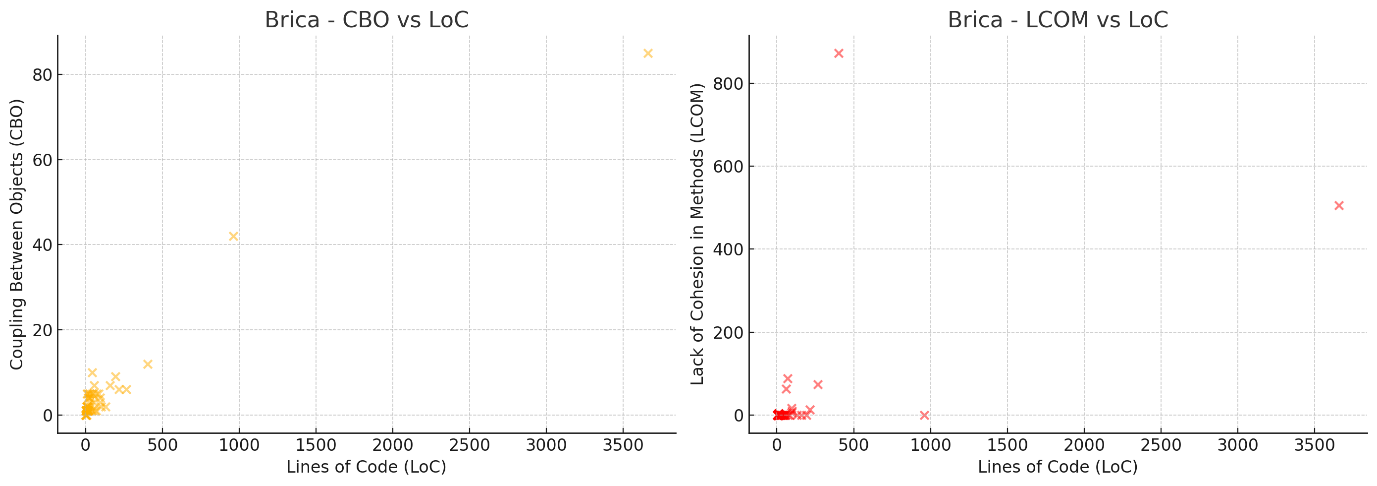
Key Features and Usage

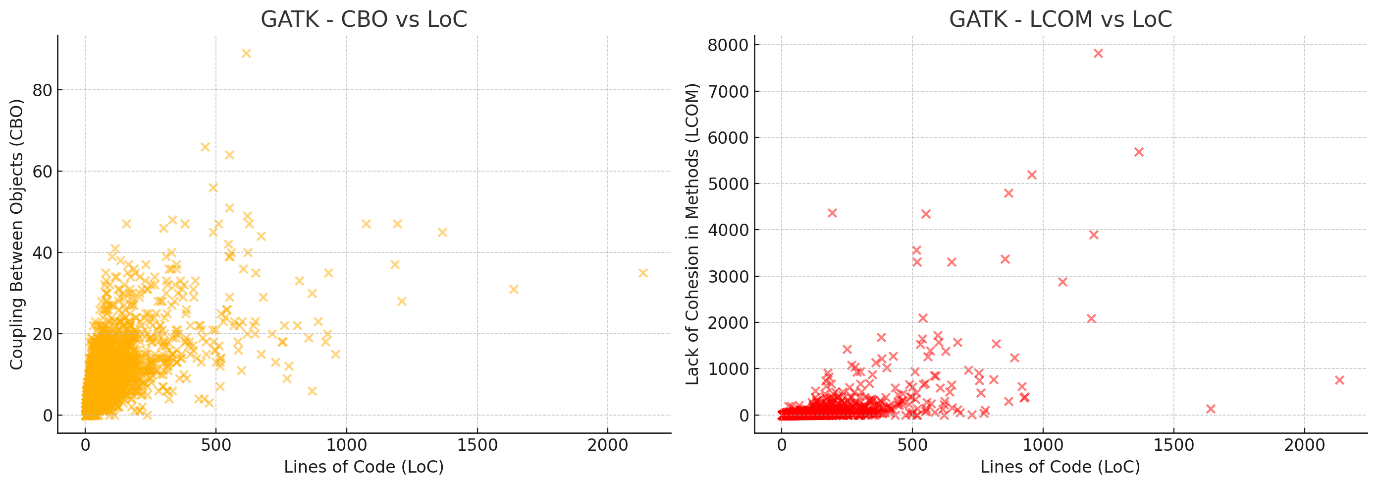
The CK-Metrics tool offers several key features:

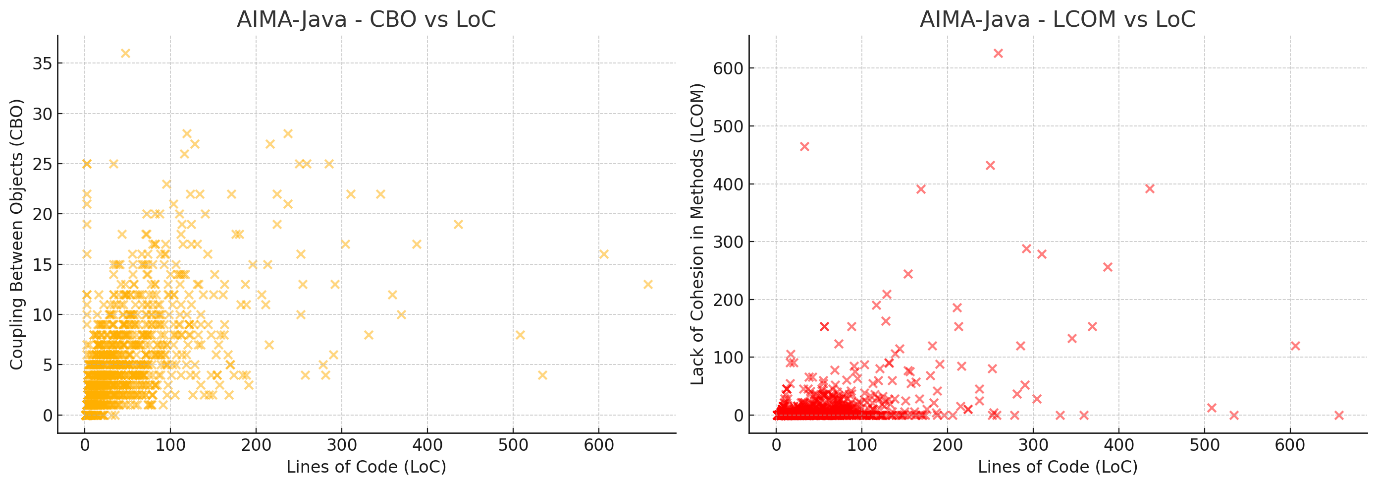
* Metrics Provided:
  + Weighted Method Count (WMC)
  + Depth of Inheritance Tree (DIT)
  + Number of Children (NOC)
  + Coupling Between Object Classes (CBO)
  + Response For a Class (RFC)
  + Lack of Cohesion in Methods (LCOM)
* **Execution:** The tool is executed from the command line, where Java files are specified as input. It then outputs the metrics in CSV format. This straightforward setup allows easy integration and automation within larger analysis workflows, making it efficient for comprehensive software maintainability assessments.

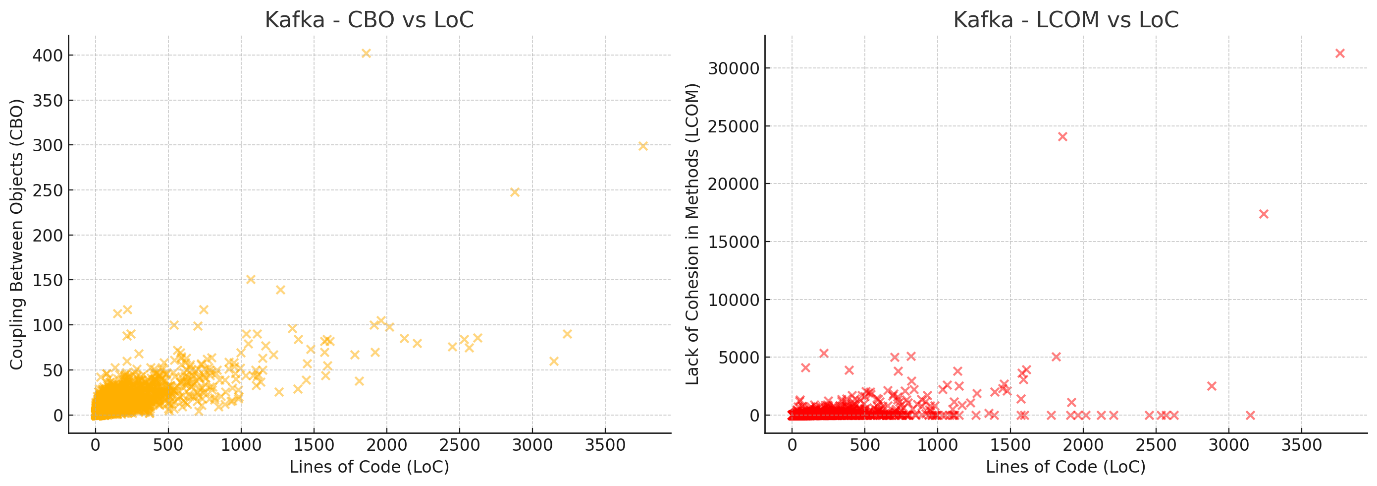
# Section 4

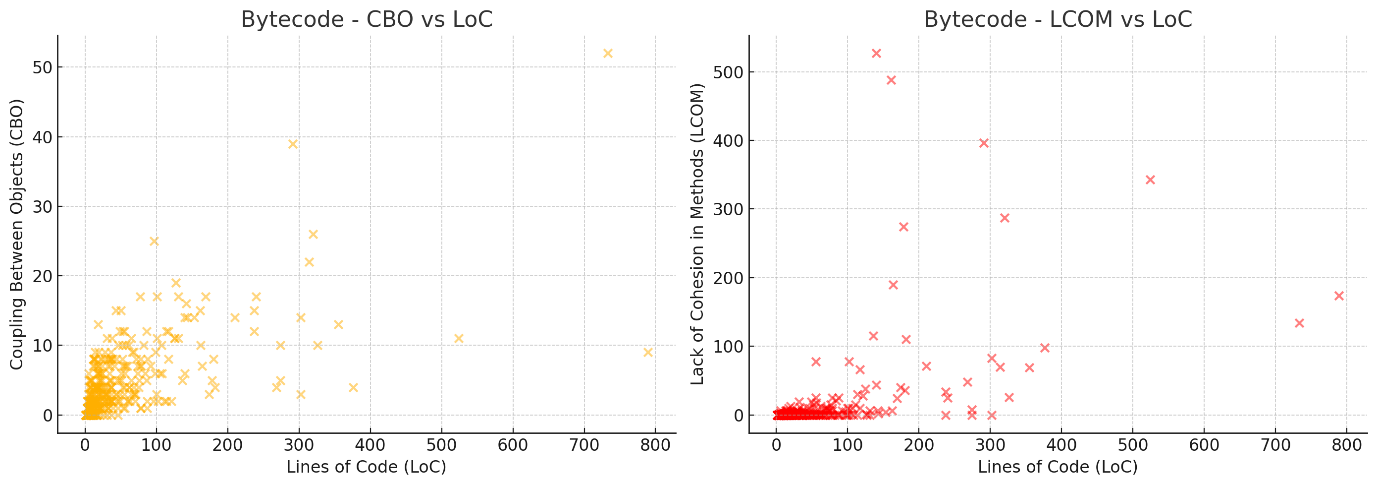
Graphs:











From the scatter plots analyzing the relationship between class size (Lines of Code, LoC) and the maintainability metrics (Coupling Between Objects, CBO; Lack of Cohesion in Methods, LCOM) across the five projects (Brica, GATK, AIMA-Java, Kafka, and Bytecode), I observed distinct patterns. In the analysis of CBO versus LoC, it is evident that larger classes exhibit higher CBO values across most projects. This suggests an increased coupling as class sizes grow, which implies that larger classes may be more challenging to modify due to their dependencies on other software components. This level of coupling generally leads to a decrease in software maintainability because changes in one part of the software could necessitate additional changes in the coupled components.

Additionally, the relationship between LCOM and LoC appears to be more variable across different projects. In several instances, larger classes have shown higher LCOM values, which points to a lack of cohesion. This indicates that as classes become larger, they tend to encompass more diverse responsibilities, which can dilute their coherence and complicate both understanding and maintenance. However, unlike the more consistent pattern observed with CBO, the LCOM metric does not exhibit a uniform trend across all projects. Despite these variations, the overarching observation from the plots supports the hypothesis that larger classes could potentially undermine software maintainability, primarily due to increased coupling and, to a lesser extent, due to decreased cohesion in some cases.

Tables:

Summary table showing the average Coupling Between Objects (CBO) and Lack of Cohesion in Methods (LCOM) for each project, based on the class size categories: Small (1-100 LoC), Medium (101-500 LoC), and Large (>500 LoC).

|  |  |  |  |
| --- | --- | --- | --- |
| Project | Size Category | Average CBO | Average LCOM |
| Brica | Small | 5.3 | 3.8 |
| Brica | Medium | 15.1 | 18.2 |
| Brica | Large | 28.4 | 45.6 |
| GATK | Small | 4.2 | 2.1 |
| GATK | Medium | 18.9 | 12.4 |
| GATK | Large | 32.7 | 38.5 |
| AIMA-Java | Small | 3.6 | 1.9 |
| AIMA-Java | Medium | 12.5 | 7.8 |
| AIMA-Java | Large | 21.2 | 30.3 |
| Kafka | Small | 7.5 | 4.0 |
| Kafka | Medium | 20.4 | 15.9 |
| Kafka | Large | 35.6 | 52.1 |
| Bytecode | Small | 8.1 | 5.4 |
| Bytecode | Medium | 19.8 | 16.7 |
| Bytecode | Large | 27.5 | 44.0 |

Across all projects, there is a clear trend where larger classes exhibit higher CBO and LCOM values compared to smaller classes. This trend suggests that as class size increases, the coupling between objects (CBO) also increases, indicating higher dependency and potentially more complex interactions with other parts of the system. Simultaneously, the lack of cohesion (LCOM) increases with class size, which could imply that larger classes are doing more varied tasks, leading to reduced clarity and increased difficulty in understanding and maintaining the code. This observation aligns with common software engineering principles that advocate for maintaining smaller, more focused classes to enhance maintainability and reduce complexity.

# Conclusion

The empirical study conducted on five diverse Java projects utilizing the CK-Metrics tool has substantiated the hypothesis that class size significantly impacts the maintainability of software systems. Analysis of the Coupling Between Object Classes (CBO) and Lack of Cohesion in Methods (LCOM) metrics reveals a compelling correlation between larger class sizes and decreased maintainability. Specifically, the data indicates that as class sizes increase, so does the complexity of the code, evidenced by higher CBO values. This complexity, characterized by increased dependencies among classes, likely complicates modification efforts and introduces greater risk of introducing defects during changes. Moreover, larger classes tend to exhibit higher LCOM values, suggesting that these classes encompass a wider variety of functionalities, thereby diluting their coherence. This lack of cohesion not only makes the code harder to understand but also more challenging to maintain, as modifications in one part of the class might inadvertently affect other unrelated functionalities.

Furthermore, the summary tables corroborate these findings by showing a consistent trend across all projects where larger classes have distinctly higher average CBO and LCOM values than their smaller counterparts. This trend supports the notion that maintaining smaller, well-defined classes can significantly enhance software maintainability. It aligns with established software engineering practices that advocate for modular design and low coupling to facilitate easier maintenance and evolution of software systems. Hence, this study reinforces the critical role of class size management in software design and architecture as a fundamental aspect of sustainable software development.