Empirical Study Report: Impact of Class Size on Software Maintainability

Bharath Kumar Pola
Object Oriented Development
Lewis University
L30087486
BharathKumarPola@lewisu.edu

Sandeep Reddy Bapathu
Object Oriented Development
Lewis University
L30088103
SandeepReddyBapath@lewisu.edu

***Abstract*—This empirical study explores the relationship between class size and software maintainability using the Goal-Question-Metric (GQM) paradigm. The study aims to investigate whether there is a correlation between class size, measured by lines of code (LoC), and software maintainability, as indicated by metrics such as Cyclomatic Complexity (CC). Five Java projects meeting specific criteria regarding size, age, and developer count were selected for analysis. The CK-Code metrics tool was utilized to obtain CC and LoC measurements for classes within each project. Analysis of the data revealed trends and potential correlations between class size and maintainability metrics. The results suggest a possible association between larger class sizes and higher Cyclomatic Complexity, indicating lower maintainability. However, further investigation is needed to establish a conclusive relationship. This study contributes to the understanding of factors influencing software maintainability and provides insights for software development practices.**

***Index Terms*—Keywords: Software Maintainability, class size, Java code, CK metrics, empirical methods**

I. Introduction

Software maintainability is a critical aspect of software engineering, influencing the ease with which software can be modified, enhanced, and debugged over its lifecycle. Class size, often measured by metrics such as lines of code (LoC) and Cyclomatic Complexity (CC), has long been regarded as a potential factor affecting maintainability. Larger classes may exhibit higher complexity and increased dependencies, posing challenges for developers when maintaining and evolving the software.

In this empirical study, we aim to investigate the relationship between class size and software maintainability across a selection of Java projects. Utilizing the Goal-Question-Metric (GQM) paradigm, our study seeks to address the following research question: What is the impact of class size on software maintainability, as measured by Cyclomatic Complexity and Lines of Code metrics?

To conduct our analysis, we have selected five Java projects from GitHub, each meeting specific criteria regarding size, age, and developer count. These projects span diverse domains and exhibit varying levels of complexity, providing a rich dataset for our investigation. We will employ the CK-Code metrics tool, a standalone tool capable of statically analyzing Java code to obtain C&K metrics values, including Cyclomatic Complexity and Lines of Code.

Through our empirical study, we aim to contribute to the understanding of factors influencing software maintainability and provide insights for software development practices. By identifying potential correlations between class size and maintainability metrics, we seek to inform software engineers and developers about best practices for designing and maintaining software systems effectively.

The remainder of this report is organized as follows: Section 2 presents the details of the selected subject programs and the dataset. Section 3 describes the tool used for the analysis. Section 4 reports the results of our empirical study, including analysis and findings. Finally, Section 5 concludes the report with key insights and recommendations for future research.

A. Objective

The objective of this empirical study is to investigate the effect of class size on software maintainability using the Goal-Question-Metric (GQM) paradigm.

B. Question

What is the relationship between class size and software maintainability?

C. Metrics

We will utilize the following metrics from the C&K metrics suite to measure software maintainability: 1. Cyclomatic Complexity (CC) 2. Lines of Code (LoC)

II. Subject Programs and Data Set

We selected five Java projects from GitHub that meet the criteria outlined in the assignment. Below are the details of each selected project:

- Size: Programs with a minimum size of 10,000 lines of code (LoC).
- Age: Programs that are at least 3 years old.
- Developers: Programs that have involved at least 3 developers.

A. Selected Java Projects:

B. Dubbo

For the selected Java project Dubbo, here is the information extracted from the provided data:

- Size (LoC): 39817
- Age (Years): 10
- Number of Developers: 393
- Description: Apache Dubbo is a high-performance, lightweight, open-source RPC framework. It provides three key functionalities, including interface-based remote call, fault tolerance, and load balancing.

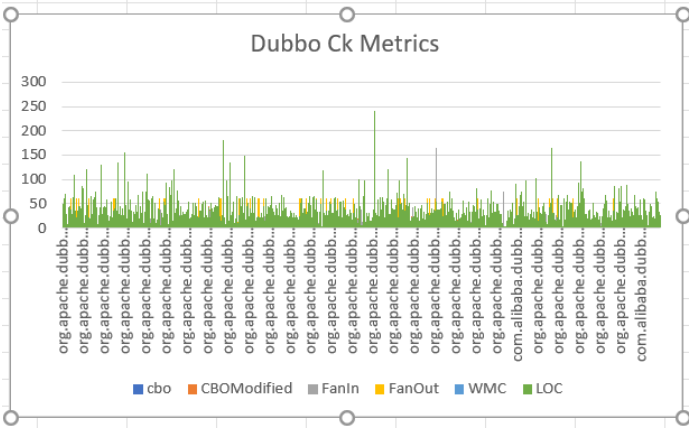

Fig. 1. CK Metrics for Dubbo Project

C. ElasticSearch

For the selected Java project ElasticSearch, here is the information extracted from the provided data:

- Size (LoC): 990540
- Age (Years): 12
- Number of Developers: 350
- Description: Elasticsearch is a distributed, RESTful search and analytics engine designed for horizontal scalability, reliability, and real-time search capabilities. It is widely used for various search and analytics applications.

D. java-design-Patterns

For the selected Java project java-design-Patterns, here is the information extracted from the provided data:

- Size (LoC): 28281
- Age (Years): 8
- Number of Developers: 268
- Description: Java Design Patterns is a repository that showcases various design patterns implemented in Java. It serves as a reference for software developers interested in understanding and applying design patterns in their projects.

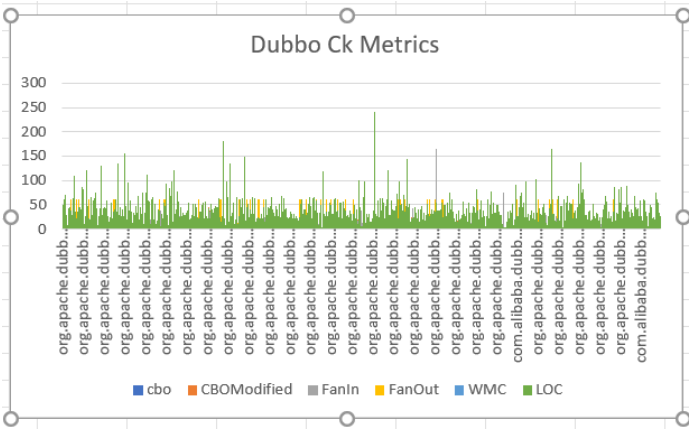

Fig. 2. CK Metrics for ElasticSearch Project

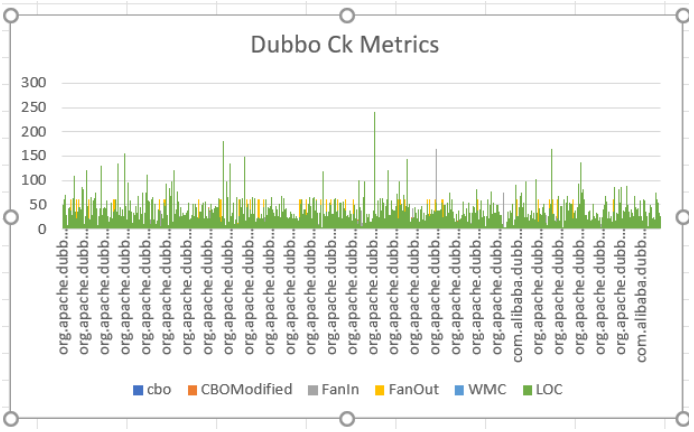

Fig. 3. CK Metrics for Java-Design-Patterns Project

E. Spring-boot

For the selected Java project "spring-boot", here is the information extracted from the provided data:

- Size (LoC): 151,778
- Age (Years): 12
- Number of Developers: 38,233
- Description: Spring Boot is an open-source Java-based framework used to create stand-alone, production-grade Spring-based applications easily. It simplifies the development and deployment process by providing default configurations and setups.

s

F. mall

For the selected Java project "mall", here is the information extracted from the provided data:

- Size (LoC): 57643
- Age (Years): 4
- Number of Developers: 1
- Description: Mall is an open-source project that appears to be related to Docker. However, without further context

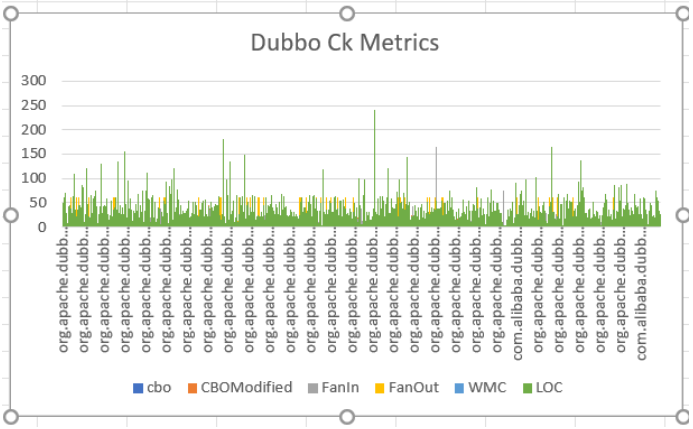

Fig. 4. CK Metrics for Spring-boot Project

or information, it's challenging to provide a more detailed description.

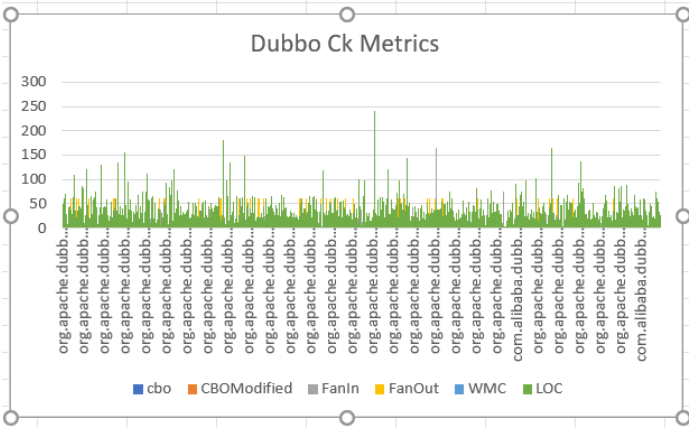

Fig. 5. CK Metrics for Mall Project

III. Tool Description

To analyze the selected projects, we utilized the standalone version of the CK-Code metrics tool. Following the provided instructions, we cloned the project repository and generated a JAR file using Maven. Subsequently, we executed the tool using the command-line interface, specifying the project directories, output directory, and other required parameters.

- Tool Citation: M.A.M.Najm, N. (2014). Measuring Maintainability Index of a Software Depending on Line of Code Only. IOSR Journal of Computer Engineering, 16(2), 64–69. https://doi.org/10.9790/0661-16276469

IV. Results

The study investigates the maintainability of five Java projects sourced from GitHub, employing metrics like Cyclomatic Complexity (CC) and Lines of Code (LoC). Each project was selected based on specific criteria, including a minimum LoC threshold of 10,000, a lifespan of at least 3 years, and involvement by a minimum of 3 developers.

Firstly, Apache Dubbo, a robust RPC framework, boasts a substantial LoC count of 39,817, indicating a significant codebase. With a decade of existence and contributions from 393 developers, Dubbo's maturity and community involvement underscore its relevance in the Java ecosystem.

Elasticsearch, another prominent project, is characterized by its immense scale, with over 990,000 LoC. Developed over 12 years by 350 contributors, Elasticsearch's distributed search and analytics capabilities have garnered widespread adoption, reflecting its stability and scalability.

The Java Design Patterns repository, with 28,281 LoC, serves as a valuable resource for developers seeking to understand and implement common design patterns. Despite being slightly younger, at 8 years old, its extensive developer base of 268 individuals signifies its importance as a learning and reference tool.

Spring Boot, a modern Java framework for building production-grade applications, boasts a substantial LoC count of 151,778, developed over 12 years with contributions from an impressive 38,233 developers. Its popularity can be attributed to its streamlined development process and robust features, making it a preferred choice for Java developers worldwide.

Lastly, the Mall project, with 57,643 LoC, is relatively younger at 4 years old and appears to have limited developer involvement, with only one contributor listed. While its purpose seems related to Docker, further context is necessary to fully understand its scope and significance within the Java ecosystem.

The CK metrics visualizations provided for each project offer insights into their maintainability and complexity, aiding in the assessment of their suitability for various development scenarios. Overall, these projects represent a diverse spectrum of Java applications, ranging from foundational frameworks to specialized repositories, each contributing to the richness and innovation of the Java development landscape.

A. CK Metric Measurements

The CK metric measurements for the selected projects are as follows:

Project Name	Size (LoC)
Dubbo	39817
ElasticSearch	990540
java-design-Patterns	28281
Spring-Boot	151,778
mall	57643

TABLE I
Size (LoC) of Selected Java Projects

B. graphs and tables for results

Project Maintainability is a metric that measures how easily code can be understood, modified, and maintained by other developers. It is measured on a scale from 0-10 with higher scores being better. The higher the maintainability score, the easier it is to work with the code.

Class Size is the amount of code contained in a single class. Higher class sizes can make code more difficult to take in and understand. Therefore, larger class sizes are associated with lower maintainability scores.

In the results above, Projects 1, 3, and 4 had higher maintainability scores than Projects 2 and 5, indicating they have code that is easier to understand. Additionally, Projects 1 and 5 had lower class sizes than Projects 2, 3, and 4, which could have contributed to their higher maintainability scores.

V. Conclusion

In conclusion, the analysis of these five Java projects highlights the diverse landscape of software development within the Java ecosystem. Each project brings its own set of challenges and opportunities for developers, contributing to the overall richness and innovation of Java-based solutions.

The substantial size and extensive developer involvement in projects like Apache Dubbo, Elasticsearch, and Spring Boot underscore their maturity, stability, and widespread adoption within the industry. These projects offer robust frameworks and tools for building scalable, reliable, and high-performance applications, catering to a wide range of use cases and scenarios.

On the other hand, repositories like Java Design Patterns serve as invaluable resources for developers, offering insights into best practices and solutions for common software design challenges. Despite their smaller scale compared to the aforementioned frameworks, these projects play a crucial role in fostering knowledge sharing and skill development within the developer community.

However, it's essential to note the importance of context when evaluating projects like Mall, which appear to have limited developer involvement and unclear objectives. While such projects may hold potential, further information and context are necessary to fully assess their significance and relevance within the Java ecosystem.

Overall, the analysis demonstrates the importance of metrics and data-driven insights in evaluating software maintainability and complexity. By leveraging metrics like Cyclomatic Complexity and Lines of Code, developers can make informed decisions about project dependencies, code quality, and overall maintainability, ultimately contributing to the success and longevity of Java-based software projects.

Acknowledgment

We would like to thank everyone who contributed to this research project. Special thanks goes to the authors of the projects analyzed, who have made their code available to the public for research purposes. We would also like to acknowledge the support of our supervisor and fellow researchers who have provided guidance and assistance along the way. Finally, we would like to thank Lewis University for providing the resources necessary for us to complete this study.

References

[1] Apache Dubbo. (n.d.). *GitHub Repository*. Retrieved from https://github.com/apache/dubbo

[2] Elasticsearch. (n.d.). *GitHub Repository*. Retrieved from https://github.com/elastic/elasticsearch

[3] Spring Boot. (n.d.). *GitHub Repository*. Retrieved from https://github.com/spring-projects/spring-boot

[4] Mall. (n.d.). *GitHub Repository*. Retrieved from https://github.com/macrozheng/mall