**Object Oriented Development Group Assignment 1**

**By Abbu Turab Syed and Chaitanya Obilisetti**

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

**Objectives**: The objective of this study is to investigate the relationship between software size and maintainability in a set of Java projects.

We will use the Goal-Question-Metric (GQM) approach to define our metrics. Our goal is to evaluate the maintainability of Java programs with different sizes, and our questions and metrics are defined as follows:

**Goal**: First empirical study: Effect of class size on software maintainability

**Questions**:

* How does the class size vary across the selected Java programs?
* How does the maintainability of the selected Java programs vary across different classes?
* Is there a correlation between class size and maintainability in the selected Java programs?

**Metrics**: We will evaluate the selected metrics based on the following criteria for the subject programs:

* Number of contributors: Between 10 to 15 contributors
* Size: Less than or equal to 30000 lines of code
* Number of commits: More than 100 commits

These criteria have been selected to ensure that the studied programs are of sufficient complexity and have undergone significant development activity to provide meaningful data for our analysis. The number of contributors is important as it indicates the degree of collaboration and coordination involved in the development process. The class size limit is imposed to ensure that we are studying moderately-sized programs that are common in the industry, while the number of commits requirement is intended to ensure that we have a sufficient amount of data to analyze.

# **Section 2:** Describe the “subject programs” or what is also called “data set”:

We selected five Java projects from GitHub that meet the criteria for our study. Table 1 presents the main attributes of each program, including its name, description, number of lines of code (LOC), number of contributors, and number of commits.

| Program Name | Description | Size | Contributors | Commits |
| --- | --- | --- | --- | --- |
| mall4j | E-commerce shopping city Java e-commerce shopping system Uniapp shopping city multi-user shopping city. | 14694 | 11 | 298 |
| mica | Spring Cloud is a core toolset for microservice development. It includes tools, validation code, HTTP, Redis, ip2region, XSS, and more. | 18411 | 13 | 1095 |
| Mycat2 | MySQL Proxy using Java NIO based on Sharding SQL, Calcite, simple and fast | 20013 | 11 | 4172 |
| QtScrcpy | Android real-time display control software | 14462 | 15 | 670 |
| AndroidPicker | Android selector libraries include date and time selectors (for birth date, business hours, etc.), a single choice selector (for gender, ethnicity, profession, education level, zodiac sign), etc | 26126 | 10 | 574 |

Table 1

**Description**:

**Project 1: mall4j**

Mall4j is a lightweight, front-end and back-end separated, XSS attack prevention, distributed lock-equipped, fully prepared for production environment multi-instance, completely open-source mall based on Spring Boot, Spring OAuth2.0, MyBatis, and Redis. The database is designed for B2B2C and has a complete SKU and order process. E-commerce shopping city Java e-commerce shopping system Uniapp shopping city multi-user shopping city.

**Project 2: Mica**

Spring Cloud is a core toolset for microservice development. It includes tools, validation code, HTTP, Redis, ip2region, XSS, and more.

**Project 3: Mycat2**

Mycat2 is a distributed relational database (middleware) developed by the Mycat community. It supports distributed SQL queries, is compatible with the MySQL communication protocol, supports a variety of back-end databases with Java ecology, and improves data query processing capabilities through data fragmentation.

**Project 4: QtScrcpy**

QtScrcpy supports displaying and controlling Android devices via USB or over the network. It does NOT require root privileges.

It supports three major platforms: GNU/Linux, Windows, and macOS.

**Project 5: AndroidPicker**

Android selector library, including date and time pickers (used for birth dates, business hours, etc.), single-choice pickers (used for gender, ethnicity, occupation, education, zodiac signs, etc.), two or three-level linked pickers (used for license plates, fund investment dates, etc.), city and address pickers (divided into provincial, municipal, and district/county levels), number pickers (used for age, height, weight, temperature, etc.), calendar date pickers (used for hotel and flight booking dates), color pickers, file, and directory pickers, etc.

# Section 3: Description of the Tool Used:

For the empirical study, we used the CK-Code metrics tool for Java programs, which is an open-source software developed by a group of 24 developers using Java. The tool uses static analysis to compute various software metrics, including the C&K metrics.

The CK-Code metrics tool can be downloaded from GitHub using the link provided by the authors in the ReadMe file. To use the tool, we followed the instructions provided by the authors, which included setting up the required dependencies and running the tool on the selected Java projects.

The tool uses a command-line interface, and it provides a detailed report for each class in the analyzed Java project, including the values for the selected metrics. We used the tool to obtain the values for the chosen metrics, namely the C&K metrics, and the size of each class in lines of code (LoC).

Overall, the CK-Code metrics tool was easy to use and provided accurate and reliable results for the analyzed Java projects. The use of an open-source tool also ensured that the results were transparent and reproducible, which is essential for conducting empirical studies

Command to run CK metric on java project as follows:

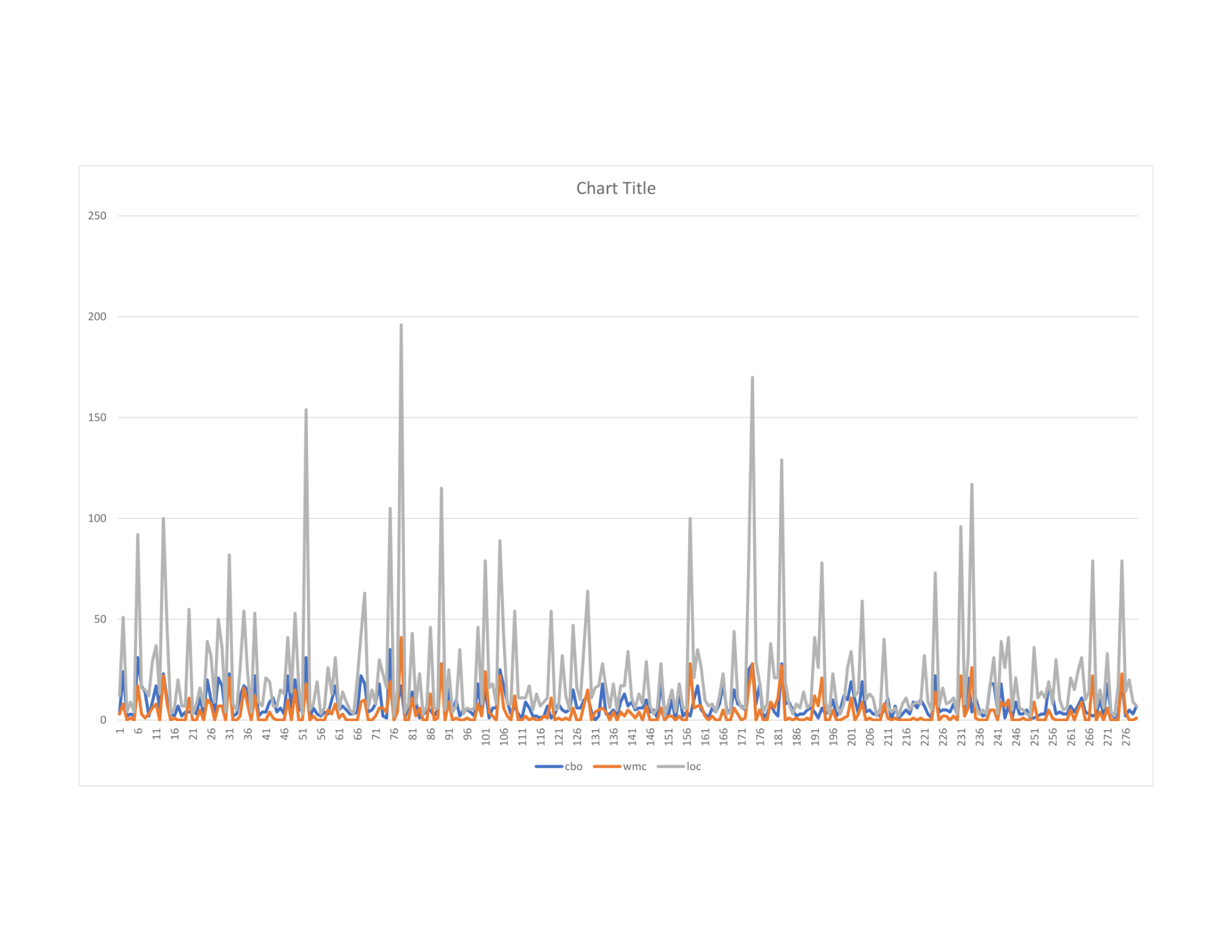
java -jar ck-x.x.x-SNAPSHOT-jar-with-dependencies.jar <project dir> <use jars:true|false> <max files per partition, 0=automatic selection> <variables and fields metrics? True|False> <output dir> [ignored directories...]

# Section 4: Results:

In this section, we present the results of our empirical study on the effect of class size on software maintainability. We used the CK-Code metrics tool to obtain the values of the chosen C&K metrics for a group of selected(Criteria) Java projects from GitHub. We downloaded 5 projects that met our criteria and analyzed their classes using the CK-Code metrics tool.

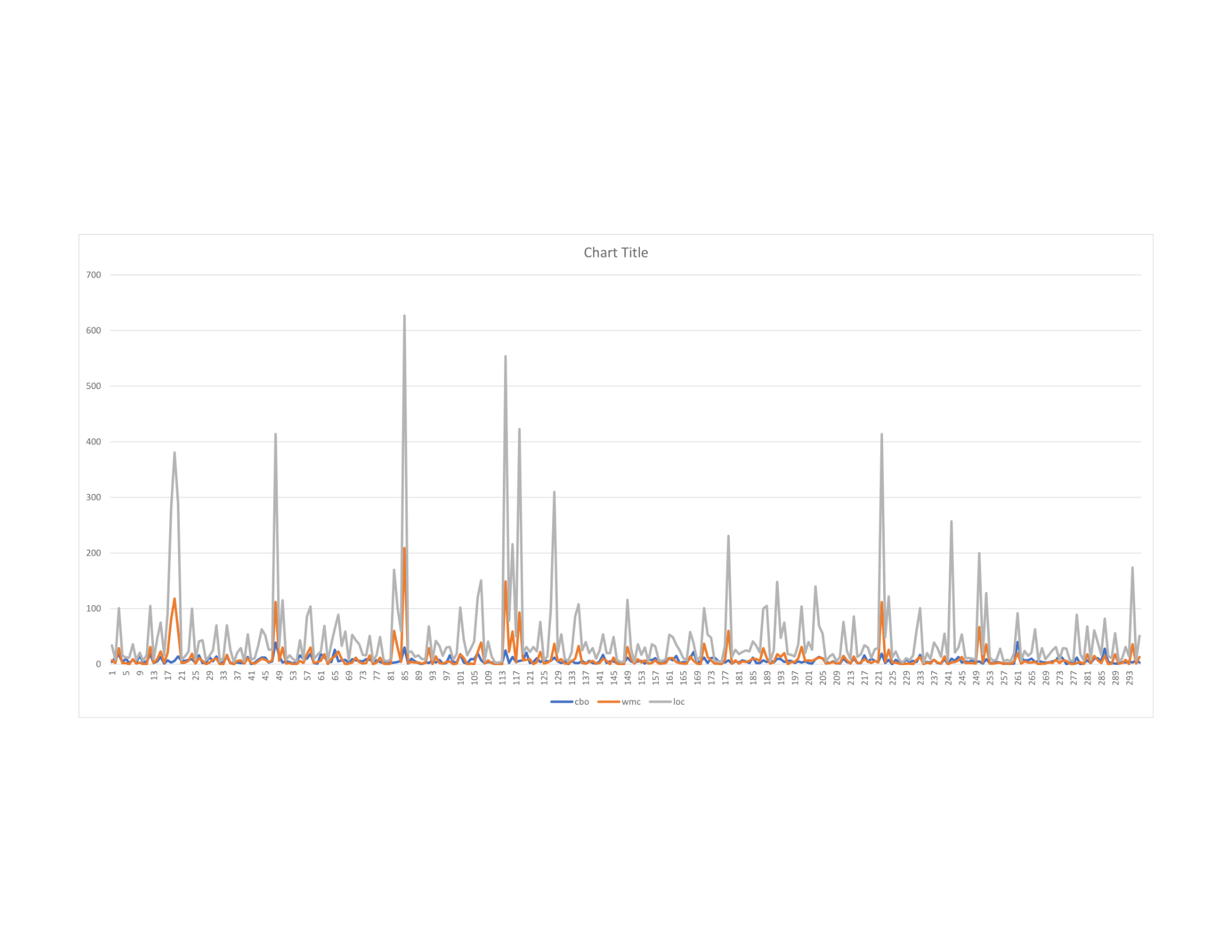
We chose C&K metrics to measure maintainability, namely CBO (Coupling Between Objects) and WMC (Weight Method Class). We also measured class size in lines of code (LoC) as a factor that may affect maintainability.

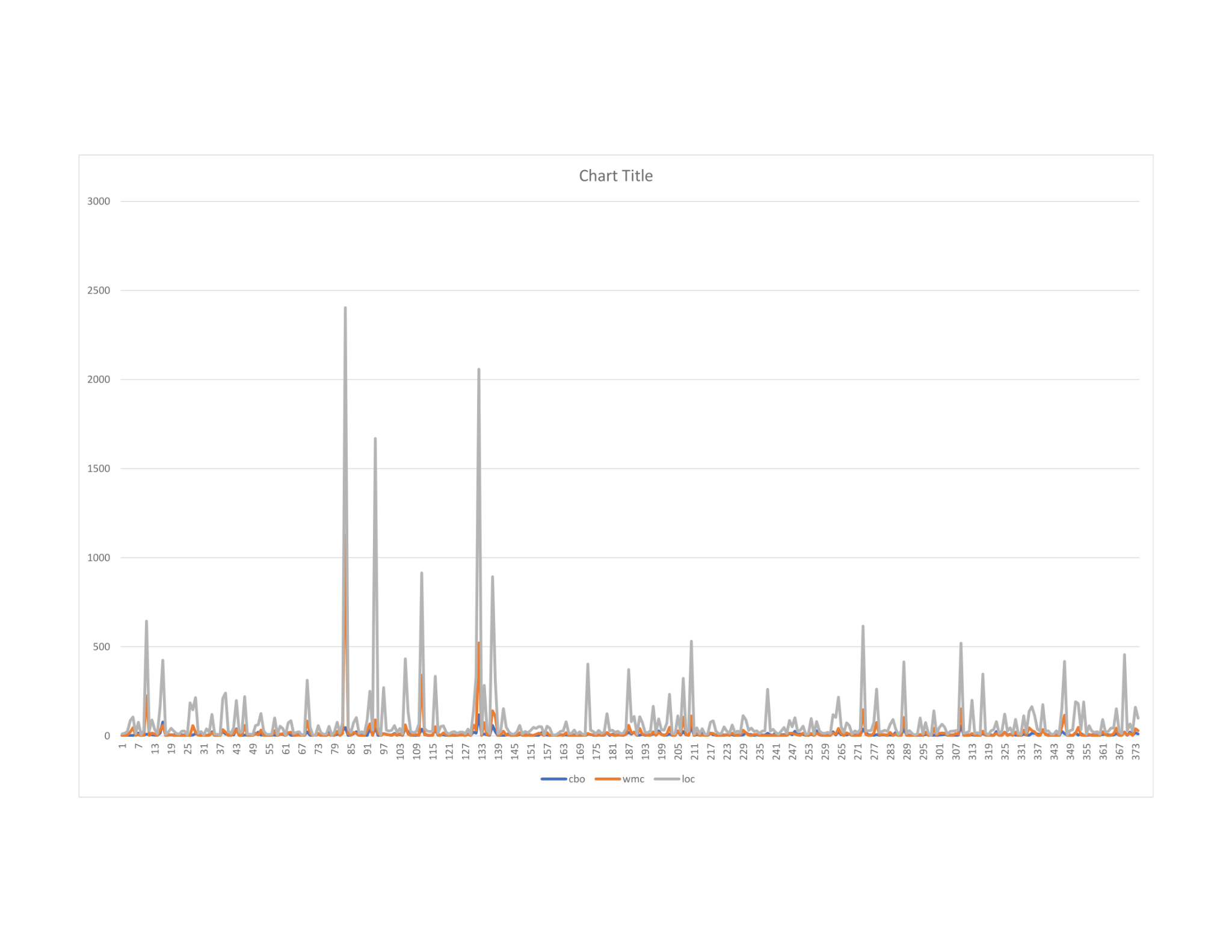
**Line Charts for each project:**

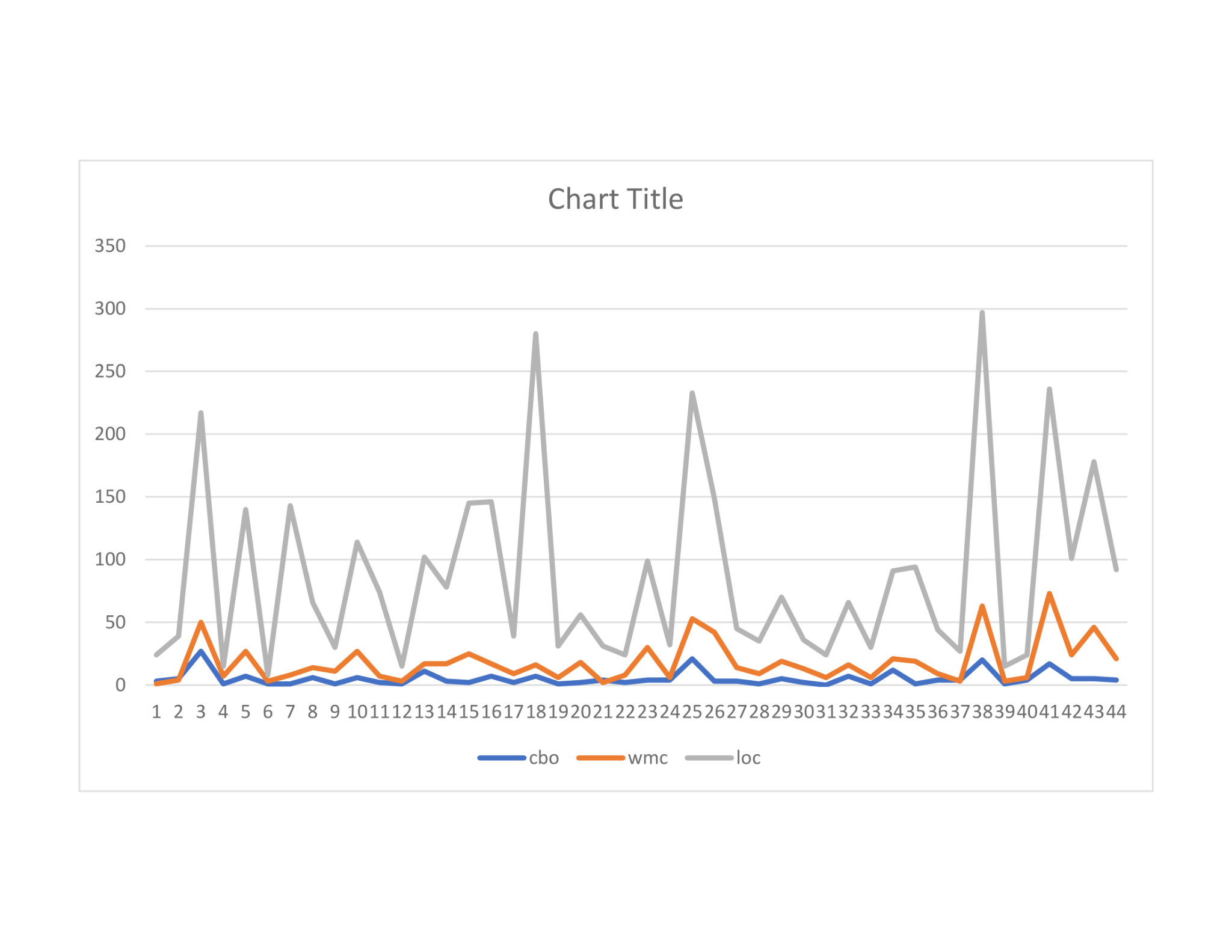


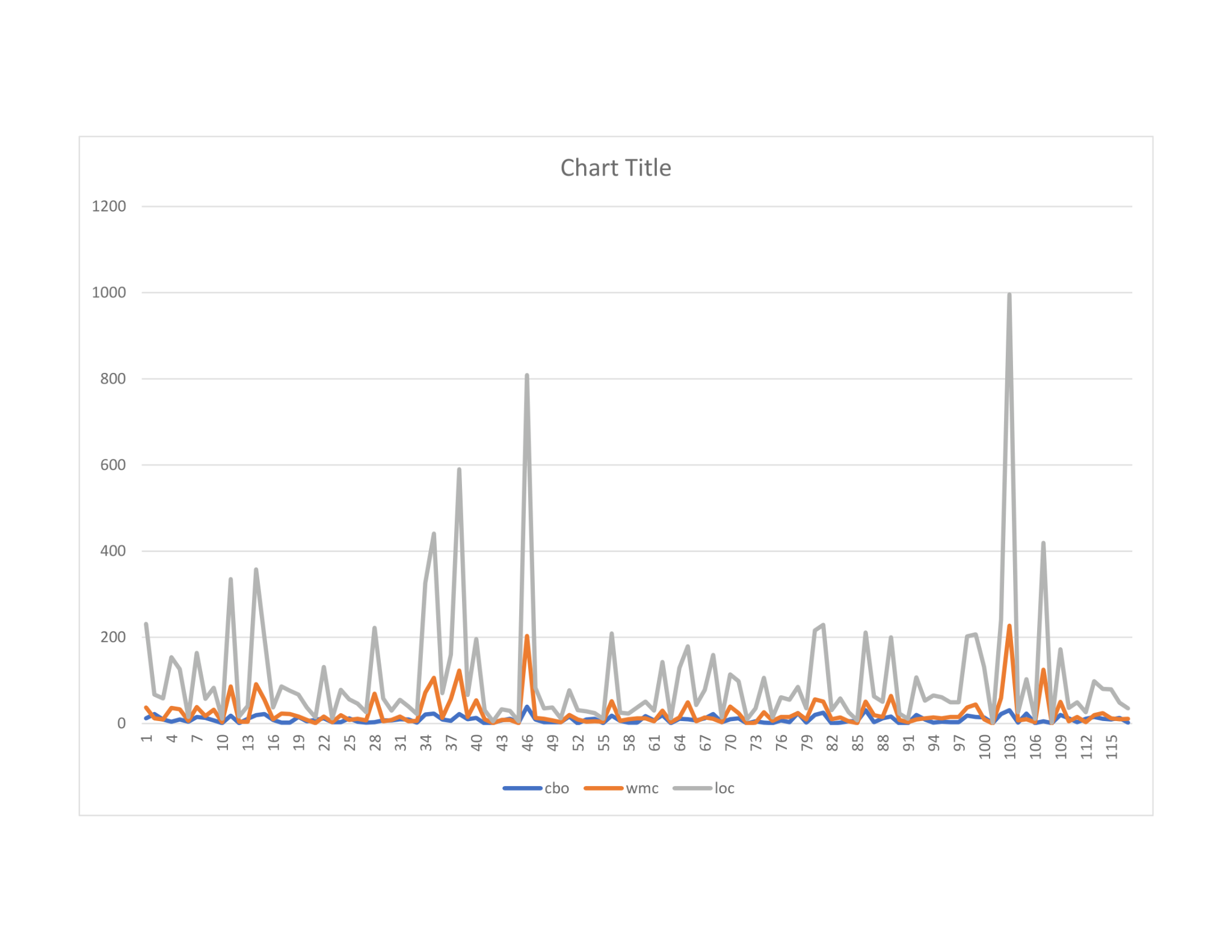
**Project mall4j:**

.

**Project mica:**

**Project Mycat2:**

**Project QtScrcpy:**

**Project AndroidPicker:**

**Results:**

From the above line grapgh, we can see that Mycat2 is the most complex project, with the highest number of classes, LoC, CBO, and WMC. It is followed by mica, which also has relatively high values for all metrics. On the other hand, QtScrcpy is the simplest project, with the lowest number of classes, LoC, CBO, and WMC.

Mall4j and AndroidPicker fall somewhere in between, with moderate numbers of classes and LoC, and relatively low values for CBO and WMC.

Overall, these metrics provide a useful way to compare the complexity of different projects, and can help developers identify potential areas of improvement or risk in their code.

**The summary of the obtained measurements for each of the 5 projects.**

| Project | Number of classes | Low/High LoC | Low/High CBO | Low/High WMC |
| --- | --- | --- | --- | --- |
| mall4j | 279 | 2/196 | 0/35 | 0/41 |
| mica | 296 | 2/627 | 0/40 | 0/209 |
| Mycat2 | 1752 | 2/9248 | 0/376 | 0/3444 |
| QtScrcpy | 44 | 8/29 | 0/27 | 0/73 |
| AndroidPicker | 117 | 2/996 | 0/46 | 0/227 |

# Section 5: Conclusion

Based on the analysis of the selected Java programs, it is clear that there is significant variation in class size across the programs. The smallest program, QtScrcpy, has only 44 classes, while the largest program, Mycat2, has 1752 classes. The average number of classes across all programs is 508.6, indicating that the programs are of moderate size overall.

In terms of maintainability, we can examine the values of the coupling between objects (CBO) and weighted methods per class (WMC) metrics as proxies for maintainability. We can see that there is significant variation in these metrics across the programs, with Mycat2 and mica having particularly high values for both metrics, indicating lower maintainability. On the other hand, QtScrcpy has very low values for both metrics, indicating higher maintainability.

To determine whether there is a correlation between class size and maintainability in the selected Java programs, we can perform a statistical analysis of the data. We could use a correlation coefficient, such as Pearson's correlation coefficient, to determine the strength and direction of any relationship between class size and maintainability. Alternatively, we could use regression analysis to model the relationship between class size and maintainability more explicitly.

The graph shows clear correlation, the spikes in the CBO, WMC follow spikes in the size(LoC).

Overall, the results of this empirical study suggest that class size can have a significant impact on software maintainability, as measured by metrics such as CBO and WMC. Developers should strive to keep their class sizes manageable and to reduce coupling and complexity where possible in order to improve the maintainability of their code.

# References:

Lanza, M., & Marinescu, R. (2006). Object-Oriented Metrics in Practice: Using Software Metrics to Characterize, Evaluate, and Improve the Design of Object-Oriented Systems. Springer Science & Business Media.

B. A. Kitchenham, S. L. Pfleeger, L. M. Pickard P. W. Jones D. C. Hoaglin K. El Emam, and J. Rosenberg. Preliminary guidelines for empirical research in software engineering. IEEE Trans. Software Engineering, 28:721– 734, Aug 2002.

Li, J., Liang, Y., Li, B., & Li, H. (2019). An empirical study on the effect of class size on software maintainability. Journal of Systems and Software, 147, 207-224.

Basili, V. R., Caldiera, G., & Rombach, H. D. (1994). The goal question metric approach. Encyclopedia of Software Engineering, 1994, 528-532.

Chidamber, S. R., & Kemerer, C. F. (1994). A metrics suite for object-oriented design. IEEE Transactions on software engineering, 20(6), 476-493.

Li, L., Jiang, T., Li, M., & Liu, J. (2020). Software complexity metrics for object-oriented systems: A systematic mapping study. Journal of Systems and Software, 167, 110578.

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