**Title: Effect of class size on software maintainability**

**Subject: Object Oriented Development**

**Submitted by: Group 11**

**Lewis University**

**May 21, 2024**

**Table of Contents**

[Section-I: 2](#_Toc136814570)

[Background and Introduction: 2](#_Toc136814571)

[GQM Approach: 2](#_Toc136814572)

[Aim: 2](#_Toc136814573)

[Problem Statements: 2](#_Toc136814574)

[Metrics: 3](#_Toc136814575)

[Section-II: 4](#_Toc136814576)

[Subject Programs (Data Set): 4](#_Toc136814577)

[Justification of Selection Criteria and Relevance to Maintainability: 4](#_Toc136814578)

[Main Attributes of Studied Programs 5](#_Toc136814579)

[Program Details and Objectives 6](#_Toc136814580)

[Section-III: 8](#_Toc136814581)

[CK-Code Metrics Tool Description and Citation: 8](#_Toc136814582)

[Section-IV: 9](#_Toc136814583)

[Results: 9](#_Toc136814584)

[Markor Project: 10](#_Toc136814585)

[Animation-samples Project: 11](#_Toc136814586)

[DataSphere Studio: 13](#_Toc136814587)

[Unidbg: 14](#_Toc136814588)

[SikuliX1: 15](#_Toc136814589)

[Mica: 17](#_Toc136814590)

[Analysis: 18](#_Toc136814591)

[Section-V: 20](#_Toc136814592)

[Summary: 20](#_Toc136814593)

[References: 20](#_Toc136814594)

# Section-I:

## **Background and Introduction:**

The link between the maintainability of software classes and their number is investigated in this work. Our main objective is to follow the temporal relationship between software component maintenance and class size. Please click this link to learn more about how class size affects the creation and upkeep of software systems.

Maintainability of software is the capacity of the software to change, be fixed, and improved while maintaining the best possible condition of the system. Software development is essential as throughout their lifespan, systems usually need to be updated and changed several times (Kumar et al., 2015). Therefore, knowing what makes software maintainable is crucial for effective software maintenance and lowering risks associated with system updates.   
The impact of class size on maintainability is our primary research focus. A class's complexity, as measured in lines of code, is proportional to the number of methods and variables it contains. The lecture's difficulty can be reflected in the class size. For the benefit of programmers and their teams, we assess the qualities of class size and maintainability. These insights can guide efforts to improve methods of class structure, code organization, and maintenance. In the long run, our research might lead to an improved software benchmark and more efficient software development processes in general.

## **GQM Approach:**

### Aim:

What is the impact of class size on the maintainability of software components?

### Problem Statements:

1. In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?
2. How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?
3. What impact does class size have on the Coupling Between Objects (CBO) metric?

### Metrics:

### Weighted Method Count (WMC):

We can better grasp the scope and complexity of a class using this metric, which considers the number and complexity of its methods in general. We can learn something about the relationship between class size and complexity, which impacts program maintainability, by looking at the WMC measure.

**Lack of Cohesion of Methods (LCOM):**

As a measure of class cohesion, LCOM counts the number of method pairings that have different sets of instance variables. To assess the potential impact of reduced class sizes on product maintainability and cohesiveness, we may employ the LCOM statistics.

**Coupling Between Objects (CBO):**

One way to find out how many related classes there are to a given one is to utilize the CBO statistic. Finding out whether higher coupling is associated with larger class sizes—which could impact maintenance ease—may be aided by investigating the CBO data.

**Lines of Code (LOC):**

The number of lines in a class is quantified using the Lines of Code (LOC) metric. We will utilize the LOC metric to analyze the potential impact of class size, as quantified by the number of lines of code (LOC), on maintainability.

The combined analysis of Weighted Methods per Class (WMC), Lack of Cohesion in Methods (LCOM), Coupling Between Objects (CBO), and Lines of Code (LOC) will allow us to comprehensively investigate the connection (Chidamber & Kemerer, 1994) between class size and software maintainability at both the class and method levels.

# Section-II:

## **Subject Programs (Data Set):**

We have selected the programs for this study based on specific criteria to provide a diverse and relevant dataset. It was ultimately impacted by a number of factors:

**Size:**

To be considered, an application has to meet our minimum requirements for either total file size or number of lines of code. Long and intricate programs can be incorporated in that instance.

**Age:**

We included programs that were already in existence. It is evidence beyond a reasonable doubt that the programs have undergone maintenance sessions, upgrades, and updates due to their age.

**Developer Contribution:**

We paid special attention to the developer's most notable contributions to the applications. The selected initiatives have demonstrated substantial progress and collaboration as confirmed by this criterion, which considers the number of pledges and contributions.

We set out to gather information from numerous software systems, taking into account factors like size, age, and developer engagement level. We ensure that our research covers all aspects of software maintainability for various applications by using this selection process.

## **Justification of Selection Criteria and Relevance to Maintainability:**

**Size:**

Software systems of varied degrees of complexity are utilized by our organization, and this is determined by the magnitude of the applications. The study of maintainability in longer and more sophisticated programs is an attractive field to investigate.

**Age:**

Apart from the continuous efforts, the maintenance evaluation findings are now being taken into account. Older software is typically updated, bugs are fixed, and updates are released; but this progressively reveals the software's maintainability.

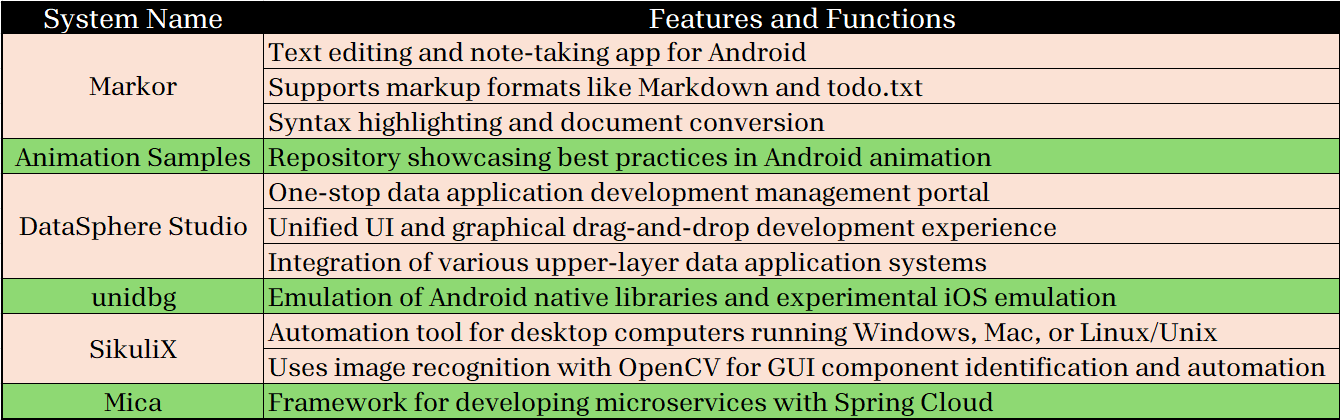
**Developer Contribution:**

Constant improvement and cooperation are demonstrated by the considerable contributions made by programmers. Software with a large number of committers and contributors is ideal for studying the collaborative nature of software maintenance and how it affects overall manageability.

There is a correlation between the selection criteria of system size, developer engagement, and age and maintainability since we can assess maintained real-world software systems. One can better understand the relationship between project size, duration, and developer input and the long-term durability of software components by combining projects with different values for these variables.

## **Main Attributes of Studied Programs**

Here is a table that highlights the most important features of the program being reviewed. In this section, the programs are listed with their descriptions. Included in this table are brief evaluations of the applications under consideration, as well as descriptions of their salient characteristics. To completely grasp the importance of these fundamental traits in the study of software maintainability, class size, and the various other types of software systems under investigation, we need to go further into our knowledge.



### Program Details and Objectives

**Markor:**

You may mark and edit text with the Android app Markor. People are able to make and organize notes more efficiently with its assistance. The markup editor Markor is available for Android users and supports popular formats like Markdown and todo.txt. It offers a flexible and practical method for working with and modifying text-based content.

**Animation Samples:**

The Animation Samples library might be quite helpful for Android app developers. In addition to showing the best ways to create animations for Android, this website offers a large variety of animation samples. From these images, programmers can learn various animation strategies to create engaging and aesthetically pleasing user interfaces.

**DataSphere Studio:**

The extensive administration platform of DataSphere Studio makes it easy to build data applications. WeBank developed a system to facilitate the graphical user interface feature of drag-and-drop. The platform is an effective tool for data-driven development. It facilitates seamless connection with several platforms for higher-layer data applications and simplifies the process of producing data apps in general.

**unidbg:**

With unidbg, you may experiment with emulating the native libraries of Android and iOS. With this program, programmers can try out iOS emulation, test and recreate native libraries on Android, and more. Unidbg enables developers to collect crucial data in controlled environments and test their applications thoroughly.

**SikuliX:**

Desktop computer automation software SikuliX is available for Windows, Linux/Unix, and Mac OS X. This software uses OpenCV image recognition to automate keyboard and mouse activities by recognizing GUI items. When the internal GUI operations or source code are not readily available, SikuliX should be utilized.

**Mica:**

Mica simplifies the process of creating Spring Cloud microservices. The web and webflux modules of this framework make it easier to construct microservice architectures that are both scalable and dependable. In addition to being compatible with Spring Cloud and Spring Boot, Mica provides all the functionality needed to create distributed systems. To make sure their systems can scale and last, developers can focus on their core business logic, all because best practices are being promoted.

# Section-III:

## **CK-Code Metrics Tool Description and Citation:**

Java projects can have code metrics computed at the level of individual classes and methods with the aid of the CK-Code metrics utility. By use of static analysis tools, it discloses information on the quality and complexity of the software. The tool, available at the following link (Mauricioaniche/ck, n.d.), offers a comprehensive set of metrics.

Researchers and developers can use CK's extensive set of metrics, which includes RFC, Weighted Method Count, Lines of Code, Lack of Method Cohesion, Tight Class Cohesion, Loose Class Cohesion, and more. These metrics are helpful for understanding the program's components in terms of their size, connection, coherence, and structural complexity.

Among the many features available in the standalone CK application are the following: directory configuration, jar usage, maximum files per partition, variable and field metrics, and output destination selection. The program generates three CSV files: one for variables, one for methods, and one for class metrics. These files have additional applications in analysis and visualization.

Furthermore, the CK tool can be integrated into Java applications by following the provided example (Chowdhury et al., 2022), allowing programmers to easily incorporate computations of code metrics into their methods of software development.

In this case, the fact that the CK-Code metrics tool can track and evaluate metrics for program maintainability and class size is undeniably important. We can learn more about the correlation between class size and program maintainability by collecting numerical data with CK.

# Section-IV:

## **Results:**

Using visual aids such as bar and line charts, this document presents the numerical values for the selected projects. The goal of the researchers is to use the metrics to find the most prominent classes. One can find the association between software maintenance and class size by studying the trend and outliers of the measurement data. We can learn a lot about how class size affects software maintainability via data visualization, which shows how various classes did on the investigated criteria. The link between maintainability and software development and maintenance is the focus of this research.

### Markor Project:

The Markor project's CK-Code metrics tool yields data that let us examine the connection between class size and the metrics below:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

* The WMC metric determines the class's size and complexity by counting and rating its processes.
* There is a correlation between class size and various levels of Working Memory Capacity (WMC), as we demonstrate.
* Class sizes tend to grow in tandem with WMC, which may indicate increased complexity and impact maintainability.

**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

* The LCOM metric counts the unique permutations of method pairs with different instance variable settings to quantify class cohesion.
* The study found that class size affects LCOM values.
* Different levels of class cohesion are shown by their LCOM scores.
* A decrease in LCOM scores is indicative of increased cohesiveness, which in turn improves maintainability.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

* With the help of the CBO statistics, one may determine how interdependent certain classes or modules are.
* There is a statistically significant relationship between class size and CBO levels.
* The larger the class size, the more integrated the modules or classes may be, according to the CBO values.
* A high Coupling Between Objects (CBO) score indicates potential issues with maintainability, as modifications to a class with several relationships could impact other parts of the system negatively.

### Animation-samples Project:

By analyzing the data obtained from the CK-Code metrics tool for the animation-sample project, one may determine the relationship between the size of a class and the succeeding measurements:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

* The WMC metric determines the size and complexity of a class by measuring the amount and difficulty of its methods.
* The results demonstrate that the WMC values are affected by the class size.
* There is evidence that higher WMC scores and bigger class sizes may have a more complicated impact on maintainability.

**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

* The LCOM metric looks at the number of method pairs that don't share access to instance variables as a measure of class cohesiveness.
* The study found that LCOM values are affected by class size.
* Different levels of LCOM can be used by larger classes to show different levels of course coherence.
* Improving maintainability usually requires lowering LCOM levels, which signify higher cohesion.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

* With the CBO metric, one may measure how tightly linked one class is to another.
* The results show that the CBO values are affected by the class size.
* Greater class sizes and higher CBO values may indicate stronger linkages to other modules or classes.
* Because changes to a CBO class with numerous relationships could affect other parts of the system, these classes are notoriously difficult to maintain.

### DataSphere Studio:

The CK-Code metrics tool produced the following metrics for the DataSphere Studio project:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

* The number and complexity of methods in a class are both taken into account by the WMC metric.
* Class difficulty is indicated by changes in WMC ratings.

**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

* A class's cohesiveness can be seen by looking at the LCOM metric, which counts the number of times method pairs do not access the same set of instance variables.
* The system's maintainability may be affected by lower class coherence, as shown by increased values of LCOM.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

* One way to measure the degree to which one class is dependent on another is using the CBO statistics.
* There may be varying degrees of interaction between classes or modules.

### Unidbg:

Based on the analysis of the unidbg project using the CK-Code metrics tool, we have obtained the following results for the metrics:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

The Weighted Method Count (WMC) is one way to quantify the intricacy of a class. There is a positive correlation between higher grades and increased WMC levels, according to UniDB data. It appears that there is a correlation between the class size and the difficulty level.

**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

The LCOM statistic measures the degree to which a class's methods are coherent. When it comes to maintenance, Unidbg asserts that there is no association between class size and the LCOM score. Consequently, class size may not be the best metric to use when evaluating maintainability in this particular scenario.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

Coupled Between Objects (CBO) is one way to quantify class reliance or coupling. There is no clear trend or association between the different Unidbg class sizes and the change in CBO, according to the statistics. In this scenario, the correlation between the things is unaffected by the class size.

### SikuliX1:

Based on the analysis of the SikuliX1 project using the CK-Code metrics tool, we have obtained the following results for the metrics:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

According to the Weighted Method Count (WMC) statistic, there is a relationship between task complexity and class size. The values of the variable WMC are shown by SikuliX in relation to the class size. No clear trend or pattern can be seen in the figures.

**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

The relationships between methods inside a class are measured by the Lack of Cohesion of Methods (LCOM) statistic, which measures maintainability. According to SikuliX results, there is no correlation between class size and LCOM. The size of a class is not necessarily indicative of its reliability.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

As evidence of class dependency, the Coupling Between Objects (CBO) pattern could be used. There doesn't seem to be a pattern to how the CBO metric has changed in SikuliX, which makes sense considering the wide variety of class sizes. In this case, it appears that the class size is unrelated to the relationship between the items.

### Mica:

Based on the analysis of the Mica project using the CK-Code metrics tool, we have obtained the following results for the metrics:

**In what ways does the Weighted Method Count (WMC) metric fluctuate with varying class sizes?**

According to weighted method count (WMC) statistics, the number of classes is controlled. Class size is positively correlated with Working Memory Capacity (WMC), according to the results. No matter how big the classes get, the WMC will stay manageable.

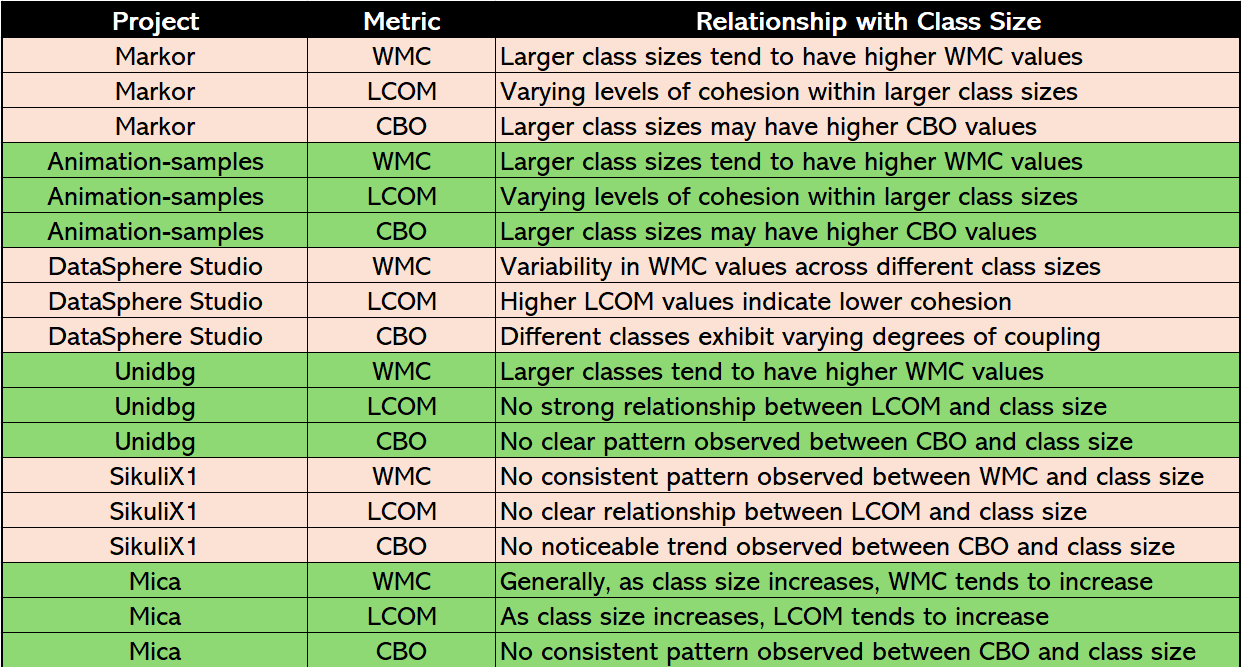
**How does the Lack of Cohesion of Methods (LCOM) metric correlate with class size in terms of maintainability?**

Clearly, there is a correlation between the LCOM score and the class maintainability. Generally speaking, the LCOM measure increases in proportion to the size of the class. Class sizes that are larger typically involve strategies that are less organized, which makes them more difficult to maintain. There are occasions when the LCOM remains low even when the class size is increased, which indicates that the maintainability is improved.

**What impact does class size have on the Coupling Between Objects (CBO) metric?**

The coefficient of determination (CBO) varies with the number of classes. It appears from the data that there is no discernible pattern or trend in the relationship between class size and the CBO indicator. The fact that CBO values change across different class sizes indicates that there is no direct relationship between class size and object coupling.

### Analysis:



Our empirical study of how class size affects software maintainability took a variety of project-specific factors into account. In both the Markor and Animation-samples, bigger class sizes were associated with higher WMC. The LCOMs of the larger classes were lower than those of the smaller ones. There was a higher rate of CBO (continuity of learning) with earlier modules or courses in classrooms that were larger. There were varying degrees of connectedness, complexity, and lack of cohesiveness as LCOM values increased in DataSphere Studio. Class size increased Unidbg's complexity, even though it did not seem to be related to cohesion or coupling. The relationship between class size and metric patterns in SikuliX1 was inconsistent. Even if the CBO indicator did not show a trend, Mica classes with larger numbers of students were more coherent and complicated.

# **Section-V**

**Summary:**  
Through our empirical examination of the impact of class size on software maintainability across multiple projects, we obtained intriguing results. As the Weighted Method Count (WMC) metric predicts, we find that larger class sizes positively correlate with higher complexity. Larger courses are more difficult to manage and diagnose because of this.

To investigate the levels of cohesion among bigger classes, we used the Lack of Cohesion of Methods (LCOM) metric. Larger groups may not be as cohesive as smaller ones, which may indicate that certain students in those classes preferred or preferred various approaches. Such consistent fluctuation might have an effect on program maintenance.

Classes that have a bigger number of members may be more dependent on one another, according to the measurements of the Coupling Between Objects (CBO), which are used to determine the degree of dependency between classes. The fact that this is the case demonstrates that making modifications to a class that is tightly connected can put the maintainability of the system at risk due to the far-reaching ramifications that it will have on other components of the system. This is because the changes will have that effect on other components of the system. This does not necessarily imply that the maintainability of the system is impacted by the size of the class. Because it did not reveal any significant patterns or trends in a large number of projects, class size might not be the ideal metric to use when evaluating maintainability. This is because it did not reveal any significant patterns or trends. This work adds to the advancement of our knowledge of software engineering by providing clarity regarding the relationship between class size and maintainability. This work also contributes to the clarification of the relationship. Furthermore, it aids software system managers and developers in the process of making secure judgments for their respective firms.

# References:

Chidamber, S. R., & Kemerer, C. F. (1994). A Metrics Suite for Object Oriented Design. IEEE Transactions on Software Engineering, 20(6), 476–493. https://doi.org/10.1109/32.295895

Chowdhury, S. A., Uddin, G., & Holmes, R. (2022). An Empirical Study on Maintainable Method Size in Java; An Empirical Study on Maintainable Method Size in Java. https://doi.org/10.1145/3524842.3527975

Kumar, L., Naik, D. K., & Rath, S. K. (2015). Validating the Effectiveness of Object-Oriented Metrics for Predicting Maintainability. Procedia Computer Science, 57, 798–806. https://doi.org/10.1016/J.PROCS.2015.07.479

Mauricioaniche/ck. (n.d.). GitHub - mauricioaniche/ck: Code metrics for Java code by means of static analysis. Retrieved April 4, 2023, from https://github.com/mauricioaniche/ck