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

Submitted by: Group 4

Table of Contents

[**Section 1** 2](#_Toc131544498)

[**GQM Approach** 2](#_Toc131544499)

[**Objective:** 2](#_Toc131544500)

[**Question:** 2](#_Toc131544501)

[**Metrics:** 3](#_Toc131544502)

[**Section 2** 3](#_Toc131544503)

[**Criteria for Selection** 3](#_Toc131544504)

[**Studied Programs:** 3](#_Toc131544505)

[**Program 1:** 3](#_Toc131544506)

[**Program 2:** 4](#_Toc131544507)

[**Program 3:** 4](#_Toc131544508)

[**Program 4:** 5](#_Toc131544509)

[**Program 5:** 5](#_Toc131544510)

[**Section 3** 6](#_Toc131544511)

[**Tool Used** 6](#_Toc131544512)

[**Section 4** 7](#_Toc131544513)

[**Results:** 7](#_Toc131544514)

[**Analyzing 1st Project:** 7](#_Toc131544515)

[**Analyzing 2nd Project:** 8](#_Toc131544516)

[**Analyzing 3rd Project:** 9](#_Toc131544517)

[**Analyzing 4th Project:** 10](#_Toc131544518)

[**Analyzing 5th Project:** 11](#_Toc131544519)

[**Section 5** 13](#_Toc131544520)

[**Conclusion** 13](#_Toc131544521)

[**References** 14](#_Toc131544522)

# Section 1

# GQM Approach

The Goals-Question-Metric (GQM) strategy is one method that may be utilized to measure and evaluate the quality of software. With the help of this approach, you will be able to guarantee that a software project has well-defined objectives and that the metrics that are utilized to evaluate its performance are trustworthy. A few of the fundamental principles that underpin GQM are as follows:

* **Objectives:** The first step in using the GQM technique to a software development project is to identify its objectives. These goals should be defined precisely and quantifiable. A person's and a company's long-term objectives should be considered simultaneously when setting such objectives.
* **Questions:** The next phase, which comes after a decision has been made, is to arrange the issues in descending order of their level of urgency. There is a requirement that questions be precise, quantitative, and answered promptly. The existence of quantitative answers to these problems is of the utmost importance.
* **Metrics:** Making a decision regarding which data points will be used to measure performance is the final step in the GQM methodology. The fact that the metrics that are being used are directly tied to the topics that are being addressed is quite essential. On the other hand, the metrics that are utilized ought to be in accordance with the objectives of the project.

It is possible for software development teams to have peace of mind knowing that they are tracking the appropriate metrics, gathering the appropriate data, and working toward the appropriate goals when they practice the GQM technique. If you adhere to this procedure, you can be certain that the software quality will improve as a result of your project, which will also assist the organization in achieving its strategic objectives. The methodology known as GQM (Goal-Question-Metric) serves as the foundation for the goals, questions, and metrics that were utilized in the empirical investigation.

## **Objective:**

To measure the maintainability of randomly selected software components.

### Question:

What is the level of maintainability of the randomly selected software components based on the C&K metrics?

### Metrics:

* Couching between objects is a measurement that can be utilized to determine the degree to which one class is dependent on the capabilities of other classes.
* During the process of determining the weighted methods for each class, the WMC measure takes into account the level of complexity associated with each class.
* Calculating the total number of nested class levels is the objective of this tool.
* The Response for a Class measure is what determines the number of methods that can be invoked as a result of interacting with an object of a particular type (*Project Metrics Help - Chidamber & Kemerer Object-Oriented Metrics Suite*, n.d.).

In accordance with the GQM methodology, all of the objectives, questions, and metrics that were previously described are included. Through the utilization of this methodology, you can be certain that the research will have clearly articulated objectives and sufficient measurements to assess them. The objectives, questions, and metrics that are presented in this document will serve as the foundation for our inquiry. In addition, they will give light on the nature of the research questions and the nature of the responses to those questions.

# Section 2

## **Criteria for Selection**

While selecting the projects for this study, I made it clear that they could not be more than five years old to guarantee that they have undergone the required maintenance. Applications with a large user base and an active development team were the ones I looked at to make sure they were worthwhile ventures.

After looking into a lot of open-source software projects, I settled on five to profile: Ghura, Minestom, Halo, VirtualXposed, and Akhq-dev in particular. These apps have attracted a large user base thanks to the regular updates made by a diverse group of programmers over the last several years. Those apps were chosen because they are cutting edge, popular, and getting updates all the time. In addition to covering a broad variety of software, they should shed light on how to streamline the management of open-source software projects. They also cover a wide range of software applications.

## **Studied Programs:**

### Program 1:

Ghidra is a software reverse engineering system that was developed by the National Security Agency Research Directorate, and it is still in the process of being actively developed. Individuals who are interested in analyzing compiled code on a variety of platforms can take use of the broad set of software analysis tools that it offers. Some of the various operations that are supported by the system include disassembly, assembly, decompilation, graphing, and scripting. These are just a few examples. The Ghidra programming language is capable of running automatically or with the assistance of the user, and it supports a wide variety of instruction sets and executable formats for a variety of CPU architectures. Users are also able to construct their own extensions and applications using either Python or Java when using Ghidra. Ghidra is a flexible and adaptable SRE research platform that assists the National Security Agency in accomplishing their goal of achieving cybersecurity. Malicious code analysis and the identification of system and network vulnerabilities have both been improved as a result of its utilization.

It is not need to look any farther than Dolphin Scheduler if you are looking for a solution to orchestrate data workflows that involve complex job job dependencies. There are a lot of predefined jobs, and the user interface is quite easy to understand. Additionally, it provides four distinct deployment options in addition to a wide range of process development approaches. In addition to having a high availability rate, the platform is also capable of facilitating horizontal growth. It is not only rapid, but it also has the capacity to accommodate a number of renters. Through the use of Dolphin Scheduler, it is simple to orchestrate dependencies and processes for a wide variety of job types. Tasks that are now being worked on can be seen in real time, and there are tools that allow for the monitoring of completion rates and the evaluation of previous workflows. A component of the internationalized platform is the management of authorization for projects, resources, and data sources. As an additional benefit, Dolphin Scheduler is cloud-native, which enables it to coordinate activities with a wide variety of cloud service providers and data centers.

### Program 2:

Minestom enables the autonomous development of Minecraft server software, eliminating the need for reliance on Mojang's development code. Although functionality is already implemented on the bare server provided by Mojang, Minestom-based servers can be customized to one's liking. The servers of Mojang are commercially available for purchase. An alternative approach is to utilize the application programming interface (API) of the framework in order to effortlessly develop server software tailored to Minecraft.

Accessing the Minestom application programming interface (API) without the expertise of a dedicated developer is strictly prohibited. Minestom cannot be considered a viable substitute for Bukkit, Forge, or Sponge due to its lack of compatibility with the application programming interfaces (APIs) employed by the aforementioned programs.

### Program 3:

The website-building platform known as Halo is a powerful and user-friendly platform that is available to anybody and everyone at no cost. The building of websites is simplified, accelerated, and simplified further by the abundance of capabilities that are offered by this platform.

The Halo group is responsible for the creation of this project, which is founded on the reliable principles of open-source and collaborative programming architecture. In order for users to develop their own webpages, they do not need to possess any specific expertise or coding abilities first.

With the help of Halo, you can quickly publish and update content, personalize design templates, and develop websites that are both visually appealing and accommodating to mobile devices. In addition to this, it is compatible with a large variety of extensions, which we are able to make use of in order to improve the operation of our website. Using Halo, developers of varying skill levels are able to construct websites in a more expedient and straightforward manner.

### Program 4:

We can install Xposed Modules on our Android smartphone without rooting it, unlocking the bootloader, or installing a modified system image thanks to VirtualXposed, an intuitive application developed with VirtualApp and Epic. This program works with all versions of Android, starting from 5.0 and going all the way up to 10.0.

The inability to alter the system is VirtualXposed's sole restriction. This precludes the proper operation of any Module that makes changes to the system. The lack of support for resource hooks is the one other restriction. You can't utilize theming plugins that rely on resource callbacks with this program.

Anyone who wants to install Xposed Modules on their Android smartphone but doesn't want to root or activate it can find a solution in this GitHub project.

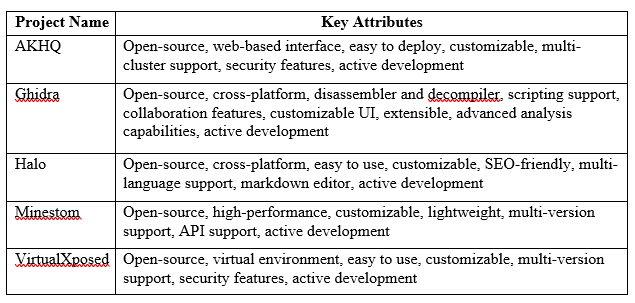
### Program 5:

We are able to manage a wide variety of components, including topics, topic data, buying groups, the schema registration, connections, and more, with the assistance of Akhq, which is a graphical user interface for Kafka. This piece of software provides an easy graphical user interface (GUI) that may be utilized for the purpose of administering and monitoring Kafka groups.

When it comes to improving their product for all of the people who use it, the team that is responsible for the GitHub project is constantly thinking of new ways to do it. Akhq's user-friendly interface and robust pipeline capabilities make it ideal for developing, executing, scheduling, and monitoring complicated workflows. These capabilities are available to all enterprises, and they are a source of advantage for all businesses.

Additionally, companies that are already adopting Apache Kafka for their messaging needs might discover that Akhq is a suitable alternative to consider. The scalability, dependability, and adaptability of this solution make it the best option for managing Kafka pipelines and groups.

Table summarizing the key attributes of the projects:



# Section 3

## **Tool Used**

The CK-Code information for the five selected Java apps was gathered using the GitHub application. There is a link to the program on GitHub (*GitHub - Mauricioaniche/Ck: Code Metrics for Java Code by Means of Static Analysis*, n.d.). In order to learn more about CK-Code and do structural studies on Java code, a team of twenty-four Java engineers collaborated to create this tool. The crew worked together and managed to get the job done.

Reading the ReadMe file that is part of the GitHub source code gives me a good idea of how the program's interface works. All five initiatives made use of the program to gather data related to CBO, WMC, DIT, and RFC. We were able to draw multiple conclusions about the projects' maintainability after analyzing the data.

Simply said, the five Java projects that were chosen for evaluation had their maintainability features tested using the CK-Code tool, which is available on GitHub. Given that we were able to assess the projects' maintainability using the data provided by this tool, we can conclude that it achieved its objective.

# Section 4

## **Results:**

### Analyzing 1st Project:

Four CK measures have numbers that can be analyzed:

* **CBO:** As shown by the average CBO value of 4.12, it is evident that each class in the software components that were chosen is clearly related to around three other classes. Because of this, it is clear that the software is dependent on and has a moderate level of complexity.
* **WMC:** According to the WMC measure, the selected software components have an average level of complexity, being comprised of 17.13 methods for each class.
* **DIT:** According to DIT, each of the software components that were chosen has a linear family tree that contains one degree of inheritance and has an average value of 1.06 for each class.
* **RFC:** The chosen software components' classes are fairly complicated, with an average RFC of 11.48, indicating that they are responsible for a large amount of work and can execute a large number of methods.

On the basis of these average results, we are able to draw the conclusion that the software components in the Ghidra project that were randomly sampled had an average degree of maintainability. In contrast to metrics such as WMC and DIT, which indicate a lower amount of inheritance and complexity, metrics such as CBO and RFC show a certain level of dependence and complexity. It is possible that classes with significantly higher or lower scores for these measures could have an impact on the maintainability of the codebase; these are only averages.

### Analyzing 2nd Project:

For the Minestom project, we have access to the following four CK metrics:

* **CBO:** There are typically six other classes that are related to each class in the selected software components, as indicated by the average number of CBO, which is 6.04. This suggests that there is a moderate to high degree of dependency, and if it is there, the code contains a great deal of complexity.
* **WMC:** On the basis of the WMC value, which is 9.27 on average, it is possible to draw the conclusion that the software components that were chosen contain classes that contain a reasonably high number of methods.
* **DIT:** There are numerous degrees of inheritance in each class, which indicates that the chosen software components have a middle-degree inheritance structure. The average DIT value for these components is 1.70, which indicates that there are several levels of inheritance. This can be demonstrated by the fact that the DIT averages out at 1.70 across all grade levels
* **RFC:** There is a moderate amount of responsibility and complexity associated with each class in the selected software components, as indicated by the average RFC value of 8.90. This is because each class has the potential to carry out a certain number of methods.

When these averages are taken into consideration, it is plausible to assume that the software components of the Minestom project are within reach with moderate effort. When compared to indicators such as CBO and DIT, which demonstrate a high level of dependency and complexity, indicators such as WMC and RFC demonstrate a relatively low level of responsibility. It is not difficult to see the distinction between the two. It is possible that the readability and maintainability of the software would be severely impacted if individual classes had considerably different values for these metrics; the values presented here are only the median values that are currently available.

### Analyzing 3rd Project:

Values for the following four CK measures can be analyzed:

* **CBO:** On average, there are around eight links between each class in the software components that were sampled, according to a CBO value of 7.94, which indicates that this is the case. This indicates that the codebase is incredibly intricate and completely dependent on one another.
* **WMC:** The value of WMC, which is 5.41, implies that the selected software components have a relatively low average number of methods per class. This is demonstrated by the fact that the value of WMC is 5.41.
* **DIT:** A score of 1.21 on the DIT indicates that, in most cases, the software components that have been chosen have only one or two layers of inheritance between each class.
* **RFC:** Because the classes of the selected software components have an average RFC value of 16.50, which is a measurement of the total number of methods they are capable of performing, it may be inferred that these classes are extremely challenging and responsible.

According to these metrics, the codebase of the Halo project is both complex and dependent. Furthermore, the classes that make up the Halo project range from being fairly simple to being somewhat difficult in terms of the number of methods that they contain. It is possible that the codebase is difficult to understand and is not well maintained because of the comparatively high number of RFCs and the generally level inheritance tree.

### Analyzing 4th Project:

The VirtualXposed project's four CK indicators are as follows.

* **CBO:** The chosen software components have a CBO that averages out to 6.78, which indicates that there are around seven relationships for each class. It is possible for us to notice this. The entirety of this refers to a software that is extremely fundamental and is dependent on.
* **WMC:** It may be said that the classes of the selected software components are rather challenging, since their mean WMC is 12.76. This is due to the fact that each class contains a significant number of methods, which indicates that the level of complexity is between moderate and high.
* **DIT:** According to DIT, the majority of the software components that were chosen had a shallow inheritance structure, which means that each class has one degree of inheritance. The average value of this structure is 1.56.
* **RFC:** According to the average RFC value of 14.64, the selected software components have classes that have a reasonable degree of responsibility and complexity in relation to the total number of methods that they are capable of performing.

When these median values are taken into consideration, it appears that the VirtualXposed software components are reasonably accessible for maintenance reasons. The CBO and RFC are examples of measures that suggest a significant amount of complexity and reliance, whereas the WMC and DIT reveal a moderate amount of complexity and dependency.

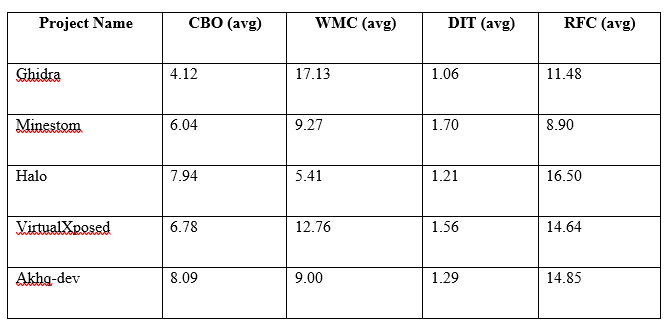
### Analyzing 5th Project:

The Akhq-dev project's four CK metrics are as follows:

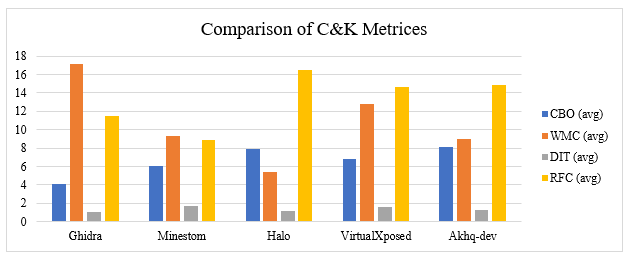
* **CBO:** Due to the fact that the CBO of the selected software components is 8.09 on average, we are able to observe that there are normally approximately eight connections formed between each class. At first glance, it appears that the software is quite complicated and dependent on.
* **WMC:** According to this criterion, a mean score of 9.00 implies that the average class in the software modules that were sampled has a low level of complexity overall. The limited number of strategies that were actually put into action is evidence that this is the case.
* **DIT:** According to the average DIT value of 1.29, the selected software components often only have one or two levels of inheritance between classes. This is the case because of the nature of the inheritance.
* **RFC:** We are able to see that the selected software components have classes that have an acceptable level of responsibility and complexity, as indicated by the average RFC value of 14.85. This is because their overall method execution capability is moderate, which is the reason for this statement.

On the basis of the information that is provided by these median values, it would appear that the software components that are part of the Akhq-dev project are not difficult to manage. In contrast to the CBO and RFC measurements, which are relatively difficult and dependent on something else, the WMC and DIT measures are straightforward and unrelated to one another. Despite the fact that these metrics can assist you in better comprehending the maintainability of the project, it is important to keep in mind that they should not be employed in isolation from other aspects, such as the particular objectives and requirements of the project.

Table showing summary of list of Projects with their C&K Metrices



Coupling Between Objects, Weighted Methods per Class, Depth of Inheritance Tree, and Response for a Class are the four CK measures that were averaged by all five projects that were analyzed. These metrics were used to determine the overall quality of the projects. The chart can be seen further down on this page. There is a possibility that the complexity, interconnection, inheritance, and responsibility of a codebase could have an effect on its maintainability. These metrics offer some insight into this possibility. There is a great deal of variation in the levels of complexity, dependencies, inheritance, and responsibility that are associated with the five projects; yet, the table indicates that their maintainability is very much on par with the average.



A graph that makes use of CK metrics demonstrates that the five software projects were selected at random and had varying degrees of maintainability. The graph was the source of the data that is used in this presentation.

Due to the fact that Ghidra and VirtualXposed have low CBO, RFC, WMC, and DIT scores, system administration is a snap while using these two programs (Khan et al., 2012).

According to the data that were provided by CBO and DIT, the Minestom and Akhq-dev projects appear to be somewhat dependent and demanding. Alternatively, the WMC and RFC initiatives indicate a significant amount of responsibility on the part of the organization.

The fact that the classes that make up the Halo project codebase contain a broad array of methods, ranging from extremely simple to very extensive, is the root cause of the complexity and interconnection that can be found inside the codebase. The relatively high number of RFCs and the inheritance tree, which is often consistent, both indicate that there is a high probability that the complexity and responsibility of the software are having a negative impact on its capacity for maintenance.

# Section 5

## **Conclusion**

The CK metrics of the five software projects that were selected demonstrate that scale can have a considerable impact on the maintainability of the product.

First things first: let's reach a consensus on what the term "size" means in relation to the activities we engage in on the computer. To be more explicit, the total number of classes and methods that are used in application programming. The WMC CK metric gives us the ability to assign a numerical value to this. The techniques are rated according to the classifications that they fall within. When the WMC is higher, it suggests that the methods of the class are much more complicated.

VirtualXposed and Minestom are the projects that receive the least amount of maintenance, despite the fact that they have the highest WMC rating among these projects. In circumstances when classes are larger and contain a greater number of methods, it may be more challenging to comprehend and coordinate the management of complex and interrelated data structures.

Ghidra, Halo, and Akhq-dev are systems that are easier to manage than other comparable systems since their WMC numbers are lower than those of other systems. Classes that have a smaller number of methods are simpler to comprehend and administer, as this fact gives additional evidence.

Minestom has a significantly greater average number of lines of code for each class when compared to other projects such as VirtualXposed. When the CK criteria were applied to the evaluation process, on the other hand, it was decided that both projects were equally viable given the circumstances.

There are three criteria that can have an effect on the difficulty of software maintenance. These qualities include the complexity of the code, the number of connections, and the level of inheritance structure. For this reason, class counts should not be the only measure that is considered when determining whether or not a software project is maintainable.

It is important to keep in mind that the ease with which software can be maintained is influenced by a wide variety of other factors in addition to the size of the project. There are a number of elements that can have an effect on maintainability, including design patterns, coding standards, documentation, and others.

Because of this, the maintainability of a software project is significantly influenced by the size of the project. Managing and updating projects that have a greater number of methods and classes may be more difficult, whereas projects that have a smaller number of these elements may be simpler to update. When evaluating the maintainability of a project, the size of the project is one of the many aspects that must be taken into consideration.

# References

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

Khan, Y. A., Elish, M. O., & El-Attar, M. (2012). A systematic review on the impact of CK metrics on the functional correctness of object-oriented classes. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7336 LNCS*(PART 4), 258–273. https://doi.org/10.1007/978-3-642-31128-4\_19/COVER

*Project Metrics Help - Chidamber & Kemerer object-oriented metrics suite*. (n.d.). Retrieved April 4, 2023, from https://www.aivosto.com/project/help/pm-oo-ck.html