Contents

[Section 1: 3](#_Toc131603597)

[Introduction 3](#_Toc131603598)

[GQM approach 3](#_Toc131603599)

[Research Objectives 3](#_Toc131603600)

[Research Questions 4](#_Toc131603601)

[Maintainability Metrics 4](#_Toc131603602)

[Section 2: 5](#_Toc131603603)

[Program selection Criteria 5](#_Toc131603604)

[Data set description 5](#_Toc131603605)

[Section 3: 6](#_Toc131603606)

[Tool description 6](#_Toc131603607)

[Section 4: 6](#_Toc131603608)

[Result Analysis 6](#_Toc131603609)

[Section 5: 9](#_Toc131603610)

[Conclusion 9](#_Toc131603611)

[References 10](#_Toc131603612)

**Assignment**

**Research Title:** First empirical study: effect of class size on software maintainability

# Section 1:

## Introduction

Software maintainability is an important nonfunctional requirement for any software products and the companies spent more time and resources to ensure code maintainability. In this research work, the focus is given to understand about the effect of class size on code maintainability [1]. As the thumb rule, it is identified that large size of class results difficulties in managing the code. The size of the class depends on the number of declarations made in the code along with the executable commands. The number of methods present in the code and the total number of attributes contributes to class size.

To understand the impact of class size on code maintainability, the Goal Question Metric (GQM) approach is used and five Java programs with preset conditions are chosen and downloaded from the repository [2]. Then the same is analyzed using the suitable tool to extract the CK metrics. On comparing the same with the optimal value, one can assess the impact of class size on code maintainability.

## GQM approach

In this research work, GQM approach is used for analysis; initially the research objectives and questions are set and then identification of the relevant dataset for the process happens. The ideal tool to extract the maintenance metrics are identified with respect to the chosen objectives [3]. By analyzing on quantitative level, one can well assess whether the mentioned objectives are achieved.

## Research Objectives

The main research objective includes

* To well understand the effect of class size on code maintainability.
* Downloading suitable Java programs as per the specified conditions.
* Extracting the CK metrics using the CK metrics tool.
* Analyzing the metrics with the optimal value.
* Generating graphs to understand about the relationship between class size and code maintainability.
* Result analysis and documentation

## Research Questions

The main research questions include:

* Does class size create an impact on code maintainability?
* How to measure the level of impact because of class size on software programs?
* What are the CK metrics related to code maintainability?
* Identify the optimal value of the CK metrics for code maintainability?

## Maintainability Metrics

The CK metrics related to code maintainability are identified as:

Coupling Between Objects (CBO) – This metrics refers to the number of classes that are coupled, in other words the methods of one class will be accessing the variables of the other. The coupling can happen through method class, class extends, class properties, method of arguments and variables in methods.

Weighted Method Class (WMC) – This metric measure the sum of the complexity of the methods in class and it is mainly used to compute the complexity of the class. Higher the value of WMC, it is the clear indication that the class is more complex [4].

Lack of Cohesion of Methods (LCOM) – This metrics clearly measures the number of not connected method pairs present in the class. It is the representation of the independent parts that does not have any cohesion with other classes. The low cohesion adds up design complexity and the code will be more likely to errors [5].

RFC (Response For Class) – This metric brings up the total number of methods that can be executed in response to the message received by the object of the class. This metric will bring up the summation of the methods of the class.

# Section 2:

## Program selection Criteria

For to understand how class size impacts code maintainability, 5 Java programs which are 3 years old are taken into consideration. The chosen codes are developed in the year 2019 and the main reason for this choice is since codes are old, it would have gone through several maintainability phases and hence analyzing the same would be related to the subject [6].

## Data set description

5 java programs are downloaded from Github and their details are mentioned as below:

|  |  |  |  |
| --- | --- | --- | --- |
| S. No | Program Name | Link | Description |
| 1 | Minestom master | <https://github.com/Minestom/Minestom> | This is the open source program that enables the developers to create their own Minecraft server software without any code from Mojang. |
| 2 | dolphinschedular | <https://github.com/apache/dolphinscheduler> | It is the modern data workflow program with powerful user interface, dedicated to solve complex task dependencies. |
| 3 | linkis master | <https://github.com/apache/incubator-linkis> | It is the program that act sa the middleware between the upper applications and underlying engines focused to provide powerful connectivity, reuse and orchestrations. |
| 4 | wildfire chat | <https://github.com/wildfirechat/im-server> | This program support server based chats and it comes with flutter plugins. |
| 5 | ghidra | <https://github.com/NationalSecurityAgency/ghidra> | This program comes with full featured, high end analysis which comes with inbuilt programs to solve scaling as well as teaming problems. |

# Section 3:

## Tool description

The CK metrics tool computes the class level and method level code metrics for Java projects by making use of the static analysis. Here there is no need for the compiled code. This tool will be making use of the various types of maintainability metrics and measures the code dependencies. This tool is essential for quick code [7] debugging and enables easy extraction of the CK metrics. It is the free source code that supports programming languages like python and Java and it is found to be much useful for real time collaboration.

Any unwanted files and folders can be excluded from the project tree by adjusting the settings [8]. It is noted that this tool comes with multiple extensions of FTP and this permits the code to be synched between the editor and the server without the need of any additional software. Thus in our research work, with the help of this tool, the CK metrics or the chosen 5 Java Programs are well extracted.

# Section 4:

## Result Analysis

5 Java programs are taken to analyze the impact of class size on code maintainability. The metrics that corresponds to code maintainability are identified as Weighted Methods Per Class (WMC), Coupling between Objects (CBO), Response for a Class (RFC) and Lack of Cohesion of Methods (LCOM).

The optimal value for the above metrics is determined as

* WMC - <50
* CBO <=14
* RFC<=43.84
* LCOM<=78.34

For analytical purpose, 2 metrics CBO and WMC are considered and this value is analyzed for all 5 downloaded Java Programs

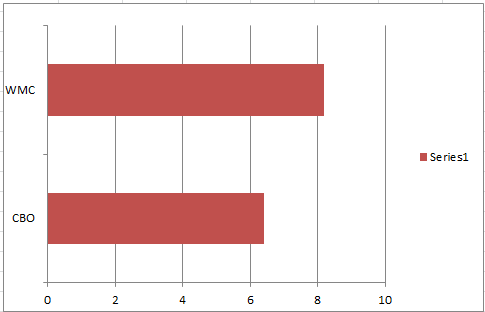
It is necessary to understand about this two metrics. CBO will be counting the number of dependencies which the class holds. All the field declaration, method return types, variable declarations etc. are taken into consideration [9].

WMC refers the number of branch instructions within the class. More the branches, the level of complexity is high and it becomes tough to manage.

**Program 1: dolphinscheduler**

On looking into the program dolphinscheduler, the average CBO value comes to 6.42 and the WMC comes to 8.188.

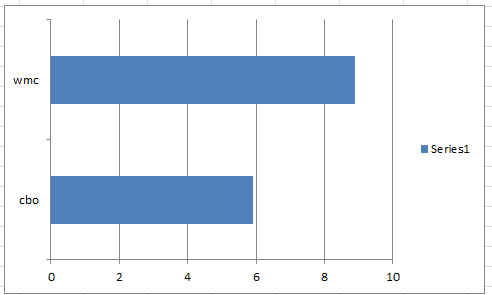
On analyzing the graph, it is identified that both the value falls within the optimal value [10]. This indicates that the code maintainability is achieved even the code is three years old.



*Figure 1: Dolphinschedular Java Program – CK Metrics evaluation*

**Program 2: Minestom–master**

In the case of Minestom–master program, which is also developed in the year 2019, on considering the CBO and WMC value, it is found to be 5.90 and 8.88 respectively.



*Figure 2: Minestom–master program - CK Metrics evaluation*

Here also on analyzing the WMC and CBO values, it comes under the optimal value.

**Program 3: linkis master**

In the case of linkis master, the CBO value comes to 4.54 and the WMC value comes to 8.84.

*Figure 3: linkis program - CK Metrics evaluation*

**Program 4: wildfire chat**

In the program wildfire chat, the CK metrics are extracted and the value for the CBO is 6.27 and for the WMC, it is 22.4.

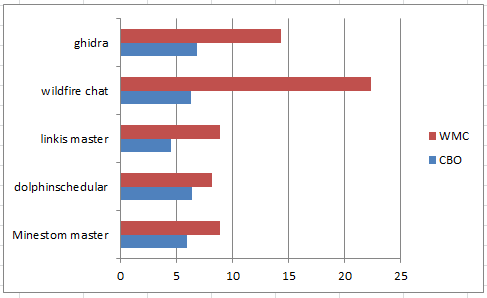
*Figure 4: wildfire chat - CK Metrics evaluation*

**Program 5: Ghidra**

In the program Ghidra, the CK metrics are extracted and the value for the CBO is 6.82 and for the WMC, it is 14.3.

*Figure 5: Ghidra - CK Metrics evaluation*

It is identified that in all the programs, this value comes within the permissible limit, which clearly indicates that the code is maintainable [11].



*Figure 6: Selected Java programs and their respective CK metrics*

This is the overall graph of the metric analytics of the 5 Java Programs and it is found that in all the programs, the code maintainability is achieved.

# Section 5:

## Conclusion

Even in this technology progressed era, software maintenance is one of the challenges faced by numerous software engineers. It is noted that in many cases that the maintainability cost is somewhere equivalent or exceeds the development cost. Under this case, it is essential that the software engineers and developers are trying their best to develop the code that is easily maintainable.

In this research report, the analysis is carried out to understand about the impact of class size on code maintainability. The class length module length has negative impact on code maintainability. For evaluation, the Goal Question Metric Paradigm approach is considered to be the best for the empirical study. The goals are clearly set and research questions are well framed. Here the analysis is carried out to understand where the total number of classes, class attributes, class method impacts the code maintenance.

5 random Java programs are selected, that was developed before 3 years and the main reason for these selection criteria is, since the code gets old, it would have been undergone with numerous maintenance measures.

It is necessary to understand about the metrics that discuss about code maintainability, The CK metrics are identified and they are WMC or McCabe’s complexity, which will be counting the number of branch instructions in the class, also the coupling between the objects also counts to code maintainability. The lines of code explain about the blank lines by ignoring the comments and empty files.

So in the research work, by downloading the selected Java programs, the CK metrics values are extracted with the help of the CK metrics tool. There were numerous CK values extracted and only CBO and WMC is considered for analysis. For each program the average value of the CK metric values are analyzed and then they are compared with the optimal usage value. The below given table clearly mentions the individual metric value (CBO and WMC) for the chosen programs.

|  |  |  |  |
| --- | --- | --- | --- |
| **S. No** | **Java Program** | **CBO** | **WMC** |
| 1 | dolphinscheduler | 6.42 | 8.188 |
| 2 | Minestom–master | 5.90 | 8.88 |
| 3 | linkis master | 4.54 | 8.84 |
| 4 | wildfire chat | 6.27 | 22.4 |
| 5 | Ghidra | 6.82 | 14.3 |

In all the 5 programs analyzed, it is understood that they fall within the optimal range that confirms that code maintainability is achieved irrespective of the number of years of code development. There is variation in the values of CBO and WMC in each code and it depends on the class dependencies, branching of classes, code length and method return types etc. It is essential to ensure that the code follows the standard coding principles to achieve easily maintainability.

# References

|  |  |
| --- | --- |
| [1] | K. E. Emam, S. Benlarbi, N. Goel, W. Melo and H. Lounis, "The optimal class size for object-oriented software," *IEEE Transactions on Software Engineering ,* pp. 1-12, 2002. |
| [2] | R. v. Solingen and G. Caldiera, "Goal Question Metric (GQM) Approach," *Open Access,* pp. 1-9, 2002. |
| [3] | L. Knight, "Why you should be using GQM to measure software quality and value," 2022. [Online]. Available: https://techbeacon.com/app-dev-testing/why-you-should-be-using-gqm-measure-software-quality-value. |
| [4] | J. A. Dallal, "Object-oriented class maintainability prediction using internal quality attributes," *Information and Software Technology 55(11),* p. 2028–2048, 2013. |
| [5] | Mauricioaniche, "mauricioaniche," 2022. [Online]. Available: https://github.com/mauricioaniche/ck. |
| [6] | S. A. Chowdhury, G. Uddin and R. Holmes, "An Empirical Study on Maintainable Method Size in Java," *Open Access,* pp. 1-13, 2022. |
| [7] | Visualstudio, "Code editing redefined," 2023. [Online]. Available: https://code.visualstudio.com/. |
| [8] | M. Schnappinger, A. Fietzke and A. Pretschner, "Defining a Software Maintainability Dataset: Collecting, Aggregating and Analysing Expert Evaluations of Software Maintainability," *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME),* pp. 1-9, 2020. |
| [9] | R. Malhotra, "Handling class imbalance problem in software maintainability prediction: an empirical investigation," *Frontiers of Computer Science,* pp. 1-9, 2022. |
| [10] | Teamscale, "Understanding the Maintainability of Your Code Base," 2023. [Online]. Available: https://docs.teamscale.com/introduction/understanding-the-maintainability-of-your-code-base/#how-not-to-measure-and-assess-maintainability. |
| [11] | Y. Lu, X. Mao and Z. Li, "Assessing Software Maintainability Based on Class Diagram Design: A Preliminary Case Study," *Open Access,* pp. 1-7, 2017. |