Table of Contents

[Project Title: 2](#_Toc131754933)

[Introduction 2](#_Toc131754934)

[The Goal-Question-Metric (GQM) paradigm 2](#_Toc131754935)

[(i). Conceptual level – GOAL 3](#_Toc131754936)

[(ii). Operation level – QUESTION 3](#_Toc131754937)

[(iii). Quantitative level – METRIC 3](#_Toc131754938)

[Methodology 4](#_Toc131754939)

[Data collection 4](#_Toc131754940)

[Description of the subject programs or data set 5](#_Toc131754941)

[Tools used 6](#_Toc131754942)

[CK Metrics tool 6](#_Toc131754943)

[Visual Studio Code 6](#_Toc131754944)

[Results & Findings 6](#_Toc131754945)

[Conclusion 12](#_Toc131754946)

[References 13](#_Toc131754947)

Assignment - 1

# Project Title:

The title upon which the research has to be conducted is **“Effect of class size on software maintainability”.**

# Introduction

Code metrics are the ones that are most widely used to estimate software code maintenance. The metrics act as a guide for software developers to minimize the challenges faced during software maintenance. In object-oriented programming, class is one of the main concepts, and the class size would indeed affect the software’s maintainability. This report focuses on conducting an empirical study based on Goal-Question-Metric (GQM) approach to measure the code metrics and their impact on software maintainability. There are several size metrics such as WMC, CBO, RFC, and LCOM listed under CK metrics and those are taken into consideration. So here maintainability metrics and the number of lines of code are accounted in which the class would be involved. About 5 Java programs are being downloaded for the demonstration and their size metrics are analyzed and evaluated. Based on the values for the metrics chosen for measuring maintainability and class size, the impact on software maintainability has been determined through the use of graphical analysis.

# The Goal-Question-Metric (GQM) paradigm

The Goal-Question-Metric approach also known as GQM is the method that helps in detecting important metrics by listing the goals, tracing the goals with respected metrics, and then using a framework to interpret the data with the stated goals. So it is important to make clear what the goal is and how it has to be focused, what metrics could be quantified, and then how the quantified information could be analyzed to check whether it has achieved the goals or not [1]. The main purpose of using this GQM approach is to design measurements that offer a clear and solid result as the measurements are obtained by considering the goal and objectives. The result of the GQM approach is the specification of the measurement system which has been targeting certain issues and rules for measuring software maintainability. The GQM schema includes 3 steps such as determining the main goal of the project, determining the question of goal to make sure that the questions are answered, and then determining the metrics to be measured to answer the stated questions [2]. The three levels are described as follows:

## (i). Conceptual level – GOAL

A goal represents an objective and the goal might be an object or entity. It could be a quality, user’s perspective, and so on. The objectives of measurement could be products, processes, and resources.

## (ii). Operation level – QUESTION

This operational level includes a set of questions that would characterize the way to achieve the goal. Questions would consider the object of measurement [1].

## (iii). Quantitative level – METRIC

The quantitative level includes a set of data that is associated with each question to answer them quantitatively.

The GQM’s hierarchical structure for the current project is given as follows:

|  |  |
| --- | --- |
| **Goal**  Purpose  Issue  Object | To study and evaluate the effect  of class size on  software maintainability |
| 1. **Question** | What metrics and methods are used to measure maintainability? |
| **Metrics** | * WMC (Weighted Methods per Class) * CBO (Coupling Between Objects) * LOC (Lines of Code) |
| 1. **Question** | Does class size matters in the case of software maintainability? |
| **Metrics** | Yes, in object-oriented programming, a smaller class size leads to higher maintainability. |
| **3. Question** | What tool is used to extract the value of CK metrics? |
| **Metrics** | * CK metrics tool * Visual Studio Code |

**Table 1: GQM approach**

# Methodology

The research methodology that has been chosen for the research project is the quantitative analysis that includes a collection of the data and evaluation of them. The main goal of the project is to conduct an empirical study through the GQM approach to analyze the effect of class size on software maintainability. The quantitative analysis would help in measuring the effect of class size on software maintainability by using the CK metrics [3].

## Data collection

Initially, to carry out the demonstration, five Java programs are to be collected from an online repository. To obtain the subject programs, certain criteria would be set that include the collection of programs that consists of at least 10K in size, at least 3 years old program, and at least 3 developers included. GitHub is an online repository and is a code hosting environment mainly used for collaboration and version control. This platform lets multiple users work together and from anywhere. It helps in storing, tracking, and collaborating on various software programs. And so here for the demonstration purpose, the online repository, GitHub is used to collect 5 Java programs. So the Java programs are downloaded based on the given criteria.

# Description of the subject programs or data set

The following table gives the details of the obtained five Java programs:

|  |  |  |
| --- | --- | --- |
| S.No | Name of the Java program | URL of the program |
| 1. | Android-master | <https://github.com/nextcloud/android> |
| 2. | [keepass2android](https://github.com/PhilippC/keepass2android) | <https://github.com/PhilippC/keepass2android> |
| 3. | Lamp-cloud-master | <https://github.com/dromara/lamp-cloud> |
| 4. | Picocli-main | <https://github.com/remkop/picocli> |
| 5. | Struts-master | <https://github.com/apache/struts> |

**Table 2: Selected Java programs**

1. ***Android-master***

This Java program has been developed for an Android client and is named the Nextcloud Android app. This app has been created in such a way that it easily works with data and helps in the easy sharing of data with others securely. This does not need CLA (Contributor License Agreement).

1. [***keepass2android***](https://github.com/PhilippC/keepass2android)

This keepass2android Java program is nothing but a password manager for an Android application. This helps in storing and retrieving passwords and other sensitive data in a file of a database. This is a strong password and is complemented with a second factor for increased security. This program is configured to work best with built-in cloud storage options and third-party apps.

1. ***Lamp-cloud-master***

This Java program includes the development of an open-source Web development program and uses Linux, and Apache as the web server. It is a popularly used environment to build web applications.

1. ***picocli-main***

This program is about a modern framework that helps to build a powerful and user-friendly app. This supports colors, auto-completion, subcommands, and so on. It is extremely fast and requires a low memory and is used as a single executable file.

1. ***struts-master***

This Apache strut is a web framework that is an open-source solution to build Java web applications. It is a modern and fully maintainable web framework.

# Tools used

Tools used for this project are:

* CK Metrics tool
* Visual Studio Code

## CK Metrics tool

CK (Chidamber and Kemerer) metric tool is used to calculate the object-oriented metrics by processing the bytecode of Java files [4]. The program helps in calculating each class of the metrics and displays the output. To run the program, jar files/class files “ck-0.7.1-SNAPSHOT-jar-with-dependencies.jar” is used. The program would be run in the following format:

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>

Here the “project dir” represents the cirectory where CK could find the source code to be parsed. Then CK will look for .java files.CK that uses project dependencies to improve the precision. The jar files tells CK to look for jar files in the directory. Then variables and field metrics indicates CK to check for the variables and fields. Finally output dir would make CK to export the CSV file with metrics. Finally, four CSV files would be generated namely class, method, filed and variable.

## Visual Studio Code

Visual Studio Code also known as VS Code is a code editor which helps to do coding very quickly. It supports several programming languages such as Python, Java, C++, and so on. Here in this report, this visual studio code is used to measure code maintainability. This visual studio code helps in analyzing the chosen Java projects and displays their code metrics [5].

# Results & Findings

The analysis includes loading 5 Java programs into the Visual Studio Code with which the developers could generate code metrics that help to measure the complexity and maintainability of the code. So once when the program is loaded into the tool, this VS code would analyze the selected project and would display the code metrics results. So from the result, the class file could be accessed to analyze the metrics data which is stored in the excel format. Then based on the CK metrics introduced by Chidamber and Kemerer (CK), the static characteristic of software design would be measured [3]. The metrics would then be divided into certain categories such as class size, class internal, and class inheritance [6]. Here the class size has been considered to measure its effect on the maintainability of software.

1. **android-master**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Average value** | | |
| **Project Name** | **WMC** | **CBO** | **LOC** |
| android-master | 16.65211 | 9.411268 | 80.39014 |

**Table 3: Metrics of android-master**

**Figure 1: Graph of android-master**

Here in the given table 3, the average value for the selected metrics are been noted. The metrics chosen for measuring the maintainability includes WMC (Weighted Methods per Class) and CBO (Coupling Between Objects). The standard value of WMC must be <50 and here it is found to be 16.56, then the CBO must have default value of <=14 and here it is found to be 9.411. Hence the Java program’s code is maintainable. The LOC is found as 80 which is the total number of lines of code in the program.

1. **keepass2android-master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Average value** | | | |
| **Project Name** | **WMC** | **CBO** | **LOC** |
| keepass2android-master | 16.43474 | 4.447368 | 70.34211 |

**Table 4: Metrics of keepass2android-master**

**Figure 2: Graph of keepass2android-master**

Here in the given table 4, the average value for the chosen metrics are been noted and area also represented in graph format. The metrics includes WMC (Weighted Methods per Class) and CBO (Coupling Between Objects) and their values are 16.43 and 4.44 respectively. The standard value of WMC is <50 and the CBO’s value is <=14. Comparing the given values, it satisfies the standard value and so the software code is maintainable. The total LOC is found as 70 that denotes the total number of lines of code in the program.

1. **lamp-cloud-master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Average value** | | | |
| **Project Name** | **WMC** | **CBO** | **LOC** |
| lamp-cloud-master | 3.599572 | 8.379015 | 20.66595 |

**Table 5: Metrics of lamp-cloud-master**

**Figure 3: Graph of lamp-cloud-master**

The table 5 and figure 3 denotes the average value for the chosen metrics to measure the code maintainability. The metrics are WMC and CBO whose standard values are <50 and <=14. The Calculated value of WMC and the CBO’s value is below the standard value and so code is maintainable.

1. **picocli-main**

|  |  |  |  |
| --- | --- | --- | --- |
| **Average value** | | | |
| **Project Name** | **WMC** | **CBO** | **LOC** |
| picocli-main | 3.917106 | 2.963119 | 33.12294 |

**Table 6: Metrics of picocli-main**

**Figure 4: Graph of picocli-main**

The table 6 and figure 4 represents the average value for the chosen metrics of WMC and CBO. The Calculated values are below the standard value and so code is maintainable.

1. **struts-master**

|  |  |  |  |
| --- | --- | --- | --- |
| **Average value** | | | |
| **Project Name** | **WMC** | **CBO** | **LOC** |
| struts-master | 11.48967 | 4.864893 | 61.89114 |

**Table 7: Metrics of struts-master**

**Figure 5: Graph of struts-master**

The above given table and graph shows that the value of WMC and CBO is below the standard value.

**Comparison of metric values of 5 programs**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Average value of metrics** | | | | |
| S. No | **Project Name** | **WMC** | **CBO** | **LOC** |
| 1 | android-master | 16.65211 | 9.411268 | 80.39014 |
| 2 | keepass2android-master | 16.43474 | 4.447368 | 70.34211 |
| 3 | lamp-cloud-master | 3.599572 | 8.379015 | 20.66595 |
| 4 | picocli-main | 3.917106 | 2.963119 | 33.12294 |
| 5 | struts-master | 11.48967 | 4.864893 | 61.89114 |

**Table 8: Comparison of metric values**

**Figure 6: Comparison of metrics – Graph**

The above graph shows the comparison of the chosen metrics WMC, CBO and LOC for 5 programs to measure the software maintainability. The three metrics value chosen to measure the software maintainability is plotted in the form of bar graph and it denotes that the average value which has been taken is lower than the standard values of WMC and CBO and so it is found that the code is maintainable. And the LOC has no particular standard value but for high maintainability the number of LOC must be small.

# Conclusion

The metrics of maintainability and class size was analyzed for about 5 Java programs and their values are compared. The LOC (Lines of Code) is a software metric that helps to measure the size of the program by counting the number of lines present in the source code [7]. The number of lines of code modified per maintenance task has been taken as a measure of the software maintenance [8].

WMC is the number of methods defined within a class and it helps in measuring how complex is the individual class. When the number of methods in a class is large, then greater would be the impact on children. So a class with more member functions tends to be more complex and so it would be more error-prone [9]. It is very clear that larger numbers of methods would affect the usability of the software. The standard value of WMC has to be lesser than 50 so that the software would be easily maintainable.

CBO is the measure of interdependence of two objects and the value of CBO is measured by counting the number of classes to which it is coupled. The CBO value must be low in a code so that it improves modularity and reusability.

It has been strongly concluded that LOC (Lines of Code) and class size are strongly correlated with the maintenance effort. LOC could be considered as the controlling factors for software maintainability [10]. And only when the class size is small, the code could be easily modifiable. Hence it is proved that class size impacts software maintainability.

# References

|  |  |
| --- | --- |
| [1] | V. R. Basili, G. Caldiera and H. D. Rombach, "The Goal Question Metric Approach," *https://www.cs.umd.edu/users/mvz/handouts/gqm.pdf,* p. 10, 2018. |
| [2] | R. Setiawan, Z. E. Rasjid and A. Effendi, "Design Metric Indicator to Improve Quality Software Development," *3rd International Conference on Computer Science and Computational Intelligence,* p. 8, 2018. |
| [3] | S. Srivastava, "Software metrics and Maintainability Relationship with CK Matrix," *International Journal of Innovations in Engineering and Technology (IJIET),* vol. 1, no. 2, p. 7, 2012. |
| [4] | A. M. T. Oo, "Implementation of Object Oriented Metric(CK-Metric) Tool for Java Program," 27 December 2017. [Online]. Available: https://onlineresource.ucsy.edu.mm/handle/123456789/1072. [Accessed 6 April 2023]. |
| [5] | T. Pal, "Improving Code Quality with Visual Studio Code Metrics," 25 October 2017. [Online]. Available: https://www.codeguru.com/csharp/improving-code-quality-with-visual-studio-code-metrics/. [Accessed 4 April 2023]. |
| [6] | S. K. Dubey and A. Rana, "Assessment of Usability Metrics for Object-Oriented Software System," *ACM SIGSOFT Software Engineering Notes,* vol. 35, no. 6, p. 5, 2010. |
| [7] | D. S. Chawla and G. Kaur, "Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development," *(IJACSA) International Journal of Advanced Computer Science and Applications,* vol. 4, no. 9, p. 4, 2013. |
| [8] | Y. Dash and S. K. Dubey, "Maintainability Measurement in Object Oriented Paradigm," *International Journal of Advanced Research in Computer Science,* vol. 3, no. 2, p. 7, 2012. |
| [9] | T. Zheng and W. Zhang, "Are Big Classes in Java Bad? The Effect of Classes Size on Software Quality in Java," *Applied Mechanics and Materials ,* Vols. 568-570, pp. 1443 - 1446, 2014. |
| [10] | L. Kumar, D. K. Naik and S. K. Rath, "Validating the Effectiveness of Object-Oriented Metrics for Predicting Maintainability," *Procedia Computer Science,* vol. 57, pp. 798 - 806, 2015. |