**Group Assignment 1**

**Subject:** Object Oriented Development

**Group Name:** New Group 8

**Group members:**

Venkata Ramana Patnam

Praveen Thumati

**Professor Name:** Fadi Wedyan

**Date:** 4 june, 2023

Table of Contents

[Section 1 2](#_Toc131547351)

[Applying GQM Approach 2](#_Toc131547352)

[Section 2 3](#_Toc131547353)

[Selected Projects: 4](#_Toc131547354)

[Section 3 6](#_Toc131547355)

[Tool Description 6](#_Toc131547356)

[Section 4 7](#_Toc131547357)

[Results: 8](#_Toc131547358)

[CatVodTVSpider Project: 8](#_Toc131547359)

[SmartTubeNext Project: 9](#_Toc131547360)

[cwa-server Project: 10](#_Toc131547361)

[RxJava Project: 11](#_Toc131547362)

[wvp-GB28181-pro Project: 12](#_Toc131547363)

[Findings 13](#_Toc131547364)

[Overall findings for each project: 14](#_Toc131547365)

[Section 5 14](#_Toc131547366)

[Conclusion 14](#_Toc131547367)

[References 15](#_Toc131547368)

# Section 1

The following is an example of the GQM, or Goal Question Metric, methodology being used to find key metrics:

* Write down the process's intended outcomes.
* Write the questions that need to be answered to obtain those outcomes.
* Follow the measures back to the objectives.

GQM enables software development teams to better prioritize their activities, plan out their work more efficiently, and monitor their overall progress toward a common goal. At the corporate, project, and procedure levels, the GQM technique has been successfully applied to streamline decision-making, improve performance, and make the best use of resources.

## The GQM technique can be used to, among other things, construct measurements for evaluating the maintainability of software, produce research questions, and decide the goal of the study. Researchers may make sure their study has distinct objectives and metrics, that their questions are focused in the proper way, and that their data is trustworthy and usable by using the GQM technique. This approach also guarantees that their data is accurate and useful.

## Applying GQM Approach

**Objective:**

The purpose of this empirical investigation is to apply the C&K measures to the question of how much of an impact class size has on software maintainability.

**Questions:**

Is there a correlation between the number of a class and how easily it can be maintained?

What correlation exists among the number of classes and the C&K measures chosen to evaluate maintainability?

**Metrics:**

Based on the objective and research questions, the following C&K metrics can be selected to measure software maintainability:

* **Weighted Methods per Class,**

This class complexity metric takes into account both the total number of methods and the typical complexity of each method.

* **Depth of family Tree (DIT):**

DIT counts the levels in a class's family hierarchy.

* **Coupling Among Objects (CBO):**

This is a measure of the degree to which different sets of objects are interconnected.

# These metrics can be used to examine class number and program maintenance ease (Dubey & Rana, 2011). We can learn more about the effect of class size on software maintainability by gathering and examining this data for a subset of randomly chosen software components. By evaluating these metrics for a sample of arbitrarily chosen software components, and then looking at the correlation between them, we can find out more about the impact of class size on software maintainability.

# Section 2

We set the following criteria for the subject programs:

* The programs should be at least 10K lines of code in size.
* The programs must be at least two years old and within the last three years of development.
* A minimum of three programmers must have contributed to the software's source code.

These criteria were chosen to make sure that the programs chosen had undergone maintenance procedures, were large enough to be typical of real-world software systems, and had been created cooperatively by several developers.

The program age criterion was established to make sure that the software systems have maintenance procedures that could affect their maintainability, but that they weren't too old to not reflect contemporary practices and technologies.

To ensure that the programs are complicated enough to produce insightful results for the examination of the impact of class size on software maintainability, the criterion for the size of the programs was established.

The requirement for the number of developers was established to guarantee that the programs were created jointly and not by a single person.

These criteria can guarantee that the programs chosen are suitable for the investigation of the impact of class size on software maintainability and are typical of real-world software systems.

## Selected Projects:

The follwing are the projects that were selected which fulfilled criteria:

1. **CatVodTVSpider (https://github.com/catvod/CatVodTVSpider)**

The project is titled CatVodTVSpider. This is the updated common crawler code package for the Maoying TV software. This open-source project supports user-defined parameters and creates a jar file that may be integrated.

Since this is an AndroidStudio project, any code modifications require Android Studio. Once the project has been debugged, the buildAndGenJar.bat script in the base directory can be used to produce a custom\_spider.jar file. The program's executable code is located in this jar file.

Sending merge requests and adding your own crawler code to the project are both welcome from users. The CatVodTVSpider project encourages collaboration and fresh thinking because it is open source, and it provides a workable solution to the data mining issue for Maoying TV.

1. **SmartTubeNext(https://github.com/yuliskov/SmartTubeNext)**

For Android TVs and set-top boxes, it is a premium YouTube program that is free and open-source. Aside from the ability to monitor live conversation, this program supports 8k video at 60 frames per second, high dynamic range (HDR), and variable playback speeds. Its goal is to provide ad-free entertainment.

SmartTubeNext's SponsorBlock feature enables users to skip advertisements while watching videos. Additionally, we can use the application without connecting to Google and create our own icons. A strong reputation for friendliness and encouragement exists among the international team working on the effort.

SmartTubeNext offers a dependable and uncomplicated choice for watching YouTube videos on Android TVs and TV boxes while also eliminating grating commercial breaks. It encourages a culture of collaboration and innovative thinking because it is freely accessible to the general population.

1. **cwa-server**

It functions as a server application for the Apple and Google exposure warning APIs and is an essential part of Germany's official Corona-Warn-App. In order to anonymously communicate secured data with other mobile phones nearby, this project seeks to develop mobile software (for both iOS and Android). This software will employ Bluetooth technology.

Each user's device stores all data given and received through the app securely and privately, shielding it from prying eyes. The cwa-server source contains the implementation of the server for encryption keys used by Corona-Warn-App.

The Corona-Warn-App project, which seeks to contain the spread of COVID-19 in Germany, relies heavily on cwa-server. This software can be a valuable tool in the fight against disease due to its usage of the Apple/Google exposure warning API and commitment to user privacy and data security.

1. **RxJava (https://github.com/yuliskov/SmartTubeNext)**

This Java framework is used to incorporate observable patterns into event- and delayed-based systems. A Java-based implementation of the Reactive Extensions library, RxJava is a framework for constructing observable sequences in an asynchronous and event-based manner.

The watcher design is expanded by RxJava to handle data and event sequences. Low-level threading, synchronization, thread-safety, and concurrent data structures are abstracted away, and operators for explicit sequence composition are added. RxJava is a crucial tool for working with asynchronous and event-based computing in Java. Its declarative handling of complex data patterns and abstraction of underlying technical details make it popular among programmers.

1. **Wvp-GB28181-pro(https://github.com/yuliskov/SmartTubeNext)**

This network video technology is based on the GB28181-2016 standard and supports NAT traversal. IPC, NVR, and DVR products from well-known manufacturers are available over the network. Hikvision, Dahua, and Uniview are a few of these. Due to the platform's support for spreading, video feeds can be forwarded to other platforms that follow governmental regulations.

Push streams like rtsp and rtmp can also be delivered to national standard systems using Wvp-GB28181-pro. In conclusion, by enabling simple interoperability with a variety of hardware and software systems, the project provides a valuable resource for video security programs. Users may access their streams no matter where they are in the world thanks to support for NAT entry and compliance with the GB28181-2016 standard.

Table 1:Comparison of features of all projects.

|  |  |  |
| --- | --- | --- |
| Project Name | Type | Function |
| CatVodTVSpider | Crawler code package | enables data crawling for use with the Maoying TV program |
| SmartTubeNext | YouTube app | enables ad-free YouTube video viewing on Android TVs and TV boxes. |
| cwa-server | Backend implementation | Provides the server for encryption keys used by the Corona-Warn-App |
| RxJava | Library | demonstrates how to create observable sequences in Java, allowing the development of event-driven and delayed applications. |
| wvp-GB28181-pro | Network video platform | based on the GB28181-2016 standard, supports access for IPC, NVR, and DVR from well-known manufacturers including Hikvision, Dahua, and Uniview. |

# Section 3

## Tool Description

On GitHub, you can find the program used to calculate CK-Code metrics for Java code.(GitHub - Mauricioaniche/Ck: Code Metrics for Java Code by Means of Static Analysis, n.d.) The CK-Code metric tool is an open-source instrument created to assess a number of software quality characteristics, including maintainability.

Cyclomatic Complexity, Lines of Code, and Lack of Cohesion in Methods are just a few of the metrics that the tool can calculate. In the software industry, these metrics are frequently used to gauge the quality and maintainability of code.

Developers can track the maintainability of their code over time with the help of the user-friendly CK-Code metric tool, which can be linked into continuous integration and delivery pipelines (Michura et al., 2013). The generated reports can be examined by developers to find out more about the data the tool tracked and potential areas for optimization.

The CK-Code metric utility should be investigated by programmers if they are interested in improving the readability, comprehension, and security of their Java code. Due to its open-source nature and simple integration, it is readily available to developers of all skill levels.

# Section 4

In this section, we go over the findings of our empirical investigation on how the size of a class affects how maintainable software is. Using the CK-Code evaluation tool, we assessed a piece of code, looking at factors such cyclomatic complexity, lines of code, and method cohesiveness among other indicators of software quality.

After gathering the data, we mathematically analyzed it, produced some charts and spreadsheets, and then presented the results. The results of our study can be put into practice to guide the decisions and practices that are implemented during software development by highlighting the relationship between class size and maintainability (Chowdhury et al., 2022).

It's likely that the results of our study will add something fresh to the ongoing discussion regarding the value of software quality and maintainability. Our findings should be helpful to both software developers and researchers, and we hope they will inspire more investigation into the variables affecting software maintainability.

## Results:

### **CatVodTVSpider Project:**

The answers to questions laid in GQM approach in Section are analyzed as follows:

1. **Is there a correlation between the number of a class and how easily it can be maintained?**

The results show that the project's Weighted Methods per Class (WMC) average of 24.93 is fairly high. Large class sizes are a sign of a complex undertaking, which could make it more difficult to manage.

The relatively high Coupling Between Objects (CBO) metric value of 4.23 suggests that there may be significant links between classes, making the code more challenging to manage.

According to the Depth of Inheritance Tree (DIT) metric, which is only 1.5, the project has a shallow hierarchy. However, by alone, this measurement might not provide a complete picture of the project's maintainability.

1. **What correlation exists among the number of classes and the C&K measures chosen to evaluate maintainability?**

The findings demonstrate a relationship between class size and maintainability, which is supported by the project's high WMC and CBO scores.

However, it should be noted that the C&K metrics used in this study only give a broad picture of maintainability and that other factors, such as code understanding, adaptability, and documentation, should also be taken into account when assessing software maintainability.

### **SmartTubeNext Project:**

The answers to questions laid in GQM approach in Section are analyzed as follows:

1. **Is there a correlation between the number of a class and how easily it can be maintained?**

This project has a subpar Weighted Methods per Class (WMC) of 11.39 when compared to similar projects. Less intricate projects tend to include fewer classes to manage, which is good for maintainability.

The code is simple to maintain with a Coupling Between Objects (CBO) score of 6.29 since there aren't many links between classes.

There aren't many inherited components in the project, according to its minimal Depth of Inheritance Tree (DIT), which is 1.74. As a byproduct, this might aid in making the code more manageable and easy.

1. **What correlation exists among the number of classes and the C&K measures chosen to evaluate maintainability?**

The project's relatively low WMC and CBO ratings suggest a potential link between maintainability and class size.

Nevertheless, it should be noted that the C&K measures used in this study only give an indication of maintainability and that other factors, such as code understanding, adaptability, and documentation, should also be taken into account when evaluating software maintainability.

### **cwa-server Project:**

The answers to questions laid in GQM approach in Section are analyzed as follows:

1. **Is there a correlation between the number of a class and how easily it can be maintained?**

When compared to similar programs, the project has a subpar Weighted Methods per Class (WMC) of 3.76 on average. As a result, it's probable that the project's low level of complexity due to the smaller class size will improve its maintainability.

The code is simple to maintain with a Coupling Between Objects (CBO) score of 4.29 since there aren't many links between classes.

A low Depth of Inheritance Tree (DIT) of 1.24 indicates that there aren't many inherited components in the project. This may also assist to simplify and manage the code, which is an added benefit.

1. **What correlation exists among the number of classes and the C&K measures chosen to evaluate maintainability?**

The project's relatively low WMC and CBO ratings suggest a possible relationship between maintainability and class size.

It should be noted, however, that other factors, such as code accessibility, flexibility, and needs, should also be taken into account when evaluating software maintainability and that the C&K metrics utilized in this research only provide a partial perspective of maintainability.

### **RxJava Project:**

The answers to questions laid in GQM approach in Section are analyzed as follows:

1. **Is there a correlation between the number of a class and how easily it can be maintained?**

When compared to comparable projects, the project has a subpar Weighted Methods per Class (WMC) of 4.08 on average. As a result, it's feasible that the project's lack of complexity as a result of the smaller class size will improve its maintainability.

Because there aren't many connections between classes, the code has a Coupling Between Objects (CBO) score of 2.52.

The Depth of Inheritance Tree (DIT) value for the project, which is 1.32, shows that there aren't many inherited sub-components.

1. **What correlation exists among the number of classes and the C&K measures chosen to evaluate maintainability?**

The results indicate that the high class sizes utilized in the analysis may have contributed to the project's poor WMC and CBO values.

However, when examining software maintainability, code clarity, modularity, and documentation should also be taken into account; the C&K scores in this study only give a partial picture.

### **wvp-GB28181-pro Project:**

Based on the CK-Code metric results for the wvp-GB28181-pro project, we can analyze the effect of class size on software maintainability as follows:

1. **How does the class size affect software maintainability?**

This project's value of 17.35, as determined by Weighted Methods per Class (WMC), is excellent when compared to other projects. Due to the larger class ability, it's probable that the project's complexity will rise, making future updates more difficult.

The Coupling Between Objects (CBO) metric, which has a value of 8.73, likewise shows that there are numerous connections between classes, making it more challenging to update the code.

According to the Depth of Heredity Tree (DIT) metric, which only has a 1.23 value for the project, there are not many levels of heredity. This might simplify the code, increasing maintainability by lowering its complexity.

1. **What is the relationship between class size and the selected C&K metrics for measuring maintainability?**

The results and the project's strong WMC and CBO values show a potential connection between class size and maintainability. The C&K scores in this study only demonstrate half of maintainability. Clarity, structure, and instructions in the code should all be taken into account when maintaining software.

|  |  |  |  |
| --- | --- | --- | --- |
| **Project Name** | **CBO** | **WMC** | **DIT** |
| CatVodTVSpider | 4.23 | 24.93 | 1.5 |
| SmartTubeNext | 6.29 | 11.39 | 1.74 |
| cwa-server | 4.29 | 3.76 | 1.24 |
| RxJava | 2.52 | 4.08 | 1.32 |
| wvp-GB28181-pro | 8.73 | 17.35 | 1.23 |

The chosen projects' Java code metrics were computed using CK-Code and shown in table above.

## Findings

The CK-Code metric tool allows us to analyze the five projects using the GQM method. The analysis based on results are:

* **Implications for software maintainability of increasing class sizes:**

The results show that larger class sizes most likely caused higher average WMC values for the tasks. Due to the added complexity, larger class sizes may negatively affect the maintainability of software.

However, the CBO and DIT project values are typically low to average, which is a sign of little interclass interaction and a low level of inheritance. If this is put into practice, the potential detrimental effects of larger class sizes on maintainability might be diminished.

* **Class size and a few key C&K measures for evaluating maintainability:**

The projects with larger class sizes had relatively high WMC scores, which may be a sign that maintainability and class size are related.

The research evaluated maintainability exclusively using the C&K measures, hence the results are plainly constrained. Additional elements like readability of the code, functionality, and documentation must be taken into account in order to conduct an appropriate analysis of the software's maintainability.

### **Overall findings for each project:**

It's possible that CatVodTVSpider has good maintainability given that it has a moderate WMC, a relatively low CBO, and a low DIT.

The moderate WMC, low CBO, and moderate DIT of SmartTubeNext suggest that its maintainability may also be relatively satisfactory.

Because there are fewer dependencies between classes and fewer classes overall, cwa-server has a low WMC, low CBO, and low DIT, which suggests that it may have respectable maintainability.

It's possible that RxJava's fewer classes and moderate amount of inheritance help to explain why it's so easy to maintain. The low WMC, low CBO, and moderate DIT of RxJava all point to its high maintainability.

The fact that wvp-GB28181-pro has a high WMC, a high CBO, and a low DIT suggests that, as a result of its larger class size and greater degree of complexity in the ode, it may have substantially worse maintainability.

**Summary:-**

The observed courses seem to have a moderate amount of complexity, with a mean WMC. Maintainability may be affected by the classes' moderate **complexity** since more complexity might make them harder to comprehend and alter.

**Message Interactions**: The observed classes have a modest amount of method interactions, with an average RFC value . This suggests that the classes are exposed to a moderate volume of incoming and outgoing messages, which may increase their complexity and create additional upkeep issues.

The average values for WMC and RFC suggest that the complexity and number of method interactions in the studied classes may provide some difficulties in terms of **maintainability**. In general, larger values for either measure imply a decline in maintainability since they point to increased complexity and possible challenges when modifying the classes.

While exact values for **DIT** are not supplied, it is possible that the existence of inheritance hierarchies will have an effect on maintainability. Complexity and dependencies introduced by several inheritance levels might make it more difficult to maintain the observed classes.

In conclusion, the data suggests that the classes' complexity, as evaluated by WMC and RFC, may have consequences for their maintainability. Maintenance activities may need considerable thought and preparation due to the system's moderate complexity and the moderate number of method interactions. Inheritance hierarchies also have the potential to add complications that detract from maintainability. A deeper knowledge of the connection between these indicators and software maintainability may be attained by further research and investigation of a bigger dataset.

# Section 5

## Conclusion

The results of the CK-Code metric application suggest that the number of classes in a program may have an impact on its maintainability. This is demonstrated by the relatively high WMC values that were found for the programs with larger class sizes. This suggests that as the number of classes grows, the code could get more complex, making it more challenging to understand and modify.

It's crucial to keep in mind that the C&K measurements utilized in this research only give a partial picture of maintainability when assessing the maintainability of software. Another crucial factor to think about is how readable the code is, as well as how it is organized and explained. To get an accurate picture of the software's quality, a variety of measurements and research techniques must be used.

The results in regard to the projects under investigation suggest that, depending on the project, the impact of class size on maintainability may differ from project to project. For instance, despite having comparable low CBO and DIT values, CatVodTVSpider and SmartTubeNext both had bigger class sizes. By doing this, you might be able to lessen the detrimental effects that bigger class sizes have on maintainability. cwa-server and RxJava, on the other hand, have comparatively fewer classes and smaller class sizes, both of which may help with their great maintainability.

Compared to the other projects, Wvp-GB28181-pro has worse maintainability due to its high WMC, high CBO, and poor DIT values. This might be as a result of its larger class size and more complex code, both of which make system maintenance more challenging.

Overall, this study's results suggest that class size may have an impact on the software's capacity to be maintained; however, there are a lot of other factors to take into account. To provide a complete picture of a program's quality, a range of metrics and analytical methods must be used. It's vital to include additional aspects while assessing maintainability, such as code accessibility, modularity, and documentation.

# References

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

Dubey, S. K., & Rana, A. (2011). Assessment of maintainability metrics for object-oriented software system. *ACM SIGSOFT Software Engineering Notes*, *36*(5), 1–7. https://doi.org/10.1145/2020976.2020983

*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

Michura, J., Capretz, M. A. M., & Wang, S. (2013). Extension of Object-Oriented Metrics Suite for Software Maintenance. *ISRN Software Engineering*, *2013*, 1–14. https://doi.org/10.1155/2013/276105