**Group Assignment 1**

**Subject:** Object Oriented Development

**Group Name:** New Group 10

**Group members:**

Abhilash Pabathi

Gopalakrishna Perla

**Professor Name:** Fadi Wedyan

**Date:** April 8, 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 allows software development teams to plan out their work more effectively, assign priorities to their tasks, and track their overall progress toward a shared objective. The GQM strategy has been successfully implemented at the corporate, project, and procedure levels to simplify decision-making, enhance performance, and make the most efficient use of available resources.

The GQM technique may be utilized to, first and foremost, determine the purpose of the study, secondly, generate research questions, and thirdly, establish measures for evaluating the maintainability of software. Using the GQM strategy, researchers can ensure that their study has clear objectives and measures, that their questions are oriented in the right direction, and that their data is reliable and actionable. Using this method also ensures that their data is reliable and actionable.

## 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,**

It is a measure of class complexity that considers both the overall number of methods and the average complexity of each method.

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

DIT measures the number of levels in a class's family structure.

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

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

Class number and the ease of maintaining a program can be analyzed using such measures (Dubey & Rana, 2011). By collecting and analyzing this data for a subset of randomly selected software components, we can gain insight into the impact of class size on software maintainability. We can learn more about the effect of class size on software maintainability by measuring these metrics for a sample of randomly selected software components and examining the correlation between them.

# 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 should have been developed within the last three years, but not less than two years old.
* There needs to be a minimum of three programmers who have contributed to the source of the software.

These criteria were selected to ensure that the selected programs are large enough to be representative of real-world software systems, have undergone maintenance tasks, and have been developed collaboratively by multiple developers.

The criterion for the age of the programs was set to ensure that the software systems have undergone maintenance tasks that could affect their maintainability, but not too old as to not reflect current practices and technologies.

The criterion for the size of the programs was set to ensure that the programs are complex enough to provide meaningful results for the analysis of the effect of class size on software maintainability.

The criterion for the number of developers was set to ensure that the programs have been developed collaboratively and are not the work of a single individual.

These criteria can ensure that the selected programs are representative of real-world software systems and are suitable for the analysis of the effect of class size on software maintainability.

## Selected Projects:

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

1. **CatVodTVSpider**

CatVodTVSpider is the name of the project. This is the updated Maoying TV software's common crawler code package. This open-source undertaking facilitates user-defined settings and generates an integratable jar file.

This is an AndroidStudio project, so we will need Android Studio to make any changes to the code. The buildAndGenJar.bat script in the base path can be used to create a custom\_spider.jar file once the project has been debugged. This jar file contains the executable code for the program.

Users are welcome to send merge requests and contribute their own crawler code to the project. Since it is open source, the CatVodTVSpider initiative fosters cooperation and new ideas, and it offers a practical answer to the problem of data mining for Maoying TV.

1. **SmartTubeNext**

It is a free and open-source, high-end YouTube software for Android TVs and set-top boxes. This program is aimed at providing ad-free content and has many useful features, including the ability to view live chat, support for 8k video at 60 frames per second, high dynamic range (HDR), and customizable playback speed.

The SponsorBlock function, available in SmartTubeNext, lets viewers ignore commercials while watching videos. Plus, we can make our own icons and use the program without having to connect to Google. The global group working on the initiative has a solid reputation for being friendly and encouraging.

When it comes to watching YouTube videos on Android TVs and TV boxes, SmartTubeNext provides a reliable and straightforward option, all while removing annoying commercial breaks. Because it is freely available to the public, it fosters a culture of cooperation and new ideas.

1. **cwa-server**

As a core component of Germany's official Corona-Warn-App, it serves as a server application of the Apple and Google exposure warning API. This project aims to create mobile software (for both iOS and Android) that makes use of Bluetooth technology to anonymously share protected data with other mobile phones in the user's immediate proximity.

All information sent and received through the app is encrypted and kept privately on each user's device, protecting it from prying eyes. The server for encryption keys used by Corona-Warn-App is implemented in the cwa-server source.

cwa-server is an essential part of the Corona-Warn-App project, which aims to slow the spread of COVID-19 in Germany. Because of its use of the Apple/Google exposure warning API and its dedication to user privacy and data security, this app can be an invaluable resource in the fight against disease.

1. **RxJava**

It is a Java framework for writing observable patterns into delayed and event-based applications. RxJava is a framework for writing observable sequences in an asynchronous and event-based fashion; it is a version of the Reactive Extensions library.

RxJava expands the watcher design to handle data and event sequences. Operators for explicit composition of sequences are added, and low-level threading, synchronization, thread-safety, and concurrent data structures are abstracted away. When dealing with asynchronous and event-based computing in Java, RxJava is an essential utility. Popular among coders is its declarative handling of complicated data patterns and abstraction of underlying implementation details.

1. **Wvp-GB28181-pro**

The GB28181-2016 standard forms the basis of this network video technology, which also allows for NAT traversal. Popular manufacturers of IPC, NVR, and DVR are accessible on the network. These include Hikvision, Dahua, and Uniview. Video feeds can be routed to other platforms that adhere to national standards due to the platform's support for spreading.

Wvp-GB28181-pro also allows push streams such as rtsp and rtmp to be transmitted to national standard systems. In sum, the project delivers a useful resource for video security apps by facilitating easy interoperability with numerous hardware and software systems. Support for NAT entry and conformance to the GB28181-2016 standard means that users can watch their streams no matter where they happen to be in the world.

Table 1:Comparison of features of all projects.

|  |  |  |
| --- | --- | --- |
| Project Name | Type | Function |
| CatVodTVSpider | Crawler code package | Allows for crawling data for use with Maoying TV software |
| SmartTubeNext | YouTube app | Allows for ad-free viewing of YouTube videos on Android TVs and TV boxes |
| cwa-server | Backend implementation | Provides the server for encryption keys used by the Corona-Warn-App |
| RxJava | Library | Provides a way to write observable sequences in Java, enabling the creation of delayed and event-driven applications. |
| wvp-GB28181-pro | Network video platform | Supports access for IPC, NVR, and DVR from popular brands such as Hikvision, Dahua, and Uniview, based on the GB28181-2016 standard |

# Section 3

## Tool Description

The tool used for calculating CK-Code metrics for Java code is available at GitHub.The CK-Code metric tool is an open-source tool that is designed to measure a variety of software quality attributes, including maintainability(*GitHub - Mauricioaniche/Ck: Code Metrics for Java Code by Means of Static Analysis*, n.d.).

The tool calculates a range of metrics, including Cyclomatic Complexity, Lines of Code, and Lack of Cohesion in Methods, among others. These metrics are widely used in the software industry to measure code quality and maintainability.

The CK-Code metric tool is easy to use and can be integrated into continuous integration and delivery pipelines, allowing developers to monitor the maintainability of their code over time(Michura et al., 2013). Developers can examine the generated reports to learn more about the data the tool tracked and where optimizations could be made.

If we as coders are interested in making our Java code more readable, understandable, and secure, we should check out the CK-Code metric utility. It is easily available to developers of all skill levels due to its open-source character and effortless integration.

# Section 4

In this section, we discuss the outcomes of our empirical research into the ways in which the number of a class influences the maintainability of software. We evaluated an example of code using the CK-Code evaluation tool, and some of the software quality characteristics we looked at included cyclomatic complexity, lines of code, and method cohesion.

Following the collection of the data, we conducted some mathematical analysis on it and then created some graphics and spreadsheets. By highlighting the association between class size and maintainability, the findings of our research can be made into action to direct the choices and procedures that are implemented during software development(Chowdhury et al., 2022).

It is possible that the findings of our research will contribute something new to the ongoing conversation about the importance of maintainability and the quality of software. We anticipate that both software developers and researchers will find our findings useful, and we also anticipate that they will stimulate further research into the factors that influence the maintainability of software.

## 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 findings indicate that the project has a reasonably high Weighted Methods per Class (WMC) average of 24.93. Large class sizes are indicative of a high level of complexity in the undertaking, which may affect its maintainability.

There may be substantial connections between classes, making the code more difficult to manage, as indicated by the Coupling Between Objects (CBO) metric's comparatively high value of 4.23.

The project has a shallow hierarchy, as measured by the Depth of Inheritance Tree (DIT) statistic (which is only 1.5). However, this measure may not paint a full image of the project's maintainability by itself.

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

The high WMC and CBO metrics for the project are indicative of a connection among class size and maintainability, as shown by the findings.

It should be noted, however, that the C&K measures utilized in this research only provide an overview of maintainability, and that other variables, including code comprehension, flexibility, and documentation, should also be considered when evaluating 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?**

When compared to similar projects, this one has a poor Weighted Methods per Class (WMC) of 11.39. Having fewer classes to manage is an indicator of a less complicated undertaking, which works well for its maintainability.

With a Coupling Between Objects (CBO) of 6.29, the code is easy to manage because there are few connections between classes.

The project has a minimal Depth of Inheritance Tree (DIT) of 1.74, suggesting that there are not many inherited components. As a side effect, this may help make the code simpler and easier to manage.

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

The comparatively low WMC and CBO values for the project are indicative of a possible connection between class size and maintainability.

It should be noted, however, that the C&K measures utilized in this research only provide an indication of maintainability, and that other variables, including code comprehension, flexibility, and documentation, should also be considered 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?**

The project has a poor Weighted Methods per Class (WMC) of 3.76 on average when compared to similar programs. Because of this, it is possible that the project's maintainability will be enhanced by its low degree of complexity due to reduced class size.

With a Coupling Between Objects (CBO) of 4.29, the code is easy to manage because there are few connections between classes.

The project has a low Depth of Inheritance Tree (DIT) of 1.24, suggesting that there are not many inherited components. As an additional advantage, this may help make the code simpler and easier to manage.

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

The comparatively low WMC and CBO values for the project are indicative of the potential link between class size and maintainability.

It should be noted, however, that the C&K measures used in this research only provide an incomplete view of maintainability, and that other variables, such as code accessibility, flexibility, and the requirements should also be considered when evaluating software 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?**

The project has a poor Weighted Methods per Class (WMC) of 4.08, on average, when compared to other initiatives. Because of this, it is possible that the project's maintainability will be enhanced by its lack of complexity due to reduced class size.

With a Coupling Between Objects (CBO) of 2.52, the code is easy to manage because there are few connections between classes.

At 1.32, the project's Depth of Inheritance Tree (DIT) measure indicates that there are not many inherited sub-components.

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

The findings show that the project's low WMC and CBO values may be related to the large class sizes used in the analysis.

However, code clarity, modularity, and documentation should also be considered when analyzing software maintainability, and this study's C&K scores only provide a limited view.

### **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?**

As measured by Weighted Methods per Class (WMC), this project has a value of 17.35, which is excellent when compared to other projects. It is possible that the project's complexity will increase because of the greater class ability, making it harder to update in the future.

At 8.73, the Coupling Between Objects (CBO) measure also indicates that there are many connections between classes, which makes the code more difficult to update.

The project has few layers of heredity, as measured by the Depth of heredity Tree (DIT) metric, which is only 1.23. This has the potential to reduce the complexity of the code, which in turn improves maintainability.

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

The project's strong WMC and CBO values, along with the findings, demonstrate a possible link between class size and maintainability. This study's C&K ratings only show half of maintainability. Software maintainability should also consider code clarity, structure, and instructions.

|  |  |  |  |
| --- | --- | --- | --- |
| **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:**

According to the findings, bigger class sizes likely contributed to higher average WMC values for the tasks. Larger class sizes may have an adverse effect on software maintainability because of the increased complexity that results.

However, CBO and DIT project values tend to be low to average, which is indicative of a low degree of inheritance, and few interclass relationships. The potential negative consequences of increased class sizes on maintainability may be reduced if this is implemented.

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

Results show relatively high WMC values for the projects with bigger class sizes, which may be indicative of a correlation between class size and maintainability.

The C&K measures are the only ones that were used in the research to evaluate maintainability, so the findings are obviously limited. To conduct an accurate analysis of the software's maintainability, additional factors like code readability, functionality, and documentation are required to be taken into consideration.

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

The fact that CatVodTVSpider has a moderate WMC, a comparatively low CBO, and a low DIT gives rise to the possibility that it has decent maintainability.

Because SmartTubeNext has a moderate WMC, comparatively low CBO, and moderate DIT, it appears that its maintainability may also be moderately satisfactory.

cwa-server has a low WMC, low CBO, and low DIT, which indicates that it may have decent maintainability because it has fewer dependencies between classes and a reduced total number of classes.

It is possible that RxJava's smaller class size and moderate degree of inheritance contribute to its good maintainability. RxJava has a low WMC, a low CBO, and a moderate DIT, all of which indicate that it has good maintainability.

The fact that wvp-GB28181-pro has a high WMC, a high CBO, and a low DIT suggests that it may have comparatively weaker maintainability as a result of its increased class size and higher level of complexity in the ode.

# Section 5

## Conclusion

According to the findings of the CK-Code metric application, the maintainability of a program might be affected by the number of its classes. The comparatively high WMC values that were observed for the programs that had greater class sizes are evidence of this point. This indicates that as the number of classes increases, the code may become more complicated, making it more difficult to comprehend and make modifications too.

When evaluating the maintainability of software, it is important to remember that the C&K measures that were used in this research only provide a partial perspective of maintainability. The readability of the code, as well as its organization and explanations, are also essential aspects to take into consideration. It is essential to make use of a wide range of measurements and research approaches to obtain a precise indication of the quality of the software.

In relation to the projects that were investigated, the findings indicate that the effect of class size on maintainability may vary from project to project depending on the project. For instance, CatVodTVSpider and SmartTubeNext both have larger class sizes, but they also have comparatively low CBO and DIT values. This could help to mitigate the negative impact that larger class sizes have on maintainability caused by the larger class sizes. On the other hand, cwa-server and RxJava have smaller class sizes and comparatively lower complexity, both of which may contribute to excellent maintainability.

Wvp-GB28181-pro has comparatively less maintainability in comparison to the other projects due to its high WMC, high CBO, and low DIT values. This could be because of its larger class size and more complicated code, both of which make it more difficult to maintain the system.

The findings of this research indicate, overall, that class size may have an effect on the maintainability of software; however, there are a number of other variables that should also be considered. It is essential to use a variety of measurements and analytic techniques to obtain a comprehensive view of the quality of the software. When evaluating maintainability, it is also important to take into consideration other factors such as the accessibility of the code, 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