**Group Assignment 1**

**Subject:** Object Oriented Development

**Group Name:** New Group 4

**September 18, 2025**

Table of Contents

[Section 1 2](#_Toc196077090)

[Applying GQM Approach 3](#_Toc196077091)

[Section 2 4](#_Toc196077092)

[Selected Projects: 5](#_Toc196077093)

[Section 3 9](#_Toc196077094)

[Tool Description 9](#_Toc196077095)

[Section 4 10](#_Toc196077096)

[Results: 11](#_Toc196077097)

[CatVodTVSpider Project: 11](#_Toc196077098)

[SmartTubeNext Project: 12](#_Toc196077099)

[cwa-server Project: 14](#_Toc196077100)

[RxJava Project: 15](#_Toc196077101)

[wvp-GB28181-pro Project: 16](#_Toc196077102)

[The Java code metrics of the selected projects were calculated using CK-Code and are presented in the table above. 18](#_Toc196077103)

[Findings 18](#_Toc196077104)

[Overall findings for each project: 19](#_Toc196077105)

[Section 5 20](#_Toc196077106)

[Conclusion 20](#_Toc196077107)

[References 22](#_Toc196077108)

# Section 1

The following shows how the GQM (Goal Question Metric) approach is used to find important measures.

* Please document the anticipated results of the process.
* Please provide the list of inquiries that must be addressed in order to achieve the desired objectives.
* Trace the steps back to the stated objectives.

Software development teams using the GQM (Goal-Question-Metric) method can improve their planning efficiency, assign priorities to tasks, and track their group progress toward a shared goal. Aiming to simplify decision-making procedures, raise general performance, and maximize resource allocation, the GQM approach has been successfully embraced at many organizational levels corporate, project, and procedural domains included.

The GQM approach can be used to first determine the goal of the study, then create research questions, and then build measures to evaluate software maintainability. Using the Goal-Question-Metric (GQM) approach, researchers can properly determine that their study has clear goals and measurable results, that their questions are suitably aligned, and that the data gathered is both reliable and able to provide actionable insights. Using this method also ensures the dependability and usefulness of their data.

## Applying GQM Approach

**Objective:**

This empirical study's main goal is to investigate how class size affects software maintainability using the C&K metrics.

**Questions:**

1. Does a correlation exist between the numerical designation of a class and its level of maintainability?
2. What is the relationship between the number of classes and the selection of C&K measures for assessing maintainability?

**Metrics:**

Based on the above objectives and research inquiries, the subsequent C&K metrics may be chosen as suitable measures for evaluating software maintainability:

* **Weighted Methods per Class,**

Class complexity is quantified by taking into account two factors: the total number of methods and the average complexity of each individual method.

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

The DIT metric quantifies the hierarchical levels present within the family structure of a given class.

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

The measure presented here pertains to the extent of interconnectivity among distinct groupings of things.

Measurements recommended by (Dubey & Rana, 2011) can be used to study class number and software maintainability. By means of collecting and analyzing data about a certain group of randomly selected software components, we may better understand how the size of a class influences the simplicity of maintaining software. Gathering data on a representative selection of software components chosen using a random sampling technique could help one investigate further how class size influences software maintainability. A deeper knowledge of the link between class size and software maintainability could be obtained by later studying the correlation between these measures.

# Section 2

The following criteria were established for the topic programs:

* The programs should possess a minimum code size of 10,000 lines.
* The applications should have been written within the preceding three years, with a minimum age of two years.
* It is imperative to ensure that a minimum of three programmers have made contributions to the source code of the software..

The criteria were selected to ensure that the selected programs were of adequate scale to accurately reflect real-world software systems, have been maintained, and have been collaboratively created by many developers.

Program age requirements were chosen with the goal of guaranteeing that software systems had experienced maintenance operations that could affect their maintainability. But it was also crucial to stay away from choosing programs that are too old and out of line with present technologies and practices.

The choice of program size requirement was meant to guarantee that the programs had enough complexity to produce notable results in the analysis of class size effect on software maintainability.

The creation of a requirement for the number of developers was meant to encourage cooperative development and to dissuade single authorship of programs.

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

## Selected Projects:

1. **CatVodTVSpider**

CatVodTVSpider is the name of the project. The common crawler in the new Maoying TV software has been updated using this code package. This open-source initiative lets people specify settings and creates a jar file that might be readily included.

This project being particularly for Android Studio, one must use Android Studio to change the code. Once the project has been debugged, a custom\_spider.jar file can be produced using the buildAndGenJar.bat script found in the root directory. The jar file contains the software's executable code.

Users are invited to provide their own crawler code to the project and file merge requests. Its open-source character encourages cooperation and creativity, thereby supporting the CatVodTVSpider project as a reasonable answer to the data mining problem for Maoying TV.

1. **SmartTubeNext**

The software in question is a high-quality, open-source application created exclusively for Android TVs and set-top boxes, with no expense connected with its use. Apart from providing a variety of useful features, this app's main goal is to provide content free of ads. These features include the ability to watch real-time chat, 8k video playback at a frame rate of 60 frames per second, high dynamic range (HDR), and the option to change the playback speed.

Available on the SmartTubeNext system, the SponsorBlock function lets viewers skip ads during video playback. It is also feasible to design unique icons and use the program on its own, without a Google link. The worldwide collaboration engaged in the project is well known for its friendly and encouraging attitude.

SmartTubeNext provides a consistent and simple approach that really removes annoying ads in the context of watching YouTube videos on Android TVs and TV boxes. Its unlimited access to the entire public fosters an atmosphere favorable to cooperation and the creation of new ideas.

1. **cwa-server**

The server application of the Apple and Google exposure warning API is an integral element of Germany's official Corona-Warn-App. The objective of this project is to develop mobile software compatible with both iOS and Android platforms, utilizing Bluetooth technology to facilitate the secure and anonymous sharing of protected data across mobile devices within close proximity to the user.

The application ensures the encryption and privacy of all data transmitted and received, securely storing it on individual users' devices to prevent unauthorized access. The implementation of the server responsible for managing encryption keys utilized by the Corona-Warn-App can be found in the cwa-server source code.

The cwa-server component plays a crucial role in the Corona-Warn-App initiative, which is designed to mitigate the transmission of COVID-19 within Germany. This application can prove to be a useful asset in the battle against disease due to its utilization of the Apple/Google exposure warning API and its unwavering commitment to safeguarding user privacy and data security.

1. **RxJava**

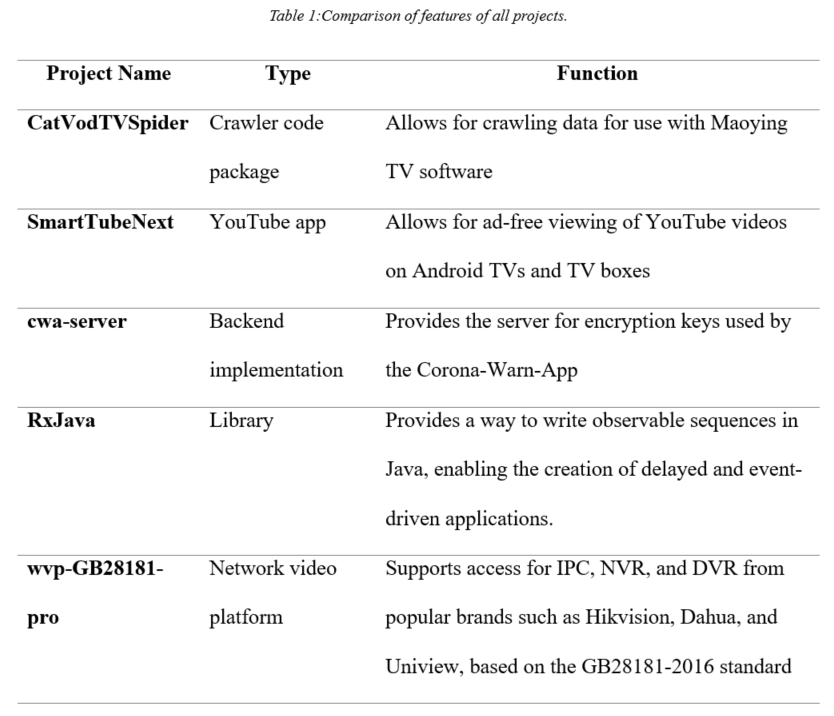
The Java framework in question supports the use of observable patterns in those with event-driven and delayed behavior. Designed to enable the generation and manipulation of observable sequences in an asynchronous and event-driven way, RxJava is a software framework. It can be seen as a version of the Reactive Extensions library.

RxJava uses the observer pattern to effectively control event and data sequences. The abstraction of low-level threading, synchronization, thread-safety, and concurrent data structures accompanies the addition of operators for explicit composition of sequences. When using event-driven and asynchronous computing in the Java programming language, RxJava is a vital tool. Programmers generally prefer the declarative administration of complex data structures and abstraction of underlying implementation concerns.

1. **Wvp-GB28181-pro**

Founded on the GB28181-2016 standard, the network video technology also supports NAT bypassing. The network gives access to well-known manufacturers of IPC (Internet Protocol Camera), NVR (Network Video Recorder), and DVR (Digital Video Recorder). Among these businesses are Uniview, Dahua, and Hikvision. The platform's capacity to permit distribution helps to route video streams to other platforms fitting national criteria.

The Wvp-GB28181-pro protocol sends push streams like rtsp and rtmp to national standard systems. Ultimately, by allowing smooth interoperability with a broad spectrum of hardware and software platforms, this study offers a useful tool for video security applications. NAT entry support and compliance to the GB28181-2016 standard guarantee users access to their streams from any geographic area.



# Section 3

## Tool Description

The GitHub platform hosts the software tool used to calculate CK-Code metrics for Java programming language.With particular emphasis on maintainability, the CK-Code metric tool is an open-source software tool specifically designed to evaluate and quantify some software quality characteristics (Mauricioaniche/Ck: Code Metrics for Java Code by Means of Static Analysis, n.d.).

Among other metrics, the software application calculates several, including Cyclomatic Complexity, Lines of Code, and Lack of Cohesion in Methods. The software industry makes great use of the above-mentioned measures to assess code quality and maintainability.

User-friendly, the CK-Code metric tool may be easily added into delivery and continuous integration systems. This lets programmers properly monitor the maintainability of their code during its lifetime. The produced reports may be examined by developers to better understand the data the tool has gathered and to spot areas where optimizations could be applied.

Programmers wishing to improve the readability, understanding, and security of their Java code should investigate the CK-Code measure tool. The platform's open-source character and smooth integration make it easily available to developers of all skill levels.

# Section 4

The results of our empirical study on the effect of class size on software maintainability are presented in this part. A particular code sample was evaluated using the CK-Code tool. Various software quality characteristics like cyclomatic complexity, lines of code, and method coherence were included in the assessment.

A mathematical study was done once the data was gathered; visualizations and spreadsheets were then produced. Research by (Chowdhury et al. 2022) shows a link between maintainability and class size. These results can help to direct the execution of alternatives and processes in software development and help to shape decisions.

The results of our study have the possibility of offering a fresh contribution to the ongoing debate on the relevance of software quality and maintainability. Our results are anticipated to benefit both software engineers and academics, and they are also anticipated to inspire more research on the factors influencing software maintainability.

## Results:

### CatVodTVSpider Project:

The analysis of the responses to questions in the GQM technique in the specified section is presented as follows:

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

With a value of 24.93, the findings imply that the project shows a rather high average of Weighted Methods per Class (WMC). Large class sizes may indicate a greater degree of complexity in the undertaking, thereby affecting its long-term viability.

Significant interdependencies across classes, as shown by the rather high score of 4.23 for the Coupling Between Objects (CBO) measure, can possibly make code maintenance more difficult.

The Depth of Inheritance Tree (DIT) measure, which shows a value of 1.5, reveals the project's low hierarchical structure. Still, taken alone, this measure could not fully reflect the general maintainability of the project.

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

Research results support a relationship between class size and maintainability suggested by the project's high Weighted Methods per Class (WMC) and Coupling Between Objects (CBO) metrics.

Though, it is crucial to note that the C&K measures used in this work provide a narrow view of maintainability. When evaluating the maintainability of software, then, one must consider other elements such code knowledge, flexibility, and documentation.

### 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?**

Compared to comparable projects, the current one shows a poor Weighted Methods per Class (WMC) rating of 11.39. A less complex task is suggested by the less classes to supervise, therefore improving its possibility for efficient upkeep.

With a low Coupling Between Objects (CBO) value of 6.29, the code shows little interdependence among classes, hence enabling simple administration.

With a Depth of Inheritance Tree (DIT) value of 1.74, the project shows a low degree of inherited components. This might therefore help to simplify and improve the control of the code.

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

The project's rather low WMC and CBO scores point to a possible link between class size and maintainability.

It should be noted, then, that the C&K measurements used in this study only suggest maintainability. When evaluating software maintainability, then, it is essential to include other elements such code understanding, flexibility, and documentation.

### 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 software systems, the project shows a less than ideal average Weighted Methods per Class (WMC) of 3.76. Consequently, one could reasonably argue that the project's maintainability could be bettered by its lower degree of complexity, which could be ascribed to the reduction in class size.

With a low Coupling Between Objects (CBO) value of 4.29, the code shows few ties between classes. This feature makes the code simple to control.

With a Depth of Inheritance Tree (DIT) value of 1.24, the project shows a quite low degree of inherited components in its structure. Furthermore, this could help to improve code simplicity and enable manageability.

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

The project's rather low WMC and CBO scores imply a likely link between class size and maintainability.

It should be noted, therefore, that the C&K criteria used in this study provide a narrow view of maintainability. A thorough evaluation of software maintainability calls for consideration of other elements such code accessibility, flexibility, and the needs..

### 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?**

Compared to comparable initiatives, the project shows a rather low average Weighted Methods per Class (WMC) of 4.08. Consequently, it is reasonable to assume that the maintainability of the project could be enhanced as a result of its simplicity stemming from a decreased class size.

A Coupling Between Objects (CBO) value of 2.52 helps the code to be more manageable since it suggests a low number of links among classes.

With a Depth of Inheritance Tree (DIT) of 1.32, the project's assessment indicates a small number of inherited sub-components..

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

The findings suggest a possible link between the project's low WMC and CBO values and the use of big class sizes throughout the study.

When assessing program maintainability, nevertheless, one must also take into account code clarity, modularity, and documentation. The C&K ratings applied in this investigation provide just a limited viewpoint.

### wvp-GB28181-pro Project:

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

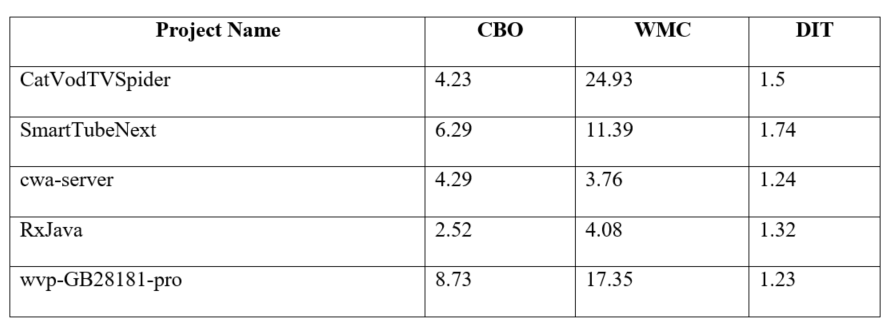
Measured at 17.35, the project's Weighted Methods per Class (WMC) score shows a great degree of quality in relation to other initiatives. Improved class features could increase the project's complexity and therefore create difficulties for future updates.

With a result of 8.73, the Coupling Between Objects (CBO) measure also indicates a large number of linkages across classes, hence complicating code maintenance.

The Depth of Inheritance Tree (DIT) measurement, at a measly 1.23, suggests the project has a low degree of heredity. Code complexity could be reduced, hence improving maintainability.

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

The strong Weighted Method Count (WMC) and Coupling Between Objects (CBO) measures of the project together with the study results show proof of a possible link between class size and maintainability. The C&K ratings in this paper offer an inadequate reflection of the general maintainability assessment. Apart from other elements, the maintainability of software should include thoughts on code clarity, structural structure, and instructional documentation.



## The Java code metrics of the selected projects were calculated using CK-Code and are presented in the table above.

## Findings

The utilization of the CK-Code metric tool facilitates the examination of the five projects through the application of the Goal-Question-Metric (GQM) method. The analysis is conducted based on the obtained results:

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

Based on the results, it can be inferred that larger class sizes are likely to have had a significant impact on the increased average working memory capacity values seen for the tasks. The potential negative impact of larger class sizes on software maintainability can be attributed to the heightened complexity that ensues.

Nevertheless, it is worth noting that the values projected by the CBO (Coupling Between Objects) and DIT (Depth of Inheritance Tree) metrics generally fall within the lower to average range. This suggests a relatively limited level of inheritance and a scarcity of interclass interactions. The possible adverse effects of larger class sizes on maintainability could potentially be mitigated with the implementation of this measure.

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

The findings indicate that projects with larger class sizes tend to exhibit higher values of WMC, suggesting a potential association between class size and maintainability.

The research exclusively employed the C&K methods for assessing maintainability, thereby resulting in inherent limitations in the conclusions. In order to provide a comprehensive evaluation of the software's maintainability, it is necessary to evaluate additional aspects such as code readability, functionality, and documentation.

### Overall findings for each project:

CatVodTVSpider's moderate Weighted Method Count (WMC), relatively low Coupling Between Objects (CBO), and low Depth of Inheritance Tree (DIT) indicate the possibility of acceptable maintainability.

SmartTubeNext's modest Weighted Method Count (WMC), fairly low Cyclomatic Complexity (CBO), and moderate Depth of Inheritance Tree (DIT) suggest that its maintainability is probably really acceptable.

The CWA-server shows low Depth of Inheritance Tree (DIT), low Coupling Between Objects (CBO), and low Weighted Methods per Class (WMC). These measures imply that the server might have good maintainability qualities given its minimal interclass dependencies and smaller total class count.

RxJava's high maintainability could be ascribed to its low class size and modest level of inheritance. The RxJava framework shows good maintainability traits as shown by its low Weighted Method Count (WMC), low Coupling Between Objects (CBO), and moderate Depth of Inheritance Tree (DIT).

Wvp-GB28181-pro's bigger class size and increased code complexity suggest that its maintainability may be somewhat hampered given higher values for WMC (Weighted Methods per Class), CBO (Coupling Between Objects), and a lower DIT (Depth of Inheritance Tree).

# Section 5

## Conclusion

The findings from the CK-Code metric application suggest that the number of classes a program has may affect its maintainability. Programs with bigger class sizes show greater working memory capacity (WMC), which supports this claim. This implies that a more complex code structure might be produced by more classes, hence creating difficulties in understanding and simplicity of modification.

Evaluating the maintainability of software using the C&K criteria used in this study provides just a narrow view on maintainability, hence it is vital to keep this in mind. The evaluation process places great importance on the assessment of code readability, organization, and explanations. A correct evaluation of the software's quality depends on using a wide range of measures and research techniques.

The results imply that, depending on the particular traits of each project under examination, class size may affect maintainability differently across projects. For instance, while showing rather low CBO (Coupling Between Objects) and DIT (Depth of Inheritance Tree), CatVodTVSpider and SmartTubeNext show larger class sizes. This approach could help to offset the negative consequences on the manageability of educational environments resulting from larger class sizes. On the other hand, the cwa-server and RxJava show smaller class sizes and less complexity, which could improve their maintainability.

Mostly because of its higher Weighted Methods per Class (WMC), Coupling between Objects (CBO), and low Depth of Inheritance Tree (DIT) ratings, the Wvp-GB28181-pro project shows poorer maintainability than comparable projects. The greater class size and more complex code of the system, which together increase difficulties in system maintenance, could help to explain this occurrence.

The results of the study imply that the maintainability of software could be affected by class size. Still, one should realize, please, that other extra elements also merit consideration. A thorough evaluation of software quality calls for the use of a varied spectrum of measures and analytical techniques. Examining many factors such as code accessibility, modularity, and documentation—helps one to evaluate a system's maintainability.

# 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