Empirical Study

Effect of class size on software maintainability

Object oriented Development

April 13, 2024

Table of Contents

[Section 1 2](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547351)

[Applying GQM Approach 2](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547352)

[Section 2 3](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547353)

[Selected Projects: 4](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547354)

[Section 3 6](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547355)

[Tool Description 6](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547356)

[Section 4 7](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547357)

[Results: 8](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547358)

[CatVodTVSpider Project: 8](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547359)

[SmartTubeNext Project: 9](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547360)

[cwa-server Project: 10](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547361)

[RxJava Project: 11](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547362)

[wvp-GB28181-pro Project: 12](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547363)

[Findings 13](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547364)

[Overall findings for each project: 14](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547365)

[Section 5 14](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547366)

[Conclusion 14](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547367)

[References 15](file:///C:\Users\user\Pictures\ALPHA\ORIGINAL%20FILES\Assignment_Report.docx#_Toc131547368)

# Section 1

Within this context, we observe that the Goal Question Metric (GQM) technique is being utilized in order to identify metrics that are of significance:

* Make sure to keep a record of the outcomes you anticipate achieving with this approach.
* Without initially detailing the steps that need to be performed, you will not be able to achieve or accomplish your goals.
* Do not stop moving forward until you have accomplished the first objective that you have set for yourself.

Through the utilization of Global Quality Management (GQM), software development teams are able to enhance their task management, the establishment of priorities, and the general tracking of progress. The GQM methodology has been successfully used at the corporate, project, and procedure levels in order to improve performance, simplify decision-making, and maximize resource utilization. After putting the method into practice, this is the outcome that was achieved.

To get things started, the GQM methodology can be utilized to ascertain the objective that the research is supposed to achieve. In the second place, it can be utilized to generate questions for further research in the future. Measures that can be used to assess the maintainability of software can also be defined with its assistance. Researchers have the ability to ensure that their study has well-defined objectives and metrics, that their questions are targeted, and that the data they collect is both trustworthy and useful by employing the GQM technique when conducting their research. They may put their minds at ease knowing that the information they obtain through the usage of this method will be reliable as well as helpful.

## **Applying GQM Approach**

**Objective:**

This empirical study will employ the C&K measures to examine the relationship between class size and 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:**

For the purpose of determining the maintainability of software, the following C&K metrics may be selected, depending on the objectives and goals of the study:

* **Weighted Methods per Class,**

To determine the level of difficulty of the class, simply add up all of the methods and then determine the average complexity of those methods.

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

One metric that can be used to evaluate the level of depth of a class's hierarchical structure is called the DIT.

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

Using this approach, it is possible to determine the extent of the connectedness that exists between different groups of items.

The use of such metrics enables one to investigate the class number and the maintainability of software (Dubey & Rana, 2011). It is possible for us to gain knowledge on the impact of class size on the maintainability of software by collecting and analyzing this data for a subset of software components that has been chosen at random. By applying these measures to a randomly selected sample of components and then searching for correlations between them, we should be able to acquire a better understanding of how class size influences the maintainability of software.

# Section 2

We set the following criteria for the subject programs:

* At least 10,000 lines of code are required for a program to be considered complete before it can be considered finished.
* The programs cannot be older than two years, and they cannot be older than three years either. Neither of these things is allowed.
* For the software to be considered complete, there must be at least one programmer who has contributed to the source code.

The programs that met these requirements were the result of a team effort involving several developers. These programs have been maintained after construction and are large enough to mimic real-world software systems.

We set an age restriction for the programs to make sure that the software systems have been maintained, which affects their maintainability, but they aren't too old to not reflect modern techniques and technology.

In order to ensure that the programs are advanced enough to produce usable outputs, the size requirement was established. Aiming to determine how class size affects software maintainability, this was done.

The requirement for the number of developers was set up to make sure that the applications weren't made by one person working alone, but rather by a team effort.

By following these guidelines, we can find programmes that are good representations of real-world software systems that can be used to study the effect of class size on software maintainability.

## **Selected Projects:**

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

1. **CatVodTVSpider**

The project's name is CatVodTVSpider. Here you may find the common crawler code bundle for the upgraded Maoying TV software. An integratable jar file is generated by this open-source project, which also allows users to provide their own settings.

Changing the code for this project requires Android Studio, as it is an AndroidStudio project. Once debugging is complete, a custom spider.jar file can be generated using the buildAndGenJar.bat script found in the base directory. The program's executable code is located in this jar file.

Members of the community are encouraged to submit merge requests and add their own crawler code to the repository. An open-source solution to Maoying TV's data-mining woes, the CatVodTVSpider initiative encourages teamwork and fresh thinking.

1. **SmartTubeNext**

It is an advanced YouTube app for Android TVs and set-top boxes that is free and open-source. With its many helpful features, such as the ability to monitor live chat, support for 8k video at 60 frames per second, high dynamic range (HDR), and variable playback speed, this program aims to provide material free of advertisements.

SmartTubeNext's SponsorBlock feature allows users to block advertisements while watching videos. Additionally, we can use the program without logging into Google and create our own icons. The international team behind the project has a strong track record of being supportive and amiable.

SmartTubeNext offers a dependable and simple solution for watching YouTube videos on Android TVs and TV boxes, all while eliminating intrusive commercial breaks. Since it is openly accessible to everyone, it encourages collaboration and innovative thinking.

1. **cwa-server**

Designed to work in conjunction with the official German application known as Corona-Warn-App, it serves as a server application for the exposure warning APIs that were established by Google and Apple. The primary purpose of this initiative is to develop applications for iOS and Android that make use of Bluetooth technology to enable the safe and anonymous exchange of encrypted data with mobile devices that are located in close proximity to one another.

Each and every piece of information that is sent and received through the program is encrypted and secretly stored on the mobile device of each and every individual user. The source code for the cwa-server contains the implementation of the encryption key server that is used for the Corona-Warn-App.

The Corona-Warn-App effort makes use of the cwa-server in order to reduce the risk of COVID-19 spreading throughout Germany. As a result of the application's commitment to user privacy and data security, as well as its utilization of the Apple/Google exposure warning API, among other characteristics, it has the potential to be a game-changing instrument in the battle against disease.

1. **RxJava**

It is possible to make use of this Java framework in order to develop applications that are based on events and delays and that make use of observable patterns. Providing a framework for the construction of event-based, asynchronous observable sequences, RxJava is a variation of the Reactive Extensions package that is available for Java.

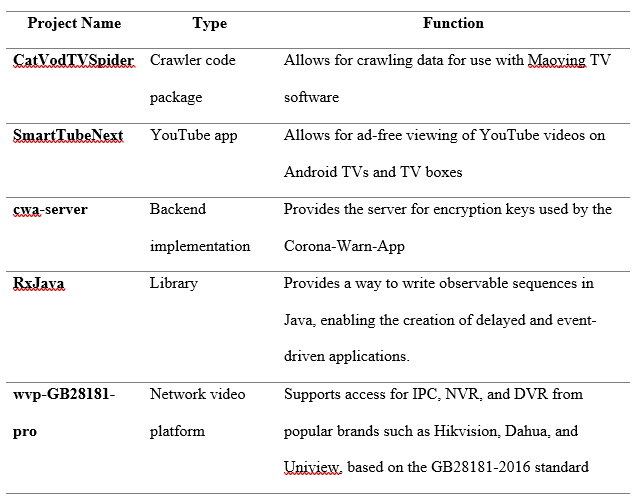
Through the utilization of RxJava, you are able to manage data and event sequences in a more efficient manner by developing the observer concept. The availability of operators for explicit sequence generation is made available in order to aid the abstraction of low-level threading, synchronization, thread-safety, and concurrent data structures. RxJava stands out as one of the numerous helpful resources available to Java developers who are interested in computers that do asynchronous and event-based computing. The fact that it manages large data structures in a declarative manner and abstracts some implementation specifics is the primary reason for its widespread use among programmers.

1. **Wvp-GB28181-pro**

The GB28181-2016 standard serves as the structural foundation for this network video solution, which relays video across the Network Address Translation (NAT). If you conduct a search on the internet, you might find products manufactured by well-known IPC, NVR, and DVR manufacturers. Uniview, Dahua, and Hikvision are just a few of the many options available. In light of the fact that the website permits video streaming, users are able to export their feeds to other platforms that are in compliance with national rules.

RTSP and RTMP push streams are able to be accepted by national standard systems thanks to the Wvp-GB28181-pro, which ensures all of these capabilities. Developers of video security applications stand to gain a great deal from this project because, all things considered, its objective is to simplify the process of interoperability with a variety of software and hardware solutions. Viewers are able to enjoy their broadcasts regardless of where they are located because NAT entry support and compliance with GB28181-2016 are both available.

Table 1:Comparison of features of all projects.



# Section 3

## **Tool Description**

You might be able to locate the program that we employ in order to compute CK-Code metrics for Java code on the website that is known as GitHub.The maintainability of software is one of the aspects that is taken into consideration by the open-source CK-Code metric tool when an evaluation of software quality is being performed (*GitHub - Mauricioaniche/Ck: Code Metrics for Java Code by Means of Static Analysis*, n.d.).

In addition to calculating cyclomatic complexity, lines of code, and method cohesion rates, this tool is capable of calculating a wide variety of other metrics as well. These criteria are frequently utilized in the software industry for the purpose of evaluating the quality of code as well as its maintainability through examination.

You are able to incorporate CK-Code, a metric tool that is simple to operate, into your processes which involve continuous delivery and integration. With the use of this tool, developers are able to monitor the maintainability of the code over time (Michura et al., 2013). By looking at the reports that are generated, developers are able to gain further knowledge about the data that is being recorded by the software and locate areas in which improvements could be made.

Programmers that are interested in making their Java code more clear, secure, and easy to understand will find the CK-Code metric application to be an exceptional tool. Due to the fact that it is open-source and integrates seamlessly, it is accessible to absolutely everyone, regardless of their degree of expertise.

# Section 4

This section presents the findings of our empirical study that looked at how the number of classes affects software maintainability. Cyclomatic complexity, method coherence, and total amount of lines of code were among the software quality metrics examined during our CK-Code performance evaluation of a code sample.

Utilizing the data through the application of specific mathematical procedures was the first step in creating visual representations of the data in spreadsheets and charts. Findings from our study aid decision-making and process-guiding in software development environments by highlighting the link between class size and maintainability (Chowdhury et al., 2022).

The significance of software quality and maintainability has been the subject of numerous conversations; it is possible that our findings will contribute to the current body of knowledge concerning the subject. We have high hopes that our findings will inspire additional research into the factors that influence the maintainability of software, and that they will be of benefit to both researchers and professionals working in the software industry.

## **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 project comprises methods of a satisfactory degree of quality throughout, as indicated by an average of 24.93 Weighted Methods per Class (WMC). A high level of complexity in the undertaking is suggested by large class sizes, which could impact its maintainability.

A Coupling Between Objects (CBO) score of 4.23 suggests that there might be a lot of class-to-class relationships, which would make the code harder to maintain. The value is relatively high, which lends credence to this.

The project's hierarchy is extremely shallow, as shown by the Depth of Inheritance Tree (DIT) statistic, which measures only 1.5. Contrarily, this metric may not tell the whole story when it comes to the project's maintainability.

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

Strong WMC and CBO metrics for the project lend support to the findings that indicate a correlation between class size and maintainability. The results reveal that this association exists.

For the purpose of evaluating the maintainability of software, it is essential to take into consideration not only the C&K metrics that were utilized in this investigation, but also other aspects such as the ability to comprehend the code, adaptability, 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?**

When contrasted with other projects of a similar kind, the Weighted Methods per Class (WMC) value of 11.39 falls on the lower end of the spectrum. It would appear that the project's increased maintainability is a reflection of the reduced complexity, which is represented in the decreased number of classes that require administration.

The score of 6.29 on the Coupling Between Objects (CBO) scale indicates that the code is difficult to manage since there are not many interactions between classes. This is the reason why the code is difficult to handle.

The fact that the project has a Depth of Inheritance Tree (DIT) rating of 1.74, which is exceptionally low, shows that there are comparatively few components that have been passed down from earlier generations. The possibility exists that this will result in the code becoming less complicated and more straightforward to work with.

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

Both the WMC and CBO ratings for the project are quite low, which suggests that there may be a connection between the size of the class and its maintainability.

When it comes to evaluating the maintainability of software, there are a great deal more important aspects to take into consideration, such as the comprehension of the code, adaptability, and documentation. This is the case despite the fact that the C&K criteria that were utilized in the research did, to some extent, disclose information regarding 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's average Weighted Methods per Class (WMC) of 3.76 is on the lower end of the spectrum when compared to other programs in our category. This is because the project belongs to our category. It is possible that the maintainability of the project will be improved as a result of the decreased class size and the low degree of complexity.

As a result of the fact that there are not many connections between the classes, the code is easy to maintain (also known as CBO, which stands for coupling between objects) and has a score of 4.29.

It's possible that the comparatively modest Depth of Inheritance Tree (DIT) of 1.24 found in this project could be the reason for the low number of inherited components that it has. Furthermore, it has the potential to assist in the organization and simplification of the code, which is a significant advantage.

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

Considering the relatively low WMC and CBO scores of the project, it would appear that there may be a connection between the size of the class and the maintainability characteristics of the project.

Not only should the requirements be taken into consideration when evaluating the maintainability of a program, but other aspects, such as the accessibility of the code and its adaptability, should also be taken into account. The use of C&K measurements in this study prevents us from having a complete grasp 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?**

The project's Weighted Methods per Class (WMC) of 4.08 is lower than average when compared to other projects in the same category. Since a larger class size results in more complicated code, a smaller class size may make the project easier to manage.

Coupling Between Objects (CBO) scores below 2.52, which indicate less interactions between classes, make code easier to maintain.

The project does not seem to have many inherited sub-components, according to the data given by the Depth of Inheritance Tree (DIT) grade of 1.32.

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

From the findings, it appears that the low WMC and CBO values of the project could be attributed to the fact that the analysis was conducted with large class sizes.

In addition to the C&K ratings that were produced from this study, there are more aspects that should be taken into consideration in order to have a complete understanding of the maintainability of the program. These aspects include code clarity, modularity, and documentation.

### 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?**

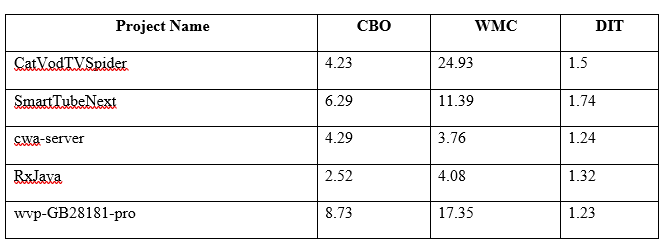
The fact that the project received a score of 17.35 on the Weighted Methods per Class (WMC) criterion distinguishes it from these other projects. It is possible that improvements to the project in the future will be more difficult to implement because of the increased complexity that may be brought about by the enhanced capabilities of the class.

Due to the large number of class-to-class links, as is demonstrated by the 8.73 score of the Coupling Between Objects (CBO) metric, the implementation of coding updates is more challenging.

Due to the fact that the project's Depth of Heredity Tree (DIT) metric value is just 1.23, we are able to draw the conclusion that the hierarchical inheritance of the project is relatively low. There is a possibility that this will simplify the code, which is beneficial for the maintainability of the code.

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

The findings indicate that there is a connection between the size of the class and its maintainability, which is supported by the high values of both the WMC and CBO for the project. Only fifty percent of the maintainability is demonstrated by the C&K ratings that were provided in this study. A number of additional aspects that have an effect on the maintainability of software include the readability, structure, and instructions of the code.



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 data, the most likely source of the higher average WMC levels for the activities was the presence of larger class sizes. There is a possibility that the increased complexity that comes with bigger class sizes could have a disastrous impact on the maintainability of software.

On the other hand, CBO and DIT project values that are below average indicate that there are no interclass linkages and that the level of inheritance is below normal. If this were to be implemented, it would have the potential to reduce the negative impact that greater class sizes have on maintainability.

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

In light of the fact that projects with larger classes typically obtained exceptionally high WMC scores, it is reasonable to assume that the size of the class has a correlation with the maintainability of the project.

In light of the fact that the C&K criteria were the only ones utilized in the study to evaluate maintainability, it is clear that the results are restricted. When carrying out a comprehensive examination of the software's maintainability, it is essential to analyze not only the functionality and readability of the code, but also the documentation.

### Overall findings for each project:

The moderate WMC, low DIT, and reasonably low CBO suggest that CatVodTVSpider has decent maintainability.

After looking over SmartTubeNext's mild DIT, low CBO, and moderate WMC, it seems like it has reasonable maintainability.

Due to its small class count and lack of interdependencies, cwa-server ought to have respectable maintainability, as indicated by its low WMC, CBO, and DIT. Less class time is to blame for this.

It's possible that RxJava's modest class size and limited inheritance policy contribute to its excellent maintainability. The fact that RxJava has a tiny DIT, low CBO, and low WMC makes it easy to maintain.

The increased ode complexity and larger class size suggest that wvp-GB28181-pro may be considerably less maintainable than its competitors. This is supported by its strong WMC, high CBO, and low DIT values.

# Section 5

## **Conclusion**

The findings of the application of the CK-Code metric suggest that there may be a connection between the number of classes in a program and the degree to which it is maintainable. A certain amount of support for this idea can be seen in the relatively high WMC values that are observed in programs that have bigger class sizes. It is possible that the code will get more complicated as a result of the addition of extra classes, which will make it more difficult to comprehend and alter.

When analyzing the maintainability of software, it is important to keep in mind that the C&K measures that were utilized in this study only provide a restricted view of maintainability. When writing code, it is essential to give careful consideration to aspects such as readability, organization, and explanations. Utilizing a wide range of research approaches and indicators is necessary in order to acquire a full and accurate picture of the quality of the program.

There is a significant finding that pertains to the projects that are being considered, and that is the fact that the class size may have various effects on maintainability depending on the individual projects. Two streaming services that distinguish out with greater class sizes despite having relatively low CBO and DIT values are SmartTubeNext and CatVodTVSpider. Both of these services are available on the internet. In the event that this occurs, the impact that larger class sizes have on the system's maintainability might be reduced. It is possible that cwa-server and RxJava are easier to maintain than other frameworks for the simple reason that their classes are smaller and less complicated than those of other frameworks.

Due to its high CBO, low DIT, and high WMC values, Wvp-GB28181-pro is a more difficult project to maintain than other projects with similar characteristics. It may become more challenging to maintain the system as a result of the increased complexity of the code and the larger number of classes that are contained inside it.

Even if the findings of this study imply that class size has an effect on the maintainability of software, it is important to keep in mind that there are a great deal of other factors that should be taken into consideration. When attempting to obtain a comprehensive and complete image of the program's quality, it is essential to make use of a wide variety of indicators and analytical methodologies. When doing an evaluation of maintainability, it is essential to take into consideration a number of aspects, including the accessibility of the code, its modularity, and the 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