

SOEN 6611: SOFTWARE MEASUREMENT

Milestone 2



|  |  |
| --- | --- |
| **Student ID** | **Name** |
| 40024381 | Amar Kumar |
| 40018973 | Charanpreet Kaur |
| 40013983 | Ajeeta Kaveti |
| 40021928 | Tarun Jeet Singh |
| 40019612 | Navnoor Multani |
| 40010983 | Lovish Tayal |
| 40031066 | Bitta Rani Rana |
| 40012480 | Mandeep Kaur |

**Table of Contents**

[LITERATURE REVIEW 1](#_Toc474528862)

[REFERENCES 4](#_Toc474528863)

[APPENDIX I: ACRONYMS & ABBREVIATIONS 5](#_Toc474528864)

# LITERATURE REVIEW

* Madhu Rohilla et al. [1] conducted a study to describe how CK’s object-oriented metric suite is useful to illustrate fault-proneness of the system. Quality of the software systems should be maintained from the very start of SDLC. This motivates us to implement such metrics, which can be used to improve the quality and reliability of the software system so that it can be easily maintained. Various hypothetical examples, to access the applicability CK metric suite (WMC, DIT, CBO, RFC, NOC, and LCOM) to predict the bugs in classes, are used. Logistic Regression Method [8] is used to study the relationship between the metrics and fault-proneness of classes. This methodology is adopted here. The empirical results suggest that the metrics provide better results as compared to traditional measures. DIT metric is the best metric to predict the fault-proneness of classes and it is the most useful to improve the quality and reliability of the design. In our study, we will be analyzing the fault-proneness and maintainability of jEdit Software over various versions using MOOD and CK metric suite. The similarity with the reviewed paper is that it predicts the fault-proneness of Object Oriented Systems using CK Metric Suite. However, the difference is that that no particular software system has been studied, rather a set of general examples is taken to understand the applicability of metrics. The data used for analysis is not from some industrial project. This turns out to be a limitation. To improve our study, we will be analyzing numerous versions of jEdit so that we have abundant data to work on and to obtain accurate and efficient results using suitable metric suites.
* Aman Kumar Sharma et al. [2] investigated the software quality of the three latest versions of JfreeChart software using CK Metrics and MOOD. The result obtained from the empirical analysis helps the software developers in improving the software quality while developing software products using the Object-oriented (OO) approach. The CK Java Metrics software tool is used on JfreeChart to compute metrics namely WMC, NOC, DIT, RFC and CBO and the Essential Metrics software tool is used to compute the MOOD suite metrics namely AHF, MHF, AIF, MIF, and POF. It can be analyzed that the versions of JfreeChart considered in this study are flexible, extendable and usable but are error-prone and lack security. It is found that WMC- RFC, CBO-RFC, and WMC-CBO are highly correlated. Among MOOD, metrics AIF – MHF and POF – AIF are negatively correlated, whereas POF – MHF is positively correlated. The similarity of our research with this related study is that we will also use MOOD metrics and CK metrics to predict maintainability and fault-proneness of the software product. In this study, the researchers have worked on three latest versions of JfreeChart while we will work on total 15 versions (first five versions, middle five versions and last five versions) of jEdit software. The limitation of this related work is that the analysis has been done on less empirical data as the researchers have used only three latest versions to predict the software quality. So, we will analyze 15 versions of jEdit software to predict maintainability and fault proneness.
* Ritu Chauhan et al. [3] investigated that the main goal of software quality estimation is to identify and minimize error-prone tasks to reduce development cost. In the software industry, corrective actions for successful software completion comes very late, which catalyze poor quality with reduced capabilities of the software. Therefore, the motivation behind this research is to overcome this by using object-oriented metrics like CK, MOOD, and QMOOD for estimating software quality early in the SDLC. In this research, they have analyzed QMOOD, CK and MOOD metrics and for the case studies, they have used JAVA RMI classes and subclasses to determine the impact of different metrics. After applying MOOD metrics indicators like MHF, AHF, MIF, AIF etc. they have shown that MIF/AIF are the measures of inheritance and by increasing this value results in low understandability and testability of the system. In a similar manner, they have shown results with different indicators of all three given metrics. The main conclusion of the study is that it is very important to measure software quality early so that by using different metrics like QMOOD, MOOD, and CK we can estimate software quality and can improve it to make the best system. Our study also focuses on to improve error-proneness of the system by using object-oriented metrics like CK and MOOD. The main difference is that in our study we are using only two metrics MOOD and CK to improve maintainability and error-proneness of the jEdit software, which is written in java but in this research, they have used QMOOD, CK and MOOD metrics to improve error-proneness to minimize development cost on JAVA RMI classes and subclasses. The main limitation of the related paper is that the researchers have not explained in details about given metrics and how they have calculated the entire values from these metrics. In our study, we have explained in detail about MOOD and CK metrics and their computation in order to reach a particular conclusion.
* Jubair Al-Ja’afer et al. [4] investigated that the problem statement is to assess the quality of the software system quantitatively and objectively using object-oriented metrics. MOOD metrics have been used to assess the quality using different indicators like AHF, MHF, MIF, AIF, COF, and POF. A range of acceptable values of the MOOD metric indicators has been created that are used to calculate the grade of the program. Then the grade of the program is analyzed to identify which indicator needs to be improved in order to increase the quality of the program. For example, the number of inherited methods in the class are very much lower than the total number of the methods. As the inherited methods, increase the quality of the program will also increase. We have learned from this study that we can analyze our project results obtained from the different releases to figure out how the program quality is varying in term of error-proneness and maintainability. The similarities of this study with our study are that they have used MOOD metrics and they have assigned weights to each indicator according to their importance. Coupling factor is the most important among others and we can use this in our study as the coupling factor increases the chances of having an error in the program also increases. The difference between this study and our study is that they have assigned lower weight to MHF and in our case, we will be considering MHF as one of the key factors. However, the limitation is that this system can be used with the large-scale java programs. In our study, we will aim to modify the weight of the indicators according to our hypothesis so we can achieve them.
* Dr Sonal Chawla et al. [5] conducted a study to investigate the impact of OO metrics and static code metrics on the quality of the software systems where software maintainability is a major concern these days. It has been indicated that modern software systems without object-oriented design are incomplete. Every system has its own complexity, which should be measured to improve the quality of the system. In this research, firstly Static code metrics are described and performed without the execution of code and object-oriented metrics are based on studying the code behavior during execution. Then, the metrics have been summarized because of relevance to find the complexity. Dynamic metrics study the code at run time. Only object-oriented metrics allow the modifications to reduce the cost effectiveness, time consumption and improve the quality. Object-oriented programs also use Halstead Metrics but some essential factors like inheritance, coupling remain uncovered using these metrics. The authors concluded that measurement of the software quality is complex for the development of the software product as metrics play a significant role in determining the complexity and maintainability of the software code. The similarity of reviewed paper with our study is that it focuses on maintainability using the object-oriented metrics and the difference is that in our paper we have not focused on static code metrics. In our study, we will analyze different versions of jEdit to achieve our hypotheses using the OO metrics.
* Jeenam Chawla et al. [6] emphasized on finding ways to predict the quality attributes of the software or system by analyzing the object-oriented metrics. Main focus is given to object oriented metrics given by CK in order to correctly find out the fault-proneness of the system. Prediction and detection of faults and defects accurately will enable the software engineers to easily correct them, thereby improving the quality and reliability of the software. Extensive study of literature by different authors was done taking into account the wide number of aspects such as, fault proneness, normalized rework, maintenance, defect density, understandability, reusability etc. to define the quality of the software or system. Class level object oriented metrics was used to perform the investigation. It has revealed that CBO metric is best in predicting fault proneness. However, LOC metric is suitable for quick fault prediction. The study suggests that one can make use of metrics as means to efficiently define the quality of a system and relate them to economic variables such as productivity, rework effort, etc. The DIT metric was found to be the best metric to predict the fault-proneness of classes and it is very useful to find and improve the quality and reliability of the design. They have stated that the class components in the MOOD metrics suite are not good class fault-proneness predictors and that only CK metrics are enough to define the quality of the system. This need not be true for every software system. Using CK metrics alone gives way to small errors or rework etc. that will lead to low quality software. Other metrics must be accounted for best quality systems. Our work will be done in around 15 open source Java versions. This range of versions will help in comprehensive research and study of the metrics to get more optimized results. We will analyze CK and MOOD metrics to define the overall quality of the system and find the best quality factor metrics.
* Mr. U. L. Kulkarni et al. [7] conducted a study on six java based open source software systems to validate CK metrics to evaluate the quality of the system and to find probable design faults at both architectural and component level. Object oriented design system has become very extensive and complex, so there is a need of quantitative measurements to assess the quality of design to improve quality of the system. CK proposed a set of six object oriented design metrics, which encapsulate the concept of coupling, cohesion and inheritance. These metrics includes WMC, RFC, LCOM, CBO, DIT, and NOC. The result of the study indicates that low values of DIT and NOC in the system implies improper use of inheritance. In contrast to it, CBO values close to two is not a problem. From the study, we have learnt that WMC, LCOM and CBO are related to each other and high values of these three increases complexity of the system. Moreover, there could be some scope to use inheritance in a more accurate manner. In this paper, authors use CK metrics suite to find out design quality of the product or system and in our study, we are also trying to reduce faults in design to enhance the quality and reduce complexity. The basic difference is that in this study authors use CK metrics and in our study, we are using CK along with MOOD metric. The limitation of the related work is that this study covers the concept of inheritance, coupling and cohesion but not encapsulation. In our study, we will try to study metrics in detail and to try them for various systems to assess the quality of the software.

# REFERENCES

1. Madhu Rohilla and P. K. Bhatia,” Prediction of Fault- Proneness Using CK Metrics,”in *International Journal of Emerging Technology and Advanced Engineering* (ISSN 2250-2459, ISO 9001:2008 Certified Journal, Volume 3, Issue 8, August 2013).
2. Aman Kumar Sharma, Arvind Kalia, and Hardeep Singh, “Empirical Analysis of Object Oriented Quality Suites,” in *International Journal of Engineering and Advanced Technology (IJEAT) ISSN: 2249 – 8958*, *Volume-1, Issue-4*, April 2012.
3. Ritu Chauhan,Rahul Singh, and Ashish Saraswat, “Estimation of Software Quality using Object Oriented Design Metrics,” in *International Journal of Innovative Research in Computer and Communication Engineering (An ISO 3297: 2007 Certified Organization)* *Vol. 2, Issue 1,* January 2014.
4. Jubair Al-Ja’afer and Khair Eddin Sabri,” Metrics for Object Oriented Design (MOOD) to Assess Java Programs,” in *International Arab Conference on Information Technology (ACIT’04)*, Algeria, 2004.
5. Dr Sonal Chawla and Gagandeep Kaur, “Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development”, *(IJACSA) International Journal of Advanced Computer Science and Applications*, Vol. 4, No. 9, 2013.
6. Jeenam Chawla and Arun Agarwal, “Object-Oriented Design Metrics to Predict Fault Proneness of Software Applications,” *(IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (3),* 2014, 3318-3321*.*
7. U. L. Kulkarni, Y. R. Kalshetty, and Vrushali G. Arde, “Validation of CK metrics for Object Oriented Design Measurement,” Third International Conference on Emerging Trends in Engineering and Technology 978-0-7695-4246-1/10 $26.00 © 2010 IEEE DOI 10.1109/ICETET.2010.159.
8. T. Gyimóthy, Rudolf Ferenc, and Istva´n Siket, “Empirical validation of object-oriented metrics on open source software for fault prediction,” in [*IEEE Transactions on Software Engineering*](http://ieeexplore.ieee.org/xpl/RecentIssue.jsp?punumber=32) (Volume: 31, [Issue: 10](http://ieeexplore.ieee.org/xpl/tocresult.jsp?isnumber=32934), Oct. 2005).

# APPENDIX I: ACRONYMS & ABBREVIATIONS

|  |  |
| --- | --- |
| **A** | |
| AHF | Attribute Hiding Factor |
| AIF | Attribute Inheritance Factor |
| **C** | |
| CBO | Coupling between Objects |
| CK | Chidamber and Kemerer |
| COF | Coupling Factor |
| **D** | |
| DIT | Depth of Inheritance Tree |
| **L** | |
| LCOM | Lack of Cohesion in Methods |
| **M** | |
| MHF | Method Hiding Factor |
| MIF | Method Inheritance Factor |
| MOOD | Metrics for Object Oriented Design |
| **N** | |
| NOC | Number of Children |
| **O** | |
| OO | Object Oriented |
| **P** | |
| POF | Polymorphism Factor |
| **Q** | |
| QMOOD | Quality Model for Object Oriented Design |
| **R** | |
| RFC | Response for Class |
| **S** | |
| SDLC | Software Development Life Cycle |
| **W** | |
| WMC | Weighted Method per Class |