Empirical Analysis of Mood Metrics, CK Metrics and Code Smell Detection of Go Project

|  |  |
| --- | --- |
| Maloy Kanti Sarker- BSSE0834 | Md. Shihab Shohrawardi-BSSE0813 |
| Abdullah Al Jubaer-BSSE0812 | Tulshi Chandra Das-BSSE0811 |

Institute of Information Technology

University of Dhaka

**Abstract**- *With the rise of the OO paradigm has come the acceptance that conventional software metrics are not adequate to measure object-oriented systems. This has inspired a number of software practitioners and academics to develop new metrics that are suited to the OO paradigm. The MOOD metrics have been subjected to much empirical evaluation, with claims made regarding the usefulness of the metrics to assess external attributes such as quality and maintainability. We evaluate the MOOD metrics and CK metrics on 5 open source GO projects and show the result in a csv file. We measured 5 MOOD metrics and 5 CK metrics and 2 code smells from the projects. This is reflected in the increasing number of industrial software tools, such as Rational Rose, that enable automated computation of these metrics. Even though this metric suite is widely, empirical validations of these metrics in real world software development setting are limited.*

**Index Terms- mood matrix, CK metrics, GO**

III.INTRODUCTION

Software quality metrics concentrate on process, product and project quality elements. The goal of software quality metrics is to identify the improvement of project, planning, process and product. Various studies on software metrics distribution share the same objective of offering a way of improving software development life cycle.

A metric is a standard unit of measure, such as meter or mile for length, or gram or ton for weight, or more generally, part of a system of parameters, or systems of measurement, or a set of ways of quantitatively and periodically measuring, assessing, controlling or selecting a person, process. [1] There were several studies to explain how metrics can be used to enhance the software quality. [2] [3] The calculation of a specific property of system performance or efficiency is a measure of the software metric (noun). A rule to measure a computer software object feature or attribute. Computer organizations, including specification reports, software object models and database structure models, can use metrics.

Since the early 1990s, object oriented (OO) techniques have beencommon in software development. Researchers have proposed se-veral metrics, including metrics for Chidamber & Kemerer (CK) [4] and object-orientated design (MOOD) [5] for quality assurance of OO software.

II. BACKGROUND

F.B Abreu presents a MOOD array of paradigms that defines the use in software code of object-oriented paradigms. These measures contribute to an object-based system assessment of efficiency and productivity. MOOD is referred to as encapsulation (MHF, AHF) as the basic structural mechanism of the object-oriented paradigm. [6], inheritance (MIF, AIF), [7] polymorphism (POF), and message passing (COF). Two main features are used in MOOD models: methods and attributes in every metric. Methods are used to perform many types of operations on objects such as achieving a status change. Attributes are used to represent the status of each entity in the system.

In the field of fault forecasting, several researchers have done significant work. The literature survey consists of developing CK parameters to explore various techniques used to model fault prediction. For object-based (OO) code, CK metric suit is the most used metrics. Chidamber et al. [8]have developed and implemented a new set of Object-Oriented Design application metrics. They found that Object Oriented can contain some of the application crisis solutions.

III. DATASET DESCRIPTION

We collected 5 Go open source project from GitHub. 4 MOOD metrics and 4 CK metrics are calculated from these projects.

IV**.** METHODOLOGY

The study includes the following metrics.

**Line of Code**

The express “lines of code” (LOC) is a metric by and large utilized to assess a software program or codebase concurring to its estimate. It could be a common identifier taken by including up the number of lines of code utilized to type in a program.

**Total Comments**

In computer programming, a comment may be a programmer-readable clarification or explanation within the source code of a computer program. They are included with the reason of making the source code less demanding for people to get it, and are by and large overlooked by compilers and translators.

**Weighted Method Count (WMC)**

The Weighted Method Count or Weighted Method per Class metric was originally defined in A Metrics Suite for Object Oriented Design. The WMC metric is characterized as the sum of complexities of all methods declared in a class. This metric is a great indicator how much exertion will be vital to preserve and create a particular class.

**Lack of Cohesion between Methods (LCOM)**

Lack of cohesion in methods. Cohesion refers to the degree of the intra-relationship between the elements in a software module such as packages and classes. It is ideal that each element has a strong relationship in the module by achieving a particular functionality. The LCOM metric indicates a set of methods in a class is not strongly connected to other methods

*𝐿𝐶𝑂𝑀= (𝑚∗𝑎/𝑣)/m-1*

Where,

*I. m: the number of methods in the class*

*II. a: the number of methods in a class that access an instance variable.*

*III. V: the number of instance variable*

**Attribute Hiding Factor (AHF)**

This is the degree of the invisibilities of attributes in classes. The invisibility of an attribute refers to the rate of the overall classes from which the attribute is invisible. It can be calculated by summing the invisibility of each property with regard to the other classes within the project. Within the previous calculation of invisibility, private=1, public=0, protected = Size of the legacy tree / Number of classes. AHF is a fraction.

*Where:*

*Ah(Ci): hidden Attributes in class Ci*

*Ad(Ci) = Av(Ci) + Ah(Ci): Attributes defined in Ci*

*Av(Ci): visible Attributes in class Ci*

*TC: Total number of Classes.*

**Method Hiding Factor (MHF)**

MHF is the measure of the invisibilities of methods in classes. The invisibility of a method refers to the percentage of the total classes from which the method is invisible. It can be calculated by summing the invisibility of each method in respect to the other classes in the project. In the previous calculation, private=1, public=0, protected = Size of the Inheritance tree / Number of classes. MHF is a fraction.

*Where:*

*Mh(Ci): hidden Methods in class Ci*

*Md(Ci) = Mv(Ci) + Mh(Ci): Methods defined in Ci*

*Mv(Ci): visible Methods in class Ci*

*TC: Total number of Classes.*

**Attribute Inheritance Factor (AHF)**

This is the fraction of class attributes that are inherited. The expression for calculating it requires summing the inherited attributes for all classes from its super-classes in a project.

*Where:*

*Ai: inherited Attributes*

*Aa(Ci) : Ad(Ci) + Ai(Ci)*

*Ad: defined Attributes*

*TC: Total number of Classes.*

**Method Inheritance Factor (MHF)**

This is obtained by dividing the total number of inherited methods by the total number of methods. The total number of inherited methods is obtained by summing the number of operations that a class has inherited from its super-classes.

*Where:*

*Ma(Ci): Md(Ci) + Mi(Ci)*

*Mi: inherited Methods*

*Md: defined Methods*

*TC: Total number of Classes.*

***Code Smell Detection:*** *We measured 2 code smells from the each project. The 2 code smells are I. God Struct and ii. Feature envy*

V.RESULT

VI**.** FINDINGS

VII**.** CONCLUSION

# References

|  |  |
| --- | --- |
| [1] | A. Kaur, K. S. Kahlon and D. P. S. Sandhu, "Empirical Analysis of CK & MOOD Metric Suit," *International Journal of Innovation, Management and Technology,* vol. 1, p. 447, 5 December 2010. |
| [2] | H. Barkmann, R. Lincke and W. Low, "Quantitative Evaluation of Software Quality Metrics in Open-Source Projects," *IEEE,* 2009. |
| [3] | V. Basili, L. Briand and W. Melo, "A Validation of ObjectOriented Design as Quality Indicators," *IEEE TSE,* vol. 22, pp. 751-761, 1996. |
| [4] | S. Chidamber and C. Kemerer, "A Metrics Suite for ObjectOriented Design," *IEEE Trans,* vol. 20, pp. 479-493, 1994. |
| [5] | R. Harrison, S. Counsell and R. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEE,* vol. 24, pp. 491-496, 1998. |
| [6] | D. Parnas, "On the Criteria to be Used in Decomposing Systems into Modules," *ACM,* vol. 15, pp. 1053-1058, 1972. |
| [7] | B. Kitchen, N. Fenton and L. Pfleeger, "Towards a Framework for Software Measurement Validation," *IEE,* vol. 21, pp. 929-944, 1995. |
| [8] | .. Shyam R, Chidamber and Chris and F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE,* vol. 20, 1994. |
| [9] | http://betterthesis.dk/research-methods/empirical-studies, "Better Thesis, Empirical Studies," Retrieved September 11, 2019. |
| [10] | Stackoverflow, "Developer survey Results 2019”, from," *https://insights.stackoverflow.com/survey/2019#technology,* 11 sepetember 2019. |
| [11] | Githut, "A SMALL PLACE TO DISCOVER LANGUAGES IN GITHUB.," *[Online]. Available: https://githut.info/,* 11 September 2019. |
| [12] | D. KOSTANJEVEC, M. PUŠNIK, M. HERIČKO and B. ŠUMAK, "A Preliminary Empirical Exploration of Quality Measurement for JavaScript Solutions.," in *6th Workshop of Software Quality, Analysis, Monitoring, Improvement, and Applications*, Belgrade, Serbia, 2017. |
| [13] | S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering,* vol. 20, no. 6, p. 476–493, 1994. |
| [14] | S. R. Chidamber, P. D. Darcy and C. F. Kemerer, "Managerial use of metrics for object-oriented software: An exploratory analysis. Software Engineering," *IEEE Transactions on,* vol. 24, no. 8, p. 629–639, 1998. |
| [15] | T. Zimmermann and N. Nagappan, "Predicting defects using network analysis on dependency graphs," in *In Proceedings of the 30th international conference on Software engineering*, Leipzig, Germany, 2008. |
| [16] | M. Tyagi, D. Bellin and M. Tyler, "Object-Oriented Metrics:An Overview," *Computer Science Department,North Carolina A ,T state University,* Vols. Greensboro, Nc 27411-0002.. |
| [17] | M. Selakovic and M. Pradel, "Performance Issues and Optimizations in JavaScript: An Empirical Study.," in *ICSE '16 Proceedings of the 38th International Conference on Software Engineering*, New York, 2016. |
| [18] | V. . R. Basili, G. Caldiera and H. D. Rombach, "THE GOAL QUESTION METRIC APPROACH," PDF retrieved, 2019. |

APPENDIX

Abbreviations

Source of Go Projects