International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)
Web Site: www.ijettcs.org Email: editor@ijettcs.org, editorijettcs@gmail.com
Volume 9, Issue 2, March - April 2020 ISSN 2278-6856

Measuring Quality Metrics using Object Oriented Approach

Lalima Soni[1], Indrajeet Singh Chouhan[2]

[1]ME, Department of Computer Engineering,
Institute of Engineering & Technology, DAVV Indore (M.P), India

[2]BE, Department of Computer Engineering,
Jai Narain College of Technology, RGPV Bhopal (M.P), India

Abstract - An object oriented approach is used in different Software project development, and in this approach different classes and objects are interconnected with each other. The interconnection between classes and objects implied that there is also some interconnection between different Object Oriented Metrics and these connecting metrics further can be used for deriving other types of metrics like complexity, quality, estimating size and project effort etc. This paper focus on a set of object oriented metrics with their optimum value that can be used to measure the quality of an object oriented design. Object oriented design focus basically to the class and design characteristics. In this paper object oriented design metrics like coupling, cohesion, polymorphism, design size etc. are used to measure quality metrics like Reusability, Flexibility, Understandability, Extendibility, and Effectiveness. This paper also proposes the relationship between object oriented metrics and quality metrics.

Keywords - Object oriented metrics (OOM), Optimum value of object oriented metrics, Quality metrics (QM), Relationship between Object oriented metrics and Quality metrics.

1. INTRODUCTION

The demand of quality software is increasing day by day. Today each software company concern more and more about the quality of software, because quality is the main feature of any long term product. Embedded software needs high level of result accuracy whereas business software requires high user productivity and usability. Both software requirements can be fulfill by good quality product only. Quality of software includes reliability, security, testability, portability, integrity, efficiency, survivability etc. some of them are totally based on the functioning of codes, and some of them depends upon the interaction of their functions. Functions are directly related to the classes. So measuring the quality metrics of classes is connected with function interaction of that class. Software metrics is a collective term used to describe the very wide range of activities concerned with measurement in software engineering. Software metrics is divided into three sub metrics, first process metrics assess the effectiveness and quality of software process, determine maturity of the process, effort required in the process, effectiveness of defect removal during development, and so on. Second product metrics is the measurement of work product produced during different phases of software development. And last project metrics illustrate the project characteristics and their execution. Software quality metrics are a subset of software metrics that focus on the quality aspects of the product, process, and project. In general, software quality metrics are more closely associated with process and product metrics than with project metrics. This paper mainly focuses to the product metrics. Product metrics assess the internal product attributes in order to know the efficiency of the analysis, design, and code model, potency of test cases, overall quality of the software under development.

2. LITERATURE SURVEY

In this paper to evaluate the quality of the object oriented software, it is needed to assess and analyze its design and implementation using appropriate metrics and evaluation techniques [4]. Literature survey in this paper has been categorize into two sub area namely object oriented metrics with desirable value and relationship between object oriented metrics and quality metrics.

2.1 Object Oriented Metrics with desirable value

Messaging: A count of number of public methods that is available as services to other classes. This is a measure of services that a class provides [1].

Design Size: Total number of classes used in a design [1].

Coupling: class coupling is a measure of how many classes a single class uses [5]. High coupling indicates a design that is difficult to reuse and maintain because of its many interdependencies on other types. Class coupling has been shown to be an accurate predictor of software failure and recent studies have shown that CBO>14 is too high [6] an upper-limit value of 9 is the most efficient [7].

Depth of Inheritance Tree (DIT): The DIT for a particular class calculates the length of the inheritance chain from the root to the deepest level of this class. If DIT increases, it means that more methods are to be expected to be inherited, which makes it more difficult to calculate a class's behavior[8]. On the other hand, a large DIT value indicates that many methods might be reused [9]. A recommended DIT is 5 or less.

Weighted Methods per Class (WMC): WMC = number of methods defined in class. Keep WMC down. A high WMC has been found to lead to more faults. A study of 30 C++ projects suggests that an increase in the average WMC

increases the density of bugs and decreases quality. High value of WMC indicates the class is more complex than that of low values. So class with less WMC is better. The study suggests "optimal" use for WMC but doesn't tell what the optimum range is [10].

Cohesion: Cohesion refers to how closely the operations in a class are related to each other. A lower value means higher cohesion between class data and methods. High cohesion indicates good class subdivision. Lack of cohesion or low cohesion increases complexity [11].

Encapsulation: Information hiding gives rise to encapsulation in object-oriented languages. The following two encapsulation measures are contained in the MOOD metrics suite [6][7][20].

Attribute Hiding Factor (AHF)

The Attribute Hiding Factor measures the invisibilities of attributes in classes. The invisibility of an attribute is the percentage of the total classes from which the attribute is not visible. It is desirable for the AHF to have a large value**.**

Method Hiding Factor (MHF)

The Method Hiding Factor measures the invisibilities of methods in classes. The invisibility of a method is the percentage of the total classes from which the method is not visible. MHF should have a large value.

Inheritance: A measure of "is –a" relationship between classes [1]. Inheritance occurs in all levels of a class hierarchy. The two metrics used to measure the amount of inheritance are the depth and breadth of the inheritance hierarchy [12].

Composition: It measures the "part-of", "has", "consist-of" or "part-whole" relationship, which are aggregation relationships in an object –oriented design [1].

Abstraction: Abstraction is a measure of the generalization-specialization aspect of the design [1]. Classes in a design which have one or more descendants exhibit this property of abstraction. Greater the number of descendants, the greater the likelihood of improper abstraction of the parent class. If a class has a large number of children, it may be a case of misuse of sub classing [5][13][14].

CCComplexity: Complexity is a measure of the degree of difficulty in understanding and comprehending the internal and external structure of classes and their relationships [1]. Cyclomatic complexity, defined by Thomas McCabe, it is easy to understand and calculate, and it gives useful results. This metric considers the control logic in a procedure. It is a measure of structural complexity.Cyclomatic complexity defined by

CC = Number of decisions + 1.

Here Decisions are caused by conditional statements. Low complexity is desirable [15].

Polymorphism: Polymorphism is the ability of an object to take on many forms. The most common use of polymorphism in OOP occurs when a parent class reference is used to refer to a child class object. any Java object that can pass more than one IS-A test is considered to be polymorphic. Polymorphism arises from inheritance. Binding (usually at run time) a common message call to one of several classes (in the same hierarchy) is supposed to reduce complexity and to allow refinement of the class hierarchy without side effects. On the other hand, to debug such a hierarchy, by tracing the control flow, this same polymorphism would make the job harder. Therefore, polymorphism ought to be bounded within a certain range [1][16].

Table 1: Desirable value of metrics

Metrics	Desirable Value
Number of classes	High
Coupling	Low
Depth of Inheritance	Low
Weighted Methods Per Class	Low
Lack of Cohesion	Low
Encapsulation (AHF +MHF)	High
Complexity	Low

2.2 Relationship between Object Oriented Metrics and Quality Metrics

The relationship between object oriented metric (OOM) and quality metrics (QM) can be driven by the following formula.

1.Reusability: The ability to reuse relies in an essential way on the ability to build larger things from smaller parts, and being able to identify commonalities among those parts. Formula for reusability is given below [1][5].

*Reusability formula = (-0.25*coupling) + (0.25*cohesion) + (0.5*messaging) + (0.5*design size)*

2.Flexibility: The ease with which a system or component can be modified for use in applications or environments other than those for which it was specifically designed [1][17].

*Flexibility formula = (0.25*encapsulation) - (0.25*coupling) + (0.5*composition) + (0.5*polymorphism)*

3.Understandability: The capability of the component to enable the user (system developer) to understand whether the component is suitable, and how it can be used for particular tasks and conditions of use [1][18].

*Understandability formula = (-0.33*abstraction) + (0.33*encapsulation) - (0.33*coupling) + (0.33*cohesion) - (0.33*polymorphism) - (0.33*complexity) - (0.33*design size)*

4.Extendibility: It is a systemic measure of the ability to extend a system and the level of effort required to implement the extension [1][19].

*Extendibility formula = (0.5*Abstraction) - (0.5*coupling) + (0.5*inheritance) + (0.5*polymorphism)*

5.Effectiveness: The degrees to which objectives are achieved and the extent to which targeted problems are solved [1][5].

*Effectiveness formula = (0.2*abstraction) + (0.2*encapsulation) + (0.2*composition) + (0.2*inheritance) + (0.2*polymorphism)*

3. PROPOSED APPROACH

In earlier work quality measurement of any Java project is find out the current OOM values. And using those OOM values it is derived the QM. In this paper the proposed approach is more focus to the optimum value of OOM. It will find out the current status of project and can be compared by optimum OOM value and identified if things

need to be improved which makes calculation of QM more efficient. So this paper express the idea about to measure OOM of Java classes which will be compared with optimal OOM values and after this it will calculate the QM using OOM.

Goal: Measuring quality metrics of Java classes using object oriented metrics.

Hypothesis: Twelve object oriented metrics are calculated for each classes present in java project. And based on that value quality will be calculated. Our approach consists of three steps which cover whole system.

1. Accepting Java project input file, and extracting all the keywords (variables, methods etc).
2. Using those keywords calculating the object oriented metrics.
3. Using object oriented metrics calculating the quality metric for classes.

In this paper we proposed a novel approach to find better quality metrics using compared object oriented metrics value for any Java class. Our approach implemented in future and shows the expected outcomes in near future.

4. CONCLUSION

This paper presents an idea how to get better quality metrics using optimal object oriented metrics value. It is helpful for the developers to check which OOM is optimal for quality of class.it helps to assist quality metrics before the deployment phase which improve the performance of product.

References

[1] Mythili Thirugnanam and Swathi.J.N. “Quality Metrics Tool for Object Oriented Programming” International Journal of Computer Theory and Engineering, Vol. 2, No. 5, October, 2010 1793-8201.

[2] B. Kitchenham and S. Pfleeger, "Software quality: the elusive target", Software, IEEE, vol. 13, no. 1, pp. 12–21, 1996.

[3] C. Neelamegam ,Dr. M. Punithavalli “A Survey – Object Oriented Quality Metrics” Global Journal of Computer Science and Technology.

[4] Sherif M. Yacoub, Hany H. Ammar, and T. Robinson, “Dynamic Metrics for Object Oriented Designs”. Dept. of Comput. Sci. & Electr. Eng., Morgantown, WV In proceeding of: 6th IEEE International software metrics Symposium (METRICS 1999), 4-6 November 1999, Boca Raton, FL, US

[5] Chidamber, S. R. & Kemerer, C. F. (1994). A Metrics Suite for Object Oriented Design (IEEE Transactions on Software Engineering, Vol. 20, No. 6). Retrieved May 14, 2011, from the University of Pittsburgh web site: Available: http://www.pitt.edu/~ckemerer/CK%20research%20papers/MetricForOOD_ChidamberKemerer94.pdf

[6] Houari A. Sahraoui, Robert Godin, Thierry Miceli: “Can Metrics Help Bridging the Gap Between the Improvement of OO Design Quality and Its Automation?” Available: http://www.iro.umontreal.ca/~sahraouh/papers/ICSM00.pdf

[7] Shatnawi, R. (2010). A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems (IEEE Transactions on Software Engineering, Vol. 36, No. 2).

[8] Daniela Glasberg, Khaled El Emam, Walcelio Melo, Nazim Madhavji: Validating Object-Oriented Design Metrics on a CommercialJava Application. 2000. Available: http://iit-iti.nrc-cnrc.gc.ca/iit-publications-iti/docs/NRC-44146.pdf

[9] Muktamyee Sarker, Jurgen Borstler “An overview of Object Oriented Design Metrics”.Department of Computer Science, Umeå University, Sweden, June 23, 2005

[10] Misra & Bhavsar: “Relationships Between Selected Software Measures and Latent Bug-Density”: Guidelines for Improving Quality. Springer-Verlag 2003.

[11] An introduction of OOM, Available: http://agile.csc.ncsu.edu/SEMaterials/OOMetrics.htm

[12] Abreu, F. B. e., "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.

[13] Daniel Rodriguez Rachel Harrison “An Overview of Object-Oriented Design Metrics” RUCS/2001/TR/A March 2001.

[14] Seyyed Mohsen Jamali “ Object Oriented Metrics(A Survey Approach)”. Department of Computer Engineering Sharif University of Technology January 2006 Tehran Iran

[15] Cyclomatic Complexity Available: http://www.aivosto.com/project/help/pm complexity.html

[16] Prof. Jubair J. Al-Ja’afer, Khair Eddin M. Sabir “Metrics for object oriented design (MOOD) to assess Javaprograms”. Available: http://www.cas.mcmaster.ca/~sabrike/wp-content/uploads/2008/02/acit2004.pdf

[17] IEEE.Standard Glossary of Software Engineering Terminology 610.12-1990, Vol. 1. LosAlamitos: IEEE Press, 1999.

[18] Manuel F. Bertoa and Antonio Vallecillo “Usability metrics for software components”. Dpto. Lenguajes y Ciencias de la Computación. Universidad de Málaga.

[19] Vibhash Yadav, Raghuraj Singh “Validating Extendibility of the Object-Oriented Software using Fuzzy Computing Techniques” International Journal of Computer Applications (0975 –8887) Volume 65– No.25, March 2013.

[20] "McCall's Software Quality Checklist arch 2013" CSE3308/DMS/2005/25 Available: http://www.csse.monash.edu.au/courseware/cse3308/cse3308_2005/assets/McCall_Checklist.pdf

AUTHOR

Lalima Soni completed B.E. and M.E. degrees in Computer Engineering in 2009 and 2014 respectively. She also qualified GATE 2012 with 99 Percentile. She is having experience as Software Engineer in DBS Bank , Singapore. Her research interest includes Software Engineering, Machine Learning, Big Data Technology.

Indrajeet Singh Chouhan completed B.E. degree in Computer Engineering in 2009. He is currently working as Team Lead in Wipro Technology Ltd. Singapore. His research interest includes Software Engineering, Software Quality, Data Mining.