Comparative Study of the Software Metrics for the complexity and Maintainability of Software Development

Dr Sonal Chawla
Associate Professor
Department of Computer Applications
Panjab University Chandigarh

Gagandeep Kaur
Assistant Professor
Department of Information Technology
GGDSD College Chandigarh

***Abstract*---Software metrics is one of the well-known topics of research in software engineering. Metrics are used to improve the quality and validity of software systems. Research in this area focus mainly on static metrics obtained by static analysis of the software. However 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. This paper describes the different types of metrics along with the static code metrics and Object oriented metrics. Then the metrics are summarized on the basis of relevance in finding the complexity and hence help in better maintainability of the software code, retaining the quality and making it cost effective.**

Keywords—Static metrics; OO metrics; MOOD

I. Introduction

Software Metrics are used to increase the quality of software since decades. For the better software development, measurement plays a very critical role for software engineering to make it a true engineering discipline. Hardware as well as software became complex day by day, so manageability is a major concern. Past were the days when only traditional metrics were used to improve the quality and technical decisions regarding softwares.

Modern systems are impossible without OO design as object-oriented programming plays a very critical role for effective and efficient software development. Software engineers developed many ways to maintain software quality and developed softwares using object-oriented programming to solve the common problems. Object-oriented design contains all the properties and quality of software that is related to any large or small project [1].

It is a degree through which a system object can hold a particular attribute or characteristics. Object-oriented is a classifying approach that is capable to classify the problem in terms of object and it may provide many paybacks on reliability, adaptability, reusability and decomposition of problem into easily understood objects and providing some future modifications [2].

II. Objective

The software quality engineering metrics are used for quality planning, process improvement, quality control, reliability estimation and analysis of customer satisfaction data. They are used to increase the efficiency of software development life cycle. For example if the number of defects are less, the effectiveness of the Development and the Testing team is improving. To make the modern application software reliable and maintainable large numbers of metrics are used these days. This paper is an attempt to understand the impact of static and OO metrics values on the complexity and maintainability of the code. In the first section, static metrics are discussed preceded by OO metrics because characteristics of object oriented design like abstraction, inheritance, modularity and polymorphism cannot be represented using traditional metrics as they play an important role in modern software applications.

Only object oriented metrics allow the modifications to reduce the cost effectiveness, time consumption and improve the quality. Additionally there is an attempt to discard the obscure metrics and use the simple ones because easy and simple ones are appreciated in software applications and also they are easy and simple to collect.

Also size measures and complexity alone cannot provide accuracy in maintaining the applications and they alone are inappropriate for predicting the defects, so other important OO metrics are used to for reducing the complexity and easier maintainability of modern applications. Moreover modern applications are incomplete without OO design.

III. Static Code Metrics

Static metrics are derived from the measurement on static analysis of the software code. It is performed without executing any of the code. Static analysis is better to understand the security problems within the program code and can easily identify nearly 85% of the flaws in the programming code.

A. Source Lines Of Code (Sloc)

Source lines of code (SLOC) is a software metric that calculate the size of a computer program by counting the number of lines in source code of program.Main types of SLOC measures are: physical SLOC (LOC) and logical SLOC (LLOC). Physical SLOC is the total count up of lines in the program's source code together with comment lines. Logical SLOC measures the number of executable statements.

B. *Comment Percentage (Cp)*

The CP is defined as a ratio of the number of comment lines to the number of non-blank LOC [3]. Software development life cycle is normally long. In any stage of the life cycle, comments will help developers and maintainers to better understand the programs. Higher comment percentages will increase understandability and maintainability [4]. It is suggested to maintain at least 8% on comment percentage to enhance the understandability [5].

C. *Halstead Metrics*

Halstead Metrics are used to measure the complexity of a program by using operands and operators. Halstead metrics is used to interpret the source code as a sequence of tokens that can be operands and operators and counted as

- number of unique (distinct) operators (n1)
- number of unique (distinct) operands (n2)
- total number of operators (N1)
- total number of operands (N2).

The number of unique operators and operands (n1 and n2) as well as the total number of operators and operands (N1 and N2) are calculated by collecting the frequencies of each operator and operand token of the source program.Though Halstead Metrics are traditional metrics but they are used to measure the modern programs like C, C++ and Java.These metrics are used to calculate the errors,programs size,volume and testing time.

D. *Mccabe's Cyclomatic Complexity*

Cyclomatic complexity is a software metric that is used to measure the complexity of a program and was measured by McCabe in 1976. It directly measures the number of free paths through the source code of program. Cyclomatic complexity is calculated using the formula.

Cyclomatic Complexity=E-N+P

Where E is the number of edges of the graph; N is the number of nodes of the graph; P is the number of connected components. These metrics are used for control quality of software products.

IV. Object Oriented Metrics

Dynamic metrics are derived from the measurement on dynamic analysis of the software code. They are based on studying the code behavior during execution. Earlier major work was focused on static metrics but now more attention has given to Dynamic metrics as they study the code at run time. Object-oriented programs can use Halstead Metrics but some essential factors like inheritance coupling remain uncovered using these metrics.

The CK metrics suite is designed for measuring object-oriented programs [6]. The suite includes six metrics discussed as follows.

A. *Chidamber And Kemerer (Ck) Metrics Suite*

Chidamber and Kemerer (CK) are the most well known object-oriented suite of measurements for Object-Oriented software. They have defined six metrics for the OO design.

a) Weighted Method Per Class (Wmc)

It is defined as the sum of the complexities of individual class. A class with more member functions than its peers is considered to be more complex and therefore more error prone [7]. As the children will inherit all the methods defined in a class, the potential impact on children will be as greater according to the number of methods in a class.

b) Depth Of Inheritance Tree (Dit)

The depth of a class in object oriented programming can be found with the inheritance. Hierarchy is the maximum extent from the node to the root of the tree. The higher the level of inheritance is greater is the value of DIT.

c) Number Of Children (Noc)

Number of immediate subclasses of a class is called its NOC. Greater number of children of a class means more reusability as inheritance is the form of reusability.

d) Coupling Between Object Class (Cbo)

It is defined as the count of the classes to which this class is coupled. Coupling is defined as: Two classes are coupled when methods declared in one class use methods or instance variables of the other class. The more independent a class is, the easier it is to reuse it in another application. The larger the number of couples, the higher the sensitivity to changes in other parts of the design, and therefore maintenance is more difficult. The higher the inter-object class coupling, the more rigorous the testing needs to be.

e) Response Of A Class (Rfc)

It is defined as number of methods in the set of all methods that can be invoked in response to a message sent to an object of a class. Greater the number of methods to be invoked, greater is the complexity of the class.

f) Lack Of Cohesion In Methods (Lcom)

It is defined as the number of different methods within a class that reference a given instance variable. To promote encapsulation, cohesiveness of methods within a class is desirable. To decrease the possibility of errors during development process, high cohesion decreases complexity.

B. *Mood (Metrics For Object Oriented Design)*

Metrics for Object Oriented Design (MOOD) are used to measure object-oriented programs. These metrics are language independent and can be obtained in the early phases of software development life cycle.

a) Method Hiding Factor (Mhf)

MHF is defined as the ratio of the sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration. The invisibility of a method is the percentage of the total classes from which this method is not visible.

b) Attribute Hiding Factor (Ahf)

AHF is defined as the ratio of the sum of the invisibilities of all attributes defined in all classes to the total number of attributes defined in the system under consideration.

c) Method Inheritance Factor (Mif)

MIF is defined as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods (locally defined plus inherited) for all classes.

d) Attribute Inheritance Factor (Aif)

AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes (locally defined plus inherited) for all classes.

e) Polymorphism Factor (Pf)

PF is defined as the ratio of the actual number of possible different polymorphic situation for class C_i to the maximum number of possible distinct polymorphic situations for class C_i.

f) Coupling Factor (Cf)

CF is defined as the ratio of the maximum possible number of couplings in the system to the actual number of couplings not imputable to inheritance.

V. Summary Of Metrics

Software metrics are becoming the basis of the software management and crucial to the accomplishment of software development. Consequently their values help in determining the complexity and hence the maintainability of the code. The below tables summarizes the above discussed metrics for the complexity and maintainability of code.

Here, the impact of increased or higher value of the metrics on the Complexity and hence the Maintainability is analyzed (Table I - Table III). It illustrates, in general, whether a high or low value is desired for the metric for better code quality [8] [9] [13]. We have marked the high value as '1' and low value as '0' to represent in a graphical form (Fig. 1 - Fig. 3). It is shown(dark line) that the higher value of metrics increase the complexity of code, while the metrics with low value and hence lower the complexity are shown in light shade lines.

TABLE I. STATIC METRICS

Static Metrics (High Value)	Complexity	Maintainability	Desired Value of Metrics
SLOC	High	High	Low
CP	Less	Low	High
HM	High	High	Low
MCC	High	High	Low

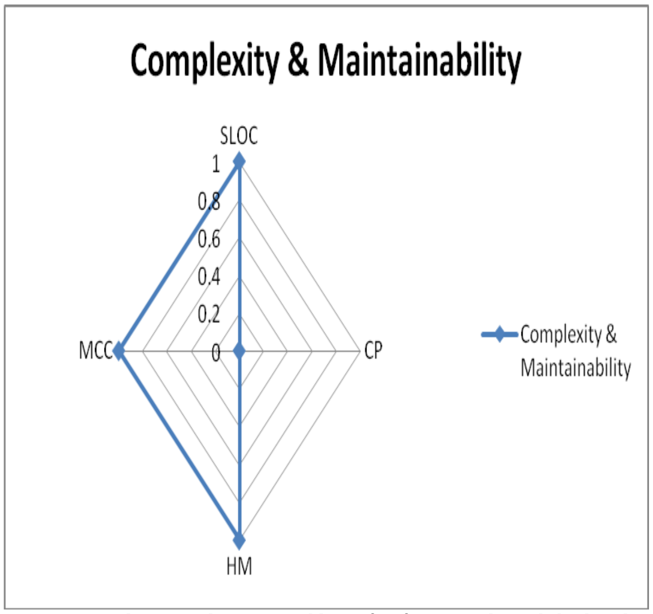

Fig. 1. Complexity and Maintainablity of software code with high value of Static metrics

TABLE II. OO METRICS

OO Metrics (High Value)	Complexity	Maintainability	Desired Value of Metrics
WMC	High	High	Low
DIT	High	High	Low
NOC	High	High	Low
CBO	High	High	Low
RFC	High	High	Low
LCOM	Less	Less	High

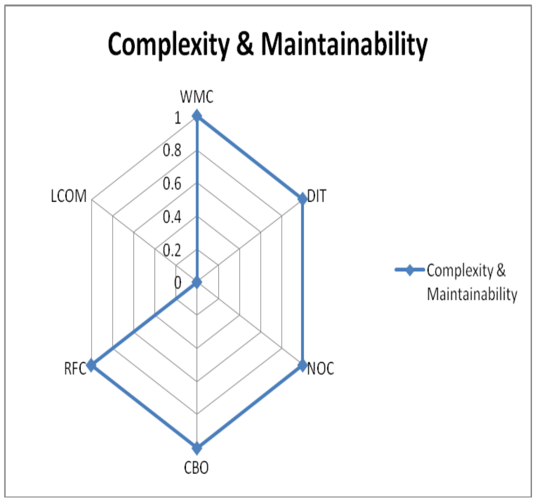

Fig. 2. Complexity and Maintainablity of software code with high value of OO metrics

TABLE III. MOOD

MOOD Metrics (High Value)	Complexity	Maintainability	Desired Value of Metrics
MHF	Less	Less	High
AHF	Less	Less	High
MIF	High	High	Low
AIF	High	High	Low
PF	High	High	Low
CF	High	High	Low

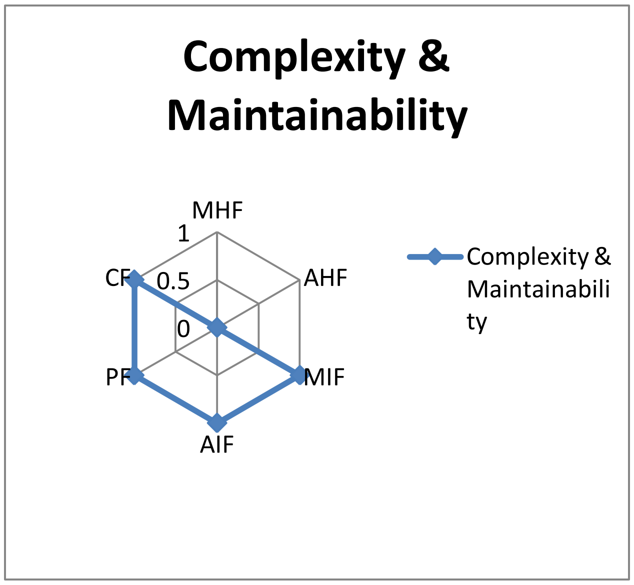

Fig. 3. Complexity and Maintainablity of software code with high value of MOOD

VI. CONCLUSION

With the advancements in the software industry, measuring the software quality is complex for the development of the software product. Therefore the need for the development of better software metrics has increased over time. Since the metrics plays a significant role in determining the complexity and thus the maintainability of the software code. Subsequently appropriate survey and study should be done to select the best metrics for the code. Each metric describes important features as, how to use it, interpretation guidelines, published thresholds whenever is possible, and assesses its appropriateness and usefulness. This would result in guiding and accessing the software to produce a robust, high-quality result, which enhances the potential reuse of the software and reduce the software maintenance cost.

REFERENCES

[1] A. Deepak, K. Pooja, T. Alpika, S. Sharma,“Software quality estimation through object oriented design metrics”, IJCSNS International journal of computer science and network security, april 2011,pp 100-104.

[2] A. Henderson, seller, “object oriented metrices:measure of complexity”,Prentice Hall,1996

[3] Lorenz, M.and Kidd, J. 1994. Object-oriented software metrics: a practical guide. Prentice-Hall, Inc.

[4] Sharble, R. C. and Cohen, S. S.1993. The Object Oriented Brewery: A Comparison of Twoobject oriented Development Methods, ACM SIGSOFT Software Engineering Notes, Vol. 18, No. 2., pp. 60 -73.

[5] McCabe Software. 2012. Metrics & Thresholds in McCabe IQ. Available at: http://www.mccabe.com/pdf/McCabe%20IQ%20Metrics.pdf

[6] Chidamber, S. R. and Kemerer, C. F. 1994. A Metrics Suite for Object Oriented Design, IEEE Transactions on Software Engineering, Vol. 20, No. 6, pp. 476–493.

[7] Watson, A. H., McCabe, T. J., and Wallace, D. R. 1996. Structured testing: A testing methodology using the cyclomatic complexity metric. National Institute of Standards and TechnologySpecial Publication 500-235.

[8] Basili, V. R., Briand, L. C., and Melo, W. L., "A Validation of Object Orient Design Metrics as Quality Indicators," IEEE Transactions on Software Engineering, vol. 21, pp. 751-761, 1996.

[9] http://www.aivosto.com/project/help/pm-oo-ck.html

[10] http://agile.csc.ncsu.edu/SEMaterials/OOMetrics.htm

[11] http://www.cc.uah.es/drg/b/RodHarRama00.English.pdf

[12] http://www.enggjournals.com/ijcse/doc/IJCSE11-03-09-003.pdf

[13] http://www.ukessays.com/essays/information-technology/polymorphism-in-object-oriented-design-information-technology-essay.php

[14] http://www.aivosto.com/project/help/pm-oo-mood.html