ISSN:0975-9646

Object Oriented Metric Tool for Measuring Quality Metrics of Classes

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

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

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

***Abstract* - In this age of object oriented language most of the softwares are developed by object oriented approach only. Object-oriented software is easier to maintain because its structure is inherently decoupled. In object oriented approach classes and objects are interlinked with each other. Using this characteristics of classes and objects many object oriented metrics like coupling, cohesion, polymorphism, inheritance etc. can be derived. For better product quality, first measure the product and then identify that improvement is required or not. In this paper object oriented metrics like encapsulation, design size, composition, abstraction etc. are used to measure quality metrics like reusability, effectiveness, extendibility, understandability etc. This paper also includes desirable value of object oriented metrics which help to identify the current state of project. And using these desirable metrics values this paper also observes the relationship between object oriented metrics and quality metrics.**

***Keywords* - Object oriented metrics, Desirable value of object oriented metrics, Quality metrics, Relationship between object oriented metrics and quality metrics.**

I. INTRODUCTION

Software quality measurement is a consecutive process. Software quality helps to improve the state of a project. For good program productivity and low maintenance better quality software is required. Software quality metrics are requisite part of any quality management system. This software quality can be achieved by better quality metrics which are interlinked with program code metrics. Quality of software includes reliability, security, testability, understandability, performance etc. And program code metrics concerned with implementation. To measure the quality it is needed to estimate and examine design and implementation of software using suitable metrics and evaluation techniques. The availability of the software metric helps manager to control the various activities of the development life cycle and contributes to the overall objective of software quality. Software metrics describes collectively very wide range of activities related with measurement of software engineering. Software metrics is split 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 [22]. 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. Quality of software comprise flexibility, correctness, maintainability, portability 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 related with function interaction of that class. This paper focuses on a set of object oriented metrics that can be used to measure the quality of object oriented design software. The metrics for object oriented design focuses on measurements that are applied to the class and design characteristics of software.

II. LITERATURE SURVEY

In this paper to evaluate the quality of the object oriented software, it is needed to analyze the product software using appropriate metrics and evaluation techniques [4]. In this paper literature survey has been categorize into two sub area namely Product metrics and Object oriented metrics.

A. Product metrics

Product metrics assess the internal attributes in order to know the efficiency of overall quality of the software under development. Various metrics formulated for products in the development process which are listed below.

1) Metrics for analysis model: These address various aspects of the analysis model such as system functionality, system size, and so on.

2) Metrics for design model: These allow software engineers to assess the quality of design and include architectural design metrics, component-level design metrics, and so on.

3) Metrics for source code: These assess source code complexity, maintainability, and other characteristics.

4) Metrics for testing: These help to design efficient and effective test cases and also evaluate the effectiveness of testing.

5) Metrics for maintenance: These assess the stability of the software product.

This paper mainly concern with metrics for source code. We choose object oriented metrics under the source code of product.

B. Object oriented metrics

Object oriented metrics further divided into five sub parts which are listed below.

1) System Metrics Level: It refers to a basic structural mechanism of the OO paradigm as Encapsulation (MHF and AHF), Inheritance (MIF and AIF), Polymorphism (PF) and Message-passing (COF).

2) Coupling Level: Coupling is the use of methods or attributes defined in a class that are used by another class. Classes interact with other classes to form a subsystem/system and this interaction can indicate the complexity of the design. Representative metrics is Coupling between Object (CBO).

3) Inheritance Level: It allows method to be inherited from super classes. Representative metrics of this set are Depth of Inheritance (DIT) and Number of Children (NOC).

4) Class Level: These metrics identify characteristics within the class, highlighting different aspects of the class abstractions and help identify where remedial action may be taken. Representative metrics of this set are Response for Class (RFC), Lack of Cohesion in method (LOCM), and Weighted Method per Class (WMC).

5) Method Level: In OO systems, traditional metrics such as Lines of Code (LOC) and Cyclomatic Complexity are usually applied to the methods level.

Using the above object oriented metrics concept we cover selective twelve metrics with their desirable value which are used for quality metrics calculation.

III. PROPOSED SYSTEM

Object oriented metric tool find out the current object oriented metric value of any Java project. These resultant values help to find out the current status of project and identified if things need to be improved. Desirable values of object oriented metrics also help to analyze the result. Resultant object oriented metrics values make calculation of quality metrics more efficient.

Goal: To develop an application tool which help to measure quality metrics of Java classes using object oriented metrics.

Hypothesis: In this approach selective twelve object oriented metrics are calculated for each class presents in java project. Based on those value quality will be calculated. This system consist three steps.

1) Accepting Java project input file, and extracting all the classes and keywords (variables, methods etc) present in Java project.

2) Using those keywords calculating the object oriented metrics for each class.

3) Using object oriented metrics calculating the Quality metric for each class.

Flow diagram of proposed system is shown in figure 1. This section further explains twelve selective object oriented metrics with their desirable value. Desirable values help to define tight bound or loose bound between objects and classes. Using this concept interconnection between objects and classes for quality metrics is shown by formulas.

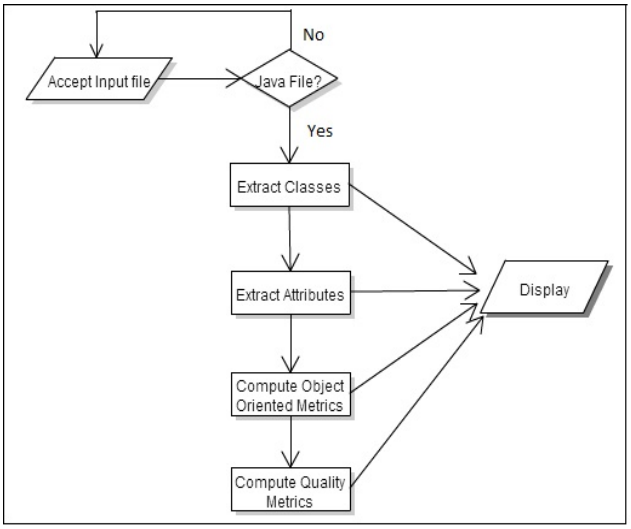

Fig. 1 Flow diagram for proposed system

A. Object Oriented Metrics with Desirable Value

1) 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].

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

3) 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].

4) 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.

5) 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].

6) 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].

7) 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.
8) 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].
9) Composition: It measures the "part-of", "has", "consist-of" or "part-whole" relationship, which are aggregation relationships in an object –oriented design [1].
10) 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].
11) 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].
12) 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].
Using the above definitions of object oriented metrics we conclude the desirable value of metrics which is shown in Table I.

B. Formulas for Quality Metrics
The relationship between object oriented metrics and quality metrics 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].

*2.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)*

TABLE I 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

IV. RESULT

Figure.2 shows the result of metrics calculation by object oriented tool. Here quality metrics is calculated for each class presents in Java project. Using the concept of McCall's Quality factor [20] we also shown relationship between Object oriented metrics and Quality metrics in figure.3. Here relationship is shown by **0(low)** and **1(high)** values. For high reusability of our classes it is needed to increase the value of cohesion, messaging and design size, but to decrease the value of coupling. Similarly for high flexibility it is needed to increase encapsulation, composition and polymorphism but to decrease the value of coupling. For high understanding it is needed to increase the value of cohesion and encapsulation but to decrease the value of coupling, design size, polymorphism,

abstraction and complexity. For high extendibility it is needed to increase polymorphism, abstraction and inheritance but to decrease the value of coupling, and for high effectiveness it is needed to increase the value of encapsulation, composition and polymorphism, abstraction and inheritance.

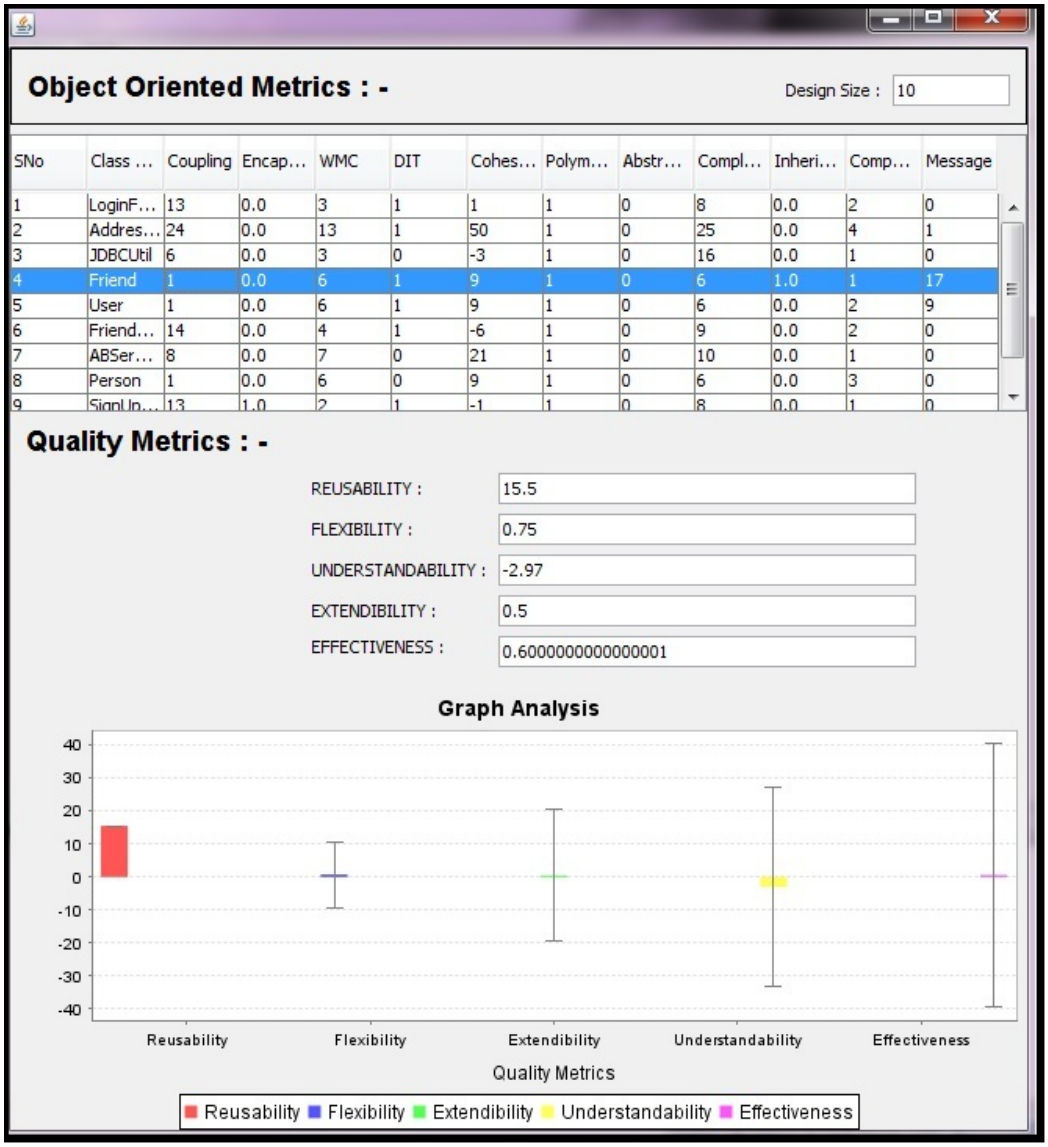

Fig. 2 Screenshot of metrics calculation generated by object oriented metric tool

	Object oriented metrics									
		Coupling	Cohesion	Messaging	Design Size	Encapsulation	Composition	Polymorphism	Abstraction	Complexity	Inheritance
Quality Metrics	Resuability	0	1	1	1						
	Flexibility	0				1	1	1			
	Understanability	0	1		0	1		0	0	0	
	Extendibility	0						1	1		1
	Effectiveness					1	1	1	1		1

Fig. 3 Relationship between object oriented metrics and quality metrics generated by result observation

V. CONCLUSION AND FUTURE WORK

The developed system is a comprehensive tool which extracts the properties of Java project, and using those properties it calculates object oriented metrics and quality metrics. This tool works like a testing tool, because implementers are able to check quality of product before the deployments which help to improve the overall product. The cost spent for maintenance will also be reduced. This tool can be enhanced to calculate quality metrics for other object oriented language. This tool can also be connected with database for storing measured records.

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”.

[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: 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? 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. 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”.

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

[11] An introduction of OOM 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”.

[15] Cyclomatic Complexity 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 Java programs”.

[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”.

[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.Available:http://www.csse.monash.edu.au/courseware/cse3308/cse3308_2005/assets/McCall_Checklist.pdf

[21] Codeproject(2011) Available: http://www.codeproject.com/Articles/179645/What-is-software-quality

[22] Sheikh Umar Farooq, S. M. K. Quadri, Nesar Ahmad,”SOFTWARE MEASUREMENTS AND METRICS: ROLE IN EFFECTIVE SOFTWARE TESTING” International Journal of Engineering Science and Technology (IJEST)