ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

Estimation of Software Quality using Object Oriented Design Metrics

Ritu Chauhan[1], Rahul Singh [2], Ashish Saraswat[3], Anwar Husain Joya[4], Vinit Kumar Gunjan[5]

Assistant Professor, Sobhasaria Group of Institution, Sikar, Rajasthan, India[1, 2, 3, 4]

BioAxis DNA Research Centre, Hyderabad, Andhra Pradesh 500068, India[5]

ABSTRACT: In software development industry the steps towards corrective actions for successful software development process comes too late resulting in ineffectiveness, late delivery, over budget and poor quality with reduced capabilities. An early estimation towards software post-release quality can be a useful remedy to maximize the business result by shortening the time and increasing the probability of project success. The development team is also a beneficiary of the software quality estimation technique as they get an early warning regarding the quality of their product. Software quality estimation has been proved to be one of the most upcoming as well as interesting research topics of the decade which aims to identify and minimize the error prone tasks to minimize the development cost. Traditional software metrics aims at the procedure-oriented development because of which it cannot fullfill the requirement of object-oriented software resulting in popularity of object-oriented design metrics in industrial software development environment as it helps in the development of higher quality products with low cost over their maintenance. Object-oriented metrics is capable of providing all the parameters to estimate the complexity and quality related issues at the early development stage of a software. In this paper we have studied and analyzed the object – oriented metrics namely MOOD Metrics, CK Metrics, and QMOOD Metrics and present the case study of how they are useful in determining the software quality developed implementing object-oriented paradigm.

Keywords: Software Quality, Object Oriented Software Development, Software Metric, MOOD Metrics, CK Metrics, QMOOD Metrics

INTRODUCTION

Quantitative measure to explain at what degree an attribute of testing or product quality or process has performed is known as software metrics. A lot of measures are used in order to provide the measurable information in the process of software development.

With the recent establishment of new regulatory bodies and eGovernment organizations, the growth of software developers and quality assurance professionals has almost doubled in the past 2-3 years. To ensure the sound and more predictable development of high quality systems, it is important for developers to gather and evaluate measurable data that guide estimation, decision-making and assessment. It is common sense that the ability to measure and analyze will lead to improved control and management. Product metrics are also referred to as software metrics. They are directly associated with the product itself and attempt to measure product quality or characteristics of the product that can be connected with product quality. Process metrics concentrate on the process of software development and measure process structures with the aim of either distinguishing problems or pushing forward effective practices. Resource metrics are essential for the development of software systems and their realization.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

Figure 1. Roles of measurement

Measurement is the process by which numbers and symbols are determined to characteristics of objects in the real world—this allows us to identify such objects according to defined rules. In software development, measurements are performed by using metrics, which are experimental designations of a value to an object aiming to characterize a definite quality of this object. Software metrics are used to measure both the process and the definitive product characteristics connected with software development. The importance of software measurement and metrics has increased over the past 30 to 40 years; nevertheless, the main attention in forming a supportable software quality measurement program is focused on following some sort of cyclical trend.

10 Steps for a Successful Metrics

1. Identify clear and measurable goals
2. Define the granularity of measurements and drill down the key variables.
3. Ask questions and select metrics.
4. Decide on periodicity of metrics.
5. Establish a measurement method.
6. Define the reporting mechanisms.
7. Generate hypotheses around key variables /factors.
8. Collect both quantitative and qualitative data.
9. Analyze metrics and take action items.
10. Track action items and ensure the improvement based on the metrics result.

A. Uses of Software Metrics

Direct Issues	Indirect Issues
Progress /Completion status	Size
Defect Analysis	Complexity
Product Quality /Stability	Cost
Structuring the Schedule	Quality

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

L.H. Rosenberg proposed nine metrics of object oriented suite. These metrics include six object-oriented metrics and three traditional metrics. A metric should have a one to one relationship with structures that is being measured or analyzed by that metric.

The proposed metrics are structure based, uses traditional metrics and is prescribed for traditional systems. In the below table it can be seen that the first three metrics are the examples of traditional metrics and applied onto the method level. Remaining six metrics are defined specifically for object oriented systems [19].

Source	Metric	OO Construct
Traditional	Cyclomatic Complexity (CC)	Method
	Lines of Code (LOC)	
	Comment Percentage (CP)	
New Object Oriented	Weighted Method Per Class (WMC)	Class/Method
	Response for Class (RFC)	Class/Message
	Lack of Cohesion of Methods (LCOM)	Class/Cohesion
	Coupling between Objects (CBO)	Coupling
	Depth of Inheritance Tree (DIT)	Inheritance
	Number of Children (NOC)	

LITERATURE SURVEY

Different metrics suits for development of software systems have been defined by various researchers as per the available literature. A small metrics suite for object-oriented designs was developed by Chidamber who defined six metrics as defined in the table below [5].

OO Construct	Metric	Output
Inheritance	Depth Inheritance Tree (DIT)	In the inheritance tree , find the depth of tree
	Number of children (NOC)	In the class, find number of decedents of the class
Coupling	Message Passing Coupling (MPC)	In a defined class, number of send statements

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

	Data Abstraction Coupling (DAC)	In a defined class, find number of abstract data type
Class	Response for a class (RFC)	To an object of the class, Set all methods that can be invoked in a response to a message
	Weighted Method Per class (WMC)	In a methods of a class, find total sum of Complexities

The metrics set defined by MOOD, includes encapsulation (MHF and AHF), inheritance (MIF and AIF), polymorphism (PF), message passing (CF) in reference of object oriented paradigm [6]. MOOD metrics can be summarized as,

Method Inheritance Factor (MIF): It is a ratio of the sum of the inherited methods in all classes to the total number of available methods. MIF has a strong capability to measure the complexity
related to message passing dependencies among various methods of different classes.

Method Hiding Factor (MHF): It is used to measure the information hiding attribute and can be represented as a ratio of the sum of the invisibilities of all methods defined in all classes
to the total number of methods defined in the system.

Attribute Hiding Factor (AHF): AHF can be defined as a ratio of the sum of the invisibilities of all the attributes defined in all classes to the total number of attributes defined in the system. It
is also helpful to determine the information hiding complexity in any object oriented system.

Coupling Factor (CF): It denotes the ratio of the maximum possible number of couplings in the system to the actual number
of couplings not imputable to inheritance.

Polymorphism Factor (PF): PF is a ratio of the actual number of possible different polymorphic situation for a class to the maximum number of possible distinct polymorphic situations for the same class. This factor is helpful to measure the level of polymorphism exhibit by a particular class.

Attribute Inheritance Factor (AIF): AIF can be represented as the ratio of the sum of inherited attributes in all classes of the system to the total number of available attributes for all classes. This explores the possibilities of attribute accessibility of different attributed from different classes.

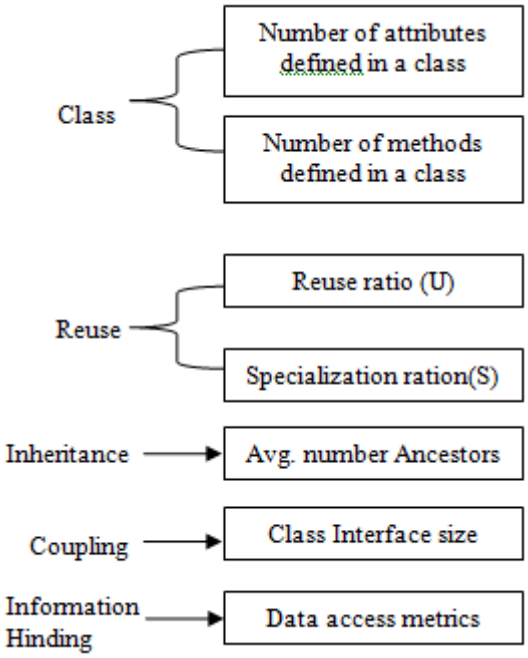
Lorenz, Kidd, Bansiya & Briand proposed some of the important metrics suit by applying object oriented metrics to the concept of Coupling, Inheritance and Classes, and given different approach for defining object-oriented metrics with their structures[8, 9, 10]. Two kinds of metrics were proposed by Hudly and Hoskins which are helpful in evaluation of the key object oriented features like: Polymorphism, Encapsulation, Data abstraction, Classes and inheritance [7]. First is based on the measurement of the class design program configuration and second based on classes. The quality model for object oriented design metrics (QMOOD) was defined by David and Bansiya based on which Total quality index of a given system can be calculated. The QMOOD class metrics are analyzed in the figure below.

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014



Subramanyam and Krishnan using CK metrics concluded that design metrics play a key role to know the design aspects and quality of software [12]. On the other side Rachel Harrison emphasized about the six properties of MOOD metrics and measured the object-oriented features like coupling, Inheritance, Encapsulation, and Polymorphism and proved that overall assessment of the system is possible using metrics [13]. Liu, K. Zhou and S. Yang by measuring the excellence of object oriented designs during development and re-development process of the software bridged the gap between the design and quality measurement of the metrics, they given the perception that quality of software plays a key role in terms of financial as well as safety aspects [11]. Visual modeling framework was defined by Booch which was capable of performing real world modeling of software and non-software products [17, 18]. Taxonomy related to the coupling and cohesion in an object oriented system was given by Eder. They also gave their approaches in terms of maintainability, extensibility and reusability to further improve their parameters.

QUALITY ASSESSMENT

In this paper we have analyzed the metrics namely CK, MOOD and QMOOD metrics. For the case study JAVA RMI classes and subclasses have been utilized to determine the impact of different metrics attributes. SD metrics tool (Quality measurement tool for UML design) is used to measure object oriented metrics. The JAVA RMI classes have been used for evaluation and the results are displayed in the below table where the value of JAVA RMI classes and subclasses metrics has been shown.

Metrics	Average
MOOD METRICS	
MHF	0.89
AHF	0.95
MIF	1.8
AIF	0.6
PF	0.1
CK METRICS	

ISSN(Online): 2320-9801
ISSN (Print): 2320-9798

International Journal of Innovative Research in Computer and Communication Engineering

(An ISO 3297: 2007 Certified Organization)

Vol. 2, Issue 1, January 2014

DIT	3
NOC	06
MPC	0
QMOOD METRICS	
NOA	9
NOM	15
ANA	3

RESULTS AND DISCUSSION

MIF/AIF are the measure of inheritance which shows relations of generalization and specialization, Increased MIF/AIF will create low understability and testability of the system. In MOOD Metrics, MHF is having value 0.89 meaning little functionality i.e., Interface is provided by classes rather than functionality. Designing of attributes or data hiding is shown by AHF 0.95 which means that using class methods data can be accessed. In the current work MIF value is 1.8 which shows system is less specialized as methods are inherited and functionalities are reused. MIF value 1.8 and AIF value 0.6 shows that reuse of functionality is higher than reuse of information or data. A PF value 0.1 indicates that system uses less polymorphism with this value and it is verified that RMI classes provide reuse of code but it doesn't support to multiple functionalities for an operation call. DIT metric value indicates maximum path from root to leaf and in our case the value is 3 which indicate average 3 levels of inheritance hence optimum reuse of code and clear understandability of system (RMI classes). NOC 16 indicates large amount of responsibility associated with a class (average 16 children per class). MPC, message passing coupling 0 indicates there is no dependency among the classes in RMI. NOA, number of attributes per class 9 and NOM (number of method per class) 15 indicates complex class design. The value 3 of ANA indicates an acceptable design complexity in JAVA RMI classes.

REFERENCES

[1] L.C.Briand, J.Wuest, J.Daly and Porter V., "Exploring the Relationships Between Design Measures and Software Quality In Object Oriented Systems", Journal of Systems and Software, 51, 2000.
[2] L.C. Briand, W.L. Melo and J.Wust, " Assessing the Applicability of Fault Proneness Models Across Object Oriented Software Projects", IEEE transactions on Software Engineering. Vol. 28, No. 7, 2002.
[3] P.Coad and E.Yourdon, "Object Oriented Analysis", Yourdon Press, 1990.
[4] L.H. Rosenberg and L.Hyatt,"Applying and interpreting object oriented metrics", Proceedings of software technology conference, utah, April 1998.
[5] S.R. Chidamber, C.F.Kemerer, "A metrics suite for Object Oriented Design,"IEEE Transactions on Software Engineering,Vol. 20, No. 6, June 1994, pp. 476-493.
[6] F.B.Abreu, "The MOOD Metrics Set", Proc.ECOOP'95 Workshop on Metrics, 1995.
[7] A.V. Hudli and R.V. Hoskins: "Software metrics for OOD", IEEE International conference, 2002.
[8] L.C. Briand, W.L. Melo and J.Wuest, "A Unified Framework for Coupling Measurement in Object Oriented Systems", IEEE Transactions on Software Engineering, 25(1), 1999.
[9] M.Lorenz and J.Kidd,"Object Oriented Software Metrics",Prentice- Hall, 1994.
[10] J.Bansiya and C.G Davis, " A Hierarchical Model for Object Oriented Design Quality Assessment", IEEE Transactions on Software Engineering, 2002.
[11] H.Lilu, K.Zhou and S.Yang: "Quality metrics of OOD for Software development and Re-development", First Asia-Pacific Conference on Quality Software, August 2002.
[12] M.Subramanyam and R.Krishnan: "Emphirical Analysis of CK metrics for OOD complexity: Implication for software defect", IEEE transaction on software engineering, 2003.
[13] R.Harrison, Steve J.Counsell and R.V.Nithi: "An evaluation of the MOOD set of OOSM", IEEE Transaction on Software Engineering, vol.24 no.6, pp.491-496, June 1998.
[14] JürgenWüst, " SD METRICS TOOL", in der Lache 17, 67308 Zellertal-Harxheim, Germany, www.sdmetrics.com, version 2.11, 2009.
[15] V.Basili, L.Briand and W.Melo, "A Validation of Object Oriented Design Metrics as Quality Indicators" , IEEE Transactions on Software Engineering, vol.22, pp.751-761, 1996.
[16] J.Eder, G.Kappel and M.Schreft, " Coupling and Cohesion in Object Oriented Systems", Technical Report University of Klagenfurt, 1994.
[17] Grady Booch, "Object Oriented Development," IEEE Trans. Software Eng., vol 2 no. 2, Feb.1986.
[18] Booch, G., Rumbaugh, J., Jacobson, I., 1999, The Unified Modeling Language User Guide, Addison Wesley, Reading, MA 1999.
[19] B.Henderson-sellers, "Object-Oriented Metrics: Measures of Complexity" Prentice Hall, 1996.