Software Quality Estimation through Object Oriented Design Metrics

Deepak Arora†, Pooja Khanna† and Alpika Tripathi†, Shipra Sharma† and Sanchika Shukla††

†Faculty of Engineering, Department of Computer Science, Amity University, Lucknow, India
†† Department of Computer Science, Amity University, Lucknow, India

Summary
Software metrics are required to measure quality in terms of software performance and reliability related characteristics like dependencies, coupling and cohesion etc. It provides a way to measure the progress of code during development and having direct relationship with cost and time incurred in the software design and development at their later stages. These major issues must be checked and informed early in the development stage, so that reliability of any software product could be ensured for any large and complex software project. Object oriented software metrics directly focuses on the issues like complexity, reliability and robustness of the software developed using object oriented design methodologies. It reflects the time, cost and effort that would be incurred in development at later stage. While the software in its development stage, it is desirable that the complexity levels at every stage should be minimized to make the end product more reliable and manageable. Object oriented metrics provides all parameters through which one can estimate the complexities and quality related issues of any software at their early stages of development. In this paper, authors have studied three object oriented metrics namely MOOD Metrics, CK Metrics, and QMOOD Metrics and given a case study to show, how these metrics are useful in determining the quality of any software designed by using object oriented paradigm.

Key words:
Software Quality, JAVA RMI, MOOD Metrics, CK Metrics, QMOOD Metrics

1. Introduction

Software Metrics can be defined by measuring property or characteristic or quality of a software objects related to any large and complex software project. In a broader term, it is a degree up to which a system object can hold a particular attribute or characteristics. Object oriented approach is capable of classifying the problem in terms of objects and provide many paybacks like reliability, reusability, decomposition of problem into easily understood object and aiding of future modifications [19].

Object-Oriented Metrics are useless if they are not mapped to software quality parameters. Many number of quality models are proposed to map parameters of the Object Oriented software like Extensibility, Reusability, efforts, manageability and cost [1, 2, 3]. To know more about the internal structure of the product one should know more about the interdependencies of parameters of metrics and Software quality parameters. Figure 1 shows the interdependencies of the metrics parameters and software quality parameters by measuring Object Oriented Metrics [15].

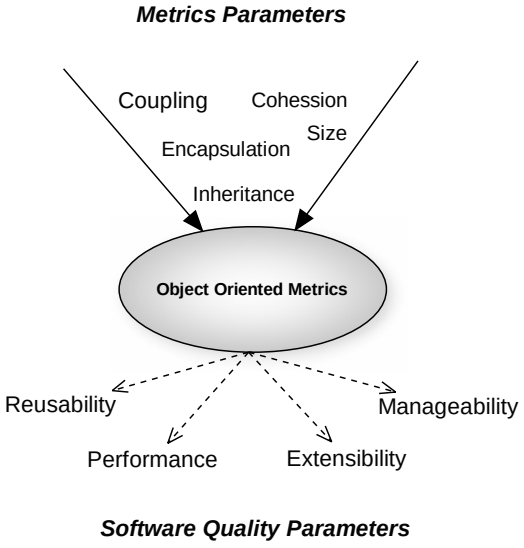

Fig. 1 Relationship between metrics and quality parameters

L.H. Rosenberg proposed various attributes related to object oriented metrics. They have proposed nine metrics for object oriented suite, which are depicted in table I. These metrics include three traditional metrics and six object-oriented metrics [4]. A metric should have a one to one relationship with structures that is being measured or analyzed by that metric.

Metrics proposed by Rosenberg, uses traditional metrics and it is structure based, prescribed for object oriented systems. Here one can see that first three metrics are the examples of traditional metrics and applied onto the

Manuscript received April 5, 2011
Manuscript revised April 20, 2011

method level. Remaining six metrics are defined specifically for object oriented systems.

Table I: Metrics proposed by Rosenberg [4] for Object Oriented Systems

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)	

2. Background

In the available literature, lots of researchers have defined different metrics suits for object-oriented software systems. Chidamber has developed a small metrics suite for object-oriented designs. They defined six metrics, which are depicted in table II [5].

Table II: CK Metrics Suite

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
	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 basic structural related metrics attributes like 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 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.
- **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.
- **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.
- **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.
- **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.

Hudly and Hoskins [7] proposed two kinds of metrics: first is based on classes and second one is to measure the class design configuration of the program. These metrics are helpful to evaluate the main features of object oriented like Polymorphism, Encapsulations, Data abstraction, Inheritance and classes. Metrics proposed by Briand et al., Lorenz and Kidd and Bansiya are some of the important metric suites. They applied object oriented metrics to the concepts of classes, coupling, and inheritance. They also have given different approaches to define the object oriented metrics and their structures [8, 9, 10]. Bansiya and Davis defined Quality Model for Object Oriented Design (QMOOD) metrics. Based on this total quality index (TQI) can be computed for a given system. The QMOOD class metrics are analyzed in Fig 2 [10].

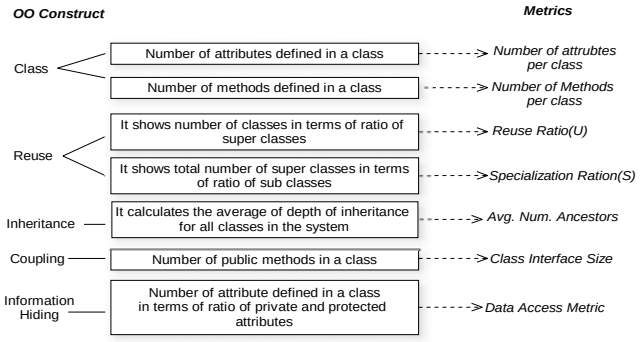

Fig 2 QMOOD Metrics

Liu, K.Zhou and S.Yang [11] have given perception that quality of software also plays an important role in terms of safety aspects and financial aspects. They bridged the gap between quality measurement and design of these metrics, with the help of measuring the excellence of Object Oriented Designs during development and re-development process of the software. On the other side Subramanyam and Krishnan [12] used CK Metrics suits and concluded that for the developers, designs metrics are very important to know the design aspects of the software and to enhance the quality of software. Rachel Harrison [13] discussed about the six properties of MOOD Metrics and measured the object-oriented features like Inheritance, coupling, encapsulation, and polymorphism. In the result they showed that the metrics could be used to provide an overall assessment of the system. Eder et al. introduces taxonomy related the coupling and cohesion in any object oriented system. They also have given their approaches to further improve these parameters in terms of maintainability, extendibility and reusability [16].

Booch has defined visual modeling framework to perform real world modeling of any software and non software systems [17, 18].

3. Quality Assessment through OO Metrics

In this paper, authors have analyzed three important metrics namely CK, MOOD and QMOOD metrics. As a case study JAVA RMI classes and subclasses has been chosen to determine the impact of different metrics attributes. In the analysis authors have used these classes to measure object-oriented metrics by using SDMetrics Tool ver. 2.11 demo [14]. This is a quality measurement tool for UML designs. The JAVA RMI classes have been used for evaluation and output has been shown in table III. It represents the value of JAVA RMI classes and subclasses metrics. In this analysis authors have analyzed three type of metrics suite i.e. CK, MOOD and QMOOD.

Table III: Metrics value for Object Oriented Metrics

Metrics	Average
MOOD METRICS	
MHF	0.89
AHF	0.95
MIF	1.8
AIF	0.6
PF	0.1
CK METRICS	
DIT	3
NOC	16
MPC	0
QMOOD METRICS	
NOA	9
NOM	15
ANA	3

4. Result and Discussion

In MOOD Metrics, first attribute is MHF which is having value 0.89 means little functionality i.e. RMI classes provide interface rather than functionality. AHF 0.95 shows proper designing of attributes or data hiding i.e. data can be accesses by the corresponding class methods. MIF/AIF are measure of inheritance this shows generalization and specialization relations. Increase of MIF/AIF will create low understandability and testability of the system. In our case MIF is 1.8, means the system is less specialized as methods are inherited or functionalities are reuse. AIF value 0.6 and MIF value 1.8 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.

5. Concluding Remarks

The authors have applied set of metrics defined by metric suit given by CK, MOOD and QMOOD. As a test data JAVA RMI classes have been chosen. Metrics are important to judge the complexities and reliability issues of any object oriented system. In this paper, authors have found that the design of JAVA RMI classes has passed various quality parameters and exhibit good design characteristics. Authors have chosen limited set of metrics, which are more important in reference of JAVA RMI classes. It shows good adoption of new changes and provides higher degree of expandability with a heavy grade of efficient message passing communication capabilities.

Available metrics confined to limited boundary, besides that there should be more emphasis on different domain related to quality parameters and some new metrics are still required to measure the hidden complexities aspects for a large and complex object oriented system. These will certainly helpful in reducing the cost and effort incurred in the design of any object oriented system and one can determine the level of its reliability and robustness, before its implementation begins.

Acknowledgment

The authors are very thankful to their respected Mr. Aseem Chauhan, Additional President, Amity University, Lucknow, Maj. Gen. K.K. Ohri, AVSM (Retd.), Director General, Amity University, Lucknow, India, for providing excellent computation facilities in the University campus. Authors also pay their regards to Prof. S.T.H. Abidi, Director and Brig. U.K. Chopra, Deputy Director, Amity School of Engineering, Amity University, Lucknow for giving their moral support and help to carry out this research work.

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.

Deepak Arora received his Ph.D. in Computer Science from Babasaheb Bhimrao Ambedkar University (A Central University), Lucknow, India in 2009. Currently he is working as an Assistant Professor in Department of Computer Science & Engineering, Amity University, Lucknow, India. His research interests include Distributed & Parallel Systems, Object Oriented Software Engineering and Data Mining. He has produced several outstanding publications on Distributed Computing Systems. He is a member of International Association of Engineers, Hong Kong and Computer Society of India, Mumbai, India.

Pooja Khanna received M.Tech. in Infromation Technology with Specialization in Software Engineering from IIIT, Allahabad India in 2008. She is currently working as Lecturer in the Department of Computer science and Engineering, Amity University, Lucknow. She has several international publications to her credit. Her research interests include Software reuse, Software performance, Software testing and cloud computing

Alpika Tripathi received M.Tech. in Computer Science from Amity University Lucknow in 2010. She is currently working as a Lecturer in Department of Computer Science Engineering, Amity University, Lucknow. Her research interests include Software Engineering and Data Mining.

Shipra Sharma M.Tech. in Computer Science Engineering from Amity University Lucknow in 2010. She is a Lecturer in Department of Computer Science Engineering, Amity University, Lucknow. Her research interests include Software Engineering and Artificial Intelligence.

Sanchika Shukla completed her B.Tech degree in IT from Chaudhary Charan Singh University 2009. She is pursuing M.Tech. in Computer Science from Amity University, Lucknow. Her research interest includes software testing and database management system.