EVALUATION OF METRICS AND ASSESSMENT OF QUALITY OF OBJECT ORIENTED SOFTWARE

Neeraj Chauhan[1], Mohan Vishal Gupta[2]

[1]*CCSIT College, TMU, Moradabad*
[2]*CCSIT College, TMU, Moradabad*

Abstract

Quality assessment of software is big issue for software development team The reason is variations of designed software in size and methodology. A huge number of metrics has designed to assess quality of software up to a level. In this paper we are discussing Object oriented metrics used to assess quality of software at design level as well as at code level. Although correct assessment of software quality is not possible but using Object oriented metrics quality can be assessed up to limit. The main focus of research is to assess software quality at design level because design level quality assessment effects coding, testing, maintaining phase of software development life cycle.

First, for evaluating metrics of design level UML diagram is used as an input. A Java Parser is designed for parsing the XML code of UML diagram .Second, Quality of same software projects also assessed at code level using same formula as at the design level. At code level Eclipse with Metrics 1.3.6 is used for assessing quality. We observed that software quality at code level moves around CC (Cyclometer Complexity), LCOM (Lack Cohesion of Methods) and LOCM (Lines of Code of method).And we find out that for increasing quality of software, CC and LCOM and LOCM are low. For decreasing quality CC, LCOM and LOCL are high.

Keywords— *Object oriented technique, object oriented, software, Object Oriented Design (OOD), Software metrics.*

----------***----------

1. INTRODUCTION

Assessment of software quality is a big issue for software engineers. Because software engineers are not very sounded in the basic quantitative laws of Physics. And question is that which factors should be taken at the different level of quality assessment of software? The solution of this problem is searched in the form of Metrics [11]. Metrics are a means for attaining more accurate estimations of project milestones, and developing a software system that contains minimal faults [7]. A huge number of metrics has been designed to assess the quality of software at different level of software development life cycle. In these metrics some metrics do not work for Object Oriented designed software. So a good set of metrics is required for assessing quality of Object Oriented software. Object Oriented metrics are used to measure properties of object oriented designs. A very popular CK metrics model and MOOD metrics model is used for assessing quality at design level [A.b CK]. The challenge for a quality engineer is to select those metrics which meet the specific needs of each software project. A quality engineer has to face the problem of selecting the appropriate set of metrics for assessing quality of his/her software. Quality assessment at design level is needed for software development. For software quality assessment we used various steps:

- Analysis of existing metrics for object oriented software.
- Selection of appropriate metrics for quality assessment at design level and code level.
- Development of a Parser for extracting metrics values from UML diagram.
- Assessment of object oriented software quality at design level.
- Verification of design level quality using Eclipse with metrics 1.3.4.

Various quality models have been developed for software quality assessment like McCall quality model(1977), Barry W. Boehm's quality model(1978), R. Geoff Dromey's quality model that is very similar to McCall quality model and Boehm's quality model and ISO 9126 model.

2. USED METHODOLOGY FOR SOFTWARE QUALITY ASSESSMENT

Quality assessment of object oriented software at design level, various tasks are identified:

- Selection of UML diagrams as well as code of object oriented software.
- Selection of metrics for software quality assessment.
- Extraction of metrics value from UML diagram.
- Evaluation of software quality attributes using an appropriate formula.
- Assessment of software quality at design level.
- Verification of software quality at product level using Eclipse with metrics1.3.4.

In this paper we are presenting software quality assessment at design level as well as code level. At design level we are extracting information from UML diagram of software. For code level quality assessment we are using software code as an input. Finally the result of both design quality and code quality are compared one by one for each selected software. For quality assessment at design and code level selected software are divided into three categories. These categories are Good, Bad and Medium. By comparing quality at design level as well as at the code level we concluded that there is positive co-relation between design quality and product quality.

3. USED METRICS FOR SOFTWARE QUALITY ASSESSMENT

Used metrics for quality assessment are discussed below:

3.1 Weighted Method Per Class (WMC)

WMC metric is used to measure the complexity of a class. Complexity of a class can be calculated by calculating the sum of complexity of the methods of a class. WMC is a predictor of how much time and effort is required to develop and to maintain the class.High value of WMC indicates the class is more complex than that of low values."Class with less WMC is better". If all methods complexities are considered unit then WMC=n, number of methods [9].

3.2 Depth of Inheritance Tree (DIT)[CK Metrics Suite]

DIT metric is used to calculate the length of the maximum path from the node to the root of the tree. This metric calculates how far down a class is declared in the inheritance hierarchy. "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 behaviour". Thus it can be hard to understand a system with many inheritance layers. On the other hand, a large DIT value indicates that many methods might be reused.

3.3 Number of Children (NOC)

NOC, number of immediate sub-classes subordinated to a class in class hierarchy [9].This metric is used to calculate number of sub-classes that are going to inherit the methods of the parent class. The size of NOC approximately indicates the level of reuse in an application. If NOC grows it means reuse increases that satisfies the condition of reusability but on the other hand, by increment in NOC, testing *and maintenance cost* will also increase because more children in a class indicate more responsibility [2].So, NOC represents the effort required to test the class and reuse. Small values of NOC may be an indicator of lack of communication between different class designers.

3.4 Coupling between Objects (CBO)

CBO for a class is count of number of other classes to which this class is coupled [3]. High coupling between classes is not good because it will prevent reuse. If a class in more independent means provides the more reusability.

3.5 Response for a Class (RFC)

RFC is used to calculate the number of methods that can be invoked in response to a message in a class. If a large numbers of methods are invoked from a class means RFC is high then testing and maintenance of the Class becomes more complex because test sequence grows. On the other hand lower values indicate greater polymorphism [3].

3.6 Lack of Cohesion in Methods (LCOM)

LCOM metric measures the dissimilarity of methods in a class via instanced variables. LCOM also measures the amount of cohesiveness present, how well a system has been designed and how complex a class is [9]. Greater values of LCOM increases complexity that does not promotes encapsulation and implies classes should probably be split into two or more subclasses.
If LCOM is high, methods may be coupled to one another via attributes and then class design will be complex.

4. EVALUATION OF SOFTWARE QUALITY ATTRIBUTES USING AN APPROPRIATE FORMULA

4.1 Selected Software Projects for Quality Assessment

For analyzing quality of object oriented software at design level, we used CK Metrics suite and selected six software projects. Focusing on coupling and cohesion in software projects. Object oriented software having high coupling and low cohesion are low quality projects in comparison of software projects having low coupling and high cohesion.

Table: 1 Selected Projects for Quality Assessment

S.No	Software Project	Quality
1	JMoney Software	High Quality
2	JUtility	High Quality
3	JFractal Applet	Medium Quality
4	ATM System	Medium Quality
5	OnlineAirTicket System	Low Quality
6	Student Project	Low Quality

4.2 Used Object Oriented concepts for Quality Assessment

Used Object oriented concepts for Quality assessment are discussed in the table given below

Table 2:. Evaluated Object oriented concepts at design level

S.NO	Object Oriented Concepts	Evaluation of Object Oriented Concepts
1	Encapsulation	Total number of members in the class.
2	Polymorphism	Actual method overrides divided by the maximum number of possible method overrides.
3	Inheritance	Total number of public and protected attributes and methods divided by total methods and attributes declared in that class.
4	Messaging	Total number of public methods and attributes in the class.
5	Design Size	Total number of classes used in software design.
6	Complexity	The number of methods in class .
7	Cohesion	R(M)=Parameters used in method /Total no of parameters in class. Cohesion of each class= Sum(R(M)) / Total no of methods in class. Software Cohesion=Sum of cohesion of classes / Total no of classes.
8	Abstraction	Ratio of the number of methods inherited by a class to the total number of methods accessible by member methods of the class.
9	Coupling	Coupling metric is evaluated by identifying type of coupling between object, data coupling, stamp coupling, common coupling, content coupling
10	Hierarchy	Total number of trees in the software that are not involved in inheritance.

4.3 Evaluation of Quality Attributes at Design level

Positive quality attributes Reusability, Flexibility, Understandability, Extensibility, Effectiveness and Functionality are selected for software quality assessment. The relationship among object oriented concepts and design quality attributes is shown in the table 3.1 and formulas used for calculating quality attributes are given below:

Reusability = -0.25*Coupling+0.25*Cohesion+0.5*Messaging+0.5*Design Size

Flexibility = 0.25*Encapsulation - 0.25*Coupling + 0.5*Composition + 0.5* Polymorphism

Understandability=-0.33*Abstraction+0.33*Encapsulation-0.33*Coupling+0.33*Cohesion -0.33*Polymorphism-0.33*Complexity-0.33*Design Size

Functionality = 0.12*Cohesion + 0.22*Polymorphism + 0.22*Messaging + 0.22*Design Size +0.22*Hierarchies

Extendibility = 0.5*Abstraction - 0.5*Coupling + 0.5*Inheritance +0.5*Polymorphism

Effectiveness = 0.2*Abstraction + 0.2*Encapsulation + 0.2*Composition+ 0.2*Inheritance + 0.2*Polymorphism

Table 3: Evaluated Object Oriented Concepts at Design Level

Project	JMoney	JUtility	JFractal	ATM	Library	Hospital
Polymorphism	2.4	2.75	1.35	1.21	0.699	0.798
Messaging	155	142	132	140	112	98
Composition	2.28	3.121	2.01	3.209	1.031	1.231
Inheritance	1.737	1.401	1.565	1.206	1.06	1.32
Coupling	2.55	2.05	4.75	3.95	5.05	5.35
Hierarchies	3	2	2	3	2	3
Complexity	3.375	2.151	4.075	4.125	6.075	5.657
Design Size	20	24	20	18	22	20
Cohesion	0.698	0.659	0.431	0.503	0.412	0.312
Abstraction	4.01	3.45	3.56	4.402	2.66	2.36
Encapsulation	243	285	225	215	220	133

Table 4: Assessed Quality Attributes and Project Quality at Design Level

Project	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness	Quality of Project
JMoney	87.0370	68.2524	56.2517	42.3237	8.5985	53.0054	46.3320
JUtility	86.6522	73.1731	56.9251	48.664	6.275	58.9441	44.0642
JFractal	74.9203	56.2429	62.5993	34.228	2.8625	46.2927	39.6996
ATM	73.138	54.9721	60.9321	33.5466	1.6601	45.0961	38.972
Library	58.840	54.5913	61.0097	27.0432	0.9844	45.6083	25.8499
Hospital	57.7405	27.926	26.120	26.8329	0.4359	23.7417	23.4394

4.4 Assessment of Quality at Code Level

As we have discussed first, quality assessment at code level is done using Eclipse with Metrics 1.3.6. Selected metrics for evaluating object oriented concepts are discussed below.

Number of Overridden Methods as a Polymorphism, Number of Methods as a Messaging, Nested Block Depth as a Composition, Depth of Inheritance Tree as a DIT, (Afferent+ Efferent) coupling as a Coupling, Number of Interfaces as a Hierarchies, McCabe Cyclomatic Complexity as a Complexity, Total Lines of Code as a Design Size, Lack of Cohesion of Methods as a Cohesion, Abstractness Distance as a Abstraction, Weighted methods per Class as a Encapsulation.

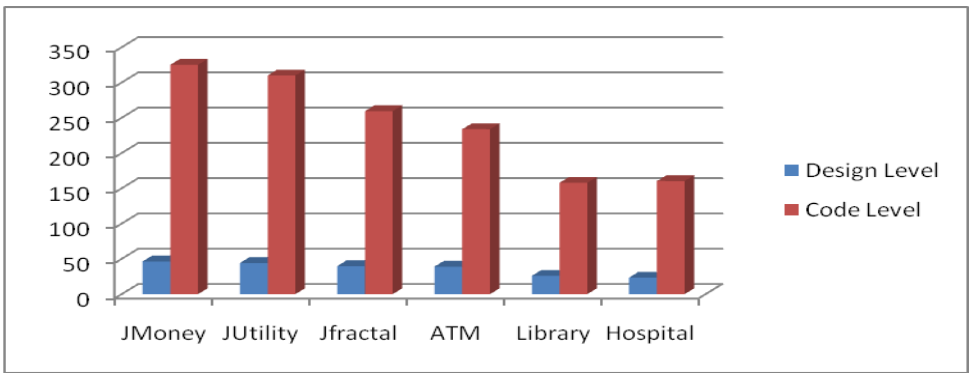
Table 5: Evaluated Object Oriented Concepts at Code Level

Project	JMoney	JUtility	JFractal	ATM	Library	Hospital
Polymorphism	48	34	28	22	18	20
Messaging	167	153	141	170	112	109
Composition	1.173	1.297	1.183	1.585	1.080	1.017
Inheritance	1.867	1.318	1.455	1.708	1.357	1.012
Coupling	4	2.667	3.76	6	4.75	5.302
Hierarchies	3	2	0	1	0	1
Complexity	2.167	2.355	4.324	3.605	5.375	5.102
Design Size	3500	3790	3000	2490	2580	1900
Cohesion	.563	0.659	.448	.432	.304	.314
Abstraction	2.5	1.5	1.5	1	0	0
Encapsulation	364	385	287	360	223	149

Table 6: Assessed Quality Attributes and Project Quality at Code Level

Project	Reusability	Flexibility	Understandability	Functionality	Extendibility	Effectiveness	Quality of Project
JMoney	1950.9152	121.2312	-1023.7140	866.6193	28.0754	84.6230	325.158
JUtility	1791.0065	114.5841	-998.4700	801.3728	24.1834	83.5079	309.988
JFractal	1569.7469	85.4261	-906.8885	697.2577	12.8970	63.5276	259.363
ATM	1328.1330	101.2925	-913.4840	590.5438	9.5831	77.4585	233.798
Libray	999.9884	37.1025	-590.8389	443.6044	5.3034	27.8879	157.850
Hospital	1003	41.4329	-594	446.63	7.85	30.205	160.2937

4.5 Quality Assessment of all Selected Six Projects at Design Level and Code Level Represented in Graph.



5. CONCLUSION

In this paper we have discussed object oriented metrics for assessing quality of object oriented software. For this we used CK metrics suite, MOOD metrics suite and QMOOD metric suite. For quality assessment we selected positive quality attributes of software like Reusability, Extendibility, Flexibility, Understandability, Functionality etc. Because all selected attributes are positive attributes, they will increase software quality.

The first work deals with quality assessment at design level of software development. For that we used UML diagram as an input and evaluated design metrics and object oriented concepts and finally assessed software quality using a specific formula of software quality attributes contribution. We also assessed quality of software at the code level and compared the two qualities. Results obtained are perfectly as per expectations.

REFERENCES

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

[2] Abreu, F. B. e. and Melo, W., "Evaluating the Impact of OO Design on Software Quality," presented at Third International Software Metrics Symposium, Berlin, 1996.

[3] Abreu F.B. and R.Carapuca."Object-Oriented Software Engineering: "Measuring and Controlling the Development Process". Proceedings of the 4th International Conference on Software Quality, McLean,Virginia USA,October ,1994

[4] Bansiya J. and C. G. Davis (2002): A Hierarchical Model for Object-Oriented Design Quality Assessment IEEE Transactions on Software Engineering, pp. 4-17, 2002

[5] 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.

[6] Bee Bee Chua and Laurel Evelyn Dyson "Applying the ISO 9126model to the evaluation of an e-learning system" 1995

[7] Boehm, B. W., "Improving Software Productivity," *IEEE Computer*, pp. 43-57, September 1987.

[8] Briand, L., Emam, K. E., and Morasca, S., "Theoretical and Empirical Validation of Software Metrics," 1995.

[9] Briand, L., Ikonomovski, S., Lounis, H., and Wust, J., "Measuring the Quality of Structured Designs," *Journal of Systems and Software*, vol. 2, pp. 113-120, 1981.

[10] Briand, L. C., Daly, J. W., and Wust, J. K., "A Unified Framework for Coupling Measurement in Object-Oriented Systems," *IEEE Transactions on Software Engineering*, vol. 25, pp. 91-121, January/February 1999.

[11] Chidamber, S. R. and Kemerer, C. F., "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, 1994.

[12] Churcher, N. I. and Shepperd, M. J., "Comments on 'A Metrics Suite for Object-Oriented Design'," *IEEE Transactions on Software Engineering*, vol. 21, pp. 263-5, 1995.

[13] Dr. Deepshikha Jamwal "Analysis of Software Quality Models for Organizations " 2010

[14] El Emam, K., "A Methodology for Validating Software Product Metrics," National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076, June 2000 June 2000.

[15] Fenton, N. E. and Pfleeger, S. L., *Software Metrics: A Rigorous and Practical Approach*: Brooks/Cole Pub Co., 1998.

[16] Fowler, M., Beck, K., Brant, J., Opdyke, W., and Roberts, d., *Refactoring: Improving the Design of Existing Code*. Reading, Massachusetts: Addison Wesley, 1999.