Validating Object Oriented Design Quality using Software Metrics

Vibhash Yadav
Computer Science & Engineering Department
Krishna Girls Engineering College,
Kanpur, India
vibhashds10@yahoo.com

Prof. Raghuraj Singh
Computer Science & Engineering Department,
Harcourt Butler Technological Institute,
Kanpur, India
rscse@rediffmail.com

***Abstract*- Software Metrics have been traditionally used as primary source for determining the Software Product Quality. In this paper, we propose an approach for determining the design quality of an Object Oriented Software using software metrics. To validate the proposed methodology, we have chosen three Open Source software projects of good, average and bad design quality which is known priori. For each of these projects, we extracted a set of chosen software metrics that play a definite role in software design quality. After applying various required normalizations on these metric values, we determined the design quality for the above stated three projects. We found that software design quality determined in this way was in confirmation with the priori known end product quality. The outcomes of the experimental study provide a strong base for the effectiveness of our proposed approach for metric based design quality measurement of object-oriented software.**

Keywords— Metrics, object oriented design,coupling, cohesion etc

I. INTRODUCTION

Measuring the design quality early during software development has been regarded as a prominent way to assure the quality of software products. Several models have been proposed to estimate the quality of software systems. They are based on prediction of fault proneness of software module [1,2,3], on detection of anti pattern [4] which is known to be bad coding practices, based on object oriented metrics [5, 6] and visualization technique [7]. Much prior work on quality measurement with several proposals of design metrics came out along with measurement data analysis for software system. However to the best of our knowledge as long as no general design standard exists. General metric threshold values are difficult to determine. Even though rules for writing code can be constructed and metrics can be used to assure that the rules are followed [8]. So there is lack of straight forward rule for selecting appropriate metrics to measure design quality of software.

In our approach, the various metric values have been evaluated using various open source tools of metric calculation like metric 1.3.4, JHawk, TeamInABox [3, 9]. Thereupon, metric are being analyzed and differentiated on the basis of their potential to indicate the design quality of the object oriented software systems. Some metrics show uniformity of results, that is to say that they are either giving high values or low values, for the design quality of the various software systems already known. But, others are random, not giving clear indication of design quality; there values do not follow any proper pattern or trend of values. Based on this demarcation, good design indicators are selected while others not paid heed, for our further analysis.

II. RELATED WORK

Various efforts have been previously made to measure the design strength and design quality of Object Oriented Software Systems. Khadim M. Breesam et al. [10] validated a set of metrics empirically that could be used to measure the quality of an object oriented design in terms of the class inheritance. Sastry et al. [11] tried to implement software metrics with aid of GUI & also analyzed relationships of metrics to determine the quality of software attributes measured with regard of object oriented software development life cycle. Shaik et al. [12] have performed statistical analysis for object oriented software metrics on CK metric suite by validating the data collected from projects of different modes. Bansia J. et. al. [13] presented a hierarchical model for assessment of design quality of Object Oriented Software Systems in quantitative terms using various lower level and higher level quality metrics / parameters. Validation of software metrics shows that metrics actually allow conclusions on the quality of software. Most studies conclude that metrics are indeed a valid indicator for quality of software, defect detection, maintainability etc. Metric data provides quick feedback for software designers and managers. Analyzing and collecting the data can predict the design quality [12]. If appropriate used, it can lead to a significant reduction in costs of the overall implementation and improvements in quality of the final product [11,12,14]. Quality metrics propose strategies on how analysis of source code with metrics can be integrated in an ongoing software development project and how metrics can be used as a practical aid in code and architecture investigations on existing systems [14, 15, 16].

III. Methodology

A. *Selection of Open Source Software*

In order to evaluate our suggested approach we performed an empirical analysis on several open source software by encouragement from some prior research works. To test our approach we selected these open source software with different design quality level. We chose these software systems on basis of their design quality obviously by considering the designer of the software and the reputation in software market.

TABLE I. CHOSEN OPEN SOURCE SOFTWARE WITH PRIOR KNOWN DESIGN QUALITY

Jdom	High design quality
Taming Java Thread	High design quality
Bonforum	Medium design quality
EviewApplet	Low design quality
Student project	Low design quality

B. *Selection of Metrics*

It is always the hardest part for design quality measurement to choose the appropriate metric suite for specific software as software systems are dissimilar in size and complexity as well as in design level [14]. As the most important measures for software quality is Cyclomatic complexity, lack of cohesion of methods, weighted method per class and lines of code are the selected metrics. The experimental Work of chalking out relationship between them through way of equations have been done.

TABLE II: CHOSEN METRICS SET THAT INFLUENCE SOFTWARE DESIGN QUALITY

Cyclomatic complexity (CC)	Low value required
Lack of cohesion of method(LCOM)	Low value required
Weighted method per class(WMC)	Low value required
Lines of codes of methods(LOCM)	Low value required

C. *Evaluation of Metric Values*

To obtain the above mentioned metric values for the chosen software, we used the tools Metric 1.3.4, Team in a Box both as plug-in for Eclipse and JHawk an open source software. Using these tools we measured the metric values for all the five chosen software. The results are summarized in this section.

1. *Metric Values Without Normalization*

Metric values obtained by applying the evaluation tools on the chosen software are detailed in Table III.

TABLE III: METRICS VALUES BEFORE NORMALIZATION

Software	**CC**	**LCOM**	**WMC**	**LOCM**
JDMO	3.13	0.241	37.17	6888
Taming	2.36	0.122	7.20	1407
Eview Applet	3.5	0.407	19.09	1364
StudentProject	3.5	0.407	19.09	999
Bonforum	5.3	0.33	47.15	3369

2. *Primary Normalization*

In our work, firstly we have normalized the metric values to a standardized form using normalization model, so that they can be compared irrespective of software sizes and complexity. At primary normalization level metric values are scaled down to standard unit of 10 kilo lines of code that provides a precise look to software systems disregarding their size i.e. line of codes. The primary normalization for each of the above mentioned software is calculated as follows:

If $M_{1,} M_{2,} M_{3,}$….. M_n are individual metric values then Primary Metric Normalization value is given by

$$N_i = (1/10k)M_i \qquad i=1,2,3\ldots n \qquad (1)$$

The results after primary normalization are summarized in Table IV.

TABLE IV: METRIC AFTER PRIMARY NORMALIZATION

Software	**CC**	**LCOM**	**WMC**
JDMO	4.54	0.35	53.89
Taming	16.77	0.86	51.12
Eview Applet	25.65	3071	265.19
Student Project	35.00	4.07	190.9
Bonforum	15.73	0.97	139.56

3. *Normalization at Comparison Level*

Comparison level normalization is performed by setting a metric threshold value among the comparing software group and by calculating the percentile metric among peers [17].
Primary Normalized Metric, N_i is set to percent for the comparing software, S_j it holds the maximum value and for the other software systems $S_{1,} S_{2,} S_{3,}$…. S_k. N_i value is calculated to percentile normalization, PN_{ik} such as:

$$PN_{ik} = (N_{ik} / PN_{max}) * N_{ij} \qquad (2)$$

If N_i is max for S_j then N_{ij} (primary normalized metric, N_i for S_j) is named as Maximum percentile normalized metric and changed as $PN_{max}=N_{ij}$ and another change done as $N_{ij=}$ 100.
The resultant metrics after comparison level normalization are summarized in Table V.

TABLE V: METRICS VALUES AFTER NORMALIZATION

Software	CC	LCOM	WMC
JDMO	12.97	8.5	20.32
Taming	47.91	21.13	19.27
Eview Applet	73.28	91.15	100
StudentProject	100	100	71.98
Bonforum	49.94	23.83	52.62

4. *Normalization At Quality Rank Level*

Finally, the design quality ranking level normalization is done with weighted normalization average calculation, defining range and providing design quality rank to each software systems to indicate their overall design quality level.

If PN_1, PN_2, PN_3, ……….. PN_n are individual percentile normalized metric values of a particular software S_j then weighted normalization average for S_j is

$$W(NA_j)= (\sum PN_i)/ n \qquad (3)$$

Range defines metric threshold value for weighted normalization average for any particular software and lies between zero and percent inclusive such as:

$$0<= w(NA) <= 100$$

Software System Normalized Quality Rank is Design Quality Level (DQL) used for the comparing software systems such that

$$DQL = \{ \begin{array}{ll} \text{High} & \text{if } 0<= w(NA) <=25 \\ \text{Medium} & \text{if } 25<= w(NA) <= 75 \\ \text{Low} & \text{if } 75<= W(NA) <= 100 \end{array} \}$$

Normalized metrics at quality rank level are summarized in Table VI.

TABLE VI: METRICS VALUES AT DESIGN QUALITY LEVEL

Software	CC	LCOM	WMC
JDMO	13.93	High	High
Taming	29.43	High	High
Eview Applet	88.14	Low	Low
StudentProject	90.66	Low	Low
Bonforum	40.46	Medium	Medium

5. *Co-relation analysis*

After applying co-relation analysis on the metrics, we concluded that the metrics CC, WMC, LCOM and LOCM are strongly co-related to each other and the relation is approximately liner. The results of co-relation analysis are summarized in Table VII.

$$R_{AB} = \frac{\sum(A-A_{mean})(B-B_{mean})}{N\, \sigma_\alpha \sigma_\beta} \qquad (4)$$

Standard deviation

$$\sigma=(\sum(A-A_{mean})^2/n)^{0.5} \qquad (5)$$

$\sigma_{cc} = 0.9653$
$\sigma_{LCOM} = 0.1325$
$\sigma_{WMC} = 14.28$

Co-relation coefficient

$$R_{A.B}=[\sum(A-A_{MEAN})(B-B_{MEAN})]/n\, \sigma_A \sigma_B \qquad (6)$$

$R_{CC, LCOM} = 0.2725$
$R_{LCOM, WMC} = 0.457$
$R_{CC, WMC} = 0.7782$
Thus, CC & LCOM are positively co-related

TABLE VII: CO-RELATION ANALYSIS FOR METRICS

Software	CC	LCOM	WMC	LOCM
JDMO	3.13	0.241	37.17	6888
Taming	2.36	0.122	7.20	1407
Eview Applet	3.5	0.507	36.18	1364
StudentProject	3.5	0.407	19.09	999
Bonforum	5.3	0.33	47.15	3369
Mean	3.558	29.35	29.35	2805.4
Std. Deviation	0.9653	0.1325	14.286	--------

IV. RESULTS

A. *Co-relation Results*

After applying co-relation analysis on the metrics, we concluded that the metrics CC, WMC, LCOM and LOCM are strongly co-related to each other and the relation is approximately liner. For example, Figure 1 depicts the relationship between LCOM and CC.

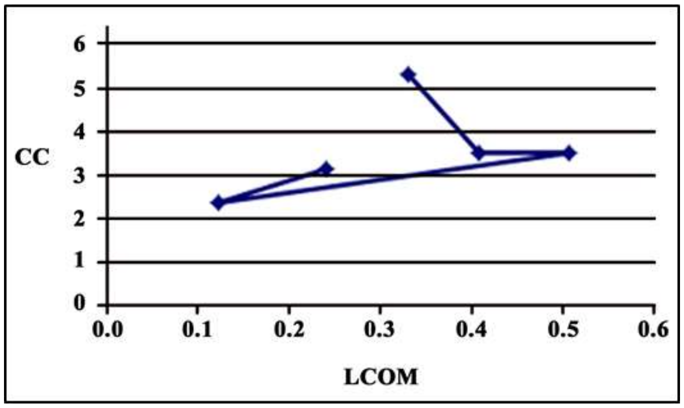

Figure 1.

B. *Normalization Result*

When we calculated the metric values without normalization for design quality measurement it could not reflect the variation of the design quality level of individual software systems much clearly. But when we introduced a slandered platform of metric measurement for all software systems we found clear indication of the variation of design quality of software systems.

C. *Regression Analysis*

The scatter diagram indicates some relationship between the two variable x and y, the dots of the scatter diagram are more or less concentrated round a curve. This curve is called the curve regression.

The straight line about which the various points may be considered as scattered is called the regression line. The relationship is linear owing to the fact that there is a strong co-relation existing between the metrics.

V. CONCLUSION

We have enunciated a straight forward approach for measuring the design quality of Object-Oriented software systems by object oriented quality metrics measurement in standardized source code. Measured metrics for a software system are scaled downed to standard unit so that measurement will have a standard platform for all software systems disregarding of their dissimilarity of size, complexity or design quality .Strong correlation between metrics is found and almost linear correlation persists in every couple of metrics. These correlations between metrics encourage us to use their percentile average values to formulate a straight forward approach to assign a design quality rank for any software systems. This work establishes that software metrics like Cyclomatic Complexity (CC), Lack of cohesion methods (LCOM), Weighted Method Per Class (WMC), Lines of codes of methods (LOCM) play a definitive role in design quality of Object Oriented Software.

REFERENCES

1 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, pp. 706-720, July 2002.

2 Shi Zhong, M. Khoshgoftaar, and Naeem Seliya, “Expert-Based Software Measurement Data Analysis with Clustering Techniques”, Accepted to IEEE Intelligent Systems, Special Issue on Data & Information Cleaning & Preprocessing, 2004.

3. Yuming Zhou and Hareton Leung, “Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults,” *IEEE Trans. Software Eng.*, vol. 32, no. 10, pp. 771-789, Oct. 2006.

4. W.J. Brown, R.C. Malveau, H.W. McCormick, III, and T.J. Mowbray, “AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis”, John Wiley Press, 1998.

5. S.R. Chidamber, C.F. Kemerer,” A metrics suit for object oriented design”, IEEE Trans. On Software eng., vol. 20, no. 6, p.p. 476-493, June 1994.

6. Santonu Sarkar, Girish Maskeri Rama, and Avinash C. Kak,“API-Based and Information-Theoretic Metrics for Measuring the Quality of Software Modularization,” *IEEE Trans. Software Eng.*, vol. 33, no. 1, pp. 14-32, Jan. 2007.

7. G. Langelier H. Sahraoui P. Poulin; Visualization based Analysis of Quality for Largescale Software Systems; ASE’05, November 7–11, 2005.

8. Raghuraj Singh, Yogesh Singh, Onakar Singh: A Hierarchical model for Design Reliability assessment of Modular Design based software, ICFAI Journal of Systems & Management, Vol. III, No. 2, PP 20-29, May 2005.

9. Fernando Brito e Abreu and Walcelio Melo: Evaluating the Impact of Object OrientedDesign on Software Quality, 3rd International S/W Metrics Symposium, March 1996, Germany.

10. Dr. Kadhim M. Breesam, “Metrics for object oriented design focusing on class inheritance metrics”, 2nd International conference on dependency of computer system IEEE, 2007.

11. Dr. B.R. sastry, M.V. Vijaya Saradhi, “Impact of software metrics on object oriented software development life cycle”, International Journal of Engineering Scienceand technology, Vol 2(2), pg 67-76, 2010.

12. Amjan shaik, Dr, CRK Reddy, Dr. A. Damodaram, “Statistical analysis for object oriented design software security metrics”, International Journal of Engineering Science & technology, vol 2(5), pg 1136-1142, 2010

13. Jagdish Bansiya and Carl G Devis: A Hierarchical Model for Object Oriented Design Quality Assessment, IEEE transactions of Software Engineering, Vol 28, No. 1, January 2002.

14. K K Agarwal, Y Singh, Arvinder Kaur and Ruchika Malhotra: Empirical study of Object Oriented Metrics, Journal of Object Technology, Vol. 5, No. 8, Nov-Dec 2006.

15. W.J. Brown, R.C. Malveau, H.W. McCormick, III, and T.J. Mowbray, “AntiPatterns: Refactoring Software, Architectures, and Projects in Crisis”, John Wiley Press, 1998.
16. Md Abul Khaer, MMA Hashem, Md Raihan Masud: On use of Design Patterns in Empirical Assessment of Software Design Quality, Proceedings of the International Conference on Computer & Communication Engineering 2008, May 13-15 Kuala Lumpur, Malaysia 2008.