International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:12, 2009

A Metric Framework for Analysis of Quality of Object Oriented Design

Amandeep Kaur, Satwinder Singh, Dr. K. S. Kahlon

***Abstract*—**The impact of OO design on software quality characteristics such as defect density and rework by mean of experimental validation. Encapsulation, inheritance, polymorphism, reusability, Data hiding and message-passing are the major attribute of an Object Oriented system. In order to evaluate the quality of an Object oriented system the above said attributes can act as indicators. The metrics are the well known quantifiable approach to express any attribute. Hence, in this paper we tried to formulate a framework of metrics representing the attributes of object oriented system. Empirical Data is collected from three different projects based on object oriented paradigms to calculate the metrics.

***Keywords*—**Object Oriented, Software metrics, Methods, Attributes, cohesion, coupling, Inheritance.

I. Introduction

RECENT years have seen the increasing use of the object oriented paradigm in software development. The use of object oriented software development techniques introduces new elements to software complexity both in software development process and in the final product. The backbone of any software system is its design. It is the skeleton where the flesh (code) will be supported. The Object-Oriented (OO) paradigm includes a set of mechanisms such as inheritance, encapsulation, and polymorphism and message-passing that is believed to allow the construction of designs where those features are enforced. However, a designer must be able to use those mechanisms in a "convenient" way.

Object-oriented analysis and design are popular concepts in today's software development environment. They are often heralded as the silver bullet for solving software problems

The concepts of software metrics are well established, and many metrics relating to product quality have been developed and used. A standardized metric set for OO does not yet exist for a metrics definition standard. Therefore, it is necessary to define metrics and to analyze them. Once a set of metrics for any type of measurement is proposed, it is necessary to systematically validate them [1]. Validating a metric means providing convincing proof that:

Amandeep Kaur is with Computer Science & Engineering Department, Govt Polytechnic College, Mohali, Distt. Mohali, Punjab, India.

Satwinder Singh is associated with Baba Banda Singh Bahadur Engineering College, Fatehgarh Sahib (Punjab)-India.

Dr. K.S.Kahlon is Professor, Deptt of Computer Science & Engineering, Guru Nanak Dev University, Amritsar, Punjab, India

- A metric measure what its purpose is to measure, i.e. the metric is well defined and consistent with the properties of the attribute that the metric claims to measure.
- The metric is associated with some important external attribute of the process or product, such as cost or maintainability.
- The metric is an improvement over existing metrics [1].

There are two types of relevant validation for purposes of this thesis, theoretical or internal validation and empirical or external validation [1].

- Theoretical validation
- Empirical validation

Theoretical validation maps to point first point in the list above, and involves clarifying the properties of the attribute to be measured, and analytically proving that the metric satisfies those properties. Such attributes are termed as internal attributes. Theoretical

Validation requires consensus among the research community regarding the properties of attributes [1] and reaching such a consensus could potentially take many years.

Empirical validation entails demonstrating points second and third above. Empirical validation requires correlating the metric to the external attribute by comparing the values of the metric with the values of the external attribute. For example, a metric that measures the number of downloads for a KB may be related to external attributes of the metrics such as awareness within the organization.

As OO technologies has some new characteristics, such as data abstraction, encapsulation, inheritance, polymorphism, information hiding and reuse, traditional software metrics do not readily lend themselves to the OO notions.

II. Motivation & Problem Formulation

The research was done by surveying the literature on object oriented metrics and then applying some object oriented metrics to meet the goal of measuring design and code quality. Many object-oriented metrics have been proposed specifically for the purpose of assessing the design of a software system. However, most of the existing approaches to measuring these design metrics involve only some of the aspects of object oriented paradigms. As a result, it is not always clear the design quality of code. We choose the metrics so that every aspect can be covered. Instead, we attempt to derive a set of indirect measures that lead to metrics that provide an indication of the quality of some representation of software.

Realizing the importance of software metrics, numbers of metrics have been defined for software [2]. These metrics try to capture different aspects of software product [1] and its process. Some of the metrics also try to capture the same aspects of software e.g., there are number of metrics to measure the coupling between different classes. Software developers need to explicitly state the relation between the different metrics measuring the same aspect of software. As an example, we might be interested to know the size of a table. There can be number of metrics related to size of a table e.g., length of the table, breadth of the table, area of the table, diagonal of the table etc. But length and breadth measures are sufficient and others measures can be derived from them if required. Similarly in software, we need to identify the necessary metrics that provide useful information, otherwise the managers will be lost into so many numbers and the purpose of metrics would be lost.

Hence, the objective of the study is to design a metric framework using structural mechanisms of the object-oriented paradigm as encapsulation, inheritance, polymorphism, reusability, Data hiding and message-passing that would be able to reflect the quality of a software system.

III. CK Metric Suit: Deficiencies & Solutions

One of the first suites of OO design measures was proposed by Chidamber and Kemerer [1] [3]. The authors of this suite of metrics claim that these measures can aid users in understanding object oriented design complexity and in predicting external software qualities such as software defects, testing, and maintenance effort. Use of the CK set of metrics and other complementary measures are gradually growing in industry acceptance. This is reflected in the increasing number of industrial software tools, such as Rational Rose®, that enable automated computation of these metrics. Even though this metric suite is widely, empirical validations of these metrics in real world software development settings are limited. Various flaws and inconsistencies have been observed in the LCOM metric as shown under.

The high value of LCOM indicates that the methods in the class are not really related to each other and vice versa. According to above definition of LCOM the high value of LCOM implies low similarity and low cohesion, but a value of LCOM = 0 doesn't implies the reverse [4].

Consider the example in figure 1 (a) the value of LCOM is 8 (as | P | =9 and | Q | = 1). Whereas in figure 1 (b) the value of LCOM is also 8 (as | P | =18 and | Q | = 10), but figure 1 (a) example is more cohesive than figure 1 (b) example. So the above said definition of CK metric for LCOM is not able to distinguish the more cohesive class from the less ones. This is simple violation of the basic axiom of measurement theory, which tells that a measure should be able to distinguish two dissimilar entities. So, this deficiency offends the purpose of metric.

.

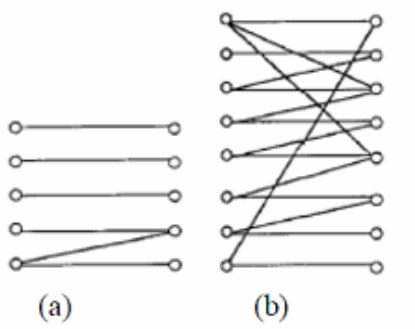

Fig. 1. Two examples of (a) Less Cohesive Class and (b) Densely Cohesive Class

In another test of validity of the LCOM metric consider the example of figure 2 as shown below:

$$I1 = \{a, b, c, d, e\}$$
$$I2 = \{a, b, e\}$$
$$I3 = \{x, y, z, d, e\}$$
$$I4 = \{x, y, z, d\}$$

Fig. 2. Example of the calculation of LCOM

Consider a class supporting the first three sets then |P| = 2, |Q| = 1 => LCOM = 1 implies less cohesion but when considering a class that supports all four sets then |P| = 3, |Q| = 3 => LCOM = 0 implies high cohesion. But this is just the reverse that we are expecting when we analyze the above sets as I1 and I2 are a pair of cohesive methods as are I3 and I4 and the good design recommends the formation of two classes, not one [5].

For solving the above problems it is recommended the alternative version which considers the number of data members of the classes, number of classes and the number of data members, denoted by $LCOM_{new}$.

It the modified form of existing CK metric for LCOM, as LCOM metric is not considering the relative importance of the elements of set P and set Q used in calculating the LCOM of a class. First component of this version is representing the normalized weightage of the "{ (Ii ,Ij) | Ii ∩ Ij }" constituents of the methods considered. The second component represents the normalized weightage of the "{ (Ii ,Ij) | Ii ∩ Ij ≠ Ø }" constituents of the methods in the class. As the | Ii ∩ Ij | is the number of data variables shared by the two methods ,on the other hand, | Ii U Ij | is the union of variables of both the methods. The basis of $LCOM_{new}$ is same as that of the LCOM metric.

$$LCOM_{new} = \frac{1}{ma}\left\{ \sum_{\text{for each element } (Ii, Ij) \text{ of } Pset} \frac{\left|I_i \cup I_j\right|}{a} - \sum_{\text{for each eleement } (Ii, Ij) \text{ of } Qset} \frac{\left|I_i \cap I_j\right|}{\left|I_i \cup I_j\right|} \right\} \quad (1)$$

IV. Proposed Metric Framework

The increasing importance of software measurement has led to development of new software measures. Many metrics have been proposed related to various constructs like class, coupling, cohesion, inheritance, information hiding and polymorphism [6] [7]. It is often difficult to determine which metric is more useful in which area. As a consequence, it is very difficult for project managers and practitioners to select measures for object-oriented systems. In the study following 22 metrics proposed by various researchers are investigated (for detail see [8]):

- Number of Attributes per Class (NOA)
- Number of Methods per Class (NOM)
- Weighted Methods per Class (WMC) [9] [10]
- Response For a Class (RFC) [10]
- Coupling Between Objects (CBO) [1] [10]
- Data Abstraction Coupling (DAC)
- Message passing Coupling (MPC)
- Coupling Factor (CF) [10]
- Lack of Cohesion in Methods (LCOM) [1] [10]
- Tight Class Cohesion (TCC)
- Loose Class Cohesion (LCC)
- Information flow based Cohesion (ICH)
- Depth of Inheritance Tree (DIT) [1] [10]
- Number of Children (NOC) [1] [10]
- Method Inheritance Factor (MIF) [10]
- Attribute Inheritance Factor (AIF) [10]
- Attribute Hiding Factor (AHF) [10]
- Method Hiding Factor (MHF)
- Polymorphism Factor (PF) [10]
- Number of Methods Overridden by a subclass (NMO)
- Reuse ratio (U)
- Specialization Ratio (S)

V. Results & Discussion

The selected metrics are implemented in Java environment and the value of the metrics are extracted from the three projects named as Proj1, Proj2 and Proj3. The metric values of the class level metrics is shown in Table 1 and the graphical representation of the metric values is shown in Figure 3.

As evidenced from the figure 3 the attribute represented by NOA, NOM, WMC, DAC, NOC and NMO are shown the same pattern that is able to identify the quality level of the software.

The metric values of the system level metrics is shown in Table 2 and the graphical representation of the metric values is shown in Figure 4. As evidenced from the figure 4 the attribute represented by CF has shown the high value fro the Proj1, medium value for the Proj2 and Low value for the Proj3. The opposite is there for the MIF, AIF and S metrics. The U metrics shows the nil value for all the projects. Hence, CF, MIF, AIF and S system metrics can be used to identify the quality level of the software.

VI. Conclusion

In this paper, work has been done to explore the quality of Object oriented design of software components using metric based approach. In this paper 22 metrics have been used to analyze various features of software component e.g inheritance ,coupling, cohesion, polymorphism, reusability etc. As evidenced from the data collected from the three projects the quality of the software can be identified using NOA, NOM, WMC, DAC, NOC and NMO class level metrics and CF, MIF, AIF and S system metrics. So, the framework of the metric has to target the Number of Attributes per Class, Number of Methods per Class, Complexity, Data Abstraction Coupling, Number of Children, Number of Methods Overridden by a subclass, Coupling Factor, Method Inheritance Factor, Attribute Inheritance Factor and Specialization Ratio as qulity indicators of a object oriented software.

In this study we have only used three projects so in the future extension one can use more projects for the empirical validation of the results.

TABLE I. Class Level Metrics

S.No.	Metric Object-Oriented	Project Name		
		Proj1	Proj2	Proj3
1	NOA	5	10	14
2	NOM	20	31	35
3	WMC	20	31	35
4	RFC	79	12	16
5	CBO	2	5	1
6	DAC	1	4	5
7	MPC	4	2	4
8	LCOM	2	1	1
9	TCC	7.142	5	5
10	LCC	7.142	5	5
11	ICH	0	1	2
12	lDIT	2	2	4
13	NOC	2	3	7
14	NMO	2	5	9

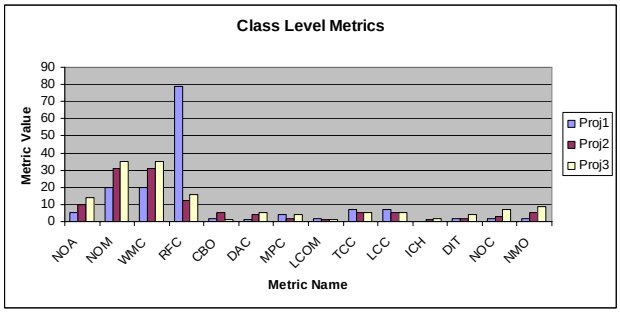

Fig. 3. Bar Chart of Class Level Metrics

TABLE II. SYSTEM LEVEL METRICS

S.No	Metrics	Project names		
		Proj1	Proj2	Proj3
1	CF	50	45	20
2	MIF	0.491	1.50	2.50
3	AIF	0.676	1	1.5
4	MHF	0.305	0.897	0.834
5	AHF	0.375	0.667	0.444
6	PF	0	0	0
7	U	0.220	0.321	0.563
8	S	3.53	4.67	5.87

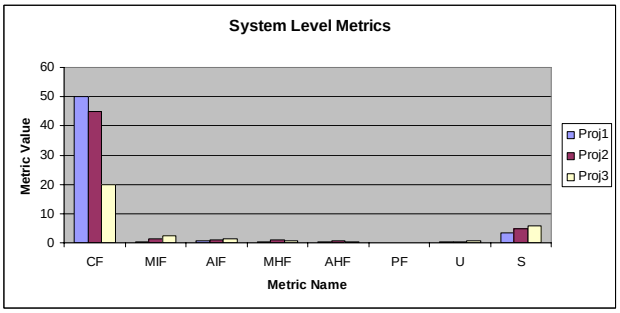

Fig. 4. Bar Chart of System Level Metrics

REFERENCES

[1] S.R. Chidamber and C.F. Kemerer, "A Metrics Suite for Object-Oriented Design," IEEE Trans. Software Eng., vol. 20, no. 6, pp. 476493, June 1994.

[2] L. Briand , W. Daly and J. Wust, Unified Framework for Cohesion Measurement in Object-Oriented Systems. Empirical Software Engineering, 3 65-117, 1998.

[3] Chidamber, S.R., and Kemerer, C. F. "Towards a Metrics Suite for Object Oriented Design," Proc. Conf. Object Oriented Programming Systems, Languages, and Applications (OOPSLA'91), vol. 26, no. 11, 1991, pp. 197-211.

[4] Mayer, T., and Hall, T. "Critical Analysis of Current OO Design Metrics", Software Quality Journal, 8, 1999, pp. 97-110,

[5] Henderson-Seller, B., and Constantine, L. L. "Coupling and Cohesion towards a valid metrics suite for object oriented analysis and Design", Object Oriented Systems, 3, 1996, pp. 143-158.

[6] L. Briand, S. Morasca, and V. Basili, Designing and Validating High-Level Design Metrics, Technical Report CS-TR-3301, Univ. of Maryland, Dept. of Computer Science, College Park, Md., 1994.

[7] L. Briand, S. Morasca, and V. Basili, "Property Based Software Engineering Measurement," IEEE Trans. Software Eng., vol. 22, no. 1, p. 68-86, Jan. 1996.

[8] Kaur Amandeep, Singh Satwinder and Kahlon K. S, "Evaluation and Metrication of Object Oriented System", Proceedings of the International MultiConference of Engineers and Computer Scientists 2009 Vol I, IMECS 2009, March 18 - 20, 2009, Hong Kong.

[9] McCabe, T. J., "A Complexity Measure", IEEE Transactions on Software Engineering, SE-2(4), pages 308-320, December 1976.

[10] Pressman, R. " A Practitioner's Approach to Software Engineering," Mc-grawhill Publications, 2001, pp. 658-662.