Volume 2, No. 7, July 2011
Journal of Global Research in Computer Science
ISSN-2229-371X
RESEARCH PAPER
Available Online at www.jgrcs.info

A METRIC FRAMEWORK FOR ANALYSIS OF OOD

Suruchi Mehra*[1] and Raman Maini[2]
*[1]Research Scholar, [2]Associate Professor, University College of Engineering, Punjabi University
Patiala, Punjab, India – 147002
srchmehra@yahoo.co.in
research_raman@yahoo.com

Abstract: Object oriented design is becoming more popular in software development environment and object oriented design metrics is an essential part of software environment. This study focus on a set of object oriented metrics that can be used to measure the quality of an object oriented design. The metrics for object oriented design focus on measurements that are applied to the class and design characteristics. These measurements permit designers to access the software early in process, making changes that will reduce complexity and improve the continuing capability of the design. Several metrics and metric-tool are presented and evaluated. An experimental study was conducted as an attempt to further validate each metric and increase knowledge about them. We present 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 already developed systems. Metrics do have a practical use and that they to some extent can reflect software systems design quality, such as: complexity of methods/classes, package structure design and the level of abstraction in a system.

Keywords: *Object Oriented , Software Metrics ,Methods, Attributes, Cohesion, Coupling, Inheritance*

INTRODUCTION

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. 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 [1][3]. The concepts of software metrics are well established, and many metrics relating to product quality have been developed and used. The metrics were selected on the basis of their ability to predict different aspects of object-oriented design (e.g. the lines of code metric predict a modules size)[9][10]. Metrics (quantitative estimates of product and project properties) can, if defined from sound engineering principles, be a precious tool for both project management and software development [15] two of the pioneers in developing metrics for measuring an object-oriented design were Shyam R. Chidamber and Chris F. Kemerer. In 1991 they proposed a metric suite of six different measurements that together could be used as a design predictor for any object-oriented language. Their original suite has been a subject of discussion for many years and the authors themselves and other researchers has continued to improve or add to the "CK" metric suite. Other language dependent metrics (in this report the Java language is the only language considered) have been developed over the past few years e.g. in ; they are products of different programming principles that describes how to write well-designed code[18][19]. One shouldn't confuse metrics with measures. A metric is a quantitative property of software products (product metrics) or processes (process metrics) whose values are numbers — either integer or real in our current framework). A measure is the value of a metric for a certain product or process [3][4]. Any metric should be relevant related to some interesting property of the processes or products being measured: cost, estimated number of bugs, ease of maintenance [6][8] A metric theory is a set of metric definitions accompanied with a set of convincing arguments to show that the metrics are relevant., Our purpose is simply to provide the basic tools that enable the development and application of good metric theories.

PROBLEM FORMULATION

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 [2]. 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.

METHODOLOGY

This paper is trying to get the data set of required Object Oriented metrics from the live projects of C++ from different software development houses. After extracting the metrics will find the correlation among the metrics and will get the set of independent metrics. After finding and removing different anomalies [17] of OO metrics. The resultant set is not measuring the redundant metrics values of projects. By the resultant set we will be able to check the quality of our object oriented language code. We will try to suggest this model set of object oriented metrics.

C.K. METRICS

Chidamber and Kemerer define the so called CK metric suite this metric suite offers informative insight into whether developers are following object oriented principles in their design. They claim that using several of their metrics collectively helps managers and designers to make better design decision. CK metrics have generated a significant amount of interest and are currently the most well known suite of measurements for OO software. Chidamber and Kemerer proposed six metrics; the following discussion shows their metrics.

Weighted Method per Class (WMC)

It relates directly to the definition of complexity of an object. The number of methods and the complexity of methods involved are indicators of how much time and effort is required to develop and maintain the object. The larger the number of methods in an object, the greater the potential impact on the children, since, children will inherit all the methods in the object. A large number of methods can result in a too application specific object, thus limiting the possibility of reuse [17]. Since WMC can be described as an extension of the CC metric (if CC is used to calculate WMC) that applies to objects, its recommended threshold value can be compared with the upper limit of the CC metric.

Depth of Inheritance Tree (DIT)

DIT metric is the length of the maximum path from the node to the root of the tree. So this metric calculates how far down a class is declared in the inheritance hierarchy. [17] 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 behavior. 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.

Number of children (NOC)

This metric measures how many sub-classes are going to inherit the methods of the parent class. If NOC grows it means reuse increases. On the other hand, as NOC increases, the amount of testing will also increase because more children in a class indicate more responsibility. So, NOC represents the effort required to test the class and reuse.

Coupling between objects (CBO)

The idea of this metrics is that an object is coupled to another object if two object act upon each other. A class is coupled with another if the methods of one class use the methods or attributes of the other class. An increase of CBO indicates the reusability of a class will decrease. Thus, the CBO values for each class should be kept as low as possible [18] .CBO metric measure the required effort to test the class.

Response for a Class (RFC)

RFC is the number of methods that can be invoked in response to a message in a class. Pressman States, since RFC increases, the effort required for testing also increases because the test sequence grows. If RFC increases, the overall design complexity of the
Class increases and becomes hard to understand. On the other hand lower values indicate greater polymorphism. [17]

Lack of Cohesion in Methods (LCOM)

This metric uses the notion of degree of similarity of methods. LCOM measures the amount of cohesiveness present, how well a system has been designed and how complex a class is. LCOM is a Count of the number of method pairs whose similarity is zero, minus the count of method pairs whose similarity is not zero.

RESULT

Here the values of metrics are calculated for the jlib software using Columbus framework [16] [20]. After calculating the values of C.K. metrics, relation between these metrics is calculated using SPSS statistics. each and every one of the metrics the minimum, maximum, mean, median and standard deviation were calculated on every source code. Parts of the results of the experimental study are presented in table.

Descriptive Statistic of Jlib

Table 1: Statistics

	NOC	DIT	CBO	RFC	WMC	LCOM
N	461	461	461	461	461	461
	3862	3862	3862	3862	3862	3862
Mean	.61	1.00	3.11	12.09	21.09	62.42
Median	.00	1.00	2.00	6.00	6.00	5.00
Std. Deviation	3.842	1.144	3.889	16.567	42.966	303.360
Minimum	0	0	0	0	0	0
Maximum	69	5	23	141	410	4892

Each metric was collected from different classes in the system. Since all metrics Measure something related to program code and its components, it's likely to expect that some correlation exists. Taking this statement into account correlation between the metrics are calculated and a significant value of these metrics is calculated.

Table 2: Correlations

		NOC	DIT	CBO	RFC	WMC	LCOM
NOC	Pearson Correlation	1	-.015	.037	.130**	.074	.226**
	Sig. (2-tailed)		.744	.424	.005	.110	.000
	N	461	461	461	461	461	461
DIT	Pearson Correlation	-.015	1	.618**	.378**	.216**	.082
	Sig. (2-tailed)	.744		.000	.000	.000	.077
	N	461	461	461	461	461	461
CBO	Pearson Correlation	.037	.618**	1	.778**	.581**	.361**
	Sig. (2-tailed)	.424	.000		.000	.000	.000
	N	461	461	461	461	461	461

RFC	Pearson Correlation	.130**	.378**	.778**	1	.798**	.691**
	Sig. (2-tailed)	.005	.000	.000		.000	.000
	N	461	461	461	461	461	461
WMC	Pearson Correlation	.074	.216**	.581**	.798**	1	.641**
	Sig. (2-tailed)	.110	.000	.000	.000		.000
	N	461	461	461	461	461	461
LCOM	Pearson Correlation	.226**	.082	.361**	.691**	.641**	1
	Sig. (2-tailed)	.000	.077	.000	.000	.000	
	N	461	461	461	461	461	461

**. Correlation is significant at the 0.01 level (2-tailed).

Like NOC is significantly correlated with RFC and LCOM.DIT is significant correlated with CBO,RFC and WMC.Now Result of Jlib calculated with all metrics. We will neglect the those metrics whose of sig. value greater than .05.

Table 3: Coefficients

Model		Unstd. Coefficients		Std. Coefficients	t	Sig.	Collinearity Statistics	
		B	Std. Error	Beta			Tolerance	VIF
1	(Constant)	.647	.273		2.370	.018		
	LCOM	-.005	.001	-.288	-5.245	.000	.415	2.407
	WMC	.120	.008	.949	15.675	.000	.341	2.934
	RFC	-.108	.029	-.330	-3.682	.000	.156	6.428
	CBO	.199	.099	.143	2.009	.045	.248	4.039
	DIT	-.286	.219	-.060	-1.305	.193	.583	1.717
	NOC	.046	.051	.033	.897	.370	.938	1.066

Result of Jlib with all metrics. We will neglect the NOC, DIT and LOC because of sig. value greater than .05.VIF value is in control.

Table 4: Coefficients

Model		Unstd Coefficients		Std Coff	t	Sig.	Collinearity Statistics	
		B	Std. Error	Beta			Tolerance	VIF
1	(Constant)	.647	.273		2.370	.018		
	LCOM	-.005	.001	-.288	-5.245	.000	.415	2.407
	WMC	.120	.008	.949	15.675	.000	.341	2.934
	RFC	-.108	.029	-.330	-3.682	.000	.156	6.428
	CBO	.199	.099	.143	2.009	.045	.248	4.039

Now in this table we are having the metrics framework to detect the quality of the code. VIF is less than 10 and sig value is less than .05. This is a correct model

CONCLUSION

By analyzing metrics, a developer can correct those areas of software process that are the cause of software defects. Regarding the practical use of metrics to improve code design the same conclusion can be drawn; it can improve the design to some extent since the use of metrics can aid a developer to easily spot simple design flaws .CK metrics suite is a set of six metrics which capture different aspects of an OO design; these metrics mainly focus on the class and the class hierarchy. It includes complexity, coupling and cohesion as well.. Many metrics have been adapted from CK metrics suite. In this literature we discussed CK metrics elaborately and we also analyzed some of the CK metrics. In our analysis we found some result, These results suggest that four of the six of CK's metrics (WMC, RFC, LCOM and CBO) are useful quality indicators for predicting fault-prone classes.

ACKNOWLEDGMENT

I would like to thank my Research Guide Raman Maini, Associate Professor, University College of Engineering, Punjabi University Patiala for his valuable assistance, help and guidance during the research process and thank to Columbus framework contain a tool set which supports metrics calculation and provides a common interface for other reverse engineering tasks as well.

REFERENCES

[1] B. Henderson-sellers, Object-Oriented Metrics, Measures of Complexity .Prentice Hall, 1996.

[2] F.B. Abreu and R. Carapuca, "Candidate Metrics for Object-Oriented Software within a Taxonomy Framework," System and Software, vol. 26, no. l, pp. 87-96, Jan. 1994.

[3] L. Briand, S. Morasca, and V. Basili, De$ning and Vdidating High- Level Design Metrics, Techtucal Report CS-TR-3301, Univ. of Maryland, Dept. of Computer Science, College Park, Md., 1994.

[4] L. Briand, S. Morasca, and V. Basili, "Property Based Software

[5] Engineering Measurement," IEEE Trans. Software Eng., vol. 22, no. 1, p. 68-86, Jan. 1996.

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

[7] L.Briand , W.Daly and J. Wust, A Unified Framework for Coupling Measurement in Object-Oriented Systems. IEEE Transactions on software Engineering, 25, 91 121,1999.

[8] L.Briand , W.Daly and J. Wust, Exploring the relationships between design measures and software quality. Journal of Systems and Software, 5 245-273, 2000.

[9] Lorenz, Mark & Kidd Jeff, Object-Oriented Software Metrics, Prentice Hall, 1994.

[10] McCabe and Associates, Using McCabe QA 7.0, 1999, 9861 Broken Land Parkway 4th Floor Columbia, MD 21046.

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

[12] Moreau, D. R., "A Programming Environment Evaluation

[13] Methodology for Object-Oriented Systems", Ph.D. Dissertation,

[14] University of Southwestern Louisiana, 1987.

[15] Moreau, D. R., and Dominick, W. D., "Object-Oriented Graphical Information Systems: Research Plan and Evaluation", Journal of Systems and Software, vol. 10, pp. 23-28, 1989.

[16] Moreau, D. R., and Dominick, W. D., "A Programming Environment Evaluation Methodology for Object-Oriented Systems: Part I – The Methodology", Journal of Object-Oriented Programming, vol. 3, pp. 38-52, 1990.

[17] I.Brooks, "Object-Oriented Metrics Collection and Evaluation with a Software Process," Proc. OOPSLA '93 Workshop Processes and Metrics for Object-Oriented Software Development, Washington, D.C., 1993.

[18] R.Harrison, S.J.Counsell, and R.V.Nithi, An Evaluation of MOOD set of ObjectOriented Software Metrics. IEEE Trans. Software Engineering, vol.SE-24, no.6, pp. 491-496 June1998.

[19] Siket, I., Ferenc, R.: Calculating Metrics from Large C++ Programs

[20] 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.

[21] S.R. Chidamber and C.F. Kemerer, "Authors Reply," lEEE Trans. Software Eng., vol. 21, no. 3, p. 265, Mar. 1995.

[22] S.R.Chidamber and C.F.Kamerer, A metrics Suite for Object-Oriented Design. IEEE Trans. Software Engineering, vol. SE-20, no.6, 476-493, 1994.

[23] Z. Balanyi and R. Ferenc. Mining Design Patterns from C++

[24] Source Code. In Proceedings of the 19th International Conference on Software Maintenance (ICSM 2003), pages 305–314. IEEE Computer Society, Sept. 2003.

SHORT BIODATA OF THE AUTHOR

Raman Maini received B.Tech (Computer Science & Engineering) from Beant College of Engineering, Gurdaspur, Punjab, India in 1999 and M.Tech(Computer Science & Engineering) from PAU, Ludhiana, India, in 2002. He got Merit certificate in his M.Tech thesis at PAU. He is currently working as an Assistant Professor in Computer Engineering at University College of Engineering, Punjabi University, Patiala, India. He is a life member of ISTE (Indian Society of Technical Education), India and IETE (Institution of Electronics & Telecommunication Engineers), India. His current area of research is Computer Vision (Specialty Noise Reduction in Medical Images, Edge Detection and Image Enhancement).

Suruchi Mehra received B.Tech(Computer Science & Engineering) from RIMT-IET , Mandigobindgarh, Punjab, India in 2009 and pursuing M.Tech(Computer Science & Engineering(2009-2011) from Punjabi university, Patiala, India.

