Taxonomy of Metrics for Assessing Software Quality

Aman Kumar Sharma
Computer Science Department,
Himachal Pradesh University
Shimla, India

Dr. Arvind Kalia
Computer Science Department
Himachal Pradesh University
Shimla, India

Dr. Hardeep Singh
Computer Science Department
Guru Nanak Dev University
Amritsar, India

Abstract

As object oriented paradigm is gaining popularity, software metrics play an important role in ensuring software quality. In this paper we first introduce the theoretical concept of object oriented metrics, specifically of CK metrics suite. Then a case study of analyzing Java based open source software using CK metrics to evaluate quality is presented. The results are interpreted to help the software developers and researchers in improving the quality of the software during the development of the software.

Index Terms— CK suite, Maintainability, Object Oriented, Software Quality, Testability.

1. Introduction

The general focus of the software industry has shifted from providing increasingly more functionality to improving the quality of software. The ease of use, security, stability and reliability form some of the factors of quality. Software quality is becoming an increasingly important area in software engineering. Software industry deals with big budgets, many man years, lots of effort and high dependence shown by customers. It gives rise to the need of having quality software. Conformance to user requirements, fitness for use, conformance to specifications and the value a customer willingly pays for the software all lead to quality of software [1]. Nevertheless, it is easier said than done, as everyone involved in the software development process is all too well aware, high quality software is not easy to produce. Measuring software quality is complex and further, improving quality of the software product during its development increases complexity.

Achieving and maintaining quality is difficult for the fact that it is impossible to maximize all quality attributes simultaneously [2]. Quality is "the degree to which a set of inherent characteristics fulfills requirements" [3]. Software quality is viewed as "customer value" and "defect levels" by Highsmith [4]. Pfleeger [5] warns against approaches that focus only on the measurement of quality of final product.

This study pertains to an empirical analysis of object oriented (oo) software domain to evaluate software quality metrics. The traits of object oriented namely inheritance, coupling and encapsulation have enabled object oriented in gaining popularity manifold in recent years. Doornik [6] has argued that "object-oriented programming can bring important benefits when used in econometric, financial and statistical computing". Chih-Min Lo et. al. [7] have discussed the application of objects, components in software development techniques and have further, highlighted the popularity of object oriented programming.

Chidamber & Kemerer proposed a CK metrics suite [8] for the purpose of measuring quality. There are numerous means of applying these metrics to practical projects to prove the validity of the metrics. However, the quality of the product is established after the product is ready for use or for sale. The object oriented metrics provide feedback to software developers and project managers to develop better software in future. As object oriented languages are becoming more and more acceptable to the software world, the requirement of measuring software quality during its development is the need of the hour. This study is aimed at identifying object oriented metrics that can be utilized to characterize the degree of quality of object oriented programs. It further, deals with potential areas to improve software quality during its development.

The Chidamber Kemerer (CK) metrics suite is designed to provide a summary of the overall quality of object oriented software and is available at the class level. The CK suite quantifies the use of four main mechanism of object oriented design namely encapsulation, inheritance, polymorphism and coupling for which metrics are identified as Weighted Method Methods per Class (WMC), Depth of Inheritance Tree (DIT), Number Of Children (NOC), Coupling Between Object (CBO), Response For a Class (RFC) and Lack of Cohesion of Method (LCOM).

Method Methods per Class (WMC): This measures the complexity of an individual class. Two different weighting functions are considered.

One approach is to measure the number of functions in each class whereas another approach is to count those functions accessible to other modules. In this study, the former approach was adopted i.e. to count all methods of a class.

Depth of Inheritance Tree (DIT): It is defined as the length of the longest path of inheritance. The deeper the inheritance tree for a class, the complex it is to ascertain the behavior due to the interaction between the inherited features.

Number Of Children (NOC): It is the number of classes that inherit directly from the current class. Moderate values for this measure indicate the scope for reuse. A high value of NOC increases complexity as it has to provide services to many classes in various contexts.

Coupling Between Objects (CBO): A class is coupled to another if it uses the member functions or variables of the other class. The CBO provides the number of other modules which are coupled to the current module. Excessive coupling indicates weakness of module encapsulation and may inhibit reuse. Highly coupled classes tend to introduce more faults caused by inter class activities.

Response For a Class (RFC): It is the number of methods that can potentially be executed in response to a message received by an object of that class. The larger the number of methods that could potentially respond to a message, the greater is the complexity of that class.

Lack of Cohesion Of Methods (LCOM): A count of the number of methods – pairs whose similarity is zero minus the count of method pairs whose similarity is not zero. However, the metric LCOM is not used in this study due to the ambiguity in the definition of LCOM metric [9]. It has been identified that the original definition of LCOM truncates all values below zero and this truncation limits the metrics ability to fully capture the lack of cohesion. It is possible that the truncation of values below zero may reduce the variability of the metric and limit its usefulness in explaining productivity or defects [10]. For this reason, the LCOM metric is omitted from this study.

Using the results of this study, practitioners, academicians and designers can identify potential problem areas and can improve their software during its development. This paper is organized as follows: Brief reviews of some of the existing literature are presented in Section 2. In Section 3, description about the objective of this study is given. The research methodology opted for this research is discussed in Section 4. Section 5 contains results and empirical analysis of object oriented software metrics used in this study and finally Section 6 sums up with conclusion and future scope.

2. Review of literature

An overwhelming majority of researchers have concentrated on theoretical and empirical validation of object oriented software metrics. Dumke [11], Campanai et al. [12] and Sellers [13] have contributed in area of object oriented design metrics however, there is inadequate information on application of proposed metrics and moreover, experimental validations are missing.

Amjan Shaik et al. [14] in their study have validated an object oriented software metric CK suite by using the data collected from projects. It was concluded in their study that if appropriately used it could lead to a significant reduction in cost of the overall implementation and improvement in quality of the software product.

Chidamber et al. [15] explored the applicability of CK metrics on practical managerial work such as productivity and rework effort. The results suggested that CK metrics were significant economic variable indicators for the three commercial object oriented systems used in the study. Other object oriented metrics are proposed to complement CK metrics. A metric suite of coupling measurement was proposed for OOD and was empirically validated using logistic regression technique [16]. The study further suggested that object oriented coupling measurement metrics are complementary quality indicators to CK object oriented metrics. The CK set of metrics are gradually gaining popularity as observed in [17].

A number of studies [18] [19] [8] [15] have used CK metric suite for analysis of quality of software. Further, cross validation study of CK metrics also exist [20] [21].

It is concluded that CK metrics suite is a good indicator of quality, analysis have been performed to evaluate quality of the ready to use software. However, the scope of improving the software during its development has not been practiced in these studies.

3. Objectives of the study

The researcher in this study has performed empirical validation of CK metric suite on different versions of the software to make the results generalized. Two software were selected to validate the experiment with three versions of each software namely Apache Ivy and Heritrix. Further, in this study five design metrics were considered for the analysis. The main objective of this study is to empirically validate the CK metrics. Also the results obtained may provide an insight into the relationship between the software quality factors and the object oriented design metrics. Thus, attempting to provide a mechanism to evaluate

quality of software during the development process is made in this study.

4. Scope of the study

In this study the selected software components are written in java. It is due to the fact that java is platform independent and is most commonly used object oriented programming language. Both Apache Ivy and Heritrix distinct software are selected for analysis comprising three versions of each. Apache Ivy is an OSS developed in Java in the year 2004 as a tool for managing project dependencies by recording, tracking, resolving and reporting. Three versions of Apache Ivy used for measurement and analysis are Apache Ivy 2.2.0, Apache Ivy 2.1.0. and Apache Ivy 2.0.0. The other software component Heritrix is also an OSS and developed using Java language. It is a web crawler and was developed in 2004. The three versions selected in this study are Heritrix 3.1.0, Heritrix 1.14.4 and Heritrix 3.0.0.

5. Research methodology

A software metrics tool CKJM was used to compute the CK suite metrics namely WMC, DIT, NOC, CBO and RFC for the selected versions of Apache Ivy and Heritrix. The correlation values between the various CK suite metrics were computed by using Public Social Private Partnership (PSPP) tool. PSPP is an OSS and statistical software tool. It produces tabular output in Hyper Text Markup Language, ASCII or in post script format. The study used bi-variate correlation with values ranging between -1 to +1. A positive correlation value denotes that the metrics are linearly correlated to each other. A positive value of 1 depicts that with every increase in one of the metric, increases the value of another metric too. Correlation values of less than -0.5 or more than +0.5 indicate a strong correlation between the metrics negatively or positively respectively. Whereas a zero correlation value means that the metrics are not related to each other.

6. Analysis of results

The CKJM tool computes the values of WMC, DIT, NOC, CBO and RFC for the software. The correlation results of the software for the three evaluated versions of Apache Ivy are represented below:

Table 1. Correlation Values of Apache Ivy 2.0.0 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	0.0	0.02	0.94	0.85	0.81
NOC		0.02	0.02	-0.01	0.05
DIT			0.04	-0.01	0.06
RFC				0.94	0.75
CBO					0.53

Table 2. Correlation Values of Apache Ivy 2.1.0 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	0.04	0.04	0.78	0.54	0.80
NOC		-0.01	0.11	-0.02	0.14
DIT			-0.02	-0.02	-0.06
RFC				0.72	0.89
CBO					0.58

Table 3. Correlation Values of Apache Ivy 2.2.0 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	0.0	0.01	0.92	0.85	0.80
NOC		0.03	0.02	-0.01	0.04
DIT			0.04	-0.01	0.04
RFC				0.95	0.74
CBO					0.55

Table 1, Table 2 and Table 3 show correlation values between different CK metrics of Apache Ivy versions. For instance in Table 1, the value of 0.81 depicts the correlation between WMC and LOC. The same analogy is applied for the remaining values shown in Table 1, Table 2 and Table 3.

Table 4. Correlation Values of Heritrix 3.0.0 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	-0.03	-0.13	0.86	0.29	0.94
NOC		-0.09	0.0	-0.06	0.0
DIT			-0.07	-0.06	-0.12
RFC				0.31	0.89

CBO					0.27

Table 5. Correlation Values of Heritrix 1.14.4 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	-0.01	-0.10	0.90	0.28	0.94
NOC		0.01	0.01	-0.04	0.0
DIT			-0.13	-0.03	-0.15
RFC				0.33	0.91
CBO					0.23

Table 6. Correlation Values of Heritrix 3.1.0 for CK Metrics

	NOC	DIT	RFC	CBO	WMC
LOC	-0.02	-0.09	0.86	0.27	0.94
NOC		-0.10	0.01	-0.06	0.01
DIT			-0.02	-0.02	-0.07
RFC				0.28	0.90
CBO					0.25

The correlation values of Heritrix 3.0.0, Heritrix 1.14.4 and Heritrix 3.1.0 are listed in Table 4, Table 5 and Table 6 respectively. A value of -0.03 in Table 4 means that the correlation between NOC and LOC is negative 0.03, hence negligible correlation. The rest of the correlation values are interpreted on the same analogy.

The results have a similar trend across all the three versions of Apache Ivy and Heritrix as depicted in Table 1, Table 2 and Table 3 for Apache Ivy and Table 4, Table 5 and Table 6 for Heritrix. Moreover, the correlation values have similar inclination in Apache Ivy and Heritrix except for correlation between CBO - LOC, and CBO - RFC. It is evident from the result that RFC – LOC, WMC – LOC and WMC – RFC are very strongly correlated to each other. Moderate correlation exists between WMC – CBO. A very weak or negligible correlation exist between NOC – LOC, DIT – NOC, DIT – LOC, RFC – NOC, RFC – DIT, CBO – NOC, CBO – DIT, WMC – NOC and WMC – DIT for all the versions of Apache Ivy and Heritrix.

The study projects that classes consisting of few lines of code are designed with long hierarchy. This results in value of LOC being consistent with fluctuating values of DIT and NOC. Thus, correlation value between LOC and (DIT and NOC) represents a negligible correlation. Further, there seems to be no relation between the breadth and depth for the classes. Higher value of DIT has no impact on NOC and vice versa, as the correlation value among DIT and NOC is insignificant. Though the factors to evaluate reusability are NOC and DIT, the study has revealed the fact the classes are not extensively reused, hence the quality has been adversely affected. With the increase in the WMC correlation value, there is potential of increased method calls hence, increased RFC. Therefore, in the study a very high correlation exists between RFC and WMC for all the versions of all the software applied in the study. To measure maintainability WMC and CBO metrics are used. In the study it is found that Apache Ivy has a strong correlation between WMC and CBO which implies that Apache Ivy is maintainable software. On the contrary, for the different versions of Heritrix the correlation value between WMC and CBO is insignificant. Thus, maintainability quality factor for Heritrix is low. The same reasoning is applied on the correlation involving CBO and RFC. In this instance too, the correlation value for various Apache Ivy versions is very high however, for Heritrix versions the correlation value was weak. This signifies that Apache Ivy is more testable and understandable as CBO and RFC are the metrics to evaluate testability and understandability quality factors. Weak correlation value between WMC and DIT is due to the fact that classes with larger number of methods may or may not have any impact on DIT i.e. more methods might be spread breadth wise or depth wise. The same analogy is applied on the correlation linking WMC and NOC.

7. Conclusion and Future Scope

CK metrics was validated using Apache Ivy and Heritrix with three versions of each. The results suggest that WMC relation with RFC and CBO is a good indicator for quality evaluation. From the observations of this study, it can be evaluated how to use CK metrics more efficiently to evaluate quality of the software in its development stage. In future, validation of CK metrics suite can be performed on several object oriented languages to strengthen the view point of this study.

8. References

[1] R. A. Khan, K. Mustafa and S. I. Ahson, *Software Quality: Concepts and Practices* 1st ed., India: Narosa Publishing House, 2008, pp. 7-18.

[2] Maria Haigh, "Software quality, non-functional software requirements and IT-business alignment," *Software Quality Journal, Springer,* vol. 18, pp. 361-385, 2010.

[3] *ISO /IEC 9000:2000*, "Quality Management Systems- Fundamentals and Vocabulary", Geneva, Switzerland: International Organization for Standardization.
[4] J. Highsmith, *Agile software development ecosystems*, Addison Wesley Professional, 2002.
[5] S. L. Pfleeger, *Software Engineering: Theory and Practice*, 2nd ed. New Jersey: Prentice Hall, 2001.
[6] Jurgen A. Doornik, *Object Oriented Programming in Econometrics and Statistics Using OX: A Comparison with C++, Java and C#*, Springer, 2002.
[7] Chih-Min Lo and Sun-Jen H Jang, "Applied Object Oriented Programming Technology to ICT Applications Development", in *2010 Proc. SICE Annual Conference*, August 2010, pp 3336-3339.
[8] S.R. Chidamber and C.F. Kemerer, "A metrics suite for object oriented design", *IEEE Transactions on Software Engineering*, vol. 20 no. 6, pp 476-493, 1994.
[9] 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.
[10] Ramanath Subramanyam and M.S. Krishnan, "Empirical analysis of CK metrics for object oriented design complexity: implications for software defects", *IEEE Transactions on Software Engineering*, vol. 29 no. 4, 2003.
[11] R. Reiner Dumke, "A measurement framework for object oriented software development", *Annals of Software Engineering*, vol. 1, 1995.
[12] M. Campanai and P. Nesi, "Supporting Object Oriented Design with Metrics", in *Proc. of Tools 94*, Europe, France 1994.
[13] B. Henderson Sellers, "Identifying Internal and External Characteristics of Classes likely to be useful as Structural Complexity Metrics", in *1994 Proc. International Conference on Object Oriented Information Systems (OOIS 94)*, London, Springer, Verlag pp. 227-230, 1994.
[14] Amjan Shaik, Dr. C. R. K. Reddy and Dr. A. Damodaran, "Statistical analysis for object oriented design software security metrics", *International Journal of Engineering Science & Technology*, vol. 2 no. 5, pp. 1136-1142, 2010.
[15] S. R. Chidamber, D. P. Darcy and C. F. Kemerer, "Managerial use of metrics for object oriented software: An exploratory analysis", *IEEE Transactions on Software Engineering*, vol. 24 no. 8, pp. 629-639, 1998.
[16] L. Briand, P. Devanbu and W. Melo, "An investigation into coupling measures for C++", Technical Report ISERN -96-08, ISERN 1996.
[17] Amandeep Kaur, Satwinder Singh, K.S. Kahlon and Parvinder S. Sandhu, "Empirical analysis of CK & MOOD metric suit", *International Journal of Innovation, Management and Technology*, vol. 1 no. 5, Dec 2010.
[18] Kuljit Kaur Chahal and Hardeep Singh, "Metrics to study symptoms of bad software designs", *ACM SIGSOFT Software Engineering*, vol. 34 no. 1, pp. 1-4, ACM New York 2009.
[19] Giulio Concas, Michele Marchesi, Alessandro Murgia and Roberto Tonelli, "An empirical study of social networks metrics in object oriented software", Advances in Software Engineering, 2010.
[20] K. El-Emam, S. Benlarbi, N. Goel and S. Rai, "A Validation of Object Oriented Metrics", NRC/ERB 1063, National Research Council of Canada, 1999.
[21] Aman Kumar Sharma, Arvind Kalia and Hardeep Singh, "Empirical analysis of object oriented quality suites", *International Journal of Engineering and Advanced Technology*, vol. 1 no. 4, 2012.