IJCSE International Journal of Computer Sciences and Engineering Open Access
Research Paper Vol.-7, Issue-1, Jan 2019 E-ISSN: 2347-2693

A Comparative Study of Software Metrics for Analysis and Its Impact on Predictability

G.Yamini[1*], Gopinath Ganapathy[2]

[1]Department of Computer Science, Bharathidasan University, Trichy, Tamil-nadu, India
[2]Bharathidasan University, Trichy, Tamil-nadu, India

**Corresponding Author: yamini.per@gmail.com, Tel.: 919489262627*

Available online at: www.ijcseonline.org

Accepted: 14/Jan/2019, Published: 31/Jan/2019

Abstract— Quality assurance is one of the important non-functional software requirements which many software products fail to satisfy. Current software market is driven mostly by urgency and competition. One of the methods to ensure software quality is a metrics-based approach. Software metrics have been used to quantitatively evaluate software products. Software metrics play an important role in developing high quality software as well as to improve the developer's productivity. Metrics can help quantify previous work in a way that can directly guide future efforts. For example, projects of different sizes can require vastly different levels of effort, organizational structure, and management discipline. There is an increasing need for metrics adapted to the Object-Oriented (OO) paradigm to help manage and foster quality in software development. Object-oriented design patterns are an emergent technology: they are reusable micro-architectures, high level building blocks. A major benefit of object-oriented software development is the support for reuse provided by object-oriented and object-based languages. The usefulness of metrics is reviewed. The reliability is one of the most important attributes of software quality. The presumed objective of the estimation of the reliability consists in the analysis of the risk and of the reliability of the software-based systems. This paper presents the study of different suite in object-oriented (OO) design metric.

Keywords— Software metrics; Object-oriented; MOOD; CK metric

I.INTRODUCTION

Software metrics are useful in analyzing, designing, coding, testing, documentation and measuring software quality and complexity. Software metric is a measure to quantify an attribute of software. In the 21 st century many software developers are dependent on object-oriented programs where above 95% are the reusable components. It is difficult to make reusable components, but if made, it gives benefits like improved quality and reduced code size. As size of the code will decrease, definitely complexity will be reduced with consequent possibility of reduction in timing and staffing of the project. Software metrics are the decisive factors to measure the quality and complexity of software. Object oriented design is becoming more popular in software development environment and object-oriented design metrics is an essential part of software environment. The metrics for object-oriented design focus on measurements that are applied to the class and design characteristics.

Various object-oriented metric suites have been proposed like CK, MOOD, Lorentz and Kidd. Object oriented metrics are most useful in early stage of software development. To minimize effort and to identify a metric relevant to measure an attribute validation of a metric is essential. Validation of a metric means to verify whether the metric is relevant to the attribute being measured. Validation technique can be empirical or analytical. CK suite was proposed in [2]. Message passing coupling (MPC), data abstraction coupling (DAC), number of methods (NOM) and some other metrics have been proposed in [3]. The metrics for object oriented design (MOOD) suite was proposed by Fernando Brito and Rogerio. Complexity is undesirable in a program. A high value of complexity is undesirable. CK suite is designed in such a way that low value implies low complexity. There is a misconception that any design metric is a complexity metric. Each CK suite is a complexity metric but has a different definition for each metric. Different researchers have pointed out regarding the reusability metrics [1] [2]. The best example in C++ is the Template which provides the reusable modules like FT (Function Template) and CT (Class Template) [3]. In earlier days the developers used traditional metrics but nowadays object-oriented metrics plays a vital role. The objective of object oriented programming is to develop a new system by using an existing system .This is possible due to reusability i.e. Inheritance [4] [5]. Nowadays CBRM (Component-Based-Reused-Metrics) are used widely as well as the component based reused programs are fully dependent on the repository for achieving success [7] [8]. Principal component analysis, Spearmans rank correlation, Logistic regression, Decision tree, Linear regression, Neural network, Multiple linear regression, Principal component analysis, Naive Bayes, and Random forest, Logistic regression, Probabilistic neural network, Bayesian poission regression, Multiple regression, and Ordinary least square, and Expert estimation, Statistical regression analysis, and Principal component analysis, Support vector machine, ANN, and Decision tree, are the various methods used by different researchers, used in different situations accordance to the nature of the software.

II. OBJECT-ORIENTED METRICES

It is widely accepted that object oriented development requires a different way of thinking than traditional structured development and software projects are shifting to object oriented design. Object oriented design is those design which contained all the properties and qualities of software that is related to any large or small project. It is a degree through which a system object can hold a particular attribute or characteristic. The main advantage of object oriented design is its modularity and reusability. Object oriented metrics are used to measure properties of object oriented designs. Metrics are a means for attaining more accurate estimations of project milestones, and developing a software system that contains minimal faults. Compared to structural development, object oriented design is a comparatively new technology. Many object oriented metrics have been proposed to assess the testability of an object oriented system. Most of the metrics focus on encapsulation, inheritance, class complexity and polymorphism.

A.C. K. Metric Suite

Chidamber and Kemerer define the so called CK metric suite [2]. CK metrics have generated a significant amount of interest and are currently the most well-known suite of measurements for object oriented software. CK suite consist of six metrics WMC, DIT, NOC, CBO, RFC and LOCM. CK metric suites have been tested in C++ and Smalltalk. This suite has come under criticism on the basis of lack of clear terminology and a number of inadequacies in the meaningfulness of the metrics. But in spite of all criticism this suite has become industry standard.

Weighted methods per class (WMC)-WMC is the number of all member functions and operators defined in each class. Friend operators are not counted. Member functions and operators inherited from the ancestors of a class are also not counted. WMC should be kept as low as possible. It is used to measure the understandability, reusability, maintainability and complexity and quality.

Depth of inheritance tree of a class (DIT)-DIT is the length of longest path from the class to the root in the inheritance hierarchy.

Number of children (NOC)- NOC is the number of classes in the inheritance tree of a class. NOC represents the effort required to test the class and reuse. NOC should be kept as low as possible. It is used to measure the quality.

Coupling between objects (CBO)- CBO provides the number of other modules that are coupled to the current module either as a client or supplier. Increase in CBO will decrease the usability. It is used to measure complexity, reusability and quality.

Response for a class (RFC)-RFC is the count of methods within a set which can be invoked in response to a message sent to an object to perform an operation. RFC should be kept as low as possible. It measures complexity.
Lack of cohesion of methods (LCOM)- LCOM is the difference between the number of methods whose similarity is zero and the methods whose similarity is not zero. It is not a good metric of quality.

B.MOOD Metric Suite

Abreu et at. defined MOOD (Metrics for Object Oriented Design) metrics [15]. MOOD suite consists of the following the following six metrics. MOOD metrics focus on system level which includes encapsulation, inheritance, polymorphism, and massage passing. This suite is applicable to all object-oriented programming languages like C++, JAVA.

Attribute Hiding Factor (AHF) - AHF is the ratio of attributes hidden to the total data members defined. It should be kept as high as possible. It is used to measure quality.
Attribute Inheritance Factor (AIF) - AIF is the ratio of the sum of inherited attributes in all classes of the system to the total number of available attributes for all classes. It should be kept as high as possible. It is used to measure quality.
Coupling Factor (CF)-CF is the ratio of the possible number of couplings in the software to the actual number of couplings. It is a measure of coupling between classes. It is not a good measure of quality.

Method Hiding Factor (MHF) –MHF is the ratio of method hidden to the total number of classes.
Metric Inheritance Factor (MIF)-MIF is the ratio of method inheritance to the total number of available methods.

Polymorphism Factor (PF)-Polymorphism is the ability to take several forms.PF is the ratio of the number of methods that redefine inherited methods to the maximum number of possible distinct polymorphic situations.

C. Lorenz and Kidd suite

In the fundamental book about software quality [14] Lorenz and Kidd introduced many metrics to quantify software quality assessment. Eleven metrics introduced by Lorenz and Kidd are applicable to class diagrams. 1. Number of public methods (NPM) 2. Number of methods (NM) 3. Number of public variables (NPV) 4. Number of variables per class (NV) 5. Number of class variables (NCV) 6. Number of class methods (NCM) 7. Number of methods inherited (NMI) 8. Number of methods overridden (NMO) 9. Number of new methods (NNM) 10. Average parameter per method (APM) 11. Specialization index (SIX). The metrics were categorized into Inheritance Metrics, Class Internals Metrics.

D. Chen Metrics

The software metrics, through which it can defined, “What is the behavior of the metrics in object-oriented design?”. All of the terminologies in object-oriented language, consider the basic components of the paradigm to be objects, classes, attributes, inheritance, method, and message passing [12]. Each describes all of the behaviors like: i. CCM (Class Coupling Metric), ii. OXM (Operating Complexity Metric), iii. OACM (Operating Argument Complexity Metric), iv. ACM (Attribute Complexity Metric), v. OCM (Operating Coupling Metric), vi. CM (Cohesion Metric), vii. CHM (Class Hierarchy of Method) and viii. RM (Reuse Metric). Metrics (i) and (iii) are very subjective in nature, Metrics (iv) and metric (vii) mostly involve the count of features; and metric (viii) is a Boolean (0 or 1) indicator metric. It is stated that, each object-oriented metrics concept implies a programming behaviour.

E.QMOOD

Quality Model for Object-Oriented Design (QMOOD).The QMOOD [6] is a comprehensive quality model that establishes a clearly defined and empirically validated model to assess object-oriented design quality attributes such as understandability and reusability, and relates it through mathematical formulas, with structural object-oriented design properties such as encapsulation and coupling. The QMOOD model consists of six equations that establish relationship between six object- oriented design quality attributes (reusability, flexibility, understandability, functionality, extendibility, and effectiveness) and eleven design properties. The whole description for QMOOD can be obtained from the Bansiya‟s thesis through which, The QMOOD metrics can further classified into two measures namely:

System Measures: System measures describe such metrics like DSC (Design Size in Metrics), NOH (Number of Hierarchies), NIC (Number of Independent classes), NSI (Number of Single Inheritance), NMI (Number of multiple Inheritance), NNC (Number of Internal Classes), NAC (Number of Abstract Classes), NLC (Number of Leaf Classes), ADI (Average Depth of Inheritance), AWI (Average Width of Classes), ANA (Average Number of Ancestors).

Class Measures:- Class measure metrics are those metrics which define metrics like, MFM (Measure of Functional Modularity), MFA (Measure of Functional Abstraction), MAA (Measure of Attribute Abstraction), MAT (Measure of Abstraction), MOA (Measure of Aggregation), MOS (Measure of Association), MRM (Modeled Relationship Measure), DAM (Data Access Metrics), OAM (Operation Access Metrics), MAM (Member Access Metrics), DOI (Depth of Inheritance), NOC (Number of Children), NOA (Number of Ancestor), NOM (Number of Methods).

F.LiW

The six metrics defined are, Number of Ancestor Classes (NAC), Number of Local Methods (NLM), Class Method Complexity (CMC), Number of Descendent Classes (NDC), Coupling Through Abstract data type (CTA), and Coupling through Message Passing (CTM).

Number of Ancestor Classes (NAC):- This metric is proposed as an alternative to the DIT metric, measures the total number of ancestor classes from which a class inherits in the class inheritance hierarchy. The theoretical basis and viewpoints both are same as the DIT metric. In this the unit for the NAC metric is “class”, justified that because the attribute that the NAC metric captures is the number of other classes‟ environments from which the class inherits.

Number of Local Methods (NLM) - The Number of Local Methods metric (NLM) is defined as the number of the local methods defined in a class which are accessible outside the class. It measures the attributes of a class that WMC metric intends to capture. The theoretical basis and viewpoints are different from the WMC metric [11]. The theoretical basis describes the attribute of a class that the NLM metric captures. This attribute is for the usage of the class in an object- oriented design because it indicates the size of a class’s local interface through which other classes can use the class.

The major three viewpoints for NLM metric are: 1) The NLM metric is directly linked to a programmer’s effort when a class is reused in an Object-Oriented design. More the local methods in a class, the more effort is required to comprehend the class behaviour. 2) The larger the local interface of a class, the more effort is needed to design, implement, test, and maintain the class. 3) The larger the local interface of a class, the more influence, the class has on its descendent classes.

Class Method Complexity (CMC) - The Class Method Complexity metric is defined as the summation of the internal structural complexity of all local methods. The CMC metric's theoretical basis and viewpoints are significantly different from WMC metric. The NLM and CMC metrics are fundamentally different as they capture two independent attributes of a class. These two metrics affect the effort required to design, implement, test and maintain a class.

Number of Descendent Classes (NDC) – This metric is an alternative to NOC and is defined as the total number of descendent classes (subclass) of a class. The stated theoretical basis and viewpoints indicate that NOC metric measures the scope of influence of the class on its sub classes because of inheritance [13]. Li claimed that the NDC metric captures the classes attribute better than NOC.

Coupling through Abstract Data Type (CTA) – This is defined as the total number of classes that are used as abstract data types in the data-attribute declaration of a class. Two classes are coupled when one class uses the other class as an abstract data type [9] [10]. The theoretical view was that the CTA metric relates to the notion of class coupling through the use of abstract data types. This metric gives the scope of how many other classes' services a class needs in order to provide its own service to others.

Coupling through Message Passing (CTM) - The Coupling through Message Passing (CTM) is defined as the number of different messages sent out from a class to other classes excluding the messages sent to the objects created as local objects in the local methods of the class. Two classes can be coupled because one class sends a message to an object of another class, without involving the two classes through inheritance or abstract data type. Theoretical view given was that the CTM metric relates to the notion of message passing in object-oriented programming. The metric gives an indication of how many methods of other classes are needed to fulfil the classes' own functionality.

G. SATC Metrics

Rosenberg Linda proposed to select object oriented metrics that supports the goal of measuring the code, quality, result and they proposed many object-oriented metrics due to lack of theoretical basis and that can be validated.These metrics may be used to evaluate the object-oriented concepts like methods, coupling and inheritance as shown in table 1.

Table 1. Types of Metrics

Name	**Source**	**Metrics**
MOOSE/CK	Chidamber et.al.	WMC, DIT, NOC, CBO, RFC, LCOM
MOOD	Abrreu et.al.	MIF, AIF, MHF, AHF, POF, COF
LK	Lorenz et.al.	CS, NOO, NOA, SI, OS, OC, NP
QMOOD	Bansiya	DSC,NOH,NSI,NMI, NNC,NAC,NLC,ADI,AWI, ANA,MFM
LiW	Li et.al.	NAC, NLM,CMC,NDC,CTA,CTM
SATC	Rosenberg et.al.	CC, LOC,WMC,RFC,LCOM,DIT,NOC

III. CONCLUSION

The increase in software development means the measurement was also so high. The increasing significance being placed software measurement which has to lead, increase the amount of research on developing the new software measures. In this paper, the various software metrics for object-oriented method is analyzed. This paper provides some help for researchers and practitioners for better understanding and selection of software metrics for their purposes. This paper presented analysis of existing major object- oriented metrics. Four out of six metrics WMC, NOC, CBO, and RFC are suitable for complexity and quality measurement. This study also advice to metrics developers that, metrics should be simple, computable and programming language independent. Future work involves identifying limited set of metrics to model quality and complexity of object-oriented design and validating the identified metric suite against prevalent metric suites using different.

Machine learning helps to gain insight from a massive amount of data which is very cumbersome to humans. Machine learning is a subfield of soft computing and a rapidly up surging topic in today's context and is expected to boom more in coming days.

REFERENCES

[1] W. Li and S. Henry, " Maintenance Metrics for the Object-Oriented Paradigm", In Proceedings of the First International Software Metrics Symposium, Baltimore Maryland, pp. 52-60, 1993.
[2] L. Etzkorn, J. Bansiya, C. Davis, "Design and Code Complexity Metrics for OO Classes", Journal of Object- Oriented Programming, pp. 35-40, 1999.
[3] K. K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, "Software Reuse Metrics for Object - Oriented Systems", In Proceedings of ACIS Third International conference on Software Engineering Research, Management and Applications, 2005.
[4] P. Gandhi, P. K. Bhatia, "Reusability Metrics for Object - Oriented System: An Alternative Approach", International Journal of Software Engineering (IJSE), Vol. 1, No 4, pp. 63 – 72, 2010.
[5] Taylor, D. (1992): Object-Oriented Information Systems: Planning and Implementation. New York, US: John Wiley & Sons, Inc.
[6] W. Frakes and C. Terry, "Reuse Level Metrics", Proceedings of the 3rd International Conference on Software Reuse: Advances in Software Reusability, IEEE, 1994.
[7] J. Guo, Luqui, "A Survey of Software Reuse Repositories", 7th IEEE International Conference and Workshop on the Engineering of Computer Based Systems, pp. 92-100, 2000.

[8] Kaur, S. Singh and K. S. Kahlon, "Evaluation and Metrication of Object Oriented System", Proceedings of the International Multi Conference of Engineers and Computer Scientists 2009, I IMECS 2009, Hong Kong, 2009.

[9] Dr B.R. Sastry, M.V. Vijaya Saradhi, "Impact of software metrics on Object Oriented Software Development life cycle", International Journal of Engineering Science and Technology, Vol. 2, No. 2, pg 67-76, 2010.

[10] Y. Singh, A. Kaur, R. Malhotra, "Software fault proneness prediction using support vector machines", Proceedings of the World Congress on Engineering, vol. 1, pp. 1–3, 2009.

[11] G.J. Pai, J.B. Dugan, "Empirical analysis of software fault content and fault proneness using Bayesian methods", IEEE Trans. Softw. Eng. 33 pp. 675–686, 2007.

[12] H.M. Olague, L.H. Etzkorn, S. Gholston, S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes", IEEE Trans. Software. Vol. 33, pp. 402–419.2007.

[13] Y. Zhou, H. Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults", IEEE Transactions on software engineering, Vol. 32, pp. 771–789, 2006.

[14] R. Subramanyam and M.S. Krishnan, "Empirical Analysis of CK metrics for Object Oriented Design Complexity: Implications of Software defects", IEEE transactions on Software Engineering, Vol. 29, No. 4, 2003.

[15] S.R.Chidamber and C.F.Kemerer. "A metrics suite for object oriented design". IEEE Transactions on Software Engieneering, pp. 476 – 493, 1994.

Authors Profile

Ms. G.Yamini pursed Bachelor of Science, Master of Science, Master of Philosophy in computer science from Bharathidasan University , Trichy in 1998, 2001 and 2004 respectively. Currently pursuing P.h.D. and currently working as Faculty in Department of Computer Science, Bharathidasan University, Trichy, since 2011. Has presented many papers in conferences. Participated in various seminars and workshops in various institutions in and around Trichy. Guided around 20 M.Phil graduates. Interested in handling subjects like Microprocessor, Software Engineering, Software Metrics, Data Structures, Programming Languages. Main research work focuses on Software Metrics, Software Reliability, Prediction of reliability in software. Has 12+ years of teaching experience and 2 years of Research Experience.

Dr Gopinath Ganapathy pursed Bachelor of Science and Master of computer applications from St.Joseph's college, Trichy. Completed Ph.D from Madurai Kamaraj University. Currently working as a Registrar, Bhararhidasan University, Trichy.He has 30 years of total experience in academia, Industry, research & consultancy services. He has around 8 year International experience in the U.S and U.K. He served as a Consultant for a few fortune 500 companies that include IBM,Lucent-Bell Labs, Merrill Lynch Toyota etc. He received the Young Scientist Fellow award for the year 1994 from the Govt. of Tamil Nadu with cash Rs 20000. His name is listed in "Who is Who in the World – 2009" by Marquis, USA for individual accomplishment He is a nominee for "Rashtriya Gouvrav" award. He is the recipient of "Best Citizens of India" award. He published and presented nearing100 research papers at international journals and conferences. He acted as academic advisor for mediating off-campus programs in University of Valley Creek, USA He is a member in several academic and technology councils in various Universities in India He convened many international conferences/workshops/seminars. He is a referee and editorial member in a few international journals. Guidance for PhD: 12 awarded, 6 progressing. He is a Professional Member in IEEE, ACM, and IAENG. He is a Life Member in Indian Science Congress, Indian Society for Technical Education, and Computer Society of India. He specialized in designing Multi-Tier and EAI technologies. His areas of Research are: Semantic Web, AI, Auto Programming, Ontology and Text Mining.