ISSN 2348-1196 (print)
International Journal of Computer Science and Information Technology Research ISSN 2348-120X (online)
Vol. 4, Issue 3, pp: (114-118), Month: July - September 2016, Available at: www.researchpublish.com

Prediction of Fault-Proneness using CK Metrics

[1]Monika, [2]Preeti Sharma

[1]M.Tech (Computer Science), M.D.U., Rohtak, Haryana, India
[2]Deptt. of Computer Science, M.D.U., Rohtak, Haryana, India

***Abstract*: Many object-oriented metrics were proposed to assess the quality of the software design such as the fault-proneness and the maintainability of classes. Software metrics can serve many purposes for software engineers. Many software metrics have been validated theoretically and empirically as good predictors of quality factors. The object-oriented metrics software provides useful information to developers and managers about the quality and object oriented structure of the design and code, but without interpretation guidelines metrics are of little value. Our main aim to analyze the object oriented metrics is that we should be able to predict the quality attributes of system so that we are able to capture the faults & defects early in the design phase. Many object-oriented metrics proposed in literature lack a theoretical basis, while other has not yet been validated. This work describes how object-oriented metrics given by CK is useful to illustrate fault-proneness of the system. Once faults are detected, then we can easily correct them out and improve the quality and reliability of the software. In this dissertation, we have measured the bugs per class per metric with the help of well known object-oriented CK metrics.**

***Keywords:* object-oriented design, fault prediction, attributes, coupling, cohesion, size, inheritance.**

I. INTRODUCTION

Metric is a standard unit of measurement that quantifies results. Metric used for evaluating the software processes, products and services. The aim to propose these metrics is to provide a way of quantitatively evaluates the quality of an object-oriented software system. Inheritance, coupling, and cohesion have been argued to significantly affect complexity. Exploratory analysis of empirical data is provided to relate the metrics to productivity, rework effort, and design effort . Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability; reusability. Software Metrics have been proposed for procedural and object oriented paradigms to measure various attributes like complexity, cohesion, software quality, and productivity. Among all of these, “Complexity” and “Cohesion” are considered to be the most important attributes. Software metrics can help to fully understand both the design and architecture information of the system.. Software metrics can help to determine the effect of object technology , especially reuse technology applied in the software development according to some quantitative evaluation such as productivity, quality, lead time, maintainability, reusability etc.

Keeping in view the present study will be carried out with the following objectives:

1. Weighted Methods per Class (WMC): This is a weighted sum of all the methods defined in a class.

2. Coupling Between Object classes (CBO): It is a count of the number of other classes to which a given class is coupled and, hence, denotes the dependency of one class on other classes in the design.

3. Depth of the Inheritance Tree (DIT): It is the length of the longest path from a given class to the root class in the inheritance hierarchy.

4. Number of Children (NOC): This is a count of the number of immediate child classes that have inherited from a given class.

In this context, it is also interesting to know if it is better and useful to combine these design attributes together to predict fault proneness, or we can achieve better accuracy and completeness by using them individually. In this paper, we aim to answer the following questions.

1) What is the relationship of existing object-oriented metrics with fault proneness in a given class?

2) What are the important designs attributes of an object oriented software that better correlated with fault proneness.

3) Is it useful and better to combine the design attributes together with each other to predict fault proneness or use them individually, when building prediction models?

To answer these above questions, we performed an empirical investigation by using class level object oriented metrics.

II. RELATED WORK

Object-oriented metrics are helpful in validating the context of fault proneness of software system.These studies investigated the objects relation and their another aspects which is useful in reducing the testing effort. Object Oriented (OO) Metrics is to predict the quality of the object oriented software products. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc. In 1974, Stevens et al. conducted an empirical first defined coupling in the context of structured development as "the measure of the strength of association established by a connection from one module to another. "Coupling is a measure of interdependence of two objects. They concluded that coupling relationship in context of inheritance.Chidamber Karmarer investigated the CK metrics for their relationship with quality artifacts of the software components and concluded that WMC, RFC, CBO metrics were good to predict the quality of software components.[Tibor Gyimo´ thy et al] empirically validated object-oriented metrics for fault prediction by using CK metrics. It concluded that CBO metric is best in predicting fault proneness. LOC metric performed fairly well for quick fault prediction, so suitable for quick fault prediction. [Pradeep Kumar Bhatia et al] conducted an empirical evaluation of object oriented metrics with three different object-oriented features like multiple inheritance, multilevel inheritance and hierarchical inheritance.[Brij Mohan Goel et al] conducted a study for measuring the reusability of object oriented program based upon CK metrics. It concluded that reusability increases with increase of DIT and NOC, reusability decreases with increase of CBO and LCOM, reusability decreases with increase of WMC and RFC.[Ahmed M. Salem et al] conducted a study for procedural and object oriented paradigm to measure complexity and cohension. [Yuming Zhou et al] conducted an empirical validation of CK metrics to investigate the prediction of error proneness in a class. It concluded that CBO, WMC, RFC, and LCOM metrics are statistically significant across fault severity, while DIT is not significant for any fault severity.

[Pradeep Bhatia et al] conducted a study to classify the quality level of object-oriented software systems. Empirical data, collected from different application columns was then analyzed using object-oriented metrics, to support theoretical validation.

Therefore, this paper first, investigates empirically the capability of these metrics in term of predicting fault proneness. Second, we investigate which design attribute is out performed, when examine individually and combination with other attributes.

III. THE SET OF METRICS

CK metrics is one of the oldest and most reliable metrics among all metrics available to software industry to evaluate OO design. The CK and QMOOD suites contain similar components and produce statistical models that are effective in detecting error-prone classes. The class components in the MOOD metrics suite are not good class fault-proneness predictors .

When code is analyzed for object-oriented metrics, often two suites of metrics are used, the Chidamber-Kemerer (CK) and MOOD suites.

Coupling: coupling in the context of structured development as "the measure of the strength of association established by a connection from one module to another. Here we use five measures of coupling in our study. They are : CBO,RFC, CA, CE and DAM.

Cohesion: Cohesion refers to how closely the operations in a class are related to each other. Cohesion of a class is the degree to which the local methods are related to the local instance variables in the class. Cohesion measure is consist three metrics in our study. They are : LCOM, LCOM3 and CAM.

Encapsulation: Information hiding is a way of designing routines such that only a subset of the module's properties, its public interface, is known to users of the module. Information hiding gives rise to encapsulation in object-oriented languages. Here Encapsulation measure consists of two metrics in our study.they are AHF,MHF.

Inheritance: Inheritance decreases complexity by reducing the number of operations and operators, but this abstraction of objects can make maintenance and design difficult. Here inheritance measure consists of five metrics in our study. They are : DIT, NOC, IC, CBM and MFA.

Complexity: A class with more member functions than its peers is considered to be more complex and therefore more error prone. The larger the number of methods in a class, the greater the potential impact on children since children will inherit all the methods defined in a class. Complexity measure here is consist three metrics. They are : WMC, AMC and CC.

IV. LITERATURE REVIEW

Object Oriented Metrics is the key area adapted by the industry so that lot of work is already done by different researchers to identify the software reliability. The work done by the earlier researchers.

[Tibor Gyimo´ thy et al] **Empirical Validation of Object-Oriented Metrics on Open Source Software for Fault Prediction** Open source software systems are becoming increasingly important these days. Many companies are investing in open source projects and lots of them are also using such software in their own work. But, because open source software is often developed with a different management style than the industrial ones, the quality and reliability of the code needs to be studied. The CBO metric seems to be the best in predicting the fault-proneness of classes. The LOC metric performed fairly well and, because it can be easily calculated, it seems to be suitable for quick fault prediction. The correctness of the LCOM metric is good, but its completeness value is low. The DIT metric is untrustworthy and NOC cannot be used at all for fault-proneness prediction

[Pradeep Kumar Bhatia et al] **Analysis of Reusability of Object Oriented Systems using Object Oriented Metrics** Reusability is the key element to reduce the cost and improve the quality of the software. This paper focuses on an empirical evaluation of object oriented metrics in C++ with three different object-oriented features. Three programs have been considered as input for the study – the first program is on multilevel inheritance, the second program is on multiple inheritance and the third program is on hierarchical inheritance. Object oriented metrics have been measured for three C++ programs under the three categories of inheritance, coupling and cohesion. The metrics were then analyzed and used to understand the various characteristics of the systems. Multilevel Inheritance has more impact on reusability.

[Rakesh kumar et al] **A Heuristics Based Review on CK Metrics** Heuristics and metrics are used to improve the quality of software development process and are interrelated. Various attributes, which determine the quality of the software, include maintainability, defect density, fault proneness, normalized rework, understandability, reusability etc.

The common goal of software metrics and heuristics based on them is the production of high quality software, which will in turn increase customer satisfaction. Software metrics are useful for planning, managing and monitoring software projects. The software metrics based heuristics are used to predict fault prone modules in software development projects thereby making the software more reliable, easy to use, maintain and debug.

[Shreya Gupta et al] **Advanced Object Oriented Metrics for Process Measurement**

Process improvement requires measurement of specific attributes of process. It provides effective ways of estimation and evaluation. Then, it is essential to develop a set metrics covering the attributes. Computed measures are used as indicators for process improvement area. The extensions in AIF and MIF are more accurate than previous definitions as they give a better idea about usage of inheritance property in the code. Results are accompanied with analysis part showing the variation in the values. Clearly, classes that have AIF, MIF values greater than threshold value needs some modification in their design. This gives clarity in estimation of actual hiding factors.

[Brij Mohan Goel et al] **Investigation of Reusability Metrics for Object-Oriented Designing** In the object-oriented environment, one of the major aspects having strong influence on the quality of resulting software system is the design complexity. Design complexity has been conjectured to play a strong role in the quality of the resulting software system in OO development environments. The organizations implement systematic software reuse programs in an effort to improve productivity and designing. Reusability increases with increase of DIT and NOC, reusability decreases with increase of CBO and LCOM, reusability decreases with increase of WMC and RFC. Since reusability is an attribute of software design, we can analyze software design by measuring software reusability. Hence, this approach is important to measure reusability of class diagram.

[Chidamber Shyam et al] **Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis** With the increasing use of object-oriented methods in new software development there is a growing need to both document and improve current practice in object-oriented design and development. First, an informal introduction to the metrics is provided by way of an extended example of their managerial use. Second, exploratory analyses of empirical data relating the metrics to productivity, rework effort, and design effort on three commercial object-oriented systems are provided. DIT and NOC tended to have minimal values. WMC, CBO, and RFC tended to be highly correlated. High levels of coupling and lack of cohesion were associated with lower productivity, greater rework, and greater design effort. Results should be of interest to both practitioners and the OO research community as to the degree to which the metrics are of practical significance. In such cases, the low metric values may indicate that appropriate design preferences are being followed. It is important to note that, although generally not able to be demonstrated in the regression models due to their low variability, these two metrics can, in theory, be helpful in highlighting egregious departures from design principles.

[R.Subramanyam et al] **Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects** To produce high quality object-oriented (OO) applications, a strong emphasis on design aspects. Design metrics play an important role in helping developers understand design aspects of software and, hence, improve software quality and developer productivity. Results are based on industry data from software developed in two popular programming languages used in OO development even after controlling for the size of the software.After controlling for size, they find that some of the measures in the CK suite of OO design complexity metrics significantly explain variance in defects. The effects of certain OO design complexity metrics, such as number of methods (WMC), coupling between objects (CBO), and inheritance depth (DIT), on defects were found to differ across the C++ and Java samples in our study.

[Ahmed M. Salem et al] **Analysis of Inconsistencies in Object Oriented Metrics** Software Metrics have been proposed for procedural and object oriented paradigms to measure various attributes like complexity, cohesion, software quality, and productivity. Among all of these, “Complexity” and “Cohesion” are considered to be the most important attributes. The aim is to compare some of the complexity and cohesion metrics and to analyze these metrics and expose their inconsistencies. CK’s cohesion metric doesn’t distinguish between two classes which have different cohesiveness; it shows both classes have same LCOM value. This clearly shows that CK’s is not able to distinguish a value of cohesion between 0 and any other non-zero value or between a class with high cohesion and a class with medium cohesion. The paper also demonstrated the need for refining these metrics and developing new object oriented metrics.

[Yuming Zhou et al] **Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults** The CBO, WMC, RFC, and LCOM metrics are statistically significant across fault severity, while DIT is not significant for any fault severity. NOC is statistically significant with regard to ungraded/low severity faults, but in an inverse direction, i.e., a class with a large NOC means a low fault-proneness. The significance of NOC with regard to high severity faults cannot be tested because of the quasi-complete separation problem. However, the results of machine learning methods indicate that the fault proneness prediction usefulness of NOC with regard to high severity faults is poor and, in this, is similar to that of DIT. When applied to fault-proneness ranking of classes in terms of low severity faults, the logistic regression model based on these metrics outperforms the simple model based on class size. In both cases, however, the prediction usefulness of these metrics with regard to high severity faults is limited.

[Pradeep Bhatia et al] **An Empirical Study for Accessing Quality of OO Code[10]** The set of object-oriented metrics in the study was applied, taking into account the recognized object-oriented features which they were intended to measure: encapsulation, coupling, inheritance and polymorphism. This evaluation is used to classify the quality level of object-oriented software systems. Empirical data, collected from different application columns was then analyzed using object-

oriented metrics, to support theoretical validation. This study provides useful information to designers and developers, especially those dealing with object-oriented software. It can be used to improve software quality by providing alternative designs. This analysis is more suitable for tightly coupled objects. In case of loosely coupled objects, results are not consistent. It is suitable for highly quality code.

V. CONCLUSION

Object-oriented metrics have become an essential part of object technology as well as software engineering. Exploratory analysis of empirical data is provided to relate the metrics to productivity, rework effort, and design effort on three commercial object-oriented systems. The empirical results suggest that the metrics provide significant explanatory power for variations in these economic variables, over and above what provided by traditional measures. We have visualized that DIT metric is the best metric to predict the fault-proneness of classes and it is the most useful to improve the quality and reliability of the design.

REFERENCES

[1] Tibor Gyimo´ thy, Rudolf Ferenc, Istva´n Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE transactions on software engineering, vol. 31, no. 10, pp.897-910, October 2005.

[2] Marcela Genero, Jose Olivas, Fransciso Romero , "Using metrics to predict OO information systems maintainablity", CAISE 2015, 388-401

[3] Rakesh Kumar, Deepali Gupta, "Heuristics based on object oriented metrics", International Journal of Applied Engineering Research, vol. 2, no. 5, pp.393-395, May 2012.

[4] Shreya Gupta, Ratna Sanyal,"Advanced object oriented metrics for process measurement", The Sixth International Conference on Software Engineering Advances, Barcelona, Spain, pp.318-324, October 2011.

[5] Brij Mohan Goel, Pradeep Kumar Bhatia, "Investigation of Reusability Metrics for Object-Oriented Designing", Proceeding of NCETCIT, GVM IT&M, Sonipat, pp. 104-110, May 2014.

[6] Shyam R. Chidamber, Chris F. Kemerer, "A metrics suite for object oriented design", IEEE transactions on software engineering, vol. 20, no. 6, pp.476-493, June 1984.

[7] Chidamber Shyam, Kemerer Chris, Darcy David, "Managerial use of metrics for object-oriented software: an exploratory analysis", IEEE Transactions on software Engineering, vol. 24, no. 8, pp.629-639, August 2008.

[8] Subramanyam, R., Krishnan, M.S., "Empirical analysis of CK metrics for object -oriented design complexity: Implications for software defects", IEEE Transactions on Software Engineering, vol. 29, no. 4, pp.297-310, April 2014.

[9] Ahmed M. Salem, Abrar A. Qureshi, "Analysis of inconsistencies in object oriented metrics", Journal of Software Engineering and Applications, vol. 4, pp.123-128, January 2011.

[10] Yuming Zhou, Hareton Leung, "Empirical analysis of object-oriented design metrics for predicting high and low severity faults", IEEE transactions on software engineering, vol. 32, no. 10, pp.771 – 789, October 2006.

[11] M.Rizwan Jameel Qureshi, Waseem Qureshi, "Evaluation of the design metric to reduce the number of defects in software development", I.J. Information Technology and Computer Science, vol. 4, pp.9-17, 2009.

[12] F. Fioravanti, P. Nesi, "A study on fault-proneness detection of object-oriented systems," Proc. Fifth European Conf. Software Maintenance and Reeng., Florence Univ., Italy, pp. 121-130, March 2001.

[13] Pradeep Bhatia, Yogesh singh, H.L. Verma, "An emperical study for accessing quality of OO code", Ultra Science, vol. 14, no.3, pp.385-400, May 2002.

[14] Jubair J. Al-Ja'afer and Khair Eddin M. Sabri. "Chidamber-Kemerer (CK) and Lorenz-Kidd (LK) metrics to assess java programs", King Abdullah II School for Information Technology, University of Jordan, Jordan.

[15] Bugzilla for Mozilla, http://bugzilla.mozilla.org, 2005.

[16] Brij Mohan Goel, Pradeep Kumar Bhatia, "Analysis of Reusability of Object Oriented Systems using CK metrics", International Journal of Computer Applications (0975 – 8887), vol. 60, no.10, pp. 32-36, December 2012.