ISSN:0975-9646

Object-Oriented Design Metrics to Predict Fault Proneness of Software Applications

Jeenam Chawla
M.Tech (Computer Science)
Amity school of engineering and Technology
Noida, India

Arun Agarwal
Deptt. Of Computer Science
Amity school of engineering and Technology
Noida, India

***Abstract*— Object-Oriented applications strongly emphasis on design aspects during the early stages of software development to get the high quality output. Object-Oriented design metrics play an important role in software engineering to understand design aspects of the software as well as to improve quality, complexity and productivity of the software. Many object-Oriented approaches have been proposed to predict the fault proneness of software metrics. To evaluate the design aspects of an application, software developers use object-oriented attributes such as coupling, cohesion, complexity, inheritance, and size with their corresponding metrics in predicting fault proneness. The goal of of this study is to evaluate software metrics using object oriented approach that are useful for predicting fault-prone classes and to investigate their capability in combination.**

***Index Terms*—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. A large number of object-oriented (OO) metrics had been proposed by the authors. The aim to propose these metrics is to provide a way of quantitatively evaluates the quality of an object-oriented software system. There are several metrics have been existed to measure the design attribute of a given class. Number of authors had been validated these object-oriented metrics in the context of their relationship with fault proneness of software systems. These studies are focused to determine the relationship among the object-oriented and the fault proneness.

The metrics discussed in this paper are based on executable OO design models from which the dynamic behavior of applications can be inferred. The importance of detection and removal of defects prior to customer delivery has received increased attention due to its potential role in influencing customer satisfaction [7] and the overall negative economic implications of shipping defective software products [12]. Design complexity has been conjectured to play a strong role in the quality of the resulting software system in OO development environments [8]. Prior research on software metrics for OO systems suggests that structural properties of software components influence the cognitive complexity for the individuals (e.g., developers, testers) involved in their development [12]. This cognitive complexity is likely to affect other aspects of these components, such as fault-proneness and maintainability [2]. Design complexity in traditional development methods involved the modeling of information flow in the application. Hence, graph-theoretic measures [13] and information-content driven measures [10] were used for representing design complexity. In the OO environment, certain integral design concepts such as inheritance, coupling, and cohesion1 have been argued to significantly affect complexity. The object-oriented metrics proposed by Chidamber and Kemerer [9] and later refined by the same authors [20] can be summarized as follows:

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

Many object-oriented class level metrics have been proposed and empirically validated in the context of the fault proneness of object-oriented (OO) software system. These studies investigated the relationship between OO

metrics and fault proneness that might be helpful to identify the faulty components in the software system to reduce the testing effort.

Basili et al. [6] conducted an empirical validation of CK metrics to investigate the prediction of error proneness in a class. They concluded that CBO, NOC, RFC, DIT, WMC were significantly correlated with error proneness. Chidamber Karmaker [1] 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. Briand et al. [1] explored the relationship between various Couplings, cohesion and inheritance metrics and fault proneness. The study showed that many coupling, cohesion and inheritance measures capture the similar dimensions in data and coupling measures based on method invocation, depth of a class, are appear to be important quality factors, on the other hand cohesion measure does not seem to be significantly correlated with fault prediction.

Emam et al. [11] conducted a study to validate the object oriented metrics by taking 24 metrics from CK, Briand metrics suits and a telecommunication system. They concluded that out of 24 only two of them are useful for fault prediction. Cartwright et al. [13] empirically investigated a subset of CK metrics and showed that those classes that participated in inheritance structure are approximately three times more defect prone than those that did not participated. Gyimothy et al. [18] empirically validated object-oriented metrics on open source system for fault prediction by using CK metrics and concluded that CBO metric is best in predicting fault proneness. LOC metric performed fairly well for quick fault prediction.

Michael et al. [22] investigated the fault prediction in open source software using CK metrics. Their study showed that DIT, NOC metrics is not useful for fault prediction and CBO, RFC along with LOC were the best prediction of fault proneness. Hector M. Olague et al. [20] empirically validated three metric suits (CK, QMOOD, MOOD) to predict fault proneness for highly iterative and agile software development process over multiple releases of software and concluded that CK metrics have been better and more reliable predictor of fault proneness than the MOOD, QMOOD metrics. Shatwani Li [21] investigated the effectiveness of software metrics to
identify error prone classes. They classified software modules into the different severity levels and showed that it is more practical to use the error severity categories than the number of errors to classify the classes.

Zhou et al. [23] investigated the ability of complexity metrics to predict fault prone classes and concluded that LOC WMC are better fault predictors than SDMC AMC metric. Mohamoud et al. [24] empirically compared the three metric suits (Martin, MOOD CK) for fault predicting in package level and suggested that models based on Martin metric are more accurate than those that are based on the MOOD CK suites. Briand [25] investigated the complexity metrics for their capability of predicting faults and shows that LOC and WMC are good for fault prediction. These studies yielded us on mixed results with some studies confirming the predictive capabilities of the metrics and others prompting questions about these metrics. In addition to this, it is also difficult to determine from these studies that which design attribute of the class is better correlated with fault proneness. 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

We use eighteen metrics of coupling, cohesion, inheritance, size and complexity of an object-oriented software system. Since here, we investigated the metrics for their relationship with fault proneness of a given class.
Therefore, we select only those metrics for our study that are available at the class level. To incorporated these metrics in our study, we divided them into five different categories according to their nature of measuring the design attributes of a given class. We divided the metrics into five different categories: coupling metrics, cohesion metrics, inheritance metrics, size metrics and complexity metrics.

1) Coupling metrics: The coupling is defined as the measure of the strength of association established by a connection from one module to another [29]. Here we use five measures of coupling in our study. They are : CBO, RFC, CA, CE and DAM.

2) Cohesion metrics: cohesion is defined as the degree of connectivity among the elements of a module" [29]. A class is cohesive if the association of elements declared in the class is focused on accomplishing a single task. Cohesion measure is consist three metrics in our study.

They are : LCOM, LCOM3 and CAM.

3) Inheritance metrics: Inheritance is a way of reusing the attributes and behavior of existing classes. Here inheritance measure consists of five metrics in our study. They are : DIT, NOC, IC, CBM and MFA.

4) Size metrics: Size of object-oriented software is a direct measure of software and the process by which it is developed. Here we use two size measures in our study : LOC and NPM.

5) Complexity metrics: Complexity is the property of a model which makes it difficult to formulate its overall behavior, even when given reasonably complete information about its atomic components and their inter relations. 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 in the same area is discussed and presented in this research work.

Lionel C. Briand has presented a suit of design measures to predict the software fault in object oriented programs or the software. Author identified the relationship

between different objects under different vectors such as cohesion, coupling, inheritance etc. Author defined a probabilistic approach for performing the fault detection in an object oriented software system. The main objective of author was to study the software product and the quality. Author defined the accurate model so that the fault analysis will be done effectively. Author defined the software analysis under the frequency analysis of the faults respective to the classes and the fault frequency. Author also study the fault proneness in the software system[1].

Erik Arisholm in year 2002. Author also presented the object relationship analysis as the main quality factor in object oriented system. This relationship was presented in the form of coupling. Author also quantifies the system under the static analysis and work with different dimensions. Author also analyzed the actual decencies between the software objects and classes and performs the message tracing so that effective and fair prediction analysis will be performed. Author presented the work for java program[2]. Another work in the fault analysis for the object oriented projects was done by Lionel C Briand in year 2002. Author identify the metric based analysis under the fault proneness. Author defined the fault prediction under the software classes for the particular software product. Author also defined the model evaluation under different tool for the java program. Author also explored the function analysis so that more accurate decision will be drawn respective to the fault proneness. Author studied the fault based analysis under the complex modelling so that more accurate probabilistic analysis will be performed[3]

Cagatay Catal In year 2007 has defined an Artificial Immune system based fault prediction program so that the system reliability and the availability will be identified under differnt software features. Author defined the real time software system analysis under the dependability assessment by taking care of the software requirement. Author also perform the classification of these modules under the fault criticality level as well as based on the fault frequency. In this study author performed the individual module analysis as well as performed the complete system analysis under the data analysis as well as class level analysis. The model presented by the author was inspired from the Immune system adapted in different applications. Author defined the fault prediction modelling at the class level. Author defined the fault prediction model analysis so that the dependability assessment will be done effectively[4].

Oral Alan in year 2009 performed the software management under different dataset with the criticality discovery as well as maintain the software system performance under the fault prediction models. Author defined the fault analysis based model using the metric analysis and the fault based evaluation. Author performed the reliability as well as the performance measurement. Author defined an outlier detection approach so that differ metric analysis can be done effectfully. Author performed the work java based software systems[5].

Cong Jin presented a computation analysis based prediction model for object oriented software so that the software quality will be identified under the software fault analysis and software cost analysis. Author defined a contructing model for the quality attribute prediction with differnt clustering approaches. Author performed the analysis under the FCM and neural network based prediction model so that the effective classification of the software metrics will be done. Author also defined the experimentation on the software system under the quality analysis as well as the fault proneness is measured. Author defined an experimentation for the effectiveness evaluation so that the more accurate system will be constructed[6].

Rachel Burrow in year 2010 defined the external characterstics of the software system under the quality analysis. Coupling metrics analysis is performed so that the fault indicators are optimized under the pivotal quality attributes so that the software assessment will be in an effective way. Author also considered the aspect oriented characterstics while performing the quality metrics analysis[7].

In year 2010, Ana Erika Camargo Cruz presented a fault prediction based study under the UML system so that the fault analysis in the software system will be done. Author defined the prediction model for the logistic regression as well defined the UML metrics analysis for the software codes. Author also performed the code measurement under the linear scaling so that the unit variance will be estimated effect fully. Author defined the prediction model under the normalized code measured so that the fault prediction will be done effectively. The normalized behaviour of the software system with result analysis across the package and projects in same model[8].

In year 2011, Bharavi Mishra presented an object oriented analysis under the defect identification in the software modules. Author defined a testing analysis based approach to reduce the testing efforts and to reduce the software development time. Author also defined the development time analysis under different other factors so that the software reliability can be measured as well a reliable software product will be delivered. Author defined the extensive test case analysis under the software enterprises. Author also defined the attribute selection and analysis in effective way so that the software classification will be done effectively. Author also defined the prediction model so that the feature selection will identify the software reliability[9].

In year 2011, Marshima Mohd Rosli defined an evolutionary algorithm to design a fault effective software application. Author defined the resource utilization approach for the software projects so that more effective time and cost based resource analysis will be performed. Author defined different techniques and application so thtat the software fault analysis will be done. Author performed the work on functional metrics as well as on object oriented metrics so that the module categorization will be identified under the faulty and non faulty software modules. Author defined the module classification under constraint specification and the optimization approach. Here the optimization is done using genetic approach. Author also performed the module separation under the faulty and non

faulty modules. Author defined a genetic based object oriented analysis approach for the UML system. Author presented the design under the automated tool so that the software module anlaysis under the problematic analysis will be done effectively[10].

In year 2012, Raed Shatnawi has defined the quality vector analysis under the software fault analysis. Author defined the prediction model under the performance analysis so that more accurate module analysis will be done. Author defined the work for java based software system. Author presented the testing mechanism under the oversampling system. Author defined the technique so that the effective software analysis will be done[11].

In year 2012, Pradeep Singh presented a fault prediction analysis based study for the open source software system. Author defined the role for the software applications as well as defined a level program so that the software effectiveness will be drawn.

Author presented the quality analysis and modelling for the open source software. Author performed the code based investigation under the characteristics analysis and the nature analysis. Author defined the system under the fault prediction capabilities[12].

Santosh Singh Rathore defined an investigation on the design metrics along with fault analysis for the object oriented software modules. Author defined the software system analysis under the machine learning algorithm so that the better result analysis will be performed.

Author performed work many authenticated datasets and experiments shows the effective validation and isolation of fault faults in these software systems[13]. Another work on the software system under the object oriented metrics analysis was done to perform the fault prediction. Author defined a class level based evaluation under the regression analysis. Author performed the study for different project dataset with different quantitative measured so that the accuracy and the reliability of the system will be estimated[14].

V. CONCLUSION

In this paper we examined a number of oject oriented attributes for predicting software defects proneness described in journals, books, web space, scientific computing. This is the primary contribution of the research. The models have been accessed on five proprietary attributes. Study shows the coupling and complexity are the two main design attributes of an object oriented software system that are more causing the faults in a given class. Along with the coupling and complexity, size is the third strongest designs attribute.

REFERENCES

[1] Lionel C. Briand," Predicting Fault-Prone Classes with Design Measures in Object-Oriented Systems".

[2] Erik Arisholm," Dynamic Coupling Measures for Object-Oriented Software", Proceedings of the Eighth IEEE Symposium on Software Metrics (METRICS.02) 0-7695-1339-5/02 © 2002 IEEE

[3] Lionel C. Briand," Assessing the Applicability of Fault-Proneness Models Across Object-Oriented Software Projects", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING 0098-5589/02@2002 IEEE

[4] Cagatay Catal," An Artificial Immune System Approach for Fault Prediction in Object-Oriented Software", 2nd International Conference on Dependability of Computer Systems(DepCoS-RELCOMEX'07) 0-7695-2850-3/07 © 2007IEEE

[5] Oral Alan," An Outlier Detection Algorithm Based on Object-Oriented Metrics Thresholds", 978-1-4244-5023-7/09©2009 IEEE

[6] Cong Jin," Quality Prediction Model of Object-Oriented Software System using Computational Intelligence", 2009 2nd International Conference on Power Electronics and Intelligent Transportation System 978-1-4244-4543-1/09 ©2009 IEEE

[7] Rachel Burrows," The Impact of Coupling on the Fault-Proneness of Aspect-Oriented Programs: An Empirical Study", 2010 IEEE 21st International Symposium on Software Reliability Engineering 1071-9458/10 © 2010 IEEE

[8] Ana Erika Camargo Cruz," Exploratory Study of a UML Metric for Fault Prediction", ICSE '10, May 2-8 2010, Cape Town, South Africa ACM 978-1-60558-719-6/10/05

[9] Bharavi Mishra," Impact of Attribute Selection on Defect Proneness Prediction in OO Software", International Conference on Computer & Communication Technology (ICCCT)-2011 978-1-4577-1386-611@2011 IEEE

[10] Marshima Mohd Rosli," The Design of a Software Fault Prone Application Using Evolutionary Algorithm", 2011 IEEE Conference on Open Systems (ICOS2011 978-1-61284-931-7/11©2011 IEEE

[11] Raed Shatnawi," Improving Software Fault-Prediction for Imbalanced Data", 2012 International Conference on Innovations in Information Technology (IIT) 978-1-4673-1101-4/12©2012 IEEE

[12] Pradeep Singh," Empirical Investigation of Fault Prediction Capability of Object Oriented Metrics of Open Source Software".

[13] Santosh Singh Rathore," Investigating Object-Oriented Design Metrics to Predict Fault-Proneness of Software Modules".