INTERNATIONAL JOURNAL OF TECHNOLOGY AND COMPUTING (IJTC)
ISSN-2455-099X,
Volume 2, Issue 5 May 2016.

A REVIEW ON SOFTWARE FAULT PREDICTION TECHNIQUE USING DIFFERENT DATASET

Heena Kapila[a], Daljit Kaur[b] , Sachin Majithia[c]

[a] heenakapila26@gmail.com, Assisstant Professor, CSE Department, Chandigarh Engineering College, Landran, Mohali

[b] er.daljitkaur@gmail.com, Assisstant Professor, CSE Department, Chandigarh Engineering College, Landran, Mohali

[c] sachinmajithia@gmail.com ,Assisstant Professor, CSE Department, Chandigarh Engineering College, Landran, Mohali

ABSTRACT

Software testing entails a number of processes that are focused on finding faults within a stipulated time. In this paper, different techniques have been discussed for finding the software faults prediction before the testing process. Numbers of researchers have been worked upon object oriented metrics and mostly concentrating on software fault prediction, very few has been published for bad smells. Bad code smells are used to identify complex classes in object-oriented software systems. Detection of bad code smell helps in refactoring. This review paper contributes to all code smell prediction techniques designed by researchers. The fault prediction model grants assistance during the software development.

Keywords: Bayesian Inference, Bayesian Regularization, Levenberg-Marquardt, Public dataset, Fault Prediction, Software reliability, CK metrics

I. INTRODUCTION

A software bug is an error, flaw, mistake, failure or fault in a computer program that prevents it from behaving as intended (e.g., producing an incorrect result). A software fault is a imperfection that causes software failure in an executable product. In software engineering, the failure of software to its requirements is commonly called a bug. Mainly bugs occur from errors done by programmer in program's source code. Some bugs are caused by compilers producing incorrect code. To increase the effectiveness and to scale down cost, we must know all possible flaws and common identified areas from the time the software application is commenced. We should have a strategy to predict early faults as it helps in evaluation and possible controlling of the same. It is essential for software development and furthermore helps in improving software quality.

II. Software Fault Prediction

Software metrics and fault data belonging to a previous software version are used to build the software fault prediction model for the next release of the software [4]. The development of these software applications is challenging because system engineers have to deal with a large number of quality requirements, dependability and performance. When the software has a defect, it leads to crisis as there is extreme human dependency on it.

Therefore, there is an increasing need for fault free software systems. The Software companies are contributing more concentration to detecting errors in software system. Preventing a software system from having errors is a difficult task. For reducing cost and improving the effectiveness of software, it is important to identify faulty software modules.

There are number of software metrics are proposed to evaluate the quality of software. In the world of object-oriented languages, software metrics are useful in providing developers with added information regarding their software quality. Software metrics can monitor the quality of software whereas Software testers can use metrics to improve the productivity of software testing [6]. Software metrics have been the subject of research over the last three decades, as they play a crucial role in making managerial decisions during the software lifecycle [2]. The Software industry is paying more attention to peremptory Error in any software system is very common and complex problem.

Software reliability can be achieved by using established techniques for Software fault prediction. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process. For preventing software system from faults software developers must know, where errors are likely to occur. Many researchers studied and designed many metrics models.

III. Review process

We have discussed in our review only if software fault prediction or software quality prediction has been investigated by the researcher. We included papers which have experimental results. The inclusion of papers was done on research topics in which similar study have been conducted on fault prediction

IV. Software Metrics

Software metrics are all about measurements which, is used to improve management of that process. Software metrics involve numbers and the use of numbers to make software better. Software metrics are related to all phase of software development life cycle from beginning to maintenance. CK metric suit is most widely used metrics for the object- oriented (OO) software. This study analyzed the CK metrics. Chidamber et al. [5] developed and implemented a new set of software metrics for Object Oriented designs.

They noticed that Object Oriented may hold some of the solutions to the software crisis. Software metrics is all about measurement and these are applicable to all the phases of software development life cycle from initiation to maintenance. Two of widely accepted metrics are CK and MOOD [1] metrics. The IEEE Standard Glossary defines metric as a "Quantitative measure of degree to which a system, component or process possess a given attribute". Metrics are grouped into three main categories [15]

Product metrics are those metrics which are used to measure the software product properties. These metrics are used to calculate its documentation, design and performance, regardless of its development stage.

Process metrics emphasize on the software development process. These metrics calculated software development time, methodology used and quality assurance techniques.

Resource metrics give emphasis to human, hardware and software resources such as developer skill level, hardware dependability, software component quality.

Metrics are designed to improve software quality by easier detection of possible error or bad design

A. Traditional Metrics

There are many metrics that are applied to traditional functional development [8]. Three metrics have been found relevant by the Assurance Technology Center (SATC) regarding object oriented development i.e complexity, size and readability.

Cyclomatic Complexity (CC): The complexity of an algorithm in a method is evaluated by cyclomatic complexity (McCabe). It is a count of the number of test cases that are needed to test the method comprehensively.

Size: Developers understanding for code can evaluated by size of a class. These include counting all physical lines of code, the number of statements, the number of blank lines, and the number of comment lines. Lines of Code (LOC) count all lines.

Comment Percentage: The line counts done to compute the Size metric can be expanded to include a count of the number of comments, both on-line (with code) and stand-alone. The comment percentage [8] is calculated by the total number of comments divided by the total lines of code less the number of blank lines. Table 1 presents an overview of the metrics applied by the SATC (Software Assurance Technology Center) [15] for object oriented systems. The SATC supports the continued use of traditional metrics, but within the structures and confines of object oriented systems. The first three metrics in Table 1 are examples of traditional metrics applied to the object oriented structure of methods instead of functions or procedures. The next six metrics are specifically for object oriented systems and the object oriented construct applicable is indicated.

Table 1: SATC Metrics for Object Oriented Systems

SOURCE	METRIC	OBJECT-ORIENTED CONSTRUCT
Traditional	Cyclomatic complexity (CC)	Method
Traditional	Lines of Code (LOC)	Method
Traditional	Comment percentage (CP)	Method
NEW Object-Oriented	Weighted methods per class (WMC)	Class/Method
NEW Object-Oriented	Response for a class (RFC)	Class/Message
NEW Object-Oriented	Lack of cohesion of methods (LCOM)	Class/Cohesion
NEW Object-Oriented	Coupling between objects (CBO)	Coupling
NEW Object-Oriented	Depth of inheritance tree (DIT)	Inheritance
NEW Object-Oriented	Number of children (NOC)	Inheritance

B. MOOD (Metrics for Object Oriented Design)

The MOOD (Metrics for Object Oriented Design) [1] metrics evaluated that how OO design mechanisms like inheritance, polymorphism, information hiding and coupling can make an influence on quality characteristics like defect density and maintainability. The MOOD set includes the following metrics: Method Inheritance Factor (MIF), Attribute Inheritance Factor (AIF),Coupling Factor (CF),Polymorphism Factor (PF), Method Hiding Factor (MHF)

Attribute Hiding Factor (AHF) Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) Method Inheritance Factor (MIF) and Attribute Inheritance Factor (AIF) are proposed to measure inheritance. MIF is defined [8] as the ratio of the sum of the inherited methods in all classes of the system under consideration to the total number of available methods for all classes. AIF is defined as the ratio of the sum of inherited attributes in all classes of the system under consideration to the total number of available attributes (locally defined plus inherited) for all classes.

Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF): The Method Hiding Factor (MHF) and Attribute Hiding Factor (AHF) were proposed together as measure of encapsulation. A class that has only private data members may not necessarily be unified. On the other hand, a fully unified class may contain only visible (public) data. MHF and AHF [16] are measures of the visibility of the properties of a class. MHF [8] is defined as the ratio of the sum of the invisibilities of all methods defined in all classes to the total number of methods defined in the system under consideration. The invisibility of a method is the percentage of the total classes from which this method is not visible.

Coupling Factor (CF): CF is defined as the ratio of the maximum possible number of couplings in the system to the actual number of couplings not imputable to inheritance.

C. Chidamber & Kemerer's Metrics Suite

The following are the CK metrics used in the classification process:

- Coupling between Objects (CBO)
- Lack of Cohesion (LCOM)
- Number of Children (NOC)
- Depth of inheritance (DIT)
- Weighted Methods per Class (WMC)
- Response for a class (RFC)

Coupling between Object Classes" (CBO) for a class is a count of the number of other classes to which it is coupled. It is measured by counting the number of distinct non-inheritance related class hierarchies on which a class depends [15]. The more independent a class is, the easier it is reuse in another application. The larger the number of couples, the higher the sensitivity to changes in other parts of the design and therefore maintenance is more difficult. Theoretical basis of CBO relates to the notion that an object is coupled to another object if one of them acts on the other. As Coupling between Object classes increases, reusability decreases and it becomes harder to modify and test the software system [11].

The high value of Lack of Cohesion in Methods (LCOM-CK) as originally proposed by Chidamber and Kemerer [5] indicates that the methods in the class are not really related to each other and vice versa. Lack of Cohesion (LCOM) measures the distinction of methods in a class by instance variable or attributes. A high LCOM-CK value indicates disparateness in the functionality identify classes that are attempting to achieve many different objectives, and consequently are likely to behave in less predictable ways than classes that have lower LCOM values. Such classes could be more error prone and more difficult to test. Classes could possibly be disaggregated into two or more classes that are well defined in their behaviour. A highly cohesive module should stand alone; high cohesion indicates good class subdivision [15]. Lack of cohesion or low cohesion increases complexity, thereby increasing the likelihood of errors during the development process. High cohesion implies simplicity and high reusability. High cohesion indicates good class subdivision. Lack of cohesion or low cohesion increases complexity, thereby increasing the likelihood of errors during the development process. It means that low value of LCOM depicts high internal strength of the class which results into high reusability. So, there should be some maximum value of LCOM after that class becomes non-reusable.

Number of children (NOC) of a class is the number of immediate sub-classes subordinated to a class in the class hierarchy. In inherit tree, count the number of direct subclass of a class. The bigger NOC is, the more affecting the class is. And therefore they should be paid substantial attention when being tested. The more children, the better the class is in reusability. Although the more reusability the better, its super class's

abstraction may become weaker. The number of subclasses creates some potential influence on the super class. The super class should put more rigorous checks for those subclasses.

Depth of Inheritance Tree (DIT): According to this metric Depth of inheritance of a class is “the maximum length from the node to the root of the tree". More is the depth of the inheritance tree greater the reusability of the class corresponding to the root of that tree as the class properties are shared by more derived classes under that class. Greater depth dilutes the abstraction and there is a need to set the minimum and maximum DIT value for a class as a contribution towards the reusability. The definition of DIT is ambiguous when multiple inheritance and multiple roots are present as the alternative length of the path is not being considered in case of multiple inheritance. If all the ancestor classes coming in common path are added to the ancestor classes of alternative paths then that will be the true representation of the theoretical basis of the DIT metric.

The Weighted Method (WMC) is a count of the methods implemented within a class or the sum of the complexities of the methods (method complexity is measured by cyclomatic complexity) [15]. The larger the number of methods in a class, the greater the potential impact on children; children inherit all of the methods defined in the parent class. The number of methods and the complexity of the methods involved is a predictor of how much time and effort is required to develop and maintain the class.

kapila et. al [9][10] have been used CK metrics for research. The CK suite of metrics database was prepared with help of Analyst4j and computed each metric of its jdt core package. Analyst4j is based on the Eclipse platform and available as a stand-alone rich client application or as an Eclipse IDE plug-in. In this research Analyst4j used as a plug-in for Eclipse to detect anti-patterns. This study employs Analyst4j for the identification of code smells. Whenever a smell is detected, the respective class is added to a set of suspicious smelly class. It is reasonable that the result of a detection strategy is re-evaluated by an engineer to verify the flaw. Detection strategy provides an overview of possible risk areas in the system. Each Detection Strategy is structured in two consecutive elements:

1) A set of code metrics.

2) A set of filtering rules, one rule for the interpretation of each metric result.

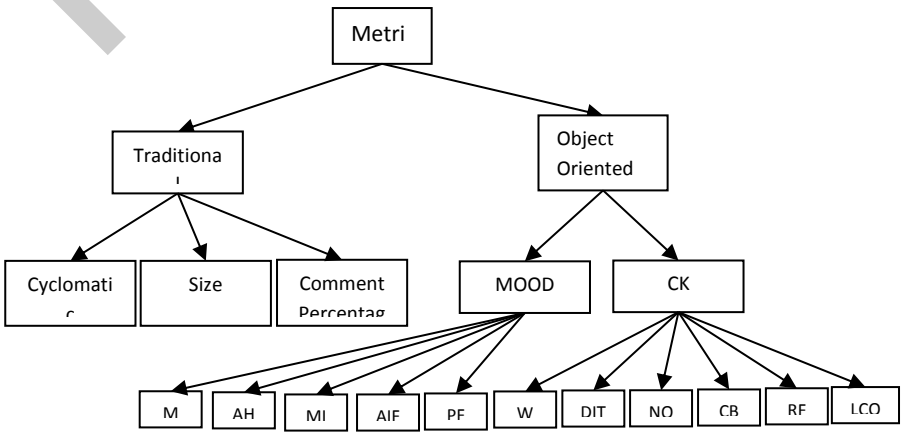

Fig. 1: Metrics Hierarchy

Mahajan et. al [14] used public dataset in research. Public dataset is that dataset which are frequently situated in Promise repositories and they are distributed freely. Mahajan et. al [14] used Ant-1.7 Public dataset. Dataset which comes from Promise repository. Ant 1.7 dataset uses class level metrics.

V. Fault Prediction Techniques

Software fault prediction has been topics of many studies .To predict the fault in software data a variety of techniques have been proposed which includes statistical method, machine learning methods and neural network techniques. These techniques are explained below:

A. Statistical methods

Statistical methods are used to find an explicit numerical formula, which determines completely how classification is performed. Kapila et. al [10] followed two statistical approaches to perform their study. Logistic Regression in which two different types of variable used to perform this experiment .one is binary dependent variable, which representing classes are erroneous or not. Logistic regression model is used to study the association between bad smell and prediction for software failure. Univariate binary logistic regression (UBR) is very useful for analyzing that data which includes binary variable.

In Bayesian inference [10] model design to relate object-oriented software metrics to software fault content and fault proneness. A Bayesian network represents a joint probability distribution over a set of stochastic variables, either discrete or continuous. Bayesian inference model is a framework for constructing posterior data by combining prior knowledge with evidence. Bayesian inference model is design on basis of Bayes' theorem. According to Bayes' theorem [10], probability theory is affected by evidence. Bayesian approach offers more intuitive and meaningful inferences to the data.

In this research [10], the analysis performed to define relationship between the bad smell and software metrics. Regression analysis technique is widely used to predict the bad smell within the code. Linear regression is used when there are only two categories of dependent variable.

The Logistic regression is used to analyze the results of the association. This technique was applied on version of Eclipse. Regression [9] is a statistical procedure which attempts to predict the values of a given variable, based on the values of one or more other variables. The result of a regression is usually a model which summarizes the relationship between the dependent and independent variable(s). Typically, the model is accompanied by summary statistics describing how well the model fits the data, the amount of variation in the outcome accounted for by the model, and a basis for comparing the existing model to other similar models. Numerous forms of regression have been developed to predict the values of a wide variety of outcome measures. Logistic Regression [9] is a particularly important technique not only because it provides a method of modelling these data, but it is also central to understanding the wider application of the generalized linear model to multi-category ordered and unordered data principally through the use of the proportional odds and multi-nominal logistic regression models

B. Machine learning:

Machine Learning is concerned with the design and development of algorithms and techniques to extract rules and patterns out of massive data sets. Neural networks which have been already applied in software engineering applications, to build reliability growth models predict the gross change or reusability metrics. A neural network is trained to reproduce a given set of correct classification examples, instead to produce formulas or rules.

According to [14] Mahajan et. al machine learning method is best method for finding the software faults because all the work is done by neural network based machine. Neural network is a machine learning approach and made up of number of artificial neurons. Each neuron in Neural Network receives a number of inputs and produces only one output. This technique consume less memory space and provide better accuracy. They have used Bayesian Regularization (BR) technique for software fault prediction. This technique has already been used for cost estimation. They have done comparison of their study with other Levenberg-Marquardt (LM) and Back Propagation Algorithm (BPA) technique.

Norman E.Fenton [6] investigated that software metrics and statistical models have been designed to search the number of defects in the software system. Also large number of prediction models utilize size and complexity metrics to find faults. A large complex multivibrate statistical model has been introduced to find a single complexity metrics.The limitations are that by using size and complexity metrics, accessible models cannot find the faults successfully.

Radial Basis Function (RBF) [12] is an important technique which is provided by neural networks and is substantially used. The main purpose of RBF is to discover the faults in the software and provide better precision. For the purpose of predicting the number of faults in the software[12,13] object oriented software systems are used. The important features of object oriented systems are inheritance and Polymorphism Dataset should be large for finding software faults.

Multilayer Perceptron (MLP) is used for ruling the defective modules and Radial Basis functions Network are used to categorized the defects on the basis of different types of faults [8]. Xing et al. 16 describes the importance of Support Vector Machine (SVM) model. For small dataset SVM model can be used. Data categorization is a significant use of SVM technique. Data categorization is a significant use of SVM technique. SVM provides better accuracy than other techniques for the prediction of quality of the software but in public datasets, the performance of SVM is poor software but in public datasets, the performance of SVM is poor.

VI. Conclusion

In this paper, we studied various techniques like Bayesian analysis based on logistic regression and Bayesian Regularization to predict faults in software systems. The main aim is to examine the performance of different techniques in software fault prediction. Fault prediction using these techniques helps in improving the quality of the software. The fault prediction in software is significant because it can help in directing test

effort, reducing cost, and increasing quality of software. Time and cost effective software development are important for today's developers. Bad code smells are strong indication of poor source code structure. With help of Logistic regression and Bayesian analysis, the probability of occurrence or non occurrence of smelly classes can be detected. An empirical study is carried out to find the association between the metrics and bad smells. Bayesian Regularization algorithm can be used to predict the faults during design phase. Bayesian algorithm having better result than Levenberg-Marquardt(LM) algorithm and Back propagation(BPA) algorithm. As a result, it is found that more algorithm and techniques may be used to increase the accuracy for fault prediction.

REFERENCES

[1] Abreu FB, and Rogério Carapuça, "Object-oriented software engineering: Measuring and controlling the development process." proceedings of the 4th International Conference on Software Quality, Vol. 186, 1994.

[2] Anscombe J.F, "Bayesian statistics." The American Statistician, Vol 15, No. 1,pp 21-24,1961

[3]C. Catal and B. Diri, "A systematic review of software fault prediction studies," Expert system with Applications, vol. 36, no. 4, pp. 7346-7354, 2009.

[4]Catal C., Sevim U., and Diri B., Member, IAENG "Software Fault Prediction of Unlabeled Program Modules" Proceedings of the World Congress on Engineering, Vol I WCE 2009, July 1 - 3,London, U.K

[5] Chidamber Shyam R. and Kemerer Chris F., "A Metrics Suite for Object Oriented Design," IEEE Transactions On Software Engineering, Vol. 20, No. 6, JUNE 1994.

[6]Fenton N.E., Neil M., "A Critique of Software Defect Prediction Models", IEEE Transactions on Software Engineering 25 (5), 675–689,1999.

[7] Gyimo´thy T., Ferenc R., and Siket I.," Empirical Validation of Object- Oriented Metricson Open Source Software for Fault Prediction", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, Vol. 31, No. 10, 2005.

[8]Jamali Mohsen Seyyed, "Object oriented metrics", A survey approach Technical report, Department of Computer Engineering, Sharif University of Technology, Tehran, 2006.

[9] Kapila, Heena, and Satwinder Singh. "Analysis of CK metrics to predict software fault-proneness using bayesian inference." International Journal of Computer Applications 74.2 (2013).

[10]Kapila, Heena, and Satwinder Singh. "Bayesian Inference to Predict Smelly classes Probability in Open Source Software." International Journal of Current Engineering and Technology 4.3 (2014): 1724-1728.

[11]Kumar Rakesh and Gupta Deepali" Heuristic based on OO metrics", International Journal of Emerging Technology and Advanced Engineering, May 2012.

[12] Mahaweerawat A., Sophasathit P., Lursinsap, C. Software fault p rediction using fuzzy clustering and radial basis function network .In proceedings of International conference on intelligent technologies, 2002;304-313.

[13] Mahaweerawat, A., Sophatsathit, P., Lursinsap, C., &Musilek, P. Fault prediction in object -oriented software using neural network techniques. Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, 2004; 1-8

[14] R Mahajan , S.K Gupta and RK Bedi "Design of Software fault prediction model using BR Technique" International Conference on Information and Communication Technologies, vol. 46, P. 849-858, 2015

[15] Rosenberg LH.," Applying and Interpreting Object Oriented Metrics", NASA, SATC.

[16]Shatnawi Raed, Li W., Swain J.," Finding software metrics threshold values using ROC curves", Journal of Software Maintenance and Evolution: Research and Practice, Vol 22,No. 1,pp 1–16,2010

[17]Xing, F., Guo, P., &Lyu, M. R.A novel method for early software quality prediction based on support vector machine.16th IEEE international symposium on Software Reliability, Vol. 37(6), 4537-4543