International Journal of Computer Applications (0975 – 8887)
Volume 74– No.2, July 2013

Analysis of CK Metrics to predict Software Fault-Proneness using Bayesian Inference

Heena Kapila
Asstt. Professor
Department of Information Technology
Chandigarh Engg. College, Landran -140307

Satwinder singh
Asstt. Professor,
Dept. Computer science & Info Tech.,
B.B.S.B. Engg. College, Fatehgarh Sahib-140407

ABSTRACT

The fault prediction model grants assistance during the software development by providing recourse to the present faults with the Bayesian Interference. All faults prediction techniques get a help in this study with the designing of Logistic regression model and Bayesian inference altogether. It is also told as fact that Bayesian inference graph can be represented for probabilistic approach for the faults both presented and identified for the upcoming release. For Probabilistic reliability analysis, Bayesian inference is intended to be evaluated for risk related data. These findings suggest that there is a relationship between faulty classes and object-oriented metrics. This study demonstrates as the performance evaluation technique for any piece of software. We examine the open source Eclipse system, which has a strong industrial usage. The focus of the study is to design Bayesian Inference graph and predict faults for next piece of software.

Keywords
Bayesian Inference, Fault Prediction, Software reliability, CK metrics

1. INTRODUCTION

Object oriented metrics were proposed to evaluate the quality of software such as the fault-proneness, reliability and maintainability of software. In the world of object-oriented languages, software metrics has been used for many years to provide developers with additional information about their software quality. Software metrics can monitor the quality of software whereas Software testers can use metrics to improve the productivity of software testing [15]. 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 [3]. Such information can give developers indications about where bad smell may affect the software and these metrics are very useful in simplifying the testing process by focusing the programmer's attention on effective parts of the program. This Information helps to reduce the programmer's effort, and may provide great overall benefit.

Software has become an integral part of most of the application domains including medical applications, power plants and air traffic control. The development of these software applications is challenging because system engineers have to deal with a large number of quality requirements and testing phases. The introduction of software testing processes to identify software faults within time period is important since corrective maintenance costs increases exponentially if faults are detected later in the software development life cycle [18]. The Software industry is paying more attention to peremptory Error in any software system is very common and complex problem.

Software fault prediction is topic of main concern. Preventing a software system from errors is such a difficult task. There are always some changes that occur in Object oriented software system design in continuous form. For cost reduction and improving the effectiveness of software, it is very important to identify the faulty module's software. In this study main focus is to obtain relation between software metric and faulty module. A software engineer should always plan the changes for software design. Although it is very difficult to choose the best method for design but in our studies experimental results show that the proposed model can establish the relation between software metrics and modules fault-proneness. A software fault prediction is a proven technique in achieving high software reliability. Software reliability can also be defined as the probability of failure-free software operation for a specified period of time in a specified environment. Prediction of fault-prone modules provides one way to support software quality. Quality of software is increasingly important for software. For improving quality we should provide more focus on testing for those portions of code which have largest number of faults. This study is an attempt to predict fault in software by applying different techniques. However, this process requires familiarity with some statistical models or machine learning methods [15].

2. RELATED LITERATURE

Many researchers have carried out significant work in the area of fault prediction. The literature survey is carried out from the designing of CK metrics to explore different techniques used for the modeling of fault prediction. CK metric suit is most widely used metrics for the object- oriented (OO) software. Chidamber et al. [7] developed and implemented a new set of software metrics for Object Oriented designs. They noticed that noted that Object Oriented may hold some of the solutions to the software crisis. These metrics were based on measurement theory and also reflect the viewpoints of experienced OO software developers. In evaluating these metrics against a set of standard criteria, they suggest some ways in which the OO approach may differ in terms of desirable or necessary design features from more traditional approaches. These metrics can help in selecting one that is most appropriate to the goals of the organization, such as reducing the cost of development, testing and maintenance over the life of the application. The study is aimed at examining the relationships between these metrics and cost, quality, and productivity. The results show that high values of CBO and LCOM are associated with lower productivity, greater rework and greater design effort. The main idea is to use measurement to improve the process of software development. Basili et al. [8] investigated the suite of object-oriented (OO) design metrics introduced by Chidamber. They define hypotheses for each metric that represented the expected connection between the metrics and the fault-proneness of the code. They tested these hypotheses and found that some of the metrics were very good predictors, while others were not. Basili has reported on the results of using the CK metrics suite to predict the quality (fault proneness) of student C++ programs. They collected data about faults from object oriented classes. Based on these data,

they verified how much fault-proneness is influenced by internal (e.g., size, cohesion) and external (e.g., coupling) design characteristics of OO classes. From their results, five out of the six CK OO metrics appear to be useful to predict class fault-proneness during the high- and low-level design phases of the life-cycle. They represented that testing of large systems is an example of a resource and time-consuming activity. Therefore, one needs to be able to identify fault-prone modules so that testing / verification effort can be concentrated on these modules They used the same Chidamber and Kemerer metrics suite and analyzed the distribution of the metrics and also the correlations between them. They used logistic regression, a standard technique to analyze the relationships between metrics and the fault proneness of classes.

R.Bender [14] Proposed a method for quantitative risk assessment and investigated about threshold value. By defining acceptable levels for the absolute risk and the risk gradient the corresponding benchmark values of the risk factor can be designed. These values can be designed by means of nonlinear functions of the logistic regression coefficients. The proposed method is implemented on medical patients data to find out value of an acceptable risk level (VARL) and value of an acceptable risk gradient (VARG).These values are used to predict about the probabilities risk level.

R.Subramanyam [6] validated the WMC, CBO, and DIT metrics as predictors of the error counts in a class. Their results indicated that the CK metrics could predict error counts. Subramanyam and Krishnan chose a large e-commerce application developed in C++ and Java. They examined the effect of the size along with the WMC, CBO, and DIT values on the faults by using multivariate regression analysis. Besides validating the usefulness of metrics, they compared the applicability of the metrics in different languages; thus, they validated their hypotheses for C++ and Java classes separately. They concluded that the size was a good predictor in both languages, but WMC and CBO could be validated only for C++. Alshayeb and Li [12] conducted a study on the relationship between some OO metrics and the changes in the source code in two client–server systems and three Java Development Kit (JDK) releases. Three of the CK metrics (WMC, DIT, and LCOM) and three of the Li metrics (NLM, CTA, and CTM)[6] were validated. They found that the OO metrics were effective to predict design effort and source lines of code added, changed, and deleted in short-cycled agile process (client–server systems); however, the metrics were ineffective predictors of those variables in long-cycled framework evolution process (JDK). TiborGyimoet. Al [16] illustrated fault-proneness detection of the source code of the open source Web. For fault proneness detection they used regression and machine learning methods to validate the usefulness of these metrics for fault-proneness prediction. They checked the values obtained against the number of bugs found in its bug database called Bugzilla. They found CBO and LOC metric seems to be the best in predicting the fault-proneness of classes but DIT metric is untrustworthy and NOC cannot be used at all for fault-proneness prediction. The LOC metric performed fairly well and because it can be easily calculated, it seems to be suitable for quick fault prediction.
C.catalet .al [4] proposed a fully automated technique which does not require an expert during the prediction process. They used X-means clustering with software metric threshold. Experiments revealed that unsupervised software fault prediction can be fully automated and effective results can be produced. Raed Shatnawi [15] introduced methodology to produced threshold values with better classification accuracy. Threshold values provide a meaningful interpretation for metrics and provide a surrogate to identify classes at risk. The classes that exceed a threshold value can be selected for more testing to improve their internal quality, which increases the testing efficiency. Hence, we have assessed the effectiveness of a statistical methodology to identify threshold values for the OO metrics. This methodology can be used to identify threshold values based upon the logistic regression model. Raed Shatnawi et. al [1] design threshold values of software metrics using receiver operating characteristic curves. They classified each error into one of seven severity categories. They used area under curve to find out threshold values for each metrics. The metrics we investigated were the following:

3. DATA COLLECTION

We define six metric that we investigated. All of these were first presented by Chidamber and Kemerer [7] but Basili et al. [8] modified these metrics to reflect the special features of the Object Oriented language. In this study the metrics CBO(Coupling Between Object classes),WMC (Weighted Methods per Class), RFC (Response For a Class), NOC (Number Of Children), LCOM (Lack of Cohesion on Methods), DIT (Depth of Inheritance Tree) were investigated. Tool which is used to collect database for implementing this study is Analyst4j tool, which help us to collect all CK metrics value and find out the bad smells presented in software code of Eclipse 3.4. After collecting and preparing the database (shown in Fig 1.), evaluation was done with Matlab. Analyst4j measures these metrics, which forms the source for analysis. Calculated values for CK metrics, is divided into two categories depend upon one binary variable. Binary variable marked as 1 if there was at least one bad smell present in the code that classes or 0(no bad smell found). Bad smells are considered a violation against software engineering principles. The binary categorization used to classifies classes into either error or non error category. The bad smells checked for include: blob classes, Spaghetti code, High risk function, complex classes, complex and undocumented code, Swiss knife classes.

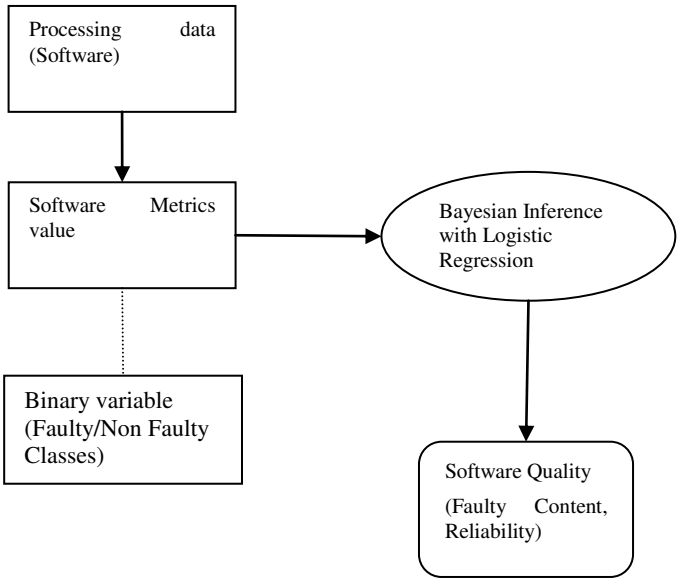

Figure 1. Model Assessment Framework

4. LOGISTIC REGRESSION

In logistic regression, the unknown variable, called the binary variable, can take only two different values. Therefore, we divided the classes into two groups according to whether a class contained at least one smell or not. Regression analysis was used to analyze the results of collected data. The 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 the data which includes binary variable.

The UBR model is as follows

$$\pi.x = \frac{e^{g(x)}}{1 + e^{g(x)}}$$

Where $g(x) = \alpha + \beta * x$ is logit function and π represents probability of a class being faulty, whereas x is object oriented metric. Logistic regression provides models were applied with a threshold value of 0.5, which means that, if $0.5 < \pi$, the class is classified as faulty, otherwise, as not faulty.

5. BAYESIAN INFERENCE

In this study Bayesian inference model is design to relate Object-Oriented software metrics to software fault content and fault proneness. Bayesian Inference represents a joint probability distribution over a set of variables, which are either discrete or continuous. Bayesian inference model is a framework for constructing posterior data by combining prior knowledge with evidence. Bayesian inference model is designed on the basis of Bayes theorem. According to Bayes theorem, Probability theory is affected by evidence. Bayesian approach offers more intuitive and meaningful inferences to the data. Bayes Theorem is as follows:

$$P(X|Y) = \frac{P(Y|X)P(X)}{P(Y)}$$

Whereas $P(Y|X)$ represents Likelihood, $P(X)$ represents Prior distribution and $P(Y)$ is known as marginal likelihood or Prior predictive distribution. Generic formulation is implemented to univariate problems with binomial distributions. Bayesian inference tends to become computationally intensive when the analysis involves multiple parameters and correspondingly high-dimensional integration

6. RESULT

In the proposed model, in order to analyze the impact of software faults, the values of the metrics from Eclipse are considered. Table 1. represents Bayesian Inference graph result with Prior and Posterior values. This table includes all statistical information about Bayesian Inference graph for CBO and NOC; we found a large difference in both graphs. In this research paper out of six metrics only two are presented. In result X axis represented the probability of fault occurrence. Whereas 0 represented as fault free classes and as graph move towards 1 probability of fault occurrence will increase.

The shape of the likelihood function represents the amount of information contained in the data. If the information it contains is small, the likelihood function will be largely distributed, whereas if the information it contains is large, the likelihood function will be closely focused around some particular value of the parameter. The representation of the likelihood is flat in Bayesian graph relative to the prior; it has little effect on the level of knowledge. In case the prior and likelihood have similar shapes so the posterior distribution is not greatly influenced by the prior knowledge. In Bayesian Inference graph if Posterior and Likelihood having higher peak than prior, data is greatly influenced by prior data. The posterior is a compromise between the information (likelihood) and prior. The posterior is more precise since we are combining two sources of information.

As shown in Figure 2. We observed for CBO metric high posterior density covering 90% of posterior Inference. It has posterior probability value 0.383. For prior probability value is same as 0.38, but that value observed only for 62 % classes. As represented in figure 2. Likelihood is very tightly across and high peak. It represented that graph contain large information. The shape of Posterior have high peak rather than Prior and likelihood, it represented that future release of tested software would be more fault free. As shown in Figure 3. for NOC metric high posterior density covering 94% of posterior Inference. It has posterior probability value 0.34 and prior probability value is 0.40 as shown in Table 2. For NOC, Likelihood largely distributed in graph which represent that information is in small amount.

7. CONCLUSION

Our objective in this research is to find empirical evidence of the association between the bad smells and class error probability. From this study we design Bayesian inference for individual metrics which provide posterior probability for fault occurrence. Bayesian graph design for the early prediction of software fault is presented in this paper. The model is based on software metrics and posterior probability. Total six metrics have calculated and using Bayesian inference system, predict probability of faults for next piece of software. In this paper Bayesian Inference graph for CBO and NOC are presented. The Bayesian Inference model is to identify posterior probability; further this study can help to identify threshold values of software metrics using receiver operating characteristic curves. We plan to conduct more studies on open-source systems to find out threshold values in identifying the faulty classes. For software developer, the model provides a methodology for assigning the resources for developing reliable and cost-effective software.

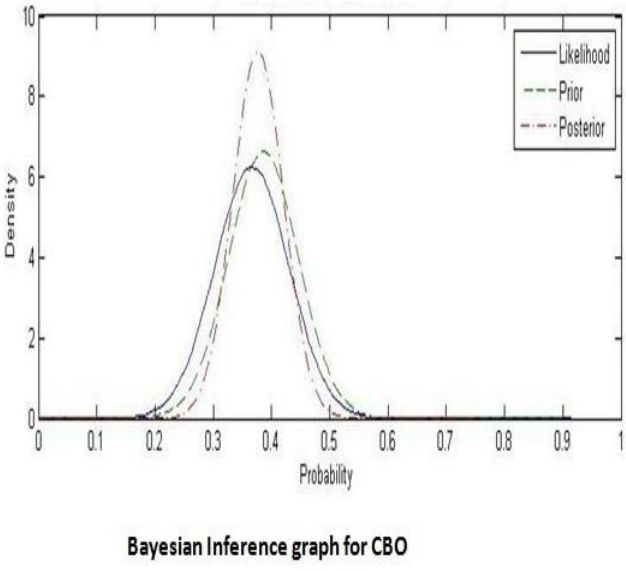

Figure 2: Bayesian Inference for representing Likelihood, Prior and posterior for CBO metric of Eclipse

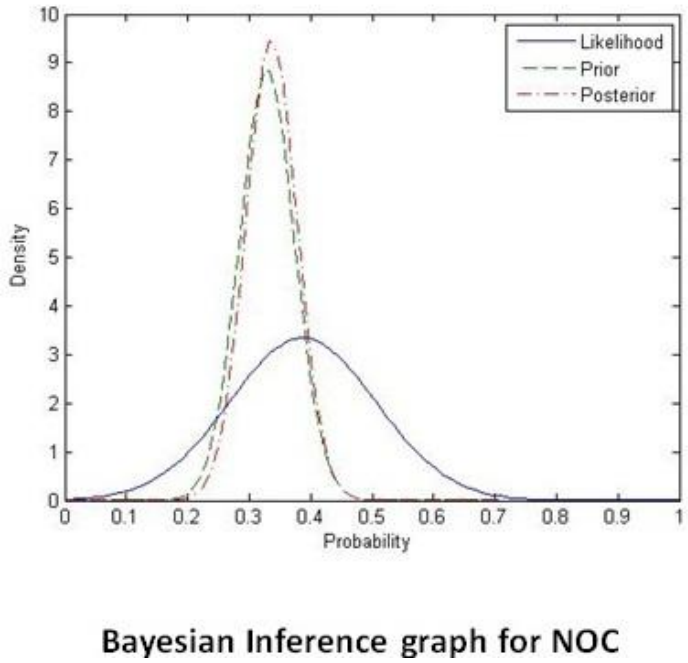

Figure 3: Bayesian Inference for representing Likelihood, Prior and posterior for NOC metric of Eclipse

Table 1: Bayesian Inference graph result CK metrics with Prior and Posterior values

Metric	Posterior value	Percentile coverage for Posterior	Prior value	Percentile coverage for Prior	Posterior Range	Prior range
CBO	0.383	90%	0.3838	62%	0.25-0.52	0.17-0.55
NOC	0.34	94%	0.40	33%	0.22-0.47	0.08-0.70

7. REFERENCES

[1] Raed Shatnawi1, Wei Li, James Swain," Finding software metrics threshold values using ROC curves" , JOURNAL OF SOFTWARE MAINTENANCE AND EVOLUTION: RESEARCH AND PRACTICE, Evol.: Res. Pract. 2010; 22:1–16

[2] Ganesh J. Pai, Member, IEEE, and Joanne Bechta Dugan ," Empirical Analysis of Software Fault Content and Fault Proneness Using Bayesian Methods",IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 33, NO. 10, OCTOBER 2007

[3] Satwinder Singh and K.S. Kahlon ," Effectiveness of Encapsulation and Object-oriented Metrics to Refactor Code and Identify Error Prone Classes using Bad Smells", ACM SIGSOFT Software Engineering Notes Volume 36, Number 5,September 2011

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

[5] Norman Fenton , Martin Neil , William Marsh , Peter Hearty , David Marquez , Paul Krause , Rajat Mishra,"Predicting software defects in varying development lifecycles using Bayesian nets", Information and Software Technology 49 (2007) 32–43

[6] Ramanath Subramanyam and M.S. Krishnan,"Empirical Analysis of CK Metrics for Object-Oriented Design Complexity: Implications for Software Defects" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 4, APRIL 2003

[7] Shyam R. Chidamber and Chris F. Kemerer, "A Metrics Suite for Object Oriented Design," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 20, NO. 6, JUNE 1994.

[8] V.R. Basili, L.C. Briand, and W.L. Melo, "A Validation of Object- Oriented Design Metrics as Quality Indicators," IEEE Trans.Software Eng., vol. 22, no. 10, pp. 751-761, Oct. 1996.

[9]. T.R. Gopalakrishnan Nair, R. Selvarani," Defect proneness estimation and feedback approach for software design quality improvement", Information and Software Technology 54 (2012) 274–285

[10] Shyam R. Chidamber, David P. Darcy, and Chris F. Kemerer, "Managerial Use of Metrics for Object-Oriented Software: An Exploratory Analysis," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 24, NO. 8, AUGUST 1998

[11] Hector M. Olague, Letha H. Etzkorn, Senior Member, IEEE, Sampson Gholston, and cxStephen 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"

[12] Mohammad Alshayeb, Member and Wei Li,"An Empirical Validation of Object-Oriented Metrics in Two Different Iterative Software Processes" , IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 29, NO. 11, NOVEMBER 2003

[13] Jiang, Y., Cukic, B., Ma, Y., 2008. Techniques for evaluating fault prediction models.Empirical Software Engineering 13 (5), 561–595.

[14] R. Bender, "Quantitative Risk Assessment in Epidemiological Studies Investigating Threshold Effects," Biometrical J., vol. 41,no. 3, pp. 305-319, 1999.

[15] Raed Shatnawi," A Quantitative Investigation of the Acceptable Risk Levels of Object-Oriented Metrics in Open-Source Systems" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 36, NO. 2, MARCH/APRIL 2010

[16] Tibor Gyimo´ thy, Rudolf Ferenc, and Istva´n Siket," Empirical Validation of Object-Oriented Metricson Open Source Software for Fault Prediction", IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 31, NO. 10, OCTOBER 2005

[17] Analyst4j," http://www.codeswat.com"

[18] Karel Dejaeger,Thomas Verbraken,Bart Baesens ,"Toward Comprehensible Software Fault Prediction Models Using Bayesian Network Classifiers" IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 39, NO. 2, FEBRUARY 2013

[19] Dr. Linda H. Rosenberg," Applying and Interpreting Object Oriented Metrics"

[20] Bayesian Inference Concepts, "http://www.epixanalytics.com"