Empirical Studies to Predict Fault Proneness: A Review

Puja Saxena
M.E Student
Department of I.T, U.I.E.T
Panjab Univ., Chandigarh

Monika Saini
Assistant Professor
Department of I.T, U.I.E.T
Panjab Univ., Chandigarh

ABSTRACT

Empirical validations of software metrics are used to predict software quality in the past years. This paper provides a review of empirical studies to predict software fault proneness with a specific focus on techniques used. The paper highlights the milestone studies done from 1995 to 2010 in this area. Results show that use of machine learning languages have started.This paper reviews works done in the field of software fault prediction studies. The study is concentrated on statistical techniques and their usage to predict fault proneness. The conclusion drawn is the future studies should use more of class level metrics and the best technique to derive fault predictors amongst statistical techniques is logistic regression.

General Terms

Statistical techniques, empirical studies, fault predictors.

Keywords

Object Oriented Metrics, Fault Proneness.

1. INTRODUCTION

The paper reviews research papers and journal articles for the empirical studies that have been conducted in the field of prediction of fault proneness. Review has been done from grass root level to the new technology which has surfaced in the field in past fifteen years.

The studies reviewed have varied aspects starting from metrics selected to techniques implemented. Firstly the aim of the researchers was to find the relationship between fault proneness and metrics. With the advent of object oriented methodology the focus of researcher have been shifted too object oriented metrics. This paper helps throws light on the areas in which people of organisation should concentrate while estimating faults in their modules. Secondly provides the significance of CK Metrics in various studies. Reviewing various paper we come to know that repository dataset is also been widely used by researchers to estimate fault proneness. The paper also provides brief description of metrics and techniques proposed in this field.

This paper is organized as section II contains a description of the techniques used and metrics used, section III describes the papers studied and section IV conclusion.

2. BACKGROUND

2.1 Statistical Techniques

Researchers have used vast variety of techniques to predict fault proneness. Only few techniques are there those are of practical relevance in organizations. This section gives the brief of some important techniques which are been used in this area.

2.1.1 Logistic Regression

This technique is base maximum likelihood estimation. A univariate logistic regression model is based on the following equation:

$$\pi(\mathrm{X}) = \frac{e^{(\beta_0+\beta_1 X_1)}}{1+e^{(\beta_0+\beta_1 X_1)}} \tag{1}$$

A multivariate logistic regression model is based on equation:

$$\pi(\mathrm{X}) = \frac{e^{(\beta_0+\beta_1 X_1+\cdots+\beta_n X_n)}}{1+e^{(\beta_0+\beta_1 X_1+\cdots+\beta_n X_n)}} \tag{2}$$

Here the dependent variable is the software metrics and the independent variable is the fault proneness. π is the probability that a fault would occur in operative period. X_i's are the independent variables. The curve between them should be S-shaped.β_i' s are the coefficients which are estimated through maximum likelihood estimation[2].

Odds Ratio represents the ratio between the probability of having a fault and the probability of not having a fault.

2.1.2 Ordinary Least Squares

This is a method for estimating the unknown parameters in a linear regression model. This method minimizes the sum of squared vertical distances between the observed responses in the dataset, and the responses predicted by the linear approximation. The resulting estimator can be expressed by a simple formula. The model depends on the equation:

$$y_i = \beta X_i + \varepsilon \tag{3}$$

2.1.3 Discriminant Analysis

Discriminant analysis develops a discriminant function or classification criterion to place each observation into one of mutually exclusive groups. These groups can be fault prone modules or not. The underlying principle of the technique is that an operational hypothesis is formulated that there exists a priori classification of multivariate observations into two or more groups or set of observations. The membership in one of these supposed groups is mutually exclusive. A criterion variable will be used for this group assignment. For predicting fault prone modules researchers code it as 0 or 1. [15]

2.2 Metrics

This section describes various metrics used by different researchers in studies.

2.2.1 C.K. Metrics Model

Chidamber and Kemmerer's Metrics Suite define the CK Metric Suite [16].The suite helps designers and testers to make better design decisions. This suite describes six metrics described below.

2.2.2 MOOD Metrics Model

Metrics for Object Oriented Design referred as MOOD Metrics Model [17] .It describes basic structural mechanism of the object oriented paradigm as encapsulation, inheritance, polymorphism and message passing. In MOOD metrics model, methods and attributes are used in every metrics. Methods are used to perform operations of several kinds. Attributes are used to represent the status of each object in the system.

2.2.3 Milestone Studies

This section briefly describes the various studies done in the field of predicting fault proneness. The studies show the progress in the field .Metrics played a pivotal role in the prediction of fault proneness.

Basili et al. [1] empirically validated CK Metrics and found no correlation among metrics. The paper stated that all metrics were effective in predicting fault proneness except LCOM.
Advantage-The paper validates metric set for predicting fault proneness modules. They can be used as quality indicators as most of them are independent.
Disadvantage- Data collected is not of industry system so there can be increased complexity when it comes to large object oriented systems of organizations. Problems can increase in large systems. The paper does not differentiate in couplings (CBO and RFC) within and across inheritance hierarchies.

Briand et al. [2] collected data from a project of a university. Paper uses GEN++ to extract the values of metrics. Validated inheritance, coupling and cohesion metrics.

Advantage-The paper validates the fact that many of the design measures present in literature are redundant. Coupling induced by method invocations, the rate of change in a class due to specialization and the depth of a class in its inheritance hierarchy appear to be important quality factors.
Disadvantage- The data collected is not of industry system so there can be increased complexity when it comes to large object oriented systems of organizations. Secondly cohesion does not have impact on fault proneness so it might be that cohesion is not been understood thoroughly.

Tang et al. [3] analyzed C K OO metrics suite on three industrial applications developed in C++. They found none of the metrics examined to be significant except RFC and WMC.
Advantage-They proposed new set of metrics which proved to be helpful in which classes need to be tested using OO testing techniques.

Emam et al. [4] took confounding effect of size is taken into consideration. Investigations regarding class sizes are done and after that a validation study of object oriented metrics is conducted. After controlling for size, none of the metrics studied were associated with fault-proneness.
Advantage-This paper demonstrates a strong size confounding effect. Questions work done by past researchers.
Disadvantage- Study did not take severity of faults in consideration.

Emam et al. [5] is a validation studies on a huge telecommunications C++ system. This investigates 24 metrics proposed by CK and Briand et al.CBO and ACMIC metrics are good indicators of fault-prone classes. They are both associated with fault-proneness after controlling for size.
Advantage- Empirically validates metrics keeping in mind confounding effects of class size. They prove accuracy of prediction models through the use of Receiver Operating Characteristic.
Disadvantage- Severity of faults not taken in consideration.

Briand et al. [6] empirically explore the relationships between existing object oriented measures. It is seen here that with size of classes, frequency of method invocations and depth of inheritance also affect fault proneness. Classes with higher WMC, CBO, DIT and RFC were more fault prone, while classes with more children (NOC) were less fault prone. LCOM was not

Table 1.Summary of Researches

Study	Year	Phase	Technique Used	Lang. Used	Dependent Variable	Independent Variable
Basili et el.[1]	1996	Testing	LR	C++	Fault Proneness	CK Metrics
Briand et al.[2]	1998	Testing	LR	C++	Fault Proneness	CK Metrics
Tang et al.[3]	1999	Testing	LR	C++	Fault Proneness	CK Metrics except NOC
Emam et al.[4]	1999	Testing	LR	C++	Fault Proneness	CKMetrics+NMO,NMA,SIX,NPAVG
Emam et al.[5]	1999	Testing	LR	C++	Fault Proneness	24 METRICS(including CK Metrics)
Briand et al.[6]	2000	Testing	LR	C++	Fault Proneness	49 Metrics(including CK Metrics)
Fioravanti et al.[7]	2001	Testing	PCA+LR	C++	Fault Proneness	226 Metrics(including CK Metrics)
Emam et al.[8]	2001	Development	LR	JAVA	Fault Proneness	24 Metrics(CK Metrics-DIT,NOC)
Yu et al.[9]	2002	Testing	RA+DA	JAVA	Fault Proneness	8 Metrics(including CK Metrics)
Subramanyam et al. [10]	2003	Testing	OLS	JAVA C++	Faults	WMC,CBO,DIT,SIZE
Gyimothy et al.[11]	2005	Development	LR+ML	C++	Fault Proneness	CK Metrics+SLOC
Zhou et al.[12]	2006	Development	LR+ML	C++	Fault Proneness	CK Metrics+SLOC
Olague et al.[13]	2007	Testing	LR	JAVA	Fault Proneness	CK,QMOOD,MOOD
Singh et al.[14]	2010	Testing	LR+ML	C++	Severity of Faults	CK Metrics+SLOC

*LR, Logistic Regression; ML, Machine Learning; OLS, Ordinary Least Square; SLOC, Source Lines of Code; DA, Discriminant Analysis

associated with defects.
Advantage-They are able to find the relationships between metrics using univariate and multivariate analysis.
Disadvantage- Cohesion does not have impact on fault proneness so there is a possibility that cohesion is not been understood thoroughly.

Fioravanti et al. [7] provides new approach to define models for fault proneness. Large set of metrics is taken and then it is being reduced using principal component analysis.
Conclusion drawn was that only few of them are relevant for identifying fault prone classes. The three models made are accurate in terms of validating metrics. The hybrid model created with 12 metrics is of use Conclusion drawn was decision trees are more flexible and robust than statistical classification models.
Advantage- Paper uses tools to extract metrics. These models can help in defining new dimensions to validation studies.
Disadvantage- The paper does not differentiate in couplings (CBO and RFC) within and across inheritance hierarchies. The models were too large to be useful in practice.

Emam et al. [8] study shows to construct predictive models using object oriented models. One version of java application is used to make a predictive model and subsequent release is used to validate that model. Results show that an export coupling metric, depth of inheritance had the strongest association with fault proneness. Coupling across inheritance hierarchies (OC) is much more significant than coupling inside inheritance hierarchies to predict fault proneness.
Advantage- Distinguished coupling inside and across inheritance hierarchies. Paid attention to locus of impact, whether the class is a user or used in coupling relationship.
Disadvantage-This study did not account for the severity of faults.

Yu et al. [9] explores relationship between metrics and fault-proneness. Data collected in java was examined for correlations among them. Highly correlated subsets were found. They found that metrics (CBO_{in}, RFC_{in}, and DIT) were significant but to a different extent.
Advantage- Distinguished coupling inside and across inheritance hierarchies.
Disadvantage- No attention to severity for faults or to the confounding effects of class size.

Subramanyam et al. [10] uses CK Metrics suite subset to determine software defects. It uses two programming languages C++ and Java. It takes care for confounding size effects on class. The paper uses defect count as measure of quality indicator. Size was good predictor in both the languages.
Advantage-This work shows that relationship between dependent and independent variable can also depend on software language. Results are different in C++ and Java.
Disadvantage-This covers subset of CK Metrics. Secondly it aims at analyzing defects.

Gyimothy et al. [11] describes how to calculate the object-oriented metrics given by Chidamber and Kemerer of open-source software system. Systems opted for validations were Bugzilla and Mozilla. Techniques used were linear regression,logistic regression and machine learning methods. All techniques revealed approximately same results while CBO metrics seems to be best predictor of metrics.NOC metric cannot be used for prediction.
Advantage-The study paves a way to a different approach for open source systems.
Disadvantage- There are lot of errors in the model which have to be rectified.

Zhou et al. [12] uses logistic regression and machine learning methods to investigate the relationship between fault proneness and CK Metrics Suite. Machine Learning methods which are incorporated in the study are Naïve Bayes network, Random Forest and NNge. It does not take account of severity of impact instead the relationship between metrics and fault proneness when fault severity is taken into account.
Advantage- Stepping stone in the field as takes severity of faults into consideration. Machine Learning methods used to predict fault proneness can be of added advantage.

Olague et al. [13] validated object oriented metrics on versions of open source agile software. They found WMC, CBO, RFC and LCOM to be very significant. The MOOD metrics is direct measure of size when used over large classes. Conclusion drawn was decision trees are more flexible and robust than statistical classification models.
Advantage-The paper reviews highly iterative or agile software development processes.
Disadvantage-When the software is in the early stages of evolution and complexity is still low, the metrics will not be very effective. Also, for highly iterative or agile systems, the metrics will not remain effective forever.

Singh et al. [14] empirically validates software metrics for different levels of severity of faults. The techniques employed are regression and machine learning methods. The paper compares both techniques in order to find which one is better. Metrics used is CK Metrics suite.ROC analysis is also done.
Advantage-Severity of faults is important aspect for predicting fault proneness. Used CK Metrics hence results could be of great help to researchers and practitioners.
Disadvantage-Study done on industrial background could have been more beneficial.

3. SIGNIFICANCE OF METRICS

As seen usage of CK Metrics in approximately all studies. Selection of metrics holds significant importance in pursuing research work. Each metrics used by study holds significance in the conclusions drawn. It depends upon the data set, language of dataset and techniques which help to draw conclusions. Hence the significance is different in different environments.

We have tabulated significance of the main metrics used by researchers in Table 2 .This tabulation can help future studies in choosing appropriate metrics. All studies have not been considered because of the different aspects of the result.

Table 2.Significance of CK Metrics in different studies

l of Systems and Software, 2000, 51(3),

STUDY	WMC	DIT	RFC	NOC	CBO	LCOM
Basili et al.[1]	+	++	++		+	0
Tang et al.[3]	+	0	+	0	0	
Briand et al.[6]	+	++	++	-	++	
Briand et al.[18]	++	--	++	0	++	
Emam et al.[8]	+	0	++		+	
Yu et al.[9]	++	0	+	++	+	
Subramanyam et al.[10]	+ 0	- -			+ -	
Gyimothy et al.[11]	++	+	++	0	++	+
Zhou et al.[12]	++	0	++	--	++	+
Olague et al. [13]	++	0	++	0	++	++
Singh et al.[14]	++	0	++	0	++	+

++, Denotes metric is significant at 0.01; +, Denotes metric is significant at 0.05; --, Denotes metric is significant at 0.01 but in an inverse nature;
- , Denotes metric is significant at 0.05 but in an inverse manner; 0, Denotes that metric is not significant. Blank entry metric was calculated in different way.

4. CONCLUSION

Logistic Regression among the statistical techniques is a good predictor of fault proneness. Among the studies included in paper CK Metrics is also highly used to predict fault proneness. Effective metrics in the milestone studies are highlighted in Table 2. Although research works using machine learning methods have not been completely explored in this paper. Statistical techniques are of practical relevance as they are used in various organizations. The review suggests that among statistical techniques logistic regression is highly used and should be preferred for future studies while opting for statistical techniques.

5. REFRENCES

[1] Basili, V., Briand, L., & Melo, W. A validation of object oriented design metrics as quality indicators. IEEE Transactions on Software Engineering, 1996, 22(10), pp.751–761.

[2] Briand, L., Daly, J., Porter, V., & Wust, J.A comprehensive empirical validation of design measures for Object Oriented Systems. Proceeding METRICS '98 Proceedings of the 5th International Symposium on Software Metrics IEEE Computer Society Washington, DC, USA. 1998.

[3] Tang, M. H., Kao, M. H., & Chen, M. H. An empirical study on object-oriented metrics. In Proceedings of 6th IEEE International Symposium on Software Metrics. 1999, pp.242–249.

[4] El Emam, K., Benlarbi, S., Goel, N., & Rai, S. The confounding effect of class size on the validity of object-oriented metrics. IEEE Transactions on Software Engineering, 1999, 27(7), pp. 630–650.

[5] El Emam, K., Benlarbi, S., Goel, N., & Rai, S. A validation of object-oriented metrics. Technical report ERB-1063, NRC, 1999.

[6] Briand, L., Daly, W., & Wust, J. Exploring the relationships between design measures and software

[7] Fioravanti, F., & Nesi, P.A Study on Fault-Proneness Detection of Object-Oriented Systems. Proc. Fifth European Conference Software Maintenance and Reengineering (CSMR 2001), 2001, pp. 121-130.

[8] El Emam, K., W.Melo, and J.Machado. The prediction of faulty classes using object-oriented design metrics. The Journal of Systems and Software, 2001, 56(1),pp.63–75

[9] Yu, P., Systa, T., & Muller, H. Predicting fault-proneness using OO metrics: An industrial case study. In Proceedings of Sixth European Conference on Software Maintenance and Reengineering, Budapest, Hungary, 2002, pp.99–107.

[10] Subramanyam, R. and M.S. Krishnan. Empirical analysis of CK metrics for object oriented design complexity. Implications for software defects. IEEE Trans. Software Eng., vol. 29, 2003, pp. 297-310.

[11] Gyimothy, T., Ferenc, R., & Siket, I. Empirical validation of object-oriented metrics on open source software for fault prediction. IEEE Transactions on Software Engineering, 31(10), 2005, pp.897–910.

[12] Zhou, Y., & Leung, H. Empirical analysis of object oriented design metrics for predicting high severity faults. IEEE Transactions on Software Engineering, 32(10), 2006, pp. 771–784.

[13] Olague, H., Etzkorn, L., Gholston, S., & Quattlebaum, S. Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes. IEEE Transactions on Software Engineering, 33(8), 2007, pp.402–419.

[14] Singh, Y., Kaur, A., & Malhotra, R. Empirical validation of object-oriented metrics for predicting fault proneness models.SoftwareQualityJournal, 2010, pp.3-35.

[15] Munson, J.C., & Khoshgoftaar, T. The Detection of Fault-Prone Programs.IEEE Trans. on Reliability, 1992, vol.51, pp.423-432.

[16] Chidamber, S., & Kemerer, C. A metrics suite for object-oriented design. IEEE Transactions on Software Engineering, 1994, 20(6), pp.476–493.

[17] Abreu, F. B., "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.

[18] Briand, L., Daly, W., & Wust, J.Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs. Empirical Software Engineering International Journal, 6(1), pp.11-58.