A Defect Prediction Model for Open Source Software

Ruchika Malhotra

***Abstract*- Defect prediction models are significantly beneficial for software systems, where testing experts need to focus their attention and resources on problematic areas in the software under development. In this paper we find the relation between object oriented metrics and fault proneness using logistic regression method. The results are analyzed using open source software. The performance of the predicted models is evaluated using Receiver Operating Characteristic (ROC) analysis. The results show that Area under Curve (calculated by ROC analysis) of the predicted model is 0.829.**

Index Terms— **Object oriented Metrics, Software quality, Empirical validation, Fault prediction, Receiver Operating Characteristics analysis**

I. INTRODUCTION

Application of software quality models early in the software development life cycle contributes to efficient defect removal and results in delivering more reliable software products. Empirical studies for predicting defects have been carried out in past and have stressed on the need to carry out more such studies in order to provide strong evidence in this important area.

Several metrics have been proposed in the literature to capture the OO design and code, constructs for example, (Aggarwal et al. [1]; Briand et al., [2, 3]; Bieman and Kang [4]; Cartwright and Shepperd [5]; Chidamber and Kemerer [6]; Harrison et al. [7]; Henderson-sellers [8]; Hitz and Montazeri [9]; Lake and Cook [10]; Li and Henry [11]; Lee et al. [12] Lorenz and Kidd [13]; Tegarden et al [14]).

These metrics provide ways to assess the quality of software and their use in early phases of software development can help software companies in evaluating large software development quickly and at a reasonable cost [1].

Manuscript received March 22, 2011.

Dr. Ruchika Malhotra (Corresponding Author phone: 91-9910290445) is with Department of Software Engineering, Delhi Technological University, Bawana Road, Delhi 110042, India, (email: ruchikamalhotra2004@yahoo.com)

There have been empirical studies evaluating the impact of OO metrics on faulty classes such as (Aggarwal et al. [15], Singh et al. [16-18]; Basili et al. [19]; Binkley and Schach [20]; Briand et al [21]; Cartwright and Shepperd [5]; El Emam et al. [22]; Gyimothy et al. [23]; Zhou et al. [24]).

The work described in this paper focuses on the use of Object Oriented (OO) metrics in predicting defect prone classes. Our results are based on open source software Ant 1.7 developed using java language [25]. Although the open source software has achieved an acceptable level of quality, but there is more to be done in order to outperform proprietary software [26]. Hence, the defect prediction models can help in improving the quality and reducing faulty classes in the open source software. The validation of the methods is carried out using Receiver Operating Characteristic (ROC) analysis. Hence, the study is divided into the following parts:

1. Descriptive statistics and outlier analysis is performed to extract useful metrics and remove irrelevant data points.
2. Multivariate logistic regression is used for model prediction and validation.
3. Defect prediction model is evaluated using performance measures including ROC analysis.

The model constructed to predict faulty classes may help in focusing testing and inspection resources on the defect prone parts of the design and code in a cost effective manner.

The paper is organized as follows: Section 2 summarizes the OO metrics studied. Section 3 describes sources from which data is collected. The results of the study are given in section 4 and the model is evaluated in section 5. Section 6 presents threats to validity of the models and the conclusions of the research are presented in section 7.

II. METRICS USED

The binary dependent variable in our study is fault proneness. Fault proneness is defined as the probability of fault detection in a class [17]. We use logistic regression, which is based on predicting probabilities. For this study, we predict fault prone classes from object oriented metrics. The metrics are collected by using a tool for calculating Chidamber and Kemerer java metrics (ckjm). The tool is available at http://gromit.iiar.pwr.wroc.pl/p_inf/ckjm/metric.html. The metrics are given in table I [6, 7, 27].

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

TABLE I
METRICS USED IN THE STUDY

Metric	Definition
Coupling between objects (CBO)	CBO for a class is a count of the number of other classes to which it is coupled and vice versa.
Lack of cohesion (LCOM)	Measures the dissimilarity of methods in a class by looking at the instance variable or attributes used by methods.
Number of children (NOC)	The number of immediate subclasses of a class in a hierarchy.
Depth of inheritance (DIT)	The depth of a class within the inheritance hierarchy is the maximum number of steps from the class node to the root of the tree and is measured by the number of ancestor classes.
Weighted methods per class (WMC)	A count of sum of complexities of all methods in a class.
Response for a class (RFC)	A set of methods that can be potentially executed in response to a message received by an object of that class.
Number of public methods (NPM)	The count of number of public methods in a class.
Afferent couplings (Ca)	It counts how many other classes use a given class.
Lack of cohesion in methods (LCOM3).	$LCOM1 = \frac{\frac{1}{N}\sum_{i=1}^{n}\mu(D_i) - m}{1-m}$
Data Access Metric (DAM)	It is defined as number of private methods divided by total number of methods.
Measure of Aggregation (MOA)	It counts abstract data types in a class.
Measure of Functional Abstraction (MFA)	It is defined as number of inherited methods divided by total number of methods accessible by its member functions.
Cohesion Among Methods of Class (CAM)	It is based upon parameters list of a method.
Inheritance Coupling (IC)	It is based upon inheritance based coupling.
Coupling Between Methods (CBM)	It counts the newly added functions with which inherited based methods are coupled.
Average Method Complexity (AMC)	It counts average size of method in a class.
Cyclomatic Complexity (CC)	CC= e-n+P, where e= number of edges in a flow graph, n=number of nodes in a flow graph, p= connected components
Lines of code (LOC)	The count of lines in the text of the source code excluding comment lines

III. EMPIRICAL DATA COLLECTION

In this study Apache Ant 1.7 open source software is used [25] for finding relationship between OO metrics and fault proneness. ANT 1.7 is a command-line tool that allows to build, compile, test and run Java applications. Ant 1.7 is developed using the Java language consisting of 745 classes. The development period of this version was December 2006 to September 2009. The details of the data set are summarized in table II.

TABLE II
DATA USED IN THE STUDY

System	Apache Ant
Language	Java
Total Classes	745
% faulty classes	22.28
Total faults	338

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

IV. ANALYSIS RESULTS

In this section, we described the analyses performed to find the relationship between OO metrics and fault proneness of the classes. We employed multivariate logistic regression analysis. The multivariate analysis is used to find the combined effect of OO metrics on fault proneness. The models predicted were applied to Apache Ant 1.7 data set consisting of 745 classes. Descriptive statistics and outlier analysis was performed to find the irrelevant data. The following measures are used to evaluate the performance of each predicted fault proneness model in the above sub sections [17]:

- Sensitivity and Specificity, Completeness, Precision, Receiver Operating Characteristic (ROC) analysis [28].
- In order to predict the accuracy of the model it should be applied to different data sets. We therefore performed k-cross validation of models [29]. The data set is randomly divided into k subsets. Each time one of the k subsets is used as the test set and the other k-1 subsets are used to form a training set. Therefore, we get the fault proneness for all the k classes.

A. Descriptive statistics

To perform research analysis, the data should be preprocessed by removing irrelevant and unnecessary attributes that have less than six data points and eliminating outliers from the data set. Thus, we obtained descriptive statistics and performed outlier analysis on the data set. The descriptive statistics of the data set are shown in tablle III.

B. Research Methodology

Logistic Regression (LR) is used to predict the dependent variable (fault proneness) from a set of independent variables (OO metrics) to determine the percent of variance in the dependent variable explained by the independent variable (a detailed description is given by [30]. LR is of two types [17]: a) Univariate LR b) Multivariate LR.

C. Multivariate LR Results

In this section, we summarize the results obtained from multivariate fault prediction model using LR method. The multivariate analysis is used to find the combined effect of OO metrics (explained above) on fault proneness. We attempted to use the backward elimination method. However, the results of the model obtained were poorer (i.e. the values of R^2 statistic were low) than the model obtained from the forward stepwise procedure. We therefore used the forward stepwise procedure in this study. The conditional number is below 30 for the model predicted. This implies that the multicollinearity of the predicted model is tolerable. Table IV provide the coefficient (B), standard error (SE), statistical significance (sig), odds ratio (exp(B)) for OO metrics included in the model. Table IV shows that two metrics, RFC and CC, are included in the predicted model. The results of model accuracy are summarized in table V. The AUC is 0.834, hence the accuracy of the model predicted is very high.

TABLE III

MODEL STATISTICS

Metrics	Min.	Max.	Mean	Std. Deviation
WMC	0	120	11.07114	11.97596
DIT	1	7	2.522148	1.398869
NOC	0	102	0.731544	4.800357
CBO	0	499	11.04698	26.34315
RFC	0	288	34.36242	36.02497
LCOM	0	6692	89.14765	349.9376
CA	0	498	5.655034	25.81422
CE	0	37	5.746309	5.653176
NPM	0	103	8.365101	9.331319
LCOM3	0	2	1.013342	0.619015
LOC	0	4541	280.0711	411.8721
DAM	0	1	0.644855	0.438138
MOA	0	11	0.726174	1.426581
MFA	0	1	0.509968	0.398696
CAM	0	1	0.474685	0.259931
IC	0	5	0.720805	0.938948
CBM	0	19	1.312752	2.332602
AMC	0	2052	23.64087	76.98608
CC	0	53	4.669799	6.276853

TABLE IV

MODEL STATISTICS

Variable	B	S.E.	Sig.	Exp(B)
RFC	0.032	0.004	0.000	1.032
CC	0.075	0.022	0.001	1.077
Constant	-2.926	0.188	0.000	0.054

TABLE V

RESULTS OF MODEL PREDICTED

Measures	Results
Cutoff	0.17
Sensitivity	75.90
Specificity	75.50
Precision	75.06
AUC	0.834
SE	0.018

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

V. MODEL EVALUATION USING ROC ANALYSIS

In this section, we present the results of model evaluation.

D. Model Evaluation

The accuracy of the models predicted is somewhat optimistic since the models are applied on same data set from which they are derived. To predict accuracy of the model it should be applied on different data sets thus we performed 10-cross validation of the models. For the 10-cross validation, the classes were randomly divided into 10 parts of approximately equal data points. We summarized the results of cross validation of predicted models via the LR approach in Table VI. Table VI shows that AUC calculated by ROC analysis is very high 0.829. Thus, the accuracy of model obtained from validation data is similar to the accuracy of model obtained from training data. The values of sensitivity, specificity and precision are also high. The researchers and software practitioners can use the model predicted in early phase of software development. This will reduce testing effort and resources.

TABLE VI

RESULTS OF 10-CROSS VALIDATION OF MODEL

Measures	Results
Cutoff	0.18
Sensitivity	75.30
Specificity	75.50
Precision	75.04
AUC	0.829
SE	0.018

The ROC curve for the model given in table VI is shown in fig. 1.

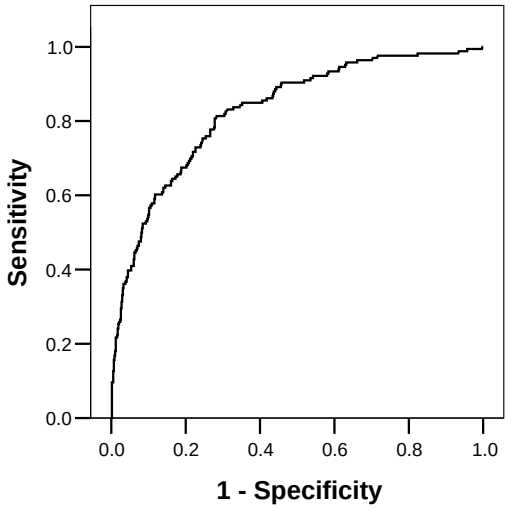

Fig. 1: ROC curve of LR Model

VI. THREATS TO VALIDITY

The study has many limitations that are common with most of the empirical studies in literature. However, it is necessary to repeat them here.

The usefulness of OO metrics for predicting fault proneness models also depends on the programming language (e.g. C++, Java) being used. Thus, similar studies with different data sets are required to be carried out in order to establish the acceptability of the model.

Our conclusions are pertinent to only dependent variable fault proneness, as it seems to be most popular dependent variable in empirical studies. We do not claim about the validity of the chosen OO metrics in this study when the dependent variable changes like maintainability or effort [17].

VII. CONCLUSION

The goal of this work is to find the effect of OO metrics on fault proneness. We also empirically analyze the performance of logistic regression method in order to predict faulty classes.

Based on the results obtained from open source software Apache Ant 1.7 data set we analyzed the performance of predicted defect model using the ROC analysis. We analyzed the OO metrics including metrics given by Chidamber and Kemerer and McCabe. Our main results are summarized as follows:

Two metrics RFC and CC were included in the model. The model predicted has higher accuracy with AUC 0.829. Thus, the model predicted shows that the model predicts faulty classes of open source software with good accuracy.

This study confirms that construction of model using logistic regression method will be effective, adaptable, and useful in predicting fault prone classes.

We plan to replicate our study to predict models based on machine learning algorithms such as support vector machines and genetic algorithms. We may carry out cost benefit analysis of models that will help to determine whether a given fault proneness model would be economically viable.

REFERENCES

[1]. K.K.Aggarwal, Yogesh Singh, Arvinder Kaur, Ruchika Malhotra, “Software Reuse Metrics for Object-Oriented Systems”, Third ACIS Int'l Conference on Software Engineering Research, Management and Applications (SERA'05), IEEE Computer Society, pp. 48-55, 2005.

[2]. L..Briand , W.Daly and J. Wust, “Unified Framework for Cohesion Measurement in Object-Oriented Systems”, Empirical Software Engineering, vol. 3, pp.65-117, 1998.

[3]. L.Briand , W.Daly and J. Wust, “A Unified Framework for Coupling Measurement in Object-Oriented Systems. IEEE Transactions on software Engineering”, Vol. 25, pp.91-121, 1999.

[4]. J.Bieman, B.Kang, “Cohesion and Reuse in an Object-Oriented System”, Proc. ACM Symp. Software Reusability (SSR’94), pp.259-262, 1995.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)

[5]. M.Cartwright, M.Shepperd, "An Empirical Investigation of an Object-Oriented Software System", IEEE Transactions of Software Engineering, 1999.
[6]. S.Chidamber and C.F.Kemerer, "A metrics Suite for Object-Oriented Design", IEEE Trans. Software Engineering, vol. SE-20, no.6, 476-493, 1994.
[7]. B.Henderson-sellers, "Object-Oriented Metrics, Measures of Complexity", Prentice Hall, 1996.
[8]. R.Harrison, S.J.Counsell, and R.V.Nithi, "An Evaluation of MOOD set of Object-Oriented Software Metrics", IEEE Trans. Software Engineering, vol. SE-24, no.6, pp. 491-496, June 1998.
[9]. B.Henderson-sellers, "Object-Oriented Metrics, Measures of Complexity", Prentice Hall, 1996.
[10]. M.Hitz, B. Montazeri, "Measuring Coupling and Cohesion in Object-Oriented Systems", Proc. Int. Symposium on Applied Corporate Computing, Monterrey, Mexico, 1995.
[11]. A.Lake, C.Cook, "Use of factor analysis to develop OOP software complexity metrics". Proc. 6th Annual Oregon Workshop on Software Metrics, Silver Falls, Oregon, 1994.
[12]. W.Li, S.Henry, "Object-Oriented Metrics that Predict Maintainability", Journal of Systems and Software, vol 23 no.2, pp.111-122, 1993.
[13]. Y.Lee, B.Liang, S.Wu and F.Wang, "Measuring the Coupling and Cohesion of an Object-Oriented program based on Information flow", 1995.
[14]. M.Lorenz, and J.Kidd, "Object-Oriented Software Metrics", Prentice-Hall, 1994.
[15]. D.Tegarden, S. Sheetz, D.Monarchi, "A Software Complexity Model of Object-Oriented Systems. Decision Support Systems", vol. 13, pp.241-262.
[16]. K.K. Aggarwal, Y. Singh, A. Kaur, R. Malhotra, "Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study", *Software Process Improvement and Practice*, John Wiley & Sons, vol. 16, no. 1, pp. 39-62, 2009.
[17]. Y. Singh, A. Kaur, and R. Malhotra, "Empirical validation of object-oriented metrics for predicting fault proneness models", *Software Quality Journal*, vol. 18, pp.3-35, Jan 2010.
[18]. Y. Singh, A. Kaur, and R. Malhotra, "Predicting Software Fault Proneness Model Using Neural Network", *Product-Focused Software Process Improvement, Lecture Notes in Computer Science*, pp. 204-214, 2008.
[19]. Y. Singh, A. Kaur, and R. Malhotra, "Application of Decision Trees for Predicting Fault Proneness", *International Conference on Information Systems, Technology and Management-Information Technology*, Ghaziabad, India, 2009.
[20]. V.Basili, L.Briand, W.Melo, "A Validation of Object-Oriented Design Metrics as Quality Indicators", IEEE Transactions on Software Engineering, vol. 22 no.10, pp. 751-761, 1996.
[21]. L. Briand, W. Daly, and J. Wust, "Exploring the relationships between design measures and software quality", *Journal of Systems and Software*,, vol. 51, no. 3, pp. 245-273, 2000.
[22]. K. El Emam, S. Benlarbi, N. Goel, and S. Rai, "A Validation of Object-Oriented Metrics", *Technical Report ERB-1063*, NRC, 1999.
[23]. T.Gyimothy , R.Ferenc , I.Siket , "Empirical validation of object-oriented metrics on open source software for fault prediction", IEEE Trans. Software Engineering, vol. 31, Issue 10, pp.897 – 910, Oct. 2005.
[24]. Y. Zhou, and H. Leung, "Empirical analysis of Object-Oriented Design Metrics for predicting high severity faults", *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 771-784, 2006.
[25]. promise. Available at http://promisedata.org/repository/
[26]. I. Samoladas, and I. Stamelos, "Assessing Free/Open Source Software Quality", Department of Informatics, Aristotle University of Thessaloniki, Greece.
[27]. T. McCabe, "A Complexity Measure", *IEEE Transactions on Software Engineering*, vol. 2, no. 4, pp. 308-320, 1976.
[28]. J. Hanley, BJ. McNeil, "The meaning and use of the area under a Receiver Operating Characteristic ROC curve", *Radiology*, vol. 143, pp.29-36, 1982.
[29]. M. Stone, "Cross-validatory choice and assessment of statistical predictions", *J. Royal Stat. Soc.*, vol. 36, pp. 111-147, 1974.
[30]. D. Hosmer, S. Lemeshow, Applied Logistic regression, John Wiley and Sons 1989.

ISBN: 978-988-19252-1-3
ISSN: 2078-0958 (Print); ISSN: 2078-0966 (Online)