Evaluation of Fault Proneness of Modules in Open Source Software Systems Using k-NN Clustering

Er. Harish Kundra[1], Er. Anamika Sharma[2], Dr. R.P.S.Bedi[3]

[1]Head of Department of Computer Science and IT, [2]M-Tech Student

Rayat Institute of Engineering and IT, Railmajra, Ropar.

[3]Joint Registrar,

Punjab Technical University, Jalandhar

hodcseit@rayatbahra.com, er.anamika87@gmail.com

ABSTRACT:*Fault-proneness of a software module is the probability that the module contains faults. A correlation exists between the fault-proneness of the software and the measurable attributes of the code (i.e. the static metrics) and of the testing (i.e. the dynamic metrics). Early detection of fault-prone software components enables verification experts to concentrate their time and resources on the problem areas of the software system under development. This paper introduces the evaluation of the fault proneness of modules in open source software system using k-NN clustering algorithm with Object-Oriented metrics and ck metrics. The contribution of this paper is that it has used Metric values of JEdit open source software for generation of the rules for the classification of software modules in the categories of Faulty and non faulty modules and thereafter empirically validation is performed. The results shows that algorithm approach can be used for finding the fault proneness in object oriented software components.*

Keywords: *k-NN Clustering Algorithms, Software Fault, Classification, Object Oriented Metrics, CK Metrics.*

1. Introduction

As the complexity and the constraints under which the software is developed are increasing, it is difficult to produce software without faults. Such faulty software classes may increase development & maintenance cost, due to software failures and decrease customer's satisfaction [11]. Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. There are many metrics and technique available for investigate the accuracy of fault prone classes which may help software organizations for planning and performing testing activities. 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. In the past, several metrics for measuring software complexity k-NN clustering algorithm is being successfully applied for solving classification problems. It is therefore important to investigate the capabilities of this algorithm in predicting software quality. In order to perform the analysis we validate the performance of the k-NN clustering method for dataset derived from open source software JEdit [15].The 274 classes in this data were developed using Java language. In this study, we investigate the capability of k-NN clustering Algorithm in predicting faulty classes. We investigate the accuracy of the fault proneness predictions using object oriented design using metrics suite given by Chidamber and Kemerer [6] and used in [2] for fault prediction. By using K-NN clustering Algorithm technique on fault prone classes may enable the software organizations for planning and performing testing by focusing on accuracy of fault prone classes. This may result in significant improvement in software quality. The objectives of the study may be described as follows:

• To find the structural code and design attributes of software systems

• Developing the Fault Prediction model using the k-NN clustering algorithm

The contributions of the paper are summarized as follows:

First open source software systems analyzed. These systems are developed with different development methods than proprietary software. In previous studies mostly proprietary software were analyzed. Second, we examine k-NN clustering method to predict the faulty classes with better accuracy.

The paper is organized as follows: section2 discusses the related work, Section3 explains about the empirical data collection and section 4 describes the k-NN clustering based methodology.

The result of the study is given in section 5. Finally conclusions of the research are presented in section6.

2. Related Work

Lanubile & Lonigro [12] presented an empirical investigation of the modeling techniques for identifying fault-prone software components early in the software life cycle.

Khoshgaftaar [10] introduced the use of the neural networks as a tool for predicting software quality. Saida Benlarbi [17] surveyed that the basic premise behind the development of object-oriented metrics is that they can serve, as early predictors of classes that contain faults or that are costly to maintain.

Runeson & Magnus [16] proposed that in striving for high-quality software, the management of faults plays an important role. The faults that reside in software products are not evenly distributed over the software modules; some modules are more fault-prone than others.

Briand [5] extracted 49 metrics to identify a suitable model for predicting fault proneness of classes. The system under investigation was medium sized C++ software system developed by undergraduate or graduate students. The eight systems under study consisted of a total of 180 classes. A. Mahaweerawat [13] analyzed that to remain competitive in the dynamic world of software development; organizations must optimize the usage of their limited resources to deliver quality products on time and within budget. P. Bellini [3] compared Fault-Proneness Estimation Models that are developed using logistic regression and the discriminate analyses.

Gyimothy [8] empirically validated CK metrics on open source software for fault prediction.

Yan Ma [21] suggested that accurate prediction of fault prone modules in software development process enables effective discovery and identification of the defects.

G. Pai [14] validated the public domain NASA data set as used in their study to predict fault proneness models with respect to two categories of

faults: high and low. In this also used the same data set using a Bayesian approach to predict fault proneness models Aggarwal [2]. Validated object-oriented metrics to predict faulty classes.

Arvinder Kaur [9] empirically evaluates the performance of RF in predicting fault-prone classes using open source software.

S.K, Dhiman [7] presented a Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes. In this paper Genetic algorithm approach can be used for finding the components.

3. EMPIRICAL EVALUATION

This section presents the goal of the study (section3.1), empirical data collection (section3.2).

3.1 Goal

The goal of the study is to evaluate the performance of k-NN clustering method. The performance is evaluating using OO metrics and CK metrics for predicting faulty classes using open source software.

3.2 Empirical Data Collection

We used JEdit open source software in this study [15]. JEdit is a programmer's text editor developed using Java language. JEdit combines the functionality of Window, Unix, and MacOS text editors. It was released as free software and the source code is available on [20]. JEdit includes 274 classes. The number of developers involved in this project was 144. The project was started in 1999. The number of bugs was computed using SVC repositories. The release point for the project was identified in 2002. The log data from that point to 2007 was collected. The header files in C++ were excluded in data collection. The word bug or fixed was counted. Details on bug collection process can be found in [19, 17].

4. K-NN CLUSTERING ALGORITHM BASED CLASSIFICATION TECHNIQUE

The k-nearest neighbor (k-NN) method is usually used to carry out classifications. The classification is performed by using a reference data

set which contains both the input and the target variables and comparing the unknown which contains only the input variables to that reference set. The distance of the unknown to the k nearest neighbors determines its class assignment by either averaging the class numbers of the k nearest reference points or by obtaining a majority vote from them. The K-nearest neighbor algorithm (KNN) is part of supervised learning that has been used in many applications in the field of data mining, statistical pattern recognition and many others.KNN is a method for classifying objects based on closest training examples in the feature space. An object is classified by a majority vote of its neighbors. K is always a positive integer. The neighbors are taken from a set of objects for which the correct classification is known. The algorithm on how to compute the K-nearest neighbors is as follows:

1. Determine the parameter K = number of nearest neighbors beforehand. This value is all up to you.
2. Calculate the distance between the query-instance and all the training samples. You can use any distance algorithm.
3. Sort the distances for all the training samples and determine the nearest neighbor based on the K-th minimum distance.
4. Since this is supervised learning, get all the Categories of your training data for the sorted value which fall under K.
5. Use the majority of nearest neighbors as the prediction value.

5. Research Methodology

This section presents the research methodology followed in this study.

5.1 Performance Evaluation Measures

Let *a* is the number of modules that actually have no fault and classifier predicts no defects in those modules, *b* is the number of modules that actually have defects and classifier predicts no defects in those modules, *c* is the number of modules that actually have no defects and classifier predicts defects in those modules and *d* is the number of modules that actually have defects and classifier predicts defects in those modules.

Table 1. The following is the details of the metrics used in the classification process

Metric	Definition
Coupling between Objects(CBO)	CBO for a class is count of the number of other classes to which it is coupled and vice versa
Lack of Cohesion (LCOM)	It measures the dissimilarity of methods in a class by looking at the instance variable or attributes used by methods
Number of Children (NOC)	The NOC is the number of immediate subclasses of a class in a hierarchy.
Depth of inheritance (DOI)	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)	The WMC is a count of sum of Complexities of all methods in a class Consider a class K1, with Methods M1…… Mn that are defined in the class. Let. C1…..Cn be the complexity of the methods
	n
	WMC = ? Ci
	i=1
	If all the methods complexities are considered to be unity, then WMC = n the number of methods in the class.
Number of Public Methods(NPM)	It is count of number of Public methods in a class
Lines of Code (LOC)	It is the count of lines in the text of the source code excluding comment lines.

5.1.1 Accuracy

It indicates approximate of measurement results to the true value, precision to the repeatability or reproducibility of the measurement [9]. The

accuracy is the proportion of true results (both true positives and true negatives) in the population.

$$Acc = (a + d) / (a + b + c + d)$$

5.1.2 Probability of detection

Probability of detection is the probability of system failure [7]. That means that the probability of a component B failing given that a component A has already failed is the same as that of B failing when A has not failed. Probability of detection is calculated as shown below:

$$Pd = d/(b+d)$$

5.1.3 Probability of false alarm

Intuitively probability of false alarm is the fraction of buggy execution that raises an alarm. The formula for Probability of false alarm is given below:

$$Pf = c/(a+c)$$

6. Analysis Result

This section presents the analysis results, following the procedure described in section 5.

6.1 Summary of Results

DIT, CBO, RFC, NPM and LOC were selected from the set of eight metrics. The percentage (%) of accuracy, probability of detection, probability of false alarm measures is shown in table3. The predicted model was applied to 274 classes. Thus, the model predicted using k-NN clustering method to achieves high accuracy (82.84percent) high probability of detection (percent),low probability of false alarm(percent)**.** Thus the results confirm that k-NN clustering can be used in constructing models for predicting faulty classes. However, more similar studies should be carried out to confirm the acceptability of the predicted model.

Table 2. A confusion matrix of prediction outcomes

		Module actually has defects	
		No	Yes
Classifier predicts no defects	No	119	26
Classifier predicts some defects	Yes	21	108

Table 3. Performance Measures

Method	Percentage (%)
Accuracy	82.84
Probability of detection	0.806
Probability of false alarm	0.15

7. Conclusion and Future Scope

This paper empirically evaluates performance of k-NN algorithm based classification technique in predicting fault prone classes using open source software. The proposed k-NN based classification technique shows 82.84 percent accuracy,0.8060% probability of detection,0-1500% probability of false alarm. This study confirms that construction of k- NN model is feasible, adaptable to Object Oriented systems and useful in predicting faulty prone classes. It is therefore concluded that model is implemented using k-NN based technique for classification of the software components into faulty/ fault-free systems is found satisfactory.

The future work can be extended in following directions:

- Most important attribute can be found for fault prediction and this work can be extended to further programming languages.
- More algorithms can be evaluated and then we can find the best algorithm. We plan to replicate our study to predict model based on soft computing techniques.

8. References

[1] Aggarwal, K. K., Singh, Y., Kaur, A. and Malhotra, R. 2008, Empirical Analysis for Investigating the Effect of Object-Oriented Metrics on Fault Proneness: A Replicated Case Study", Published online in Software Process Improvement and Practice, Wiley, 2008, pp.34-41.

[2] Aggarwal, K.K., Singh, Y., Kaur, A. and Malhotra, R.(2006), "Investigating the Effect of Coupling Metrics on Fault Proneness in Object-Oriented Systems", Software Quality Professional,8(4), 2006, pp.4-16.

[3] Bellini.(2005), "Comparing Fault-Proneness Estimation Models", 10th IEEE International Conference on Engineering of Complex Computer Systems (ICECCS'05), vol. 0, 2005, pp. 205-214.

[4] Bremner, D., Demaine, E., Erickson, J., Iacono, J., Langerman, S., Morin P. and Toussaint, G. (2005),"Output-sensitive algorithms for computing nearest- neighbor decision boundaries", Discrete and Computational Geometry 33 (4), 2005, pp.593–604.

[5] Briand, L., Wilst, J. and Lounis, H. (2001), "Replicated Case Studies for Investigating Quality Factors in Object-Oriented Designs", Empirical Software Engineering: An International Journal, 6(1), 2001, pp.11-58.

[6] Chidamber and Kemerer (1994), "A metrics suite for object-oriented design", IEEE Transactions on Software Engineering, 20(6), 1994, pp.476-493.

[7] Dhiman, S.K. and Goyal, A. (2009),"A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes", World Academy of Science, Engineering and Technology, 60, 2009, pp.485-488.

[8] Gyimothy, T., Ferenc,R. and Siket (2005),"Empirical validation of object-oriented metrics", IEEE open Transaction on Software Engineering, 31 ,2005,pp.897- 910.

[9] Kaur, A. and Malhotra. R. (2008), "Application of Random Forest in Predicting Fault-Prone classes", International conference on Advanced

Computer Theory and Engineering (ICACTE), Pukhet, Dec. 2008, pp. 37-43.

[10] Khoshgaftaar, T.M., Allen, E.D., Hudepohl, J. P. and Aud, S. J. (1997), "Application of neural networks to software quality modeling of a very large telecommunications system", IEEE Transactions on Neural Networks, 8(4), 1997, pp. 902-909.

[11] Koru, H. Liu, "Building effective defect- prediction models in practice", IEEE Software, 2005, pp.23-29.

[12] Lanubile, F., Lonigro, A. and Visaggio, G.(1995),"Comparing Models for Identifying Fault-Prone Software Components", Proceedings of Seventh International Conference on Software Engineering and Knowledge Engineering, June 1995, pp. 12-19.

[13] Mahaweerawat, A.(2004), "Fault-Prediction in object oriented software's using neural network techniques", Advanced Virtual and Intelligent Computing Center (AVIC), Department of Mathematics, Faculty of Science, Chulalongkorn University, Bangkok, Thailand, pp. 1-8.

[14] Pai, G.(2007),"Empirical analysis of Software Fault Content and Fault Proneness Using Bayesian Methods", IEEE Transactions on software Engineering, 33(10), 2007, pp.675-686.

[15] Promise. http://promisedata.org/repository/.

[16] Runeson, C. and Magnus, C. O. (2001), "A Proposal for Comparison of Models for Identification of Fault-Proneness", Dept. of Communication Systems, Lund University, Profes, LNLS 2188, 2001, pp. 341-355.

[17] Saida, B., Khaled, E. E. and Geol, N. (1999), "Issues in Validating Object-Oriented Metrics for Early Risk Prediction", By Cistel Technology 210 Colonnade Road Suite 204 Nepean, Ontario Canada K2E 7L5.

[18] Watanabe, S., Kaiya, H. and Kaijiri, K. (2008)," Adapting a Fault Prediction Model to Allow Inter Language Reuse", PROMISE, Leipzig, Germany, 2008, pp. 12-13.

[19] Watanabe, H. Kaiya, K. Kaijiri, Adapting a Fault Prediction Model to Allow Inter Language Reuse, PROMISE'08, May 12-13, Leipzig, Germany, 2008.

[20] Website sourceforge: www.sourceforge.net/projects/jedit.

[21] Yuming, Z. and Hareton, L.(2006), "Empirical analysis of Object-Oriented Design Metrics for predicting high severity faults", IEEE Transactions on Software Engineering, 32(10),2006, pp.771-784.