International Journal of Information, Control and Computer Sciences
ISSN: 2517-9942
Vol:3, No:12, 2009

A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes

Parvinder S. Sandhu, Satish Kumar Dhiman, Anmol Goyal

***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 Genetic Algorithm based software fault prediction models with Object-Oriented 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 Genetic algorithm approach can be used for finding the fault proneness in object oriented software components.

***Keywords*—**Genetic Algorithms, Software Fault, Classification, Object Oriented Metrics.

I. 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 [1]. 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 and testing thoroughness have been proposed. Static metrics, e.g., the McCabe's cyclomatic number or the Halstead's Software Science, statically computed on the source code and tried to quantify software complexity. Dynamic metrics, e.g., structural and data flow coverage, measure the thoroughness of testing as the amount of elements of the program covered by test executions.

Genetic algorithm is being successfully applied for solving both classification and regression 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 GA method for dataset derived from open source software JEdit [http://promisedata.org/repository/]. The 274 classes in this data were developed using Java language. In this study, we investigate the capability of Genetic 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 [2] and used in [3] for fault prediction. By using Genetic 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 Genetic 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 GA method to predict the faulty classes with better accuracy.

The paper is organized as follows: section 2 discusses the related work, Section 3 explains about the empirical data collection and section 4 describes the GA based methodology. The result of the study is given in section 5. Finally conclusions of the research are presented in section 6.

II. Related Work

Lanubile et. al. [4] presented an empirical investigation of the modeling techniques for identifying fault-prone software components early in the software life cycle. Using software complexity measures, the techniques build models, which classify components as likely to contain faults or not. The modeling techniques applied in their study cover the main

Dr. Parvinder S. Sandhu is Professor with Computer Science & Engineering Department, Rayat & Bahra Institute of Engineering & Bio-Technology, Sahauran, Distt. Mohali (Punjab)-140104 INDIA

Mr. Anmol Goyal is working as Asstt. Professor (Deptt. Of Electronics & Communication engineering), Rayat & Bahra Institute of Engineering & Bio-Technology, Sahauran, Distt. Mohali (Punjab)-140104 INDIA

Mr. Satish Kumar Dhiman is doing his Masters in .Philosphy from M.M. Universiyt, ullana, Ambala(Haryana) INDIA.

classification paradigms, including principal component analysis, discriminate analysis, logistic regression and logical classification models.

Saida Benlarbi et al [5] 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. They showed that common empirical validation methods to date could provide misleading conclusions as to the validity of object-oriented metrics. They presented the results of an empirical validation study of a set of object-oriented metrics. In their paper they have shown that size can have an important confounding effect on the validity of object-oriented metrics.

Runeson et al [6] 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. Various models are presented in the literature, which identify fault-prone modules, which can be used in quality management processes. They proposed schemes that would help researchers to compare the capabilities of different models and to choose a suitable model.

Atchara Mahaweerawat [7] 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. This requires prevention of fault introduction and quick discovery and repair of residual faults. In particular, faults due to the use of inheritance and polymorphism are considered, as they account for significant portion of faults in object-oriented systems. The proposed fault prediction model is based on supervised learning using Multilayer Perceptron Neural Network and the results are analyzed in terms of classification correctness and based on the results of classification, faulty classes are further analyzed and classified according to the particular type of fault.

P. Bellini et. al. [8] compared Fault-Proneness Estimation Models that are developed using logistic regression and the discriminate analyses. It is concluded that over the last years, software quality has become one of the most important requirement in the development of systems and fault-proneness estimation could play a key role in quality control of software products. In this area, much effort has been spent in defining metrics and identifying models for system assessment. Using these metrics to assess which parts of the system are more fault-proneness is of primary importance. The objective has been to find a compromise between the fault-proneness estimation rate and the size of the estimation model in terms of number of metrics used in the model itself. The methodologies were the.

Yan Ma et. al. [9] suggested that accurate prediction of fault prone modules in software development process enables effective discovery and identification of the defects. Such prediction models are especially valuable for the large-scale systems, where verification experts need to focus their attention and resources to problem areas in the system under development. The paper [9] presents a methodology for predicting fault-prone modules using a modified random forests algorithm which improve classification accuracy by growing an ensemble of trees and letting them vote on the classification decision.

Eric Rotenberg [10] speculates that technology trends pose new challenges for fault tolerance in microprocessors. The paper proposed a new time redundancy fault-tolerance approach in which a program is duplicated and the two redundant programs simultaneously run on the processor. The technique exploits several significant micro architectural trends to provide broad coverage of transient faults and restricted coverage of permanent faults. These trends are simultaneous multithreading, control flow and data flow prediction, and hierarchical processors - all of which are intended for higher performance, but which can be easily leveraged for the specified fault tolerance goals.

Arvinder Kaur et al [3] empirically evaluates the performance of RF in predicting fault-prone classes using open source software. The Random Forest algorithm is evaluated using Object Oriented metrics proposed by Chidamber and Kemerer [2].

Briand et al [11] 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. They used univariate and multivariate analysis to find individual and combined impact of OO metrics and fault proneness.

Gyimothy et al empirically validated CK metrics on open source software for fault prediction. They employed regression (linear and logistic regression) and machine learning methods (neural network and decision tree) for model prediction [12].

Zhou and Leung 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 [13]. Pai also used the same data set using a Bayesian approach to predict fault proneness models [14]. Aggarwal et al. validated object-oriented metrics to predict faulty classes [15, 16].

Khoshgaftaar et al. introduced the use of the neural networks as a tool for predicting software quality [17]. They presented a large telecommunication system, classifying modules as fault prone or not fault prone. In this paper ANN model is compared with a non-parametric discriminant model.

III. Empirical Data Collection

First of all, find the structural code and design attributes of software systems. Thereafter, select the suitable metric values as representation of statement. Next step is to analyze, refine metrics and normalize the metric values. We used JEdit open source software in this study [18]. 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 [19]. 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 [20]. The following is the details of the metrics used in the classification process:

TABLE I METRIC SUIT USED IN THE STUDY

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. C_1, C_2....C_n be the complexity of the methods $WMC = \sum_{i=1}^{n} C_i$ If all the methods complexities are considered to be unity , then WMC = n the number of methods in the class.
Response for a class (RFC)	The response set of a class (RFC) is defined as set of methods that can be potentially executed in response to a message received by an object of that class. It is given by RFC= |RS|, where RS, the response set of the class $RS = M_i \cup_{all\ j} \{R_{ij}\}$
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

IV. GENETIC ALGORITHM BASED CLASSIFICATION TECHNIQUE

A genetic algorithm (GA) is a search technique used in computing to find exact or approximate solutions to optimization and search problems. Genetic algorithms are categorized as global search heuristics. Genetic algorithms are a particular class of Evolutionary Algorithms that use techniques inspired by evolutionary biology such as inheritance, mutation, selection, and crossover. This Technique used the feature of random search. Random search feature selection searches the best possible solution over a range of data. Random features and input given produce good result. In the beginning start with a large "population" of randomly generated "attempted solutions" to a problem then repeatedly do the following:

- Evaluate each of the attempted solutions
- Keep a subset of these solutions (the "best" ones)
- Use these solutions to generate a new population
- Quit when you have a satisfactory solution (or you run out of time)

With help of Genetic algorithm classification of the software components into faulty/fault-free systems is performed. The flowchart of the Genetic Algorithm based approach is shown in the following figure:

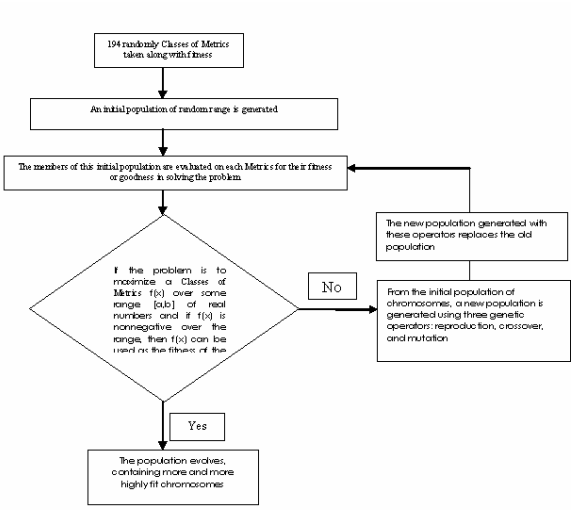

Fig. 1 Flowchart of use of GA

V. RESULT & DISCUSSION

The rules for the classification of modules are generated with help of Genetic Algorithm based approach as shown in the flowchart of figure 1 and implemented in Visual Basic 6.0 environment. The snapshot of the GUI developed is shown in figure 2.

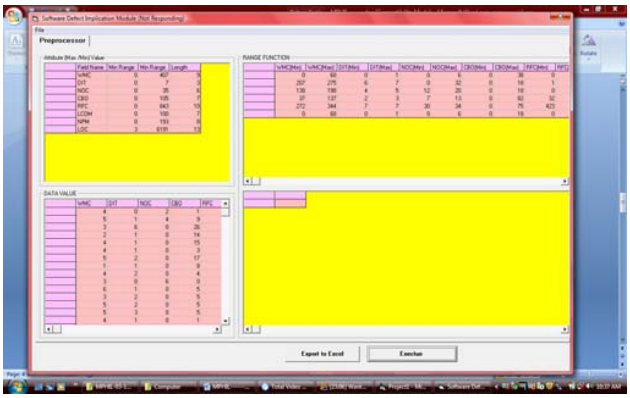

Fig. 2 Snapshot of the GUI of Developed System

The accuracy of the developed system is measured as 80.14

percent as shown in the figure 3.

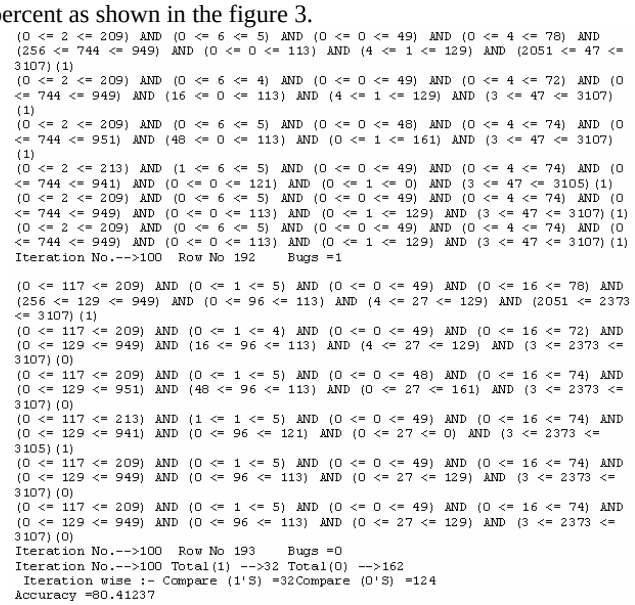

Fig. 3 Snapshot of the Output of the Developed System

VI. Conclusion

This paper empirically evaluates performance of genetic algorithm based classification technique in predicting fault-prone classes using open source software. The proposed GA based classification technique shows 80.14 percent accuracy. This study confirms that construction of GA model is feasible, adaptable to Object Oriented systems and useful in predicting faulty prone classes. It is therefore concluded that model is implemented using GA 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 hybrid genetic algorithms or soft computing techniques.

References

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

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

[3] Arvinder Kaur and Ruchika Malhotra, “Application of Random Forest in Predicting Fault-Prone Classes”, 2008 International Conference on Advanced Computer Theory and Engineering ICACTE 2008, Pukhet, Dec. 2008, pp. 37-43.

[4] 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.

[5] Saida Benlarbi,Khaled El Emam, Nishith Geol (1999), “Issues in Validating Object-Oriented Metrics for Early Risk Prediction”, by Cistel Technology 210 Colonnade Road Suite 204 Nepean, Ontario Canada K2E 7L5.

[6] Runeson, Claes Wohlin and Magnus C. Ohlsson (2001), “A Proposal for Comparison of Models for Identification of Fault-Proneness”, Dept. of Communication Systems, Lund University, Profes 2001, LNLS 2188, pp. 341-355.

[7] 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.

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

[9] Ma, Y., Guo, L. (2006), “A Statistical Framework for the Prediction of Fault-Proneness”, West Virginia University, Morgantown.

[10] Eric Rotenberg (1999), “AR-SMT: A Microarchitectural Approach to Fault Tolerance in Microprocessors”, Proceedings of the Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing, June 15-18, pp. 84-90.

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

[12] T. Gyimothy, R. Ferenc, 1. Siket, "Empirical validation of object-oriented metrics on open Trans. Software Engineering, 31 (10), 2005, pp. 897 —910.

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

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

[15] 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", Published online in Software Process Improvement and Practice, Wiley, 2008.

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

[17] T.M. Khoshgaftaar, E.D. Allen, J.P. Hudepohl, S.J. Aud, 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.

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

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

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