2011 International Conference on Computer and Software Modeling
IPCSIT vol.14 (2011) © (2011) IACSIT Press, Singapore

Fault Prediction Model for Web Application Using Genetic Algorithm

Marshima M. Rosli, Noor Hasimah Ibrahim Teo, Nor Shahida M. Yusop and N. Shahriman Mohamad

Faculty of Computer and Mathematical Sciences, Universiti Teknologi MARA, Malaysia

Abstract. The biggest challenge in the software development industry is to deliver an application with 100% defects free. However, this challenge is difficult to achieve by the software industries because it involve humans and is not an automated process done by applications, which having faults is a common things. Fault prediction is identified as one major area to predict the probability that the software contains faults. The objective of the fault prediction is to classify the software modules in the categories of faulty and non-faulty modules as early as possible in software development life cycle. In this paper, we propose fault prediction model using object oriented metrics values from web application as input values to the genetic algorithm to predict the fault probability. The aim of the proposed design model is to develop an automated tool for software development group to discover the most likely software modules in web applications to be high problematic in the future.

Keywords: software fault prone, fault proneness, fault prediction, software testing

1. Introduction

The quality of web applications is generally poor due to the weak design and poor management in the application development process [1]. The emergence and evolution of defects in the web applications may lead to faults and failures of the whole web application [2]. To improve the reliability of web applications, software development group should be supported with a fault prone prediction model that can help to detect failures as early as possible.

Fault prone prediction model requires quantitative information of an application classified as software metrics which includes number of classes, line of codes, executable statements, files, functions and blank lines as input values to predict which modules are likely to be faulty during the software development and operations. However, most of current software fault prediction model do not consider the particular attributes of object oriented design metrics that possibly will impact on the quality of web applications.

The object oriented design approach for web applications features includes size of the software, coupling, cohesion, inheritance and reuse. In this research, a set of object oriented metrics [3] was chosen to measure the design attributes particularly for web applications and will be used as input values to the fault prone prediction model. In order to validate the accuracy of proposed fault prone prediction model, genetic algorithm was selected as search based approach on natural evolution of selection [4] will be applied. So that, in the end a prototype of fault prediction model for web applications using a genetic algorithm can be shown.

2. Fault Prediction Model using Object Oriented Metrics with Genetic Algorithm

Software metrics has been widely used and proven as one of the critical attribute to measure the fault proneness of applications [5][6][7]. Past research had proposed several software product metrics include of both static and dynamic for fault proneness prediction and measuring the testing thoroughness [8]. Static metrics involve code structure measurement to identify software complexity, e.g., the McCabe's Cyclomatic number [9] or the Halstead's Software Science [10]. Dynamic metrics used structural and data flow coverage [11] to measure the testing thoroughness.

In this research, eight internal product metrics was chosen to illustrate the principal design attributes of object oriented web applications. Among them, six metrics proposed [3] are the first object oriented metrics that has been discovered to capture concept of inheritance, coupling and cohesion. They are Depth of the Inheritance Tree (DIT), Number of Children (NOC), Weighted Methods per Class (WMC), Coupling between Object classes (CBO), Lack of Cohesion (LCOM) and Response for a Class (RFC). Another two metrics are from the traditional metrics which are Number of Public Methods(NPM) and Lines of Code (LOC).

Genetic Algorithm classified as search based approach has been extensively used to solve classification [12], optimization [13] and regression [14] problems in software testing. Researches reporting the use of genetic algorithm in fault proneness prediction application are few and recent. Prediction fault proneness using genetic algorithm has been experimented and the result shall represent as influential factors to refine effort and cost estimation in inspection and testing phase [15][16][17][18].

The first application for fault prediction model [17] used genetic algorithm combined with neural networks. The experiment used nine software metrics for each module from a large scale of telecommunications system. The result from the experiments determines that genetic algorithm suitable to be a constraint to find an optimum solution to satisfy a subjective set of classification criteria for a large scale system. Another research [19] mentioned that time and test coverage are important software metrics to the software reliability growth. Therefore, the author implemented two experiments to compare the genetic programming models with other traditional and artificial neural network models. The experiment of genetic programming models based on time presented that genetic programming adapt better to the software reliability curve. The second experiments that based on test coverage data and used the Kolmogorov-Smirnov models for evaluation also favor to genetic programming and artificial neural network models. An improved result from an experiment that combined genetic programming with boosting technique was reported [20] for improvement of the software reliability growth.

Another related study [21] also adopted genetic programming algorithm to establish a software reliability model based on mean time between failure (MTBF) time series. The experiment evaluated the proposed genetic programming model with other traditional and artificial neural network models. Again, the results showed that the proposed genetic programming model has great prediction precision and better applicability. A similar and complementary study [4] about fault proneness prediction application experiment using multi-objective genetic algorithm evaluated the predictive capability of single model and multiple models using multi-objective genetic algorithm. The result showed that multiple models performed better than the single model because of the Genetic Algorithm learned weights that affect to the contribution of each model.

3. The Requirement Modeling for Fault Prediction Model using Genetic Algorithm

3.1. Application System Architecture

This section discussed on the details system architecture of the Fault Proneness Prediction application (FPP). Fig. 1 shows the components as well as the sources needed to execute the application.

There are three components needed to perform specific task which are Software Metrics Information Extractor (SMIE), Fault Classes Detection System (FCDS) and Genetic Algorithm Generator (GAG). There are two types of information that will be extracted from source files and Subversion [22] log revision which are Software Metrics Information (SMI) and percentage of Fault Class Information (FCI). SMI can be extracted using SMIE such as [23], while FCI count the number of bugs for each class in the source files. The percentage of FCI will be calculated to indicate the objective function in generating genetic algorithm by using FCDS. Finally, optimal metrics combination is determined by using GAG. GAG applies Genetic Algorithm (GA) as to its strength to find optimal solution based on the population generation. GA is chosen due to its capability in finding optimal solution as global search method [24]. Following discussed the steps involve in GA for use with FPP:

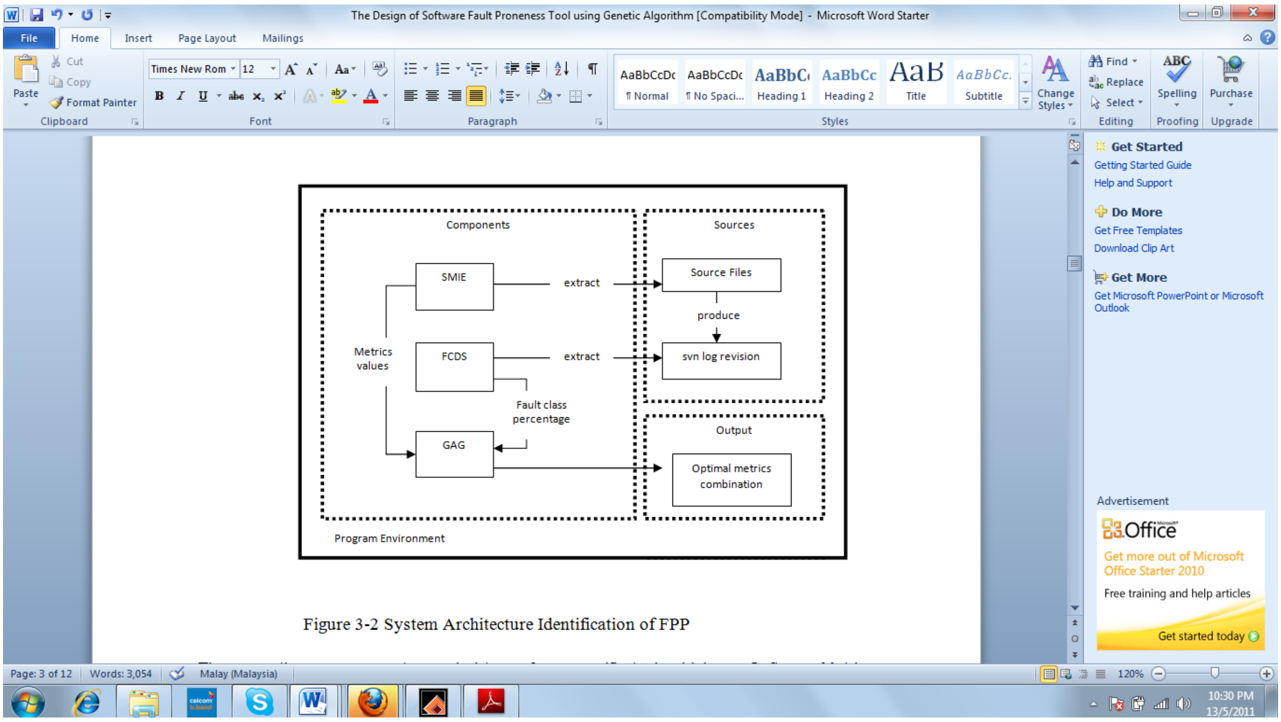


Fig. 1: System architecture identification of FPP

Chromosome representation. Chromosome is represented by SMI real values in FPP. In this research, SMI consist of eight metrics and each of it holds certain value. Each of the metrics is known as gene.

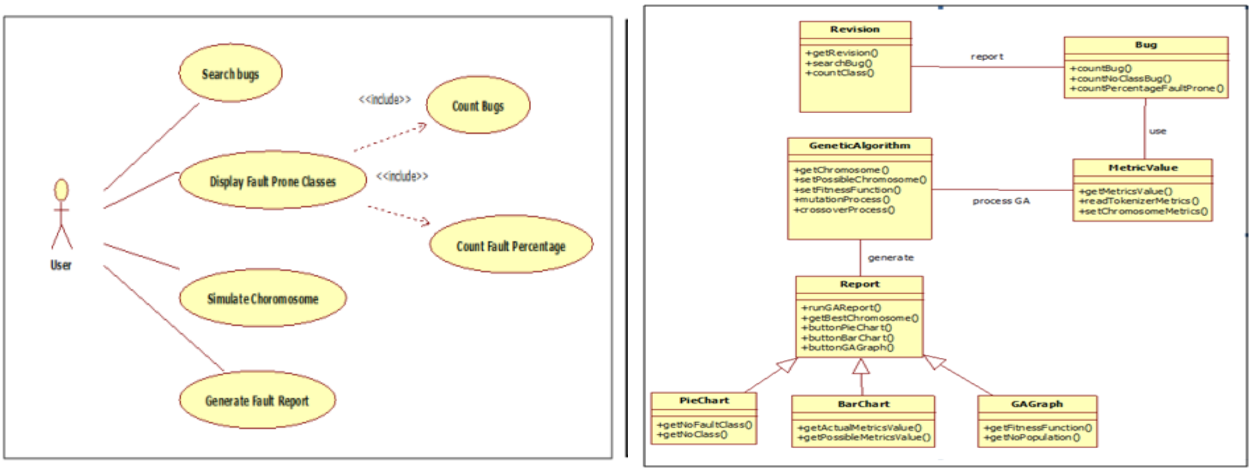
Generate initial population. Each of the chromosomes represents a population. A number of population need to be generated in order to produce better metrics combination for each of the chromosome. Each of the log revision will produce a population. Therefore, the number of the initial population is based on the number of log revision.

Calculate Fitness Function. Fitness function used to determine the optimal combination of metrics. In this research, single-constrained fitness function is applied. The fitness function used is the percentage of FCI. The fitness function objective is to maximize the percentage of FCI. Maximum FCI mean less bugs in each class. Each of the population has predefined fitness function taking along with it. Normally, two populations will be chosen randomly to be processed in order to obtain new metrics combination. Fitness function of these two populations will be compared to find the fittest population. Later, the weak population will reproduce by adapting some gene from the fittest population. The process can be obtained by using crossover and mutation.

Crossover and Mutation. Crossover process involves both weak and fit population. Crossover is done by exchanging some gene between both populations. New temporary population will be produced after this process called as offspring. This new offspring is actually a new metrics combination. Mutation will be applied to this new population with certain rate. Lower mutation rate would be better. Mutation is the final process of reproducing new population. Now, this new population fitness function needs to be calculated again and ensure that it is the highest fitness value compare to other population in dataset. This process will be iterated until the objective function has been met or the maximum iteration set has been reached.

3.2 Application Functional Requirement

This section discusses the functional requirements of Fault Proneness Prediction application. The functions of the application are captured through use case diagram and class diagram as shown in Fig. 2.



(a) Use Case Diagram (b) Class Diagram

Fig. 2: Use case diagram and class diagram for fault proneness prediction application.

3.2.1 Use Case Diagram

There are six use cases in Fig. 2(a) identified to address the functionalities as describe below:

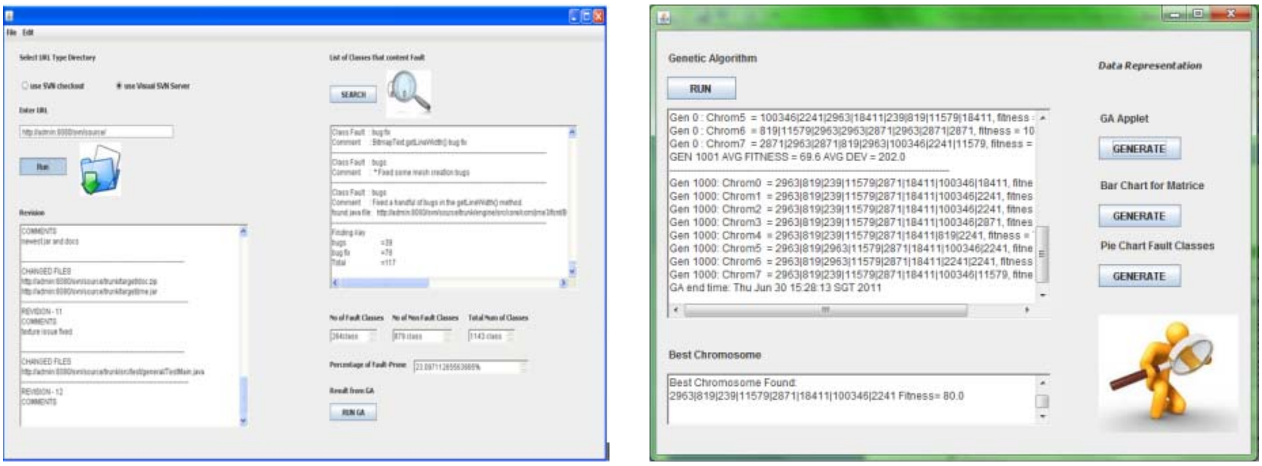
- *Search Bugs.* Search bugs from the URL specified by the user. There are two options to download the Revision either using SVN checkout or Visual SVN Sever to browse the log revision.
- *Display Fault Proneness Classes.* Display the path of class fault, the number of class fault and the number of attributes that been detected.
- *Count Bugs.* The Revision data will be used as an input to search the contents with match to the attributes of findings key in the comments. All attributes that had been found is counted and saved. The attributes that had been detected will display class fault, comment and class path.
- *Count Fault Percentage.* The total number of fault class is getting by counting the class path for the only attributes that been detected. The total number of class is process by counting the file format .java in the directory file. The data is store for the calculation of percentage purpose in Genetic Algorithm.
- *Simulate Chromosome.* The metrics value (consists of 8 bits) is used as the chromosome. This chromosome is manipulated by using mutation and crossover method in Genetic Algorithm. Genetic Algorithm will be run until the actual chromosome is getting the best chromosome fitness.
- *Generate Fault Report.* There are three options to display the fault reports which are GA Applet, Bar Chart and Pie Chart.

3.2.2 Class Diagram.

The UML class diagrams represent the static view of an application which consists of classes and their relationship. Fig. 2(b) shows of design classes with methods and relationships with eight classes identified; *Revision, Bug, GeneticAlgorithm, MetricValue* and *Report*. While *PieChart, BarChart* and *GAGraph* are subclasses that inherits all of attributes and methods of parent *Report* class. In the diagram, Revision class stored java source code file that is used for filtering process and the result will be used for bugs' calculation in Bug class. In order to identify which classes contain fault-prone, genetic algorithm will be applied as the searching technique. In simulating the chromosome, eight metrics values which are stored in MetricValue class will be used.

4. The Fault Proneness Prediction Prototype

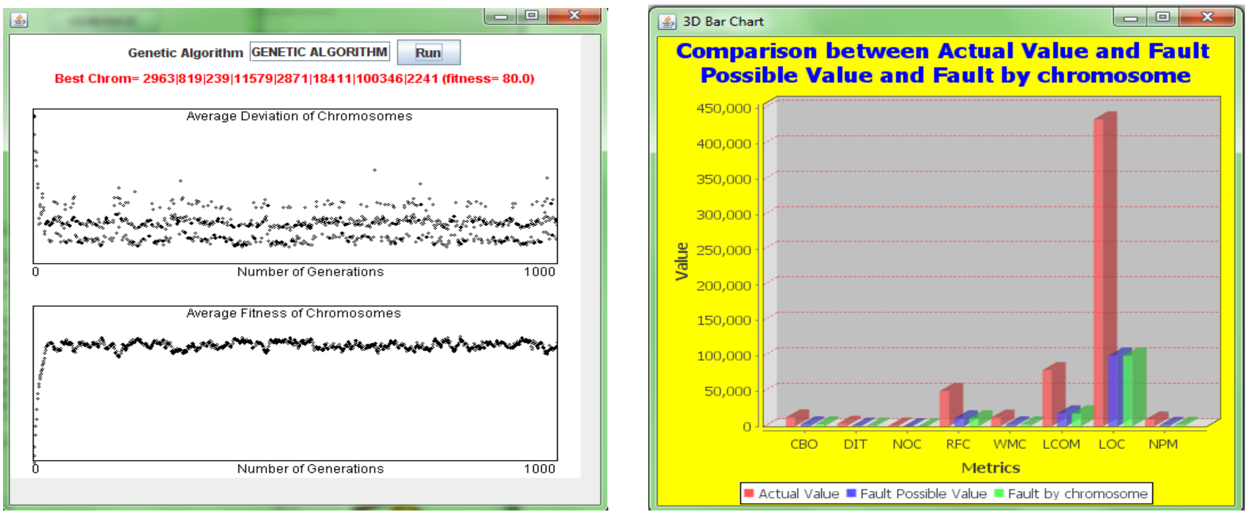
The defect data applied in the Fault Proneness Prediction Application is from OpenCms, an open source Web Content Management [25] application based on J2EE platform. The OpenCms project was released in 2000 and applied the Concurrent Version Software (CVS) repository to manage the source code and fault tracking process. The user interface of Fault Proneness Prediction prototype was written in Java is shown in Fig. 3.



(a) Main Interface (b) Genetic Algorithm Process

Fig. 3: User interface of fault proneness prediction prototype

The metrics values from the defect data source was manipulated using the mutation and crossover method in Genetic Algorithm as shown in Fig. 3(b). This process will be run iteratively until it reached the best chromosome fitness. The prototype visualizes the result by using Genetic Algorithm Applet and 3D Pie chart as shown in Fig. 4.



(a) Genetic Algorithm Applet (b) 3D Bar Chart

Fig. 4: Result from the fault proneness prediction prototype

The Genetic Algorithm Applet in Fig. 4(a) shows the average deviation and average fitness of chromosomes and the bold lines visualized the chromosomes are approach toward the best chromosomes. The 3D Bar chart in Fig. 4 (b) shows the comparison between actual values of object oriented metrics and possible fault values of object oriented metrics and chromosome predict by the genetic algorithm for the OpenCms application.

5. Conclusions

The contribution of this paper is to present the framework of fault proneness prediction application, a requirement model to show components interaction and a prototype to verify the proposed fault prediction model. The proposed model result shows that construction of fault proneness prediction using genetic algorithm is feasible, adaptable to object oriented metrics and significant for web applications. For future work, further investigation can be studied on the specific web applications metrics and the quality of web applications defect data to be use in the proposed fault prediction model.

6. Acknowledgements

The authors would like to thank Universiti Teknologi MARA Malaysia for the financial support.

7. References

[1] A. Ginige and S. Murugesan. Web Engineering: An introduction. *Multimedia, IEEE.* 2001, 8:14-18.

[2] S. Pertet and P. Narasimhan. Causes of Failure in Web Applications. *Carnegie Mellon University Parallel Data Laboratory technical report CMU-PDL-05-109*. 2005.

[3] S. R. Chidamber and C. F. Kemerer. A Metrics Suite for Object Oriented Design. *IEEE Transactions on Software Engineering.* 1994, 20(6):476-493.

[4] S. H. Aljahdali and M. E. El-Telbany. Software Reliability Prediction using Multi-objective Genetic Algorithm. *Proc*. *IEEE/ACS International Conference on Computer Systems and Applications*. Morocco, 2009: 293 - 300

[5] J. F. Ramil and M. M. Lehman. Metrics of Software Evolution as Effort Predictors-a Case Study. *Proc. International Conference on software Maintenance*. San Jose, CA, 2000: 163 - 172.

[6] V. Y. Shen, T. J. Yu, S. M. Thebaut, and L. R. Paulsen. Identifying Error-prone Software—An Empirical Study. *Software Engineering, IEEE Transactions on Software Engineering.* 1985, 11(4):317-324.

[7] R. W. Selby and V. R. Basili. Analyzing error-prone system structure. *IEEE Transactions on Software Engineering.* 1991, 17(2): 141-152.

[8] P. Yu, T. Systa, and H. Muller. Predicting fault-proneness using OO metrics. An Industrial Case Study. *Proc. 6th European Conference on Software Maintenance and Reengineering*. Budapest, Hungary, 2002: 99-107.

[9] T. J. McCabe and C. W. Butler. Design Complexity Measurement and Testing. *Communications of the ACM.* 1989, 32: 1415-1425.

[10] N. E. Fenton and S. L. Pfleeger. *Software metrics: A rigorous and Practical Approach.* London: International Thomson Computer Press, 1997

[11] P. G. Frankl and E. J. Weyuker. Provable improvements on branch testing. *IEEE Transactions on Software Engineering.* 1993: 962-975.

[12] D. Doval, S. Mancoridis, and B. S. Mitchell, "Automatic clustering of software systems using a genetic algorithm. *Proc. Conference of Software Technology and Engineering Practice*. England ,1999: 73-81.

[13] D. Berndt, J. Fisher, L. Johnson, J. Pinglikar, and A. Watkins. Breeding Software Test Cases with Genetic Algorithms. *Proc. 36th Haiwaii International Conference on System Sciences (HICSS 36).* Hawai, 2003: 338a

[14] Z. Li, M. Harman, and R. M. Hierons. Search Algorithms for Regression Test Case Prioritization. *IEEE Transactions on Software Engineering.* 2007: 225-237.

[15] L. Tian and A. Noore. Evolutionary neural network modeling for software cumulative failure time prediction. *Reliability Engineering & System Safety.* 2005*,* 87: 45-51.

[16] J. R. Birt and R. Sitte. Identifying Error Proneness in Path Strata with Genetic Algorithms. *Proc. 12th Asia Pacific Software Engineering Conference (ASPEC)*. Taiwan, 2005: 8

[17] R. Hochman, T.M. Khoshgoftaar, E.B. Allen and J.P. Hudepohl. Using the Genetic Algorithm to Build Optimal Neural Networks for Fault-Prone Module Detection. *Proc. 7th International Symposium of Software Reliability Engineering*. New York, 1996:152-162.

[18] L. Tian and A. Noore. On-line Prediction of Software Reliability using an Evolutionary Connectionist Model. *Journal of Systems and Software.* 2005*,* 77: 173-180.

[19] E. O. Costa, S. R. Vergilio, A. Pozo, and G. Souza. Modeling software reliability growth with genetic programming. *Proc. 16th IEEE International Symposium of Software Reliability Engineering*. Washington, DC, USA, November 2005:171 - 180

[20] E. O. Costa, G. A. de Souza, A. T. R. Pozo, and S. R. Vergilio. Exploring Genetic Programming and Boosting Techniques to Model Software Reliability. *IEEE Transactions on Reliability*. 2007, 56: 422-434.

[21] Y. Zhang and H. Chen. Predicting for MTBF Failure Data Series of Software Reliability by Genetic Programming Algorithm. *Proc. 6th International Conference on Intelligent Systems Design and Applications*. Jinan, 2006: 666 - 670

[22] B. Collins-Sussman, B. W. Fitzpatrick, and C. M. Pilato, *Version control with subversion*.2nd ed. O'Reilly Media, Inc., Sebastopol, 2008: pp.121 - 122.

[23] Rüdiger Lincke, Jonas Lundberg, Welf Löwe. *Proc of the 2008 international symposium on Software testing and analysis, Washington,* 2008:131-142.

[24] B. Pratibha and K. Manoj. Genetic Algorithm - an Approach to Solve Global Optimization Problems. *Indian Journal of Computer Science and Engineering*. 2010, 1(3): 199-206.

[25] D. Liliedahl. *OpenCms 7 Development*. Birmingham, U.K.:Packt Publishing Limited, 2008.