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

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:
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.
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 Server 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.
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.

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


