Investigation of Grid partitioning and Subtractive Clustering based Neuro-Fuzzy Systems for Evaluation of Fault Proneness in Open source software system

Raminder Preet Kaur and Amanjot Singh Klair

Abstract—Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly important and testing related issues are becoming crucial for software. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. 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.

Keywords— Fault-prone modules; scheduling; project control; Quality; Testing.

I. INTRODUCTION

A software fault is a defect that causes software failure in an executable product. Faults in software systems continue to be a major problem. Many systems are delivered to users with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. A limited amount of valuable work in this area has been carried out previously. Despite this it is difficult to identify a reliable approach to identifying fault-prone software components. Using software complexity measures, the techniques build models, which classify components as likely to contain faults or not. The modeling techniques applied cover the main classification paradigms, including principle component analysis, discriminate analysis, logistic regression, layered neural networks etc. Timely predictions of faults in software modules can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults. Prediction models based on software metrics, can estimate number of faults in software modules. Software metrics are attributes of the software system and may include process, product, and execution metrics.

II. FAULT AND FAILURE

Software is said to contain a fault if for some input data the output is incorrect. For each execution of the software program where the output is incorrect, we observe a failure. Software engineers distinguish software faults from software failures. In case of a failure, the software does not do what the user expects but on the other hand fault is a hidden programming error that may or may not actually manifest as a failure. A fault can also be described as an error in the correctness of the semantic of a computer program. A fault will become a failure if the exact computation conditions are met, one of them being that the faulty portion of computer software executes on the CPU. A fault can also turn into a failure when the software is ported to a different hardware platform or a different compiler, or when the software gets extended. The classification of failure is listed in Table 1.1. Software faults are all due to human errors in creating the software where as the hardware faults are due to random phenomena such as aging, external intervention etc.

<table>
<thead>
<tr>
<th>Failure class</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transient</td>
<td>Occurs only with certain inputs</td>
</tr>
<tr>
<td>Permanent</td>
<td>Occurs with all inputs</td>
</tr>
<tr>
<td>Recoverable</td>
<td>System can recover without operator intervention</td>
</tr>
<tr>
<td>Unrecoverable</td>
<td>Operator intervention is required to recover from failure</td>
</tr>
<tr>
<td>Non-corrupting</td>
<td>Failure does not corrupt system state or data</td>
</tr>
<tr>
<td>Corrupting</td>
<td>Failure corrupts system state or data</td>
</tr>
</tbody>
</table>

Table 1: CLASSIFICATION OF FAILURE

Raminder Preet Kaur is M.Tech. CSE Student at IITT(Pojewal), Punjab Technical University, Jalandhar. (e-mail:raminder.punjab5@yahoo.co.in)
Amanjot Singh Klair is M.Tech. CSE Student at IITT(Pojewal), Punjab Technical University, Jalandhar. (e-mail:jotklair@gmail.com).
III. FAULT PREDICTION

Many systems are delivered to users with excessive faults. This is despite a huge amount of development effort going into fault reduction in terms of quality control and testing. It has long been recognized that seeking out fault-prone parts of the system and targeting those parts for increased quality control and testing is an effective approach to fault reduction. Lanubile et al. [1] presented an empirical investigation of the modeling techniques for identifying fault-prone software components early in the software life cycle. The objective of a fault-proneness model is to identify faulty classes and focus testing effort on them. When a software system is developed, the majority of faults are found in a few of its modules. In most of the cases, 55% of faults exist within 20% of source code. It is, therefore, much of interest is to find out fault-prone software modules at early stage of a project (Saida and Nishith Geol, 1999). The faults that reside in software products are not evenly distributed over the software modules; some modules are more fault-prone than others [2]. Using software complexity measures, the techniques build models, which classify components as likely to contain faults or not. Quality will be improved as more faults will be detected. Predicting faults early in the software life cycle can be used to improve software process control and achieve high software reliability. Timely predictions of faults in software modules can be used to direct cost-effective quality enhancement efforts to modules that are likely to have a high number of faults. Prediction models based on software metrics, can estimate number of faults in software modules.

IV. FAULT PREDICTION TECHNIQUES

To predict the fault in software data a variety of techniques have been proposed which includes statistical method, machine learning methods and neural network techniques. Statistical methods are used to find an explicit numerical formula, which determines completely how classification is performed.

Machine learning is concerned with the design and development of algorithms and techniques to extract rules and patterns out of massive data sets. Machine Learning is part of Machine Intelligence but addresses a more specialized purpose and scope. Machine learning algorithms and applications adapt themselves to the behavior of a system usually through the discovery of time-varying patterns in the data. These algorithms typically fuse linear and nonlinear regression, adaptive control theory, neural networks, statistical learning theory, rule induction, and decision tree generation. Because of the very close relationship between learning and intelligence, nearly all machine intelligence systems incorporate some form of learning.

Neural networks, which have been already applied in software engineering applications, to build reliability growth models predict the gross change or reusability metrics. A neural network is trained to reproduce a given set of correct classification examples, instead to produce formulas or rules [3]. Neural networks are non-linear sophisticated modeling techniques that are able to model complex functions. Neural network techniques are used when exact nature of input and outputs is not known. A key feature is that they learn the relationship between input and output through training.

V. SYSTEMS UNDER STUDY

Clustering is used to determine the intrinsic grouping in a set of unlabeled data. It is the process of organizing objects into groups whose members are similar in some way. Among various clustering techniques available in literature Grid partitioning and Subtractive Clustering Based Approach is most widely being used. Hence, in this study, a Grid partitioning and Subtractive clustering based Neuro-fuzzy systems [4][5] is used for finding faulty Modules in Open Source Software System JEdit[6]. 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 [7]. Hence investigate the accuracy of the fault proneness predictions using object oriented design using metrics suite given by Chidamber and Kemerer [8] and used for fault prediction.

Use of Grid partitioning and Subtractive clustering based Neuro-fuzzy systems

Use Fuzzy Clustering algorithm for clustering of the software components into faulty/fault-free systems. Here, fuzzy logic will be employed to capture the broad categories identified during clustering into a Fuzzy Inference System (FIS). The FIS will then act as a model that will reflect the relationship between the different input parameters.

Subtractive clustering is a fast, one-pass algorithm for estimating the number of clusters and the cluster centers in a set of data. This means that the computation is now proportional to the problem size instead of the problem dimension. Since each data point is a candidate for cluster centers, a density measure at data point xi is defined as:

\[ D_{c_1} = \min (D_{c_2}, \ldots, D_{c_k}) \]

Where \( r_a \) is a positive constant representing a neighborhood radius. Hence, a data point will have a high density value if it has many neighboring data points. The first cluster center \( x_{c_1} \) is chosen as the point having the largest density value \( D_{c_1} \). Next, the density measure of each data point xi is revised as follows:

\[ D_{c_1} = \min (D_{c_2}, \ldots, D_{c_k}) \]

Where \( b_r \) is a positive constant which defines a neighborhood that has measurable reductions in density measure. Therefore, the data points near the first cluster center \( x_{c_1} \) will have significantly reduced density measure. After revising the density function, the next cluster center is selected as the point having the greatest density value. This process continues until a sufficient number of clusters are obtained.

VI. NEURO-FUZZY SYSTEM

To predict the results, we have used confusion matrix as
shown in Table II. The confusion matrix has four categories: True positives (TP) are the modules correctly classified as faulty modules. False positives (FP) refer to fault-free modules incorrectly labeled as faulty. True negatives (TN) are the fault-free modules correctly labeled as such. False negatives (FN) refer to faulty modules incorrectly classified as fault-free modules.

A confusion matrix of prediction outcomes

<table>
<thead>
<tr>
<th>Predicted</th>
<th>Real Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fault</td>
<td>TP</td>
</tr>
<tr>
<td>Fault</td>
<td>FP</td>
</tr>
<tr>
<td>No Fault</td>
<td>FN</td>
</tr>
<tr>
<td>No Fault</td>
<td>TN</td>
</tr>
</tbody>
</table>

The following set of evaluation measures are being used to find the results:

Probability of Detection (PD), also called recall or specificity, is defined as the probability of correct classification of a module that contains a fault.

\[ PD = \frac{TP}{TP + FN} \]

Probability of False Alarms (PF) is defined as the ratio of false positives to all non-defect modules.

\[ PF = \frac{FP}{FP + TN} \]

The comparisons are also made on the basis of the more accuracy and least value of MAE and RMSE error values. Accuracy value of the prediction model is the major criteria used for comparison. The mean absolute error is chosen as the standard error. The technique having lower value of mean absolute error is chosen as the best fault prediction technique.

Mean absolute error

Mean absolute error, MAE is the average of the difference between predicted and actual value in all test cases; it is the average prediction error [23]. The formula for calculating MAE is given in eq. (5).

\[ \text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |a_i - c_i| \]

Assuming that the actual output is \( a \), expected output is \( c \).

Root mean-squared error

\( \text{RMSE} \) is frequently used measure of differences between values predicted by a model or estimator and the values actually observed from the thing being modeled or estimated [23]. It is just the square root of the mean square error as shown in eq. (6).

\[ \text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (a_i - c_i)^2} \]

The mean-squared error is one of the most commonly used measures of success for numeric prediction. This value is computed by taking the average of the squared differences between each computed value and its corresponding correct value. The root mean-squared error is simply the square root of the mean-squared-error. The root mean-squared error gives the error value the same dimensionality as the actual and predicted values.

VII. CONCLUSION AND FUTURE SCOPE

This paper explains the methodology to be used for empirically evaluates performance of Grid partitioning and Subtractive clustering based Neuro-fuzzy systems in predicting fault-prone classes using open source software.

The contributions of the study can be 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 Grid partitioning and Subtractive clustering based Neuro-fuzzy systems to predict the faulty classes with better accuracy.

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