FAULT PREDICTION OF OBJECT ORIENTED DESIGN USING A HYBRID ANFIS PREDICTION MODEL

N. Rajkumar, C. Viji, S. Duraisamy

ABSTRACT

The necessity to increase the quality of a product has been gaining wide significance with new methodologies aiming to increase the efficiency of the existing quality enhancing methodologies. A major attribute to quality enhancement is the ability to detect fault prone modules in the developed system. The proposed work exploits the relationship between object oriented metrics (OOM) and the fault proneness of the design system in an empirical manner. The prominent design metrics may be cohesion, inheritance, coupling etc. Two aspects of fault prediction have been investigated in the proposed work. A statistical prediction for classifying faults and conceptuality to relate fault with classes have been suggested. A neuro-fuzzy approach is utilised for predicting and classifying the faults. Based on the observations, it is concluded, that the proposed model provides high accuracy in discrimination between faulty and fault-free classes. Besides the size of classes, the frequency of method invocations and the depth of inheritance hierarchies seem to be the main driving factors of fault proneness. An ANFIS based model for predicting faulty classes with a data set of 965 members have been experimented in this paper. Training has been done using LM model and gradient descent methods and set of 15 rules developed for this FIS. The observed results show good prediction of faulty classes.

KEYWORDS: Faulty classes, Object Oriented Metrics, Neural Fuzzification, Quality Analysis.

I. INTRODUCTION

Assessment of quality for a software product is an important protocol in almost all fields of software engineering. Increasing complexity in the designed software to cope with the recent demands make the quality assessment all the more important. At the same time, critical metrics of object oriented systems like fault, coupling, cohesion, reliability, inheritance play a vital role in quality assessment. In spite of several software testing techniques available, quality of the software does not reach up to 100%. Conventional testing methods also fail in testing certain areas of the design due to incompatibility issues thus reducing the efficiency of testing algorithms. A great deal of time could be reduced if only those areas like classes, inheritance which are fault prone could be tested for faults. The focus of the proposed work is to improve the quality of the product using an efficient fault prediction technique. Identification of faulty areas could be located with techniques like regression and learning methods with metrics like CK (Chidamber and Kemerer (1993)) metric suite, Briand metrics, MOOD (Metrics for object oriented designs), QMOOD (Quality metric for object oriented design), L&K (Lorenz and Kidd (1994)) etc. for prediction. Predictive models of software faults use historical and current development data to make predictions about faultiness of software subsystems/modules [1] where a segment of previously known data is analysed against other the other segments for predicting performance. Fault prediction is an important class as it not only predicts fault but also provides suitable verification and validation methodologies. A wide range of metrics are available to evaluate the software quality like MOOD metrics [2], CK metric [3]. This study mostly focuses on the aspect of improving reliability of a software by predicting and reducing the number of faults in the software using a Neuro-fuzzy approach which have the ability to determine the properties of a class which is very vital in fault prediction.

II. RELATED WORK

Significant work has been done in the area of fault detection of software systems varying from statistical to machine learning methods. Prominent statistical methods involve principal component analysis and discriminant analysis [4] and a model based fault detection [5] using object oriented metrics. A strong relationship between size and fault proneness [6] was establish by El Emam et al. A univariate and multi-variate analysis for fault detection was proposed by Briand et al [7] considering 180 classes. Spatial clustering techniques for fault detection [15] have been shown to have accurate detection rates. Tang et al. [17] conducted an empirical study on three industrial real time systems and validated the CK object oriented metric suite and found WMC (Weighted methods per class) and RFC (Response for Class) to be strong predictors of faulty classes. On the other hand, a number of machine learning methods [13] have also been developed to increase the efficiency of the fault detection techniques. Zhou et al. [9] have used logistic regression and machine learning methods to show how object oriented metrics and fault using CK metrics. Bieman and Ott developed a set of functional cohesion measures based on program slices [16]. These measures apply only to individual functions. 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 as high and low. Aggarwal [18] validated object oriented metrics to predict faulty classes. An empirical evaluation [19] for predicting the performance of RF in predicting fault-prone classes using open source software and also a Genetic Algorithm based classification approach [20] for finding Fault Prone Classes have been able to provide significant fault prediction models. In this paper Genetic algorithm approach can be used for finding the components and Kemerer developed an effective method of identifying the
most non-cohesive classes, but it is not effective at distinguishing between partially cohesive classes known as LCOM (Lack of Cohesion). LCOM indicates lack of cohesion only when, compared pair wise, fewer than half of the paired methods use the same instance variables. WMC, CBO (Coupling between objects) and SLOC (Source lines of code) are also found to be strong predictors using various machine learning methods. Other techniques involve the Bayesian belief method [10], artificial neural network scheme and support vector machine [11] for classification of faults. Conceptual cohesion of classes is used for fault prediction in object oriented software systems [13]. Fuzzy Logic [14] has been successfully applied to determine faults by developing a set of rules for fault detection. The works of Khoshgoftaar et al. Based on ANN (artificial neural network) model for predicting fault classes have shown good accuracy and since then boosted the usage of ANN for prediction modelling. A study of the related work shows that the object oriented metrics are positively associated with the fault proneness of a software system [12] and [22].

III. PROPOSED WORK
Object-Oriented metrics play a crucial role in predicting faults. In literature, prediction models are mostly developed using statistical models. The proposed work aims to establish the relationship between object oriented metrics and fault proneness at class level. Fault has been taken as a function of WMC, NOC, DIT, RFC, CBO and LCOM of the CK metric suite. Neural networks (NN) are an ideal choice for problem definitions involving prediction and classification. The neuro-fuzzy based system approach learns the rules and membership functions from data. It is proven in many proposed techniques that neuro-fuzzy give the better results as compared to standalone FIS or ANN because it uses the power of rules decision of FIS and adaptive nature of ANN in a single system together. In addition the results of this work show that conceptual relations between classes could also be used as a good metric for prediction of fault proneness. Figure 1 depicts a neuro fuzzy modelling for fault prediction.

**Figure 1: Fault Prediction using ANFIS model**
Adaptive neuro-fuzzy inference system (ANFIS) constructs a fuzzy inference system (FIS) for the given input/output parameters whose membership function parameters are adjusted using either a back propagation algorithm alone or in combination with a least squares type of method. This adjustment allows the Fuzzy systems to learn from the data they are modelling. ANFIS is applicable to the modelling situations where it is difficult to discern what the membership functions should look like simply from looking at data. A 3 layer ANFIS structure is shown below in figure 2.

**Figure 2: Scheme of two layer of ANFIS**
Six CK metrics have been used as the input nodes and prediction rate as the achieved output. The proposed hybrid ANFIS along with the Levinson-Marquardt updation has been used to design a classifier to for detecting fault proneness. The ANN consists of three layers namely input layer, hidden and output layer. An activation function is used in the input layer and an S or sigmoideal function used in the next two layers. The neural network output is specified as

\[ y^1 = f(W, X) \]  

where X represents the input vector and W represents the weight function. The objective is to reduce the mean squared error by updating the weights. The weights are updated using Levinson Marquardt method (L-M method) during the learning phase. The update equation is given by

\[ W_{k+1} = W_k - (J^T J + \mu I)^{-1} J e_k \]  

Where J is the Jacobian matrix and \( \mu \) is the combination coefficient.

The circular nodes represent nodes that are fixed whereas the square nodes are nodes that have parameters to be learnt. One of the neuro-fuzzy advantages is that it uses a hybrid learning procedure for estimation of the premise and consequent parameters. The algorithm could be summarized as

**step 1: Data collection from repository**
**step 2: Normalization and categorization of data set**
**step 3: Model design and training phase**
**step 4: Weight updation**
**assign an initial weight to each metric**
**W_i = 0**
**determine pair of fault and fault free classes from training set**
\[ X = \{ x_1, x_2, \ldots \}; Y = \{ y_1, y_2, \ldots \} \]  
where \( X \) = faulty class and \( Y \) = fault free class.
**Update weight using**
\[ W_{k+1} = W_k - (J^T J + \mu I)^{-1} J e_k \]
**step 5: validation of results and classification based on threshold**
In this process by keeping fixed the premise parameters, it estimates them in a forward pass and then in a backward pass by keeping fixed the consequent parameters the process would be continued.

**Figure 3: Fuzzy controller for fault detection (WMC, NOC, and DIT)**
The grade estimation could be made more accurate by utilizing both neural network and fuzzy logic. The membership function \( \mu_A(x_i) \) corresponds to the input \( x = [x_1, x_2, ..., x_k] \) and the rule set is given as follows:

Rule1: if \( x \) is \( A_1 \) and \( y \) is \( B_2 \),
then \( f_1 = p_1 x + q_1 y + 1 \)

Rule2: if \( x \) is \( A_2 \) and \( y \) is \( B_3 \),
then \( f_2 = p_2 x + q_2 y + 2 \)

The implication and rule consequences are given as:

\[
f = \frac{w_1 f_1 + w_2 f_2}{w_1 + w_2}
\]  

(3)

The performance of the model with respect to the predictions made on the test data set would make the network to perform more accurate that both the other methods. Back-propagation is the most commonly used learning algorithm in order to train multilayer feed forward networks. A training input data set is applied, weights of neurons are fixed and a set of outputs are produced as response. During the backward pass an error is generated as the difference between the networks actual and desired output. The error is taken through feedback and weights are adjusted to minimize the error through successive iterations. Depending upon the input values of the metric, some rules out of the total 27 rules from the knowledge base gets fired. The Mamdani inference engine is used to determine the degree of membership of firing.

IV. RESULTS AND DISCUSSION

The datasets used in this study come from the NASA Metrics Data Program (MDP) data repository named as \( C_3 \). The \( C_3 \) software is written in Java programming Language and six CK metrics are taken as input, and output is the fault prediction accuracy rate required for developing the software. The network is trained using Gradient descent method and Levenberg Marquardt method. Since the proposed work deals with computation of \( C_3 \), the conceptual similarities between the methods in the class are shown in Table 1. Similarity of methods in classes represents the conceptual cohesion of classes which measures the degree to which the methods in a class are conceptually related. This is used to determine whether a class represents a single concept. A class is viewed of as a set of methods \( \{M_1, M_2, M_3, M_4, ..., M_k\} \). The methods from the class are connected by set of weighted edges and the weight of the edge is defined as the conceptual similarity measure (CSM). The CSM has been extended to compute the average conceptual similarity of methods of a class \( \text{ACSM}(c) \) determined as:

\[
\text{ACSM}(c) = 1/N(\sum_{i=1}^{N} \text{CSM}(M_i, M_k)) \tag{4}
\]

where a \( \text{ACSM}(c) = 0.5 \) is considered for pairs of different methods. \( C_3 \) is an average measure where some methods are closely related while some are not and the average is around 0.5.

### Table 1: Similarity of methods in classes

<table>
<thead>
<tr>
<th></th>
<th>( M_1 )</th>
<th>( M_2 )</th>
<th>( M_3 )</th>
<th>( M_4 )</th>
<th>( M_5 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( M_1 )</td>
<td>1</td>
<td>0.23</td>
<td>0.71</td>
<td>0.36</td>
<td>0.41</td>
</tr>
<tr>
<td>( M_2 )</td>
<td>0.30</td>
<td>1</td>
<td>0.85</td>
<td>0.57</td>
<td>0.58</td>
</tr>
<tr>
<td>( M_3 )</td>
<td>0.39</td>
<td>0.39</td>
<td>1</td>
<td>0.22</td>
<td></td>
</tr>
<tr>
<td>( M_4 )</td>
<td>1</td>
<td>0.88</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( M_5 )</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>1</td>
</tr>
</tbody>
</table>

The above table is used as an indicator to depict the closeness of the methods in classes. To begin with the proportion of the total variation in the dependant variable \( y \) based on the regression model is computed. Precision, Correctness and Completeness are used to evaluate the logistic regression model developed above. Precision here is used to evaluate how well the model classifies faulty and non-faulty classes. 965 numbers of classes were used out of which 775 contain zero faults and WMC was found to have the highest number of classes containing the same name. The results of the univariate logistic regression indicate that the model based on \( C_3 \) is better than any other model except that of LCOM.

The data set is normalized using Min – Max normalization and segregated into three categories as training set, validation set and testing set. The Min – Max normalization is defined as:

\[
n = \frac{n - \text{min(attr)}}{\text{max(attr)} - \text{min(attr)}}
\]

Where \( attr \) indicate the attribute value.

The output data set is defined as the target data set. After the training process is completed, the model is reapplied to classify the test data. The output values so obtained ranging between 0 and 1 which are indecisive for data classification. A fault is used as a dependent variable and each of the CK metric is an independent variable. It is intended to develop a function between fault of a class and CK metrics suite. Fault is a function of WMC, NOC, DIT, RFC, CBO and LCOM. Logistic regression is the commonly used statistical technique. Logistic Regression is used as the predictor of the outcome as a function of input variables. It is basically used for construction a predictor for fault proneness of classes. Most common evaluation parameters are sensitivity, specificity, precision, region of convergence. Sensitivity is a measure of how many classes are predicted to be fault prone to the actual number of faulty classes while specificity is the measure of predicted non faultiness to actual number of non-faulty classes.

### Table 2: Statistics of Metrics under consideration

<table>
<thead>
<tr>
<th>Metric</th>
<th>Mean</th>
<th>Median</th>
<th>St. Deviation</th>
<th>Min</th>
<th>Max</th>
</tr>
</thead>
<tbody>
<tr>
<td>WMC (Weighted methods/Class)</td>
<td>9.34</td>
<td>6.02</td>
<td>9.063</td>
<td>0</td>
<td>56</td>
</tr>
<tr>
<td>NOC (Number of Children)</td>
<td>0.197</td>
<td>0</td>
<td>1.602</td>
<td>0</td>
<td>22</td>
</tr>
<tr>
<td>DIT (Depth of inheritance)</td>
<td>1.344</td>
<td>1</td>
<td>1.099</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>RFC (Response for a class)</td>
<td>22.021</td>
<td>13</td>
<td>27.01</td>
<td>0</td>
<td>164</td>
</tr>
<tr>
<td>CBO (Coupling between objects)</td>
<td>7.652</td>
<td>4</td>
<td>8.702</td>
<td>0</td>
<td>76</td>
</tr>
<tr>
<td>LCOM (Lack of cohesion in methods)</td>
<td>55.09</td>
<td>6</td>
<td>138.05</td>
<td>0</td>
<td>1420</td>
</tr>
</tbody>
</table>

Table 2 summarizes the performance measures determined for a set of classes. The above values denote percentile measures with which correlation between the observed class and other classes could be established. The correlation measure is used to analyse the dependency of the one metric with the other classes. Correlation normally takes a range of 0 – 1 where a value nearer to 1 indicates a good correlation or dependence while the values towards 0 indicate a poor dependence. Table 3 illustrates the evaluation metrics for an ANN based and ANFIS based technique.

---

<table>
<thead>
<tr>
<th>Evaluation Measure</th>
<th>ANN model</th>
<th>Proposed model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sensitivity</td>
<td>67.18</td>
<td>67.91</td>
</tr>
<tr>
<td>Specificity</td>
<td>62.09</td>
<td>63.56</td>
</tr>
<tr>
<td>Precision</td>
<td>62.99</td>
<td>63.79</td>
</tr>
</tbody>
</table>

Region of convergence is a graphical analysis between sensitivity and specificity.

The area under the convergence curve arrives at 0.71 indicating a good prediction accuracy. The above determined results indicate that the classes WMC, CBO, RFC metrics are good predictors of fault proneness and can be used in machine learning based prediction techniques. The proposed approach is able to predict faultiness of a class with more than 90% accuracy.

Figures 5 and 7 illustrate the measured precision, correctness and completeness of various fault models and it could be seen that out of all fault metrics WMC and CBO have a significant effect on fault prediction of classes. Figure 6 shows the variation of mean square error with respect to number of epochs required for training the network.

To predict the best model, we used six machine learning techniques that measured the accuracy in terms of sensitivity, specificity, precision, and AUC (area under the curve). The cut-off point was also selected such that a balance is maintained between the numbers of classes predicted as fault and not faults prone. The ROC curve was used to calculate the cut-off point.

Fault prediction using statistical and machine learning methods were carried out by coding in MATLAB environment. Statistical methods such as linear regression and logistic regression were applied. Also machine learning techniques such as hybrid artificial neural network using gradient descent and Levenberg Marquardt methods were applied for fault prediction analysis. It can be concluded from the statistical regression analysis that out of six CK metrics, WMC appears to be more useful in predicting faults. It is to be noted that the results analysed from the literature survey involve hybrid classifiers such as neural network – genetic algorithm, neural network – particle swarm optimisation have a good classification rate while the
proposed work has an advantage of a fast convergence by utilisation of the levinson–markovard update of coefficients. More similar type of studies can be carried out on different datasets to give generalized results across different organizations.

Figure 8. Delay Analysis

Figure 9. Efficiency for various fault models

Figure 8 show that the delay analysis and it is defined as the time taken to transmit the data to the required destination. This delay has to analyse by varying the simulation time from 25 to 200 seconds. Compared to the existing ANN model, proposed NAFIS model achieves lesser delay. Figure 9 shows that the efficiency for various fault model models and it is defined as the ratio of number of received data to the average number of forwarded data. Her WMC, NOC and DIT fault modes achieves less than 75% of efficiency, CBO and RFC models has achieved between 75% and 85% of efficiency. But the proposed ANFIS fault prediction model has increases the efficiency of 95% compared to the ANN model.

V. CONCLUSION

Multivariate analysis results show that by using some of the coupling and inheritance measures, very accurate models can be derived to predict in which classes most of the faults actually lie. When predicting fault prone classes, the best model shows a percentage of correct classifications about 80% and finds more than 90% of faulty classes. It could be seen from the results obtained that the object oriented metrics were validated and the purpose of validation to prove the associativity of the metrics to the fault proneness have been met. However, WMC and ACSM cannot be taken to be standalone parameters for predicting system faults since validation with industrial systems are required to draw stronger conclusions. The severity of faults is not taken into account. There can be number of faults which can leave the system in various states e.g. a failure that is caused by a fault may lead to a system crash or an inability to open a file. An ANFIS based predictor for prediction of fault proneness and relationship between the conceptuality between methods in a class has been projected in this work.

REFERENCES