Software Defect Prediction Using Neural Networks

Rajni Jindal [#], Ruchika Malhotra[*], Abha Jain[*]

[#]*Department of Information Technology, Indira Gandhi Delhi Technical University for Women*
Kashmere Gate, New Delhi- 110006, India
[1]rajni.jindal@yahoo.co.in

[*]*Department of Software Engineering, Delhi Technological University*
Shahbad Daulatpur, Main Bawana Road, Delhi- 110042, India
[2]ruchikamalhotra2004@yahoo.com
[3]me_abha@yahoo.com

***Abstract*— Defect severity assessment is the most crucial step in large industries and organizations where the complexity of the software is increasing at an exponential rate. Assigning the correct severity level to the defects encountered in large and complex software, would help the software practitioners to allocate their resources and plan for subsequent defect fixing activities. In order to accomplish this, we have developed a model based on text mining techniques that will be used to assign the severity level to each defect report based on the classification of existing reports done using the machine learning method namely, Radial Basis Function of neural network. The proposed model is validated using an open source NASA dataset available in the PITS database. Receiver Operating Characteristics (ROC) analysis is done to interpret the results obtained from model prediction by using the value of Area Under the Curve (AUC), sensitivity and a suitable threshold criterion known as the cut-off point. It is evident from the results that the model has performed very well in predicting high severity defects than in predicting the defects of the other severity levels. This observation is irrespective of the number of words taken into consideration as independent variables.**

***Keywords*— Receiver Operating Characteristics, Text mining, Machine Learning, Defect, Severity, Neural-Network, Radial Basis Function**

I. INTRODUCTION

In today's scenario, we are at a situation where the size and thereby the complexity of the software is increasing at an exponential rate. This causes the defects to enter the software, thereby leading to functional failures [15]. The defects which are introduced in the software are of varying severity levels. As a result, these defect reports present in various bug tracking systems contain detailed information about the defects along with their ID's and associated severity level. A severity level is used by many organizations to measure the effect of defect on the software. This impact may range from mild to catastrophic wherein catastrophic defects are most severe defects which may lead the entire system to go to a crash state [1], [6]. Therefore, it becomes very essential to detect the severity level of the defects being introduced in the software.

978-1-4799-6896-1/14/$31.00 ©2014 IEEE

This would directly impact the allocation of resources and thus preparation of consequent defect setting activities. The most common issue with defect tracking systems is that; they are used for capturing daily basis information and not suitable for the fulfilling the business needs. Secondly, the data present in such systems is of highly unstructured form which is not suitable for large-scale report analysis.

Hence, to remedy these issues, an automated tool is required to collect the data present in the software repositories so that it can be analyzed and interpreted in order to make generalized conclusions. In order to do this, data mining techniques in combination with the machine learning techniques are used to mine the information available in such systems.

In this paper, we mine the information available in the PITS (Project and Issue Tracking System) database popularly used by the NASA's engineers. The proposed tool will first extract the relevant information from PITS database using a series of text mining techniques. After extraction, the tool will then predict the defect severities using machine learning techniques. The defects are classified into five categories of severity by NASA's engineers as very high, high, medium, low and very low. In this work, we have used radial basis function of neural network to predict the defects at various levels of severity. The prediction of defect severity will help the researchers and software practitioners to allocate their testing resources on more severe areas of the software. The performance measures used in the paper are Area Under the Curve (AUC) and sensitivity obtained from Receiver Operating Characteristics (ROC) analysis.

The paper is sub-divided into the following sections: Section 2 highlights the key points of available literature in the domain. Section 3 describes the methodology of the research being carried out, including a brief explanation of the Info-Gain measure and the machine learning method used for the study. The section concludes with the highlights about the parameters used for model evaluation. Section 4 presents the result analysis. Section 5 summarizes the paper and presents the scope for work which can be carried out in future.

II. LITERATURE REVIEW

A lot of work has been done in the past regarding the analysis of the defect reports present in the defect tracking systems. Cubranic and Murphy [4] proposed an automated method that would assist in bug triage based on the analysis of an incoming bug report. This kind of bug triage would help in predicting the developer that would work on the bug based on the bug description. A very effective tool based on the techniques of Natural Language Processing (NLP) was developed by Runeson et al. [19] and Wang et al. [23] that was used to detect duplicate reports. Canfora and Cerulo [3] discussed the need for software repositories in managing a new change request which may be either in a form of a bug or an enhancement feature. Also, a lot of empirical work has been carried out in finding out the faulty classes in object-oriented (OO) software systems using a number of OO design metrics [2], [6], [8], [9], [12], [13], [16], [17], [24], [26]. Although, these studies were based on finding how the OO metrics and faulty classes are related to each other, but did not focus on the severity of faults. Till date, there are only a few studies which were based on finding such relationship taking severity of faults into account.

The most popular study in field of fault severity has been done by the authors Singh et al. [22]. They have analyzed the performance of models at three levels of severity viz. high, medium and low severity faults and found that the model predicted at high severity faults is less accurate as compared to the models predicted at other severities. It was summarized that the models predicted using DT and ANN methods are better than the models predicted using LR method. Also, it was seen that the metrics such as CBO, WMC, RFC and SLOC are important for the faults at all the severity levels. On the other hand, DIT metric is not important for the faults at any of the severity levels. Somewhat same results were also concluded in the paper by Zhou and Leung [25]. They have investigated the fault-proneness prediction performance of OO design metrics with respect to ungraded, high, and low severity faults by employing statistical (LR) and machine learning (Naïve Bayes, Random Forest, and NNge) methods. Bayesian approach was also used by the author Pai [18] in his work to see how software product metrics and fault proneness are related to each other. Shatnawi and Li [21] focused on recognizing the classes which are prone to error once the software has been released in the market. They studied the effectiveness of software metrics using three releases of the Eclipse project. They observed that, the accuracy of the model in predicting the classes kept on decreasing as they moved from one release to the other.

The work proposed in this paper is similar to the work done by Menzies and Marcus [14]. The authors have presented an automated method named SEVERIS (SEVERity Issue assessment) which is used to assign severity levels to the defect reports by using the data from NASA's Project and Issue Tracking System (PITS). Their method is based on extracting and then analyzing the documented descriptions from the defect issues available in PITS dataset by using various text mining techniques. They have used a rule learning method as their classification method to assign the features with appropriate levels of severity on the basis of classification of the available reports wherein the severity of the defects is already known. Similar work has also been done by Sari and Siahaan [20]. They have also developed a model for the assignment of the bug severity level. They have used the same pre-processing tasks (tokenization, stop words removal and stemming) and feature selection method (InfoGain), but, have used SVM as their classification method. Lamkanfi et al. [11] have also analyzed the textual matter of the defect documents using text mining algorithms in order to propose a technique that is used for severity prediction.

III. RESEARCH METHODOLOGY

In this section, we will first give a brief overview of the dataset used in our study. This will include the text classification process done in order to extract relevant words from the defect reports. Thereafter, we explain the working of Info Gain measure being incorporated in the text classification framework. Then we give a brief overview of the machine learning method used for classification. Finally, we highlight on the performance evaluation measures used to validate the results.

A. Data Source

In this work, we have used PITS (Project and Issue Tracking System) dataset being supplied by NASA's Software Verification and Validation (IV & V) Program. This data consists of the defect reports with the detailed information that includes the ID of the defect, summary of the defect and associated severity level of the defect.

There are in total 5 levels of severity into which the defects have been classified. Severity 1 defects have been classified as Very High, severity 2 defects as High, severity 3 defects as Medium, severity 4 defects as Low and severity 5 defects as Very Low. We are interested only in the last four levels – severity 2, severity 3, severity 4 and severity 5. This is so because there are no severity one issues in the defect data as these defects are very rare. However, there are 23 issues of severity 2, 523 issues of severity 3, 382 issues of severity 4 and 59 issues of severity 5.

The summary of all these issues were extracted from the defect reports for textual analysis. In other words, the textual descriptions of all the issues were analyzed and a series of text mining techniques were applied to extract the relevant words from each report. At a later stage, machine learning technique was used to assign the severity level to each defect based on the classifications of the available reports. As we know that the standard machine learning methods work well only for the data with fewer number of attributes. Therefore, we need to reduce the number of attributes in the given data. Hence, we have applied different methods of text mining for dimensionality reduction starting with tokenization, followed by stop words removal, then stemming and feature selection and finally weighting the selected words. The terms

tokenization, stop words removal and stemming come under the first and foremost step of text mining called pre-processing where the task is to remove the irrelevant words from the document. Tokenization is done by converting a stream of characters into a sequence of tokens. Thereafter, all the stop words like prepositions, articles, conjunctions, verbs, pronouns, nouns, adjectives and adverbs are removed from the document. Finally, stemming is done which is defined as the procedure of removing the words which have the same stem, thereby retaining the stem as the selected feature. This feature then undergoes a process of feature selection. For instance, the words like "move", "moves", "moved" and "moving" can be replaced with a single word as "move". All the words obtained after preprocessing were called as 'features'. Even after carrying out these pre-processing tasks, size of the feature set is still very large, that is not suitable for the machine learning algorithm. So, the next most crucial step is extraction of some useful words/features from the feature set using an appropriate feature selection method such as InfoGain. This method will rank all the features obtained after pre-processing and then the top 'N' scoring features are selected based on the rank. Thereafter, each document is represented in the form of a vector consisting of the above 'N' selected features. A complete set of all such vectors corresponding to each document under consideration is called a vector space model or VSM. Each term in the vector is weighted using a tf-idf approach which stands for Term Frequency Inverse Document Frequency. Term frequency is defined as the occurrence of the jth term in any given document, say i. It is denoted as 't_j'.This value indicates that the terms which occur frequently in a particular document are more significant than the other terms. The second value i.e. inverse document frequency is given by log (n/n_j) where n_j is the number of documents containing term t_j and n is the total number of documents. This value indicates that the terms which occur rarely among a group of documents are more significant than the other terms. Finally, the vectors are normalized before they are submitted to the learning algorithms.

B. *Feature Selection using Info-Gain Measure*

InfoGain measure is one of the most widely used feature selection method which identifies those words from the document which aim to simplify the target concept. In our study, the target concept is the severities of the defects [14]. Now, the total number of bits required to code any particular class distribution C_0 is $B(C_0)$. It is given by the following formula:

$$B(C) = -\sum_{c \in C} \left[{}^{n(c)}/_{N} \right] \log_2 \left[{}^{n(c)}/_{N} \right] \quad (1)$$

Where,

$$N = \sum_{c \in C} n(c) \quad (2)$$

Now, suppose A is a group of attributes, then the total number of bits needed to code a class once an attribute has been observed is given by the following formula:

$$B(C|A) = -\sum_{a \in A} \left[{}^{n(a)}/_{N} \right] \sum_{c \in C} p(c|a) \log_2 (p(c|a)) \quad (3)$$

Where,

$$p(c) = {}^{n(c)}/_{N} \quad (4)$$

Now, the attribute which obtains the highest information gain is considered to be the highest ranked attribute which is denoted by the symbol A_i.

$$Infogain(A_i) = B(C) - B(C|A_i) \quad (5)$$

C. *Model prediction using Radial Basis function of Neural Network*

A radial basis function network (RBF network) is an artificial neural network having single layer architecture. As the name is suggesting, the activation functions of this network are the radial functions. Thus, the network provides a linear combination of radial functions applied on the inputs [5]. RBF networks are being used in variety of different ways. Few of their applications include approximation of functions, prediction of time series, classification, system control etc. Fig 1 below shows the traditional RBF network consisting of 'n' inputs. Radial basis functions are applied on each of these 'n' inputs. The outputs of this layer are then linearly combined with weights w_j where j= 1 to m to produce the final output of the network f(x).

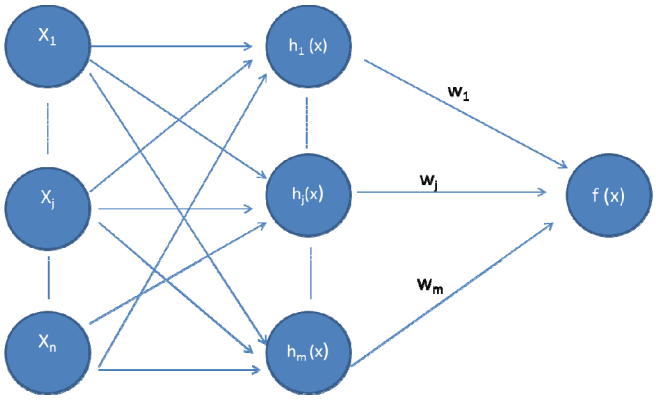

FIG 1. THE TRADITIONAL RADIAL BASIS FUNCTION NETWORK

The characteristic feature of RBFs is that their response has a tendency to decrease (or increase) monotonically as we move away from the central point [5]. On the basis of this characteristic, we can categorize RBF into two types viz. Gaussian RBF and a multiquadric RBF. Gaussian-like RBFs give a local response which means that they give a substantial response whenever they happen to lie in an area near the centre. They are more frequently used than multiquadric-type RBFs which give a global response. The key feature of Gaussian RBF is that they have a finite response due to which they become more acceptable in nature. The centre, the distance scale, and the precise shape of the radial function are the different parameters of the model. All these parameters are fixed if it is linear.

1) A Gaussian RBF decreases monotonically as we move away from the centre. It is defined as:

$$h(x) = \exp\left[\frac{-(x-c)^2}{r^2}\right] \quad (6)$$

Where, centre 'c' and radius 'r' are its parameters.

2) A multiquadric RBF increases monotonically with distance from the centre. It is defined as:

$$h(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r} \quad (7)$$

D. Model Evaluation

To measure the performance of the predicted model, we have used the following performance evaluation measures:

1) Sensitivity

It measures the correctness of the predicted model and is defined as the percentage of the documents correctly predicted to be defect prone [10]. It is also referred to as Recall. Mathematically we can define sensitivity as,

$$Sensitivity = \frac{TP}{TP + FN} * 100 \quad (8)$$

Where,

- TP are the correctly classified defect-prone documents, referred to as True Positive documents;
- FN are the wrongly classified defect-prone documents, referred to as False Negative documents;
- TN are the correctly classified defect-free documents, referred to as True Negative documents;
- FP are the wrongly classified defect-free documents, referred to as False Positive documents.

2) Receiver Operating Characteristics (ROC) analysis

ROC curve is defined as a plot of sensitivity values (true positive) on the y- axis versus its 1-specificity values (false positive) on the x- axis [7]. The main objective of constructing ROC curves is to obtain the required optimal cut-off point that maximizes both sensitivity and specificity. An overall indication of the accuracy of a ROC curve is the area under the curve (AUC). Values of AUC range from 0 to 1 and higher values indicate better prediction results.

3) Validation method used

We have used Hold-out validation method (70-30 ratio) in which the entire dataset is divided into 70% training data and remaining 30% as test data. We have used a partitioning variable that splits the given dataset into training and testing samples in 70-30 ratio. This variable can have the value either 1 or 0. All the cases with the value of 1 for the variable are assigned to the training samples and all the other cases are assigned to the testing samples. To get more generalized and accurate results, we have done validation using 10 separate partitioning variables.

IV. RESULT ANALYSIS

In this section, we have analyzed the results corresponding to the top-5, 25, 50 and 100 words in order to predict the best model that gives the highest accuracy using sensitivity, AUC and the cut-off point as the performance measures. Tables I-IV present and summarize the results with respect to high, medium, low and very low severity levels.

TABLE I. RESULTS OF NN FOR TOP-5 WORDS

	High Severity Defects			Medium Severity Defects			Low Severity Defects			Very Low Severity Defects		
Runs	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off
1	0.81	100.0	0.04	0.65	61.0	0.58	0.59	56.0	0.37	0.70	63.0	0.04
2	0.92	100.0	0.04	0.68	63.0	0.60	0.63	57.0	0.37	0.72	67.0	0.05
3	0.73	71.0	0.04	0.71	63.0	0.59	0.66	62.0	0.37	0.74	71.0	0.07
4	0.73	75.0	0.04	0.71	65.0	0.57	0.65	59.0	0.36	0.74	76.0	0.08
5	0.77	100.0	0.04	0.72	66.0	0.55	0.66	64.0	0.37	0.72	68.0	0.07
6	0.67	60.0	0.02	0.67	61.0	0.58	0.6	56.0	0.37	0.77	75.0	0.07
7	0.69	86.0	0.02	0.71	64.0	0.54	0.65	60.0	0.42	0.75	77.0	0.09
8	0.67	67.0	0.03	0.66	61.0	0.51	0.64	60.0	0.39	0.58	64.0	0.09
9	0.83	88.0	0.03	0.70	66.0	0.54	0.63	61.0	0.43	0.67	67.0	0.03
10	0.80	50.0	0.03	0.72	67.0	0.53	0.67	65.0	0.40	0.72	73.0	0.06

TABLE II. RESULTS OF NN FOR TOP-25 WORDS

	High Severity Defects			Medium Severity Defects			Low Severity Defects			Very Low Severity Defects		
Runs	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off
1	0.81	75.0	0.04	0.67	63.0	0.56	0.60	61.0	0.38	0.68	68.0	0.09
2	0.87	80.0	0.05	0.72	64.0	0.54	0.67	65.0	0.39	0.68	67.0	0.07
3	0.85	71.0	0.02	0.71	68.0	0.60	0.66	61.0	0.36	0.73	71.0	0.09
4	0.77	75.0	0.03	0.69	65.0	0.60	0.61	58.0	0.35	0.72	67.0	0.06
5	0.65	67.0	0.02	0.72	66.0	0.60	0.64	60.0	0.35	0.71	63.0	0.06
6	0.82	80.0	0.04	0.71	65.0	0.58	0.64	61.0	0.38	0.67	65.0	0.08
7	0.82	71.0	0.02	0.71	69.0	0.53	0.68	62.0	0.41	0.74	71.0	0.09
8	0.78	80.0	0.04	0.66	63.0	0.52	0.61	56.0	0.39	0.65	65.0	0.08
9	0.81	67.0	0.02	0.70	66.0	0.59	0.65	61.0	0.37	0.65	60.0	0.06
10	0.86	67.0	0.02	0.70	65.0	0.53	0.63	57.0	0.41	0.63	60.0	0.05

TABLE III. RESULTS OF NN FOR TOP-50 WORDS

	High Severity Defects			Medium Severity Defects			Low Severity Defects			Very Low Severity Defects		
Runs	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off
1	0.90	75.0	0.04	0.68	61.0	0.57	0.62	57.0	0.37	0.71	68.0	0.07
2	0.88	80.0	0.05	0.74	65.0	0.56	0.68	60.0	0.39	0.79	72.0	0.08
3	0.72	71.0	0.01	0.70	67.0	0.58	0.63	57.0	0.38	0.74	71.0	0.06
4	0.65	50.0	0.02	0.70	63.0	0.53	0.63	59.0	0.40	0.76	71.0	0.08
5	0.87	67.0	0.05	0.73	66.0	0.51	0.65	58.0	0.40	0.75	68.0	0.06
6	0.68	60.0	0.02	0.68	63.0	0.56	0.62	55.0	0.39	0.78	70.0	0.07
7	0.72	71.0	0.04	0.73	67.0	0.50	0.66	60.0	0.42	0.80	77.0	0.11
8	0.87	80.0	0.05	0.69	63.0	0.56	0.60	59.0	0.38	0.74	70.0	0.08
9	0.81	67.0	0.03	0.68	62.0	0.56	0.59	55.0	0.39	0.69	65.0	0.07
10	0.82	83.0	0.04	0.69	65.0	0.54	0.68	61.0	0.38	0.70	65.0	0.07

TABLE IV. RESULTS OF NN FOR TOP-100 WORDS

	High Severity Defects			Medium Severity Defects			Low Severity Defects			Very Low Severity Defects		
Runs	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off	AUC	Sens	Cut-Off
1	0.84	75.0	0.03	0.69	62.0	0.57	0.64	61.0	0.39	0.74	68.0	0.07
2	0.94	80.0	0.07	0.70	63.0	0.56	0.60	57.0	0.38	0.77	72.0	0.06
3	0.79	71.0	0.03	0.75	68.0	0.58	0.68	61.0	0.37	0.73	71.0	0.07
4	0.58	63.0	0.01	0.66	63.0	0.57	0.56	55.0	0.36	0.83	76.0	0.08
5	0.79	67.0	0.04	0.72	66.0	0.53	0.69	63.0	0.36	0.71	63.0	0.06
6	0.79	70.0	0.02	0.71	67.0	0.54	0.64	62.0	0.40	0.75	70.0	0.07
7	0.81	86.0	0.04	0.73	67.0	0.52	0.67	63.0	0.40	0.78	65.0	0.07
8	0.80	67.0	0.03	0.67	61.0	0.55	0.62	56.0	0.35	0.64	55.0	0.07
9	0.59	50.0	0.01	0.73	65.0	0.54	0.67	63.0	0.39	0.69	63.0	0.04
10	0.87	83.0	0.06	0.70	65.0	0.49	0.64	59.0	0.43	0.79	73.0	0.08

From table I it is clear, that the model has performed quite well in predicting high severity defects as the maximum value of sensitivity for high severity defects is 100% as compared to its maximum value which is 75% corresponding to the remaining severity defects. In other words, the values of sensitivity for medium, low and very low severity defects are in the range of 56% to 75%, in contrast to its values for high severity defects which is in the range of 70% to 100% for most of the runs. But from AUC, we can also see that the model has even predicted the defects with very low severity defects quite nicely with the maximum value of AUC as 0.83. On the other hand, performance of the model is consistent in predicting medium and low severity defects as the values for both AUC and sensitivity are almost the same for both the severity levels. A similar kind of trend is observed from table II when the number of words taken into consideration is increased to 25. The AUC values (range- 0.75 to 0.87 approx.) for high severity defects are better as compared to the AUC values corresponding to the other severity defects. That is to say, that the values of AUC corresponding to the remaining three severity level defects are consistent and lie in the range of 0.60 to 0.72 approx. This trend is suggesting that the model can predict the defects with high severity level more accurately the defects with other severity levels. The tables corresponding to top 50 and 100 words (table III and IV respectively) show that the model has performed well in predicting the defects either of high severity level or of very low severity level as indicated by their AUC values. Thus, we can conclude that the model corresponding to high and very low severity defects should be preferred in contrast to the model with medium and low severity defects.

V. CONCLUSION

Now-a-days the use of defect tracking systems has become indispensable due to the enormous growth of software repositories. However, a common issue in such systems is that they are used for maintaining daily-basis information. Moreover, the information contained within such systems is generally of unstructured form.

Hence in this paper, we have made use of text mining and machine learning techniques in order to analyze the defect data present in the defect tracking systems. We have used text mining techniques to mine the information from the database and thereafter Radial Basis function network was used to

predict the defect severities. The results were validated using NASA dataset available in the PITS database and were analyzed using the value of Area Under the Curve (AUC), sensitivity and a cut-off point obtained from Receiver Operating Characteristics (ROC) analysis. It was evident from the results that the model has performed very well in predicting high severity defects than in predicting the defects of the other severity levels. This observation is irrespective of the number of words taken into consideration as independent variables.

Till now, we have analyzed only one project of PITS in our study, but the results obtained from the research can be generalized to the other projects available in the PITS database. So, our future work involves replication of this work in other projects of PITS.

REFERENCES

[1] K.K. Aggarwal, Y. Singh, A. Kaur and R. Malhotra, "Empirical analysis for investigating the effect of object-oriented metrics on fault proneness: A replicated case study," *Software Process: Improvement and Practice*, vol. 16, no.1, pp. 39-62, 2009.

[2] C. Catal and B. Diri, "A systematic review of software fault prediction studies," *Expert Systems with Applications*,vol.36, pp. 7346-7354, 2009.

[3] G. Canfora and L. Cerulo, "How Software Repositories can Help in Resolving a New Change Request," *Workshop on Empirical Studies in Reverse Engineering***,** 2005.

[4] D**.** Cubranic and G.C. Murphy, "Automatic bug triage using text categorization," *Proceedings of the Sixteenth International Conference on Software Engineering and Knowledge Engineering*, 2004.

[5] D.S. Broomhead and D. Lowe, "Multivariate functional interpolation and adaptive networks," *Complex Systems, vol. 2, pp. 321-355, 1988*.

[6] K.E. Emam and W. Melo, "The Prediction of Faulty Classes Using Object-Oriented Design Metrics," *Technical report: NRC 43609*, 1999.

[7] K.E. Emam, S. Benlarbi, N. Goel and S. Rai, "A validation of object-oriented metrics," *NRC Technical report ERB-1063*, 1999.

[8] I. Gondra, "Applying machine learning to software fault-proneness prediction," *The Journal of Systems and Software*, vol.81, pp. 186-195, 2008.

[9] T. Gyimothy, R. Ferenc and I. Siket, "Empirical validation of object-oriented metrics on open source software for fault prediction," *IEEE Transactions on Software Engineering*, vol.31, no.10, pp. 897-910, 2005.

[10] Y. Jiang, B. Cukic and Y. Ma,"Techniques for evaluating fault prediction models," *Empirical. Software. Engineering*, vol.13, no. 15, pp. 561-595, 2008.

[11] A. Lamkanfi, D. Serge, E. Giger and B. Goethals, "Predicting the Severity of a Reported Bug," *7th IEEE working conference on Mining Software Repositories (MSR),* pp. 1-10, 2010.

[12] R. Malhotra and Y. Singh, "On the Applicability of Machine Learning Techniques for Object- Oriented Software Fault Prediction," *Software Engineering: An International Journal,* vol.1, no.1, pp. 24-37, 2011.

[13] R. Malhotra and A. Jain, "Fault Prediction Using Statistical and Machine Learning Methods for Improving Software Quality," *Journal of Information Processing Systems*,vol. 8, no.2, pp. 241- 262, 2012.

[14]T. Menzies and A. Marcus, "Automated Severity Assessment of Software Defect Reports," *IEEE International Conference on Software Maintenance (ICSM)*, 2008.

[15] G. Myers, T. Badgett, T. Thomas and C. Sandler, "The Art of Software Testing," second ed., John Wiley & Sons, Inc., Hoboken, NJ, 2004.

[16] N. Ohlsson, M. Zhao, M and M. Helander , "Application of multivariate analysis for software fault prediction," *Software Quality Journal*, vol.7, pp.51-66, 1998.

[17] H. Olague, L. Etzkorn, S. Gholston and S. Quattlebaum, "Empirical validation of three software metrics suites to predict fault-proneness of object-oriented classes developed using highly iterative or agile software development processes," *IEEE Transactions on Software Engineering*, vol.33, no.8, pp. 402-419, 2007.

[18] G. Pai, "Empirical analysis of software fault content and fault proneness using Bayesian methods," *IEEE Transactions on Software Engineering*, vol. 33, no. 10, pp. 675-686, 2007.

[19] P. Runeson, M. Alexandersson and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing," *29th IEEE International Conference on Software Engineering (ICSE),*pp. 499 – 508, 2007.

[20] G.I.P. Sari and D.O. Siahaan, "An attribute Selection For Severity level Determination According To The Support Vector Machine Classification Result," *Proceedings of The 1st International Conference on Information Systems For Business Competitiveness (ICISBC),* 2011.

[21] R. Shatnawi and W. Li, "The effectiveness of software metrics in identifying error-prone classes in post-release software evolution process," *The Journal of Systems and Software*,vol. 81, pp. 1868-1882, 2008.

[22] Y. Singh, A. Kaur and R. Malhotra,"Empirical validation of object-oriented metrics for predicting fault proneness models," *Software Quality Journal*, vol.18, pp. 3-35, 2010.

[23] X. Wang, L. Zhang, T. Xie, J. Anvik and J. Sun, "An Approach to Detecting Duplicate Bug Reports using Natural Language and Execution Information," *Association for Computing Machinery*, 2008.

[24] P. Yu, T. Systa, and H. Muller, "Predicting fault-proneness using OO metrics: An industrial case study," *In Proceedings of Sixth European Conference on Software Maintenance and Reengineering*, Budapest, Hungary, pp.99-107, 2002.

[25] Y. Zhou, and H. Leung,"Empirical Analysis of Object-Oriented Design Metrics for Predicting High and Low Severity Faults," *IEEE Transactions on Software Engineering*, vol. 32, no. 10, pp. 771-789, 2006.

[26] Y. Zhou, B. Xu, and H. Leung,"On the ability of complexity metrics to predict fault-prone classes in object -oriented systems," *The journal of Systems and Software,* vol.83, pp.660-674, 2010.