Defect Prediction by Analysing Software Project Reports Using Logistic Regression

Abha Jain

Assistant Professor

Department of Computer Science

Shaheed Rajguru College of Applied Sciences for Women
University of Delhi

Vasundhara Enclave, Delhi- 110096, India

Abstract**—**The large and complex software in today's scenario has led to unavoidable circumstances wherein defects are entering the system at an enormous rate. This has caused an emerging need to assess the severity of these defects considering the shortage of resources due to which equal amount of resources cannot be allocated to all the defects. Thus, in this paper, we intend to propose a model that will be used to assess the severity level of the defect introduced in the system so that defects having higher priority can be dealt with utmost attention. The model is validated against an open source NASA dataset of PITS database using a learning based technique namely Multinominal Multivariate Logistic Regression (MMLR). A series of text mining techniques were applied for pre-processing the data. Finally, validation of the data was conducted using Receiver Operating Characteristics (ROC) characterstics.

IndexTerms **- Receiver Operating Characteristics, Text mining, Defect, Severity, Multinominal Multivariate Logistic Regression**

I. Introduction

In today's complex software environment, a situation has arisen whereby defects are bound to enter the system, leading to failures of the functional system [15]. The defects entering the system have different severity levels associated with them. The defect ID along with its corresponding severity level is contained in the defect reports maintained by bug tracking systems.A severity level of a defect determines the impact of the defect on the software system. The extent to which the defect can hamper the system ranges from mild to severe [1], [6]. The higher severity defects can lead to an entire system crash. Thus, it is very crucial to determine the levels of severity of the defects introduced in the software. This will lead to proper utilization of resources in terms of time, money and manpower which are limited and need to be used carefully.

In this work, the information in the Project and Issue Tracking System (PITS) database has been mined using data mining methodologies in order to represent the data in a structured form suitable for carrying out large-scale analysis of the reports. This structured information is the relevant data which has been extracted using text mining techniques so that it can be further used for assessing the severity of the defects. The severity of the defects has been assessed using one of the most popularly used learning technique namely Multinominal Multivariate Logistic Regression (MMLR). NASA's engineers have classified the defects based on the five severity levels viz. very high, high, medium, low and very low.

The correct prediction of defect severity would be highly useful for software researchers in allocating their testing resources carefully leading to appropriate defect setting activities, thus having a huge impact on the business needs of the software society. The proposed model has been evaluated using the performance measures derived from Receiver Operating Characteristics (ROC) analysis.

This paper has the following sections: Section 2 highlights the available literature in the domain. Section 3 describes the methodology of the research. 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

Till date, various defect reports have been analyzed. An automated technique was proposed by Cubranic and Murphy [4] that dealt in bug triage based on the analysis of an incoming bug report containing the bug description. An effective tool used for the detection of duplicate reports based on Natural Language Processing was developed by Runeson et al. [19] and Wang et al. [23]. Canfora and Cerulo [3] presented the necessity for software repositories to manage a new change request. Apart from this, empirical work to find out defective classes in object-oriented software systems has been conducted incorporating the usage of design metrics pertaining to object-oriented software [2], [6], [8], [9], [12], [13], [16], [17], [24], [26]. But, these studies did not focus on the severity of the defects which is an emerging issue.

The authors Singh et al. [22] discussed the need for predicting fault severity considering three levels of severity viz. high, medium and low severity and how it drastically impacts the software development. Similar kind of study was conducted in the papers by Zhou and Leung [25] and Pai [18]. Shatnawi and Li [21] focused on error prone classes upon releasing the software in the market. The authorsMenzies and Marcus [14] presented an automated method named SEVERIS (SEVERity Issue assessment) that dealt with defect reports and how are these reports being assigned with correct defect severity levels. The result was validated using the data from NASA's Project and Issue Tracking System (PITS). Sari and Siahaan [20] and Lamkanfi et al. [11] also developed the model intended for assigning the bug severity level. The paper by the authors Yang et al. [27] discussed the effectiveness of feature selection approaches in predicting the defect severity with three commonly used feature selection schemes i.e. Information Gain, Chi-Square, and Correlation Coefficient, based on the Multinomial Naive Bayes classification approach. The results were validated against four open-source components from Eclipse and Mozilla. The authors Iliev et al. [28] studied the severity prediction through reasoning about the requirements and the design of a system using the concept of ontologies. A Semi-automatic approach was proposed by the authors Yang et al. [29] for Bug Triage and Severity Prediction Based on Topic Model and Multi-feature of Bug Reports. The authors Chaturvedi and Singh [30] demonstrated the applicability of machine learning algorithms (Naïve Bayes, k-Nearest Neighbor, Naïve Bayes Multinomial, Support Vector Machine, J48 and RIPPER) in determining the class of bug severity of bug report data of NASA from PROMISE repository.

III. RESEARCH METHODOLOGY

In this section we discuss the research methodology being employed in this work. As can be seen from figure 1, there are three steps used to accomplish the required work of predicting the defect severity.

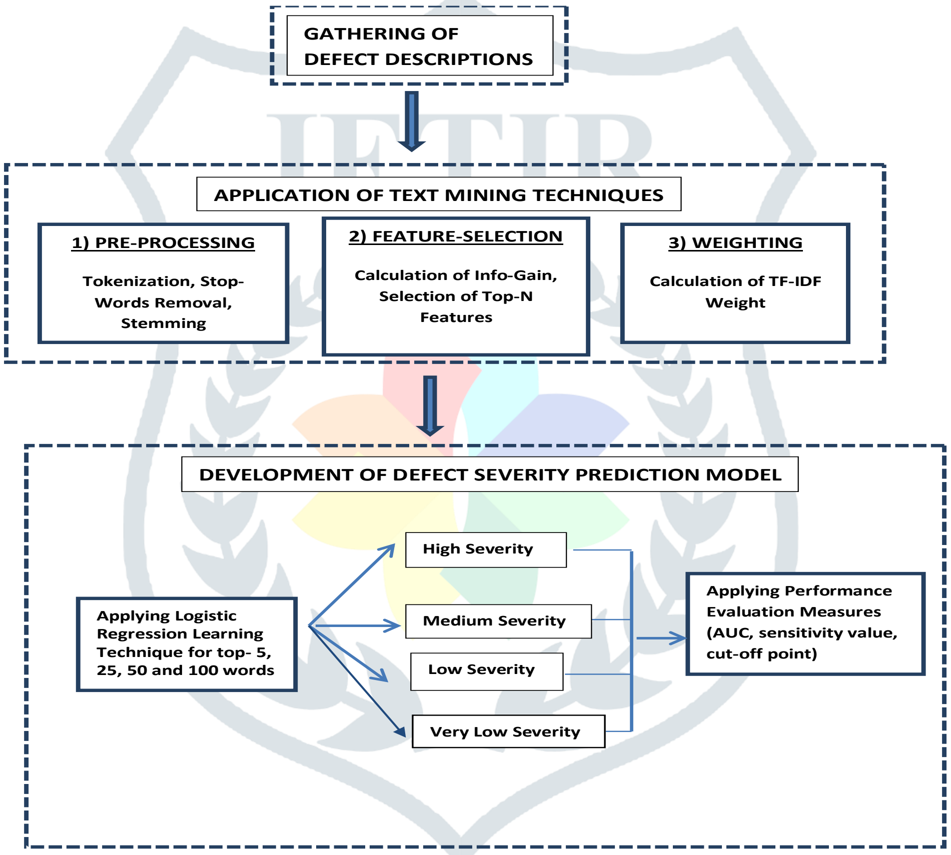
GATHERING OF DEFECT DESCRIPTIONS
APPLICATION OF TEXT MINING TECHNIQUES
1) PRE-PROCESSING
Tokenization, Stop-Words Removal, Stemming
2) FEATURE-SELECTION
Calculation of Info-Gain, Selection of Top-N Features
3) WEIGHTING
Calculation of TF-IDF Weight
DEVELOPMENT OF DEFECT SEVERITY PREDICTION MODEL
Applying Logistic Regression Learning Technique for top- 5, 25, 50 and 100 words
High Severity
Medium Severity
Low Severity
Very Low Severity
Applying Performance Evaluation Measures (AUC, sensitivity value, cut-off point)

Figure 1: Framework for Predicting Severity of defects

1. Firstly, the description of all the defects was extracted from the defect reports. These defect reports were retrieved from the PITS dataset which has been explained in the subsequent section.
2. Secondly, these descriptions were applied to a series of text mining techniques. As can be seen from the figure, application of text mining techniques begins with pre-processing tasks that include tokenization, stop words removal and stemming. Thereafter, feature selection approach was applied and finally weighting the selected words was performed. Feature selection method for dimensionality reduction used in this work is Information Gain (IG) measure which selects the top 'N' scoring features based on the rank [14]. In this paper, we have considered top-5, 25, 50 and 100 words obtained on the basis of ranking done by Information Gain measure. The weighting approach used for weighting the selected features is Term Frequency Inverse Document Frequency (Tf-Idf). The first term (Tf) 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. Application of text mining techniques was done in order to extract some relevant words so that suitable learning technique can then be applied to assign the severity level to the defect based on the classifications of the available reports.
3. Finally, model prediction was done using Multinominal Multivariate Logistic Regression (MMLR) technique [5] which has been discussed in the subsequent section. The performance of the model has been evaluated using the performance measures derived from ROC analysis viz. sensitivity [10], Area Under the Curve (AUC) and cut-off point [7]. Hold-out validation method has been used to divide the dataset in the ratio of 70% as training data and 30% as test data. Validation has been done using 10 separate partitioning variables for more generalized results.

IV. Research Background

This section highlights on the basics behind the research carried out in this work. The section begins by first explaining the data source used for conducting the experiment. Thereafter, the technique used for features selection i.e. Information Gain has been explained. This is followed by elaborating on the technique which has been used for model prediction in this work (MMLR).

4.1 Data Source

This work incorporates the usage of Project and Issue Tracking System (PITS) dataset which is supplied by NASA's Software Verification and Validation (IV & V) Program. The data is the information pertaining to the defect reports that includes the defect ID, description of the defect and its severity levels. NASA's engineers have classified severity into 5 levels viz. 'very high', 'high', 'medium', 'low' and 'very low'. Figure 2 represents the number of defects pertaining to each severity level. As can be seen from the figure, there are no defects pertaining to 'very high' severity levels and therefore this severity level has not been taken into account in our study. The maximum number of defects is pertaining to 'medium' severity level whereas the least number of defects belong to 'high' severity level.

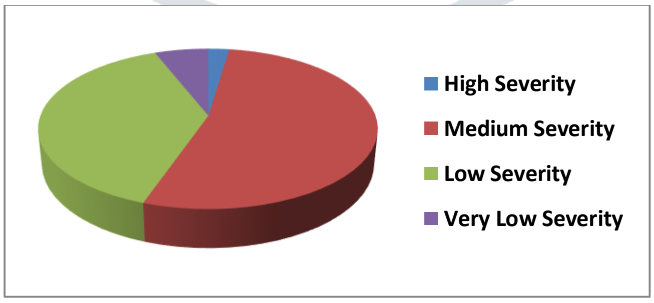

Figure 2: Number of Defects Belonging to Each Severity Level

4.2 Feature Selection using Information Gain Measure

Information Gain measure is one of the most widely used feature selection method which identifies those words from the document which aim to simplifythe target concept [14]. The number of bits required to encode an arbitrary class distribution C_0 is $H(C_0)$. It is given by the following formula:

$$N = \sum_{c \in C} n(c) \qquad (1)$$

Where, n(c) = Number of instances belonging to class c

$$p(c) = {}^{n(c)}/_{N} \qquad (2)$$

$$H(C) = -\sum_{c\in C} p(c)\log_2 p(c) \qquad (3)$$

Now, suppose Ais 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:

$$H(C|A) = -\sum_{a\in A} p(a)\sum_{c\in C} p(c|a)\log_2(p(c|a)) \qquad (4)$$

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.

$$\mathrm{Infogain}(A_i) = H(C) - H(C|A_i) \qquad (5)$$

4.3 Model Prediction using Multi-nominal Multivariate Logistic Regression (MMLR)

In this paper, MMLR method has been used which is a variant of logistic regression method. Logistic regression has been one of the most popular statistical methods being used in the literature so far. In MMLR method, all the independent variables together are used to predict different categories of the dependent variable [5]. In other words, here the dependent variable is not binary and can have more than two values. For instance, the dependent variable in this study has four levels of category viz. very low, low, medium and high. Now, in order to predict the model using MMLR, all the independent variables are used together on each of these abovementioned categories. This is in contrast to UMR model (Univariate Multinominal Regression), wherein each independent variable is used at a time in order to predict each category of the dependent variable. Much similar to UMR and MMLR models, we have two additional models viz. ULR (Univariate Logistic Regression) and MLR (Multivariate Logistic Regression) models wherein the dependent variable is of binary type. In MLR model, all the independent variables are used together in order to predict the binary dependent variable and in ULR model, the effectiveness of a single independent variable is examined at a time. The general MLR model is based on the following equation:

$$Prob(Y1, Y2, \dots\dots, Yn) = \frac{e^{B0+B1Y1+\dots\dots+BnYn}}{1+e^{B0+B1Y1+\dots+BnYn}} \qquad (6)$$

where, Yi's are the independent variables, Bi's are the regression coefficients corresponding to Yi's and Prob is probability that a defect was found during validation.

V. Result Analysis

In this section, we present the defect prediction results in terms of 'high', 'medium', 'low' and 'very low' severity defects using MMLR for different set of words. Tables I-IV present and summarize the result analysis with respect to these severity levels corresponding to the top-5, 25, 50 and 100 words respectively.

TABLE I. Results Of MMLR 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	0.84	100	0.05	0.63	58.5	0.52	0.54	53.8	0.41	0.73	68.4	0.08
2	0.81	80.0	0.04	0.68	65.8	0.50	0.62	58.9	0.409	0.72	72.2	0.08
3	0.74	71.4	0.03	0.72	68.8	0.533	0.66	61.3	0.402	0.71	70.6	0.08
4	0.75	75.0	0.03	0.68	60.5	0.514	0.56	56.1	0.40	0.80	76.2	0.10
5	0.90	66.7	0.05	0.71	69.7	0.531	0.63	63.2	0.41	0.75	73.7	0.08
6	0.79	80.0	0.03	0.65	60.1	0.523	0.60	55.7	0.40	0.69	65.0	0.08
7	0.78	85.7	0.03	0.71	67.0	0.487	0.60	59.5	0.42	0.74	70.6	0.09
8	0.78	66.7	0.03	0.65	62.0	0.511	0.62	58.7	0.41	0.61	54.5	0.07
9	0.86	87.5	0.04	0.69	66.0	0.532	0.61	60.2	0.42	0.68	66.7	0.05
10	0.83	100	0.04	0.67	63.2	0.464	0.61	56.2	0.43	0.75	66.7	0.11

TABLE II. RESULTS OF MMLR 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	0.78	75.0	0.02	0.68	61.2	0.52	0.62	59.8	0.40	0.70	68.4	0.06
2	0.81	80.0	0.02	0.75	65.8	0.49	0.72	60.7	0.42	0.76	66.7	0.05
3	0.64	57.1	0.03	0.72	67.5	0.54	0.72	66.9	0.34	0.62	58.8	0.04
4	0.70	62.5	0.00	0.70	63.8	0.50	0.66	61.8	0.41	0.70	61.9	0.04
5	0.86	100	0.01	0.71	61.8	0.53	0.65	59.0	0.39	0.77	73.7	0.06
6	0.61	50.0	0.01	0.69	62.8	0.53	0.63	58.3	0.40	0.74	70.0	0.05
7	0.80	71.4	0.02	0.70	65.9	0.49	0.66	62.1	0.42	0.65	64.7	0.03
8	0.66	60.0	0.01	0.70	63.9	0.48	0.61	56.3	0.40	0.71	70.0	0.08
9	0.88	77.8	0.07	0.71	65.0	0.51	0.65	61.5	0.37	0.72	65.0	0.06
10	0.68	50.0	0.00	0.73	68.5	0.49	0.69	61.8	0.42	0.65	65.0	0.03

TABLE III. RESULTS OF MMLR 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	0.93	100	0.02	0.73	69.4	0.57	0.69	65.0	0.39	0.83	73.7	0.03
2	0.82	80.0	0.00	0.73	69.7	0.53	0.66	62.6	0.34	0.82	77.8	0.05
3	0.76	71.4	0.00	0.76	71.9	0.56	0.73	66.9	0.36	0.74	64.7	0.02
4	0.58	50.0	0.00	0.73	69.1	0.53	0.66	60.2	0.35	0.77	71.4	0.02
5	0.81	66.7	0.00	0.77	70.4	0.55	0.71	63.2	0.36	0.85	78.9	0.05
6	0.69	70.0	0.00	0.76	71.6	0.55	0.71	67.8	0.39	0.84	75.0	0.03
7	0.85	71.4	0.00	0.74	70.5	0.49	0.69	63.8	0.37	0.82	82.4	0.06
8	0.73	60.0	0.01	0.70	63.9	0.48	0.65	60.5	0.37	0.75	70.0	0.05
9	0.80	66.7	0.00	0.73	67.9	0.55	0.67	63.1	0.34	0.70	60.0	0.05
10	0.89	83.3	0.01	0.73	68.5	0.47	0.68	63.6	0.41	0.79	70.0	0.02

TABLE IV. RESULTS OF MMLR 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	0.69	50.0	0.00	0.74	70.7	0.53	0.67	62.4	0.33	0.63	68.4	0.00
2	0.87	80.0	0.00	0.71	67.8	0.45	0.62	61.3	0.37	0.70	66.7	0.00
3	0.69	71.4	0.00	0.73	68.1	0.46	0.69	69.4	0.34	0.67	64.7	0.00
4	0.99	100	0.99	0.81	73.7	0.50	0.77	71.5	0.36	0.93	90.5	0.01
5	0.91	100	0.00	0.70	65.8	0.45	0.60	58.1	0.27	0.71	68.4	0.00
6	0.93	90.0	0.00	0.73	70.3	0.50	0.69	64.3	0.36	0.76	70.0	0.00
7	0.89	85.7	0.00	0.74	69.3	0.47	0.68	66.4	0.35	0.76	70.6	0.00
8	0.87	83.3	0.00	0.73	68.0	0.52	0.59	59.5	0.25	0.69	63.6	0.00
9	0.86	80.0	0.00	0.71	68.7	0.37	0.71	64.2	0.44	0.67	61.9	0.00
10	0.53	50.0	0.00	0.70	65.8	0.49	0.69	66.7	0.31	0.78	73.3	0.00

Table I shows that the maximum value of AUC for 'high' and 'very low' severity categories is 0.90 and 0.80 respectively as compared to its maximum value for 'medium' and 'low' severity categories which is 0.72 and 0.66 respectively. Also, sensitivity attains a 100% value for 'high' severity defects and a maximum value of 76.2% for 'very low' severity defects. In contrast to this, the maximum value of sensitivity for 'medium' and 'low' severity defects is just 69.7% and 63.2% respectively. This trend suggests that the MMLR model can predict either 'very high' severity defects or 'very low' severity defects better than the defects having 'medium' severity level when the number of words is less. If we compare the model prediction for 'medium' severity defects with 'low' severity defects, we can see that the values for AUC and sensitivity for 'medium' category is better than the 'low' category. So we can conclude that testers and researchers should focus their work on 'very high' severe and 'very low' severe defects without concentrating much on low severity defects. As we increase the number of words to 25, we can see from table II that the prediction capability of the model remains almost the same for 'medium', 'low' and 'very low' severity level. But the model again performs the best in terms of 'high' severity level with the maximum value of AUC as 0.86 and a 100% sensitivity value. This trend is suggesting that there should be maximum focus on the defects having highest severity when the number of words considered is nominal. A similar kind of trend can be seen from tables III and IV, where the model has performed exceptionally well in predicting the 'high' severity defects. The maximum value of AUC is 0.93 when top-50 words were considered and 0.99 when top-100 words were considered for prediction. In both the cases sensitivity has attained a value of 100%. The sensitivity and AUC values are also good for 'very low' severity defects for both the above mentioned cases. The maximum value of sensitivity and AUC for top-50 words is 82.4% and 0.85. On much similar lines, sensitivity and AUC (for 'very low' severity defects) attains their maximum value as 90.5% and 0.93 respectively when top-100 words were considered for prediction. This observation is indicative of the fact that no matter how many words are taken into account for model prediction, the performance of the model is consistent in predicting 'very high' severity defects more accurately than the 'low' and 'medium' severity defects. We can conclude the analysis by saying that there is a consistent phenomenon in the classification of the reports as it is noticeable that the AUC and the sensitivity values are better in the 'high' and 'very low' categories than in the 'medium' level categories irrespective of the number of words considered for classification.

VI. CONCLUSION

The tracking of defects has become highly crucial in today's scenario considering an immense growth in software industry.
In this paper, we incorporated text mining techniques to analyze the data pertaining to defect present in these defect tracking systems. The text mining techniques havebeen used to pre-process the information from the database and thereafter Logistic Regression model was appliedto predict the defect severities. The results were validated using NASA dataset available in the PITS database and were analyzed using thevalue of AUC, sensitivity and a cut-off point obtained from 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.

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. Hosmer and S. Lemeshow, "Applied logistic regression", New York: Wiley, 1989.

[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.

[27] C.Z. Yang, C.C. Hou , W.C. Kao , I.X Chen, "An Empirical Study on Improving Severity Prediction of Defect Reports Using Feature Selection", 19th Asia-Pacific Software Engineering Conference, 2012.

[28] M. Iliev, B. Karasneh , M. R. V. Chaudron , E. Essenius, "Automated prediction of defect severity based on codifying design knowledge using ontologies", First International Workshop on Realizing AI Synergies in Software Engineering (RAISE), 2012.

[29] G. Yang , T. Zhang , B. Lee, "Towards Semi-automatic Bug Triage and Severity Prediction Based on Topic Model and Multi-feature of Bug Reports", IEEE 38th Annual Computer Software and Applications Conference, 2014.

[30] K. K. Chaturvedi, V. B. Singh, " Determining Bug severity using machine learning techniques", CSI Sixth International Conference on Software Engineering (CONSEG), 2012.