J. Appl. Environ. Biol. Sci., 5(6)53-59, 2015
© 2015, TextRoad Publication

ISSN: 2090-4274
Journal of Applied Environmental and Biological Sciences
www.textroad.com

Using Smote for Convalescing Software Defect Prediction

Bushra Hamid[1], Inayat ur-Rehman[2], Abdul Rauf[3], Tamim Ahmed Khan[4]

University Institute of Information Technology, PMAS Arid Agriculture University Rawalpindi, Pakistan[1],
Department of Computer Science, COMSATS Institute of Information Technology, Islamabad, Pakistan[2],
Department of Computer Science, College of Computer & Information Sciences, Al-Imam Mohammad Ibn Saud Islamic University (IMSIU), Riyadh, KSA, Saudi Arabia[3],
Department of Computer and Software Engineering, Bahria University, Islamabad[4]

Received: February 26, 2015
Accepted: April 24, 2015

ABSTRACT

Software fault prediction models use software metrics and fault occurrences statistics, collected from previous versions of software products. These models attempt to predict the defect status (probability) of the software components. Developing an accurate and efficient software defect prediction model remains a challenging issue due to existence of outliers in data sets used in defect prediction process and that the unbalanced data sets badly influence the performance of software fault prediction model. Most of the prediction model make use of all the software metrics collected from previous projects , although there is no need to employ all software metrics collected for fault prediction as some of these software metrics are redundant and should not be used to develop prediction model due to curse of dimension.
We previously proposed [23] a model to deal with two issues i.e. outlier detection and attribute selection. In this paper, we have improved our previous model and apply Synthetic Minority Over-sampling Technique (SMOTE) on the datasets to deal with class imbalance. We have compared the classification results of both models to validate the strength of improved approach.
KEYWORDS: Defect Prediction, Optimization, Software Quality, SMOTE, Classifier Ensembles, Genetic Algorithm

INTRODUCTION

Software systems are becoming very important in all field of life. Developing a quality software product is not an easy task. In software product quality is directly related with number of defects in the software components [1]. The term defect or fault is defined as divergence from requirements which may cause failures in process. Defects scope diverge from the minute annoying misspellings in the user interface to the more distressing defects, originating flawed behavior, to the catastrophic defects that break down not only the in progress program, but also obliterate data and stop other program. Software quality is also defined as conformance to requirements, because if the software contains too many functional defects, the basic requirement of providing the desired functionality is not met.

There are many aspects which may lead to software defects in its software life cycle, such as Software requirements, software design, software coding and software testing and so on. However, software defects which are produced during the stages of Software Requirements, Software Design and Software Testing, will finally expressed in software Source Codes. Therefore, detecting software defect by software source codes is the most popular method to predict software defect [28]. The most frequently metric that researcher used to predict the software fault in software source codes is Size and Complexity metrics. Size and complexity metrics is an abstract expression of software source codes complexity, such as line of code, cyclomatic complexity and design complexity and so on [4].

Accurate prediction of faulty software modules greatly help in reducing testing effort, by assisting software testers to concentrate on the defective modules. As we know software testing is a very costly and timing consuming activity. In software development process testing usually requires 40% of the whole project schedule [5]. There is no direct method to measure the fault proneness [4], [5]. Based on Software metrics, faulty software modules are detected using software defect prediction module. Unfortunately, there is no generic technique for estimating software module as faulty or non-faulty [6].

With the availability of repositories that contains actual data collected from earlier software releases such as PROMISE repository make it possible to learn from real data. However, the available data repositories are of poor quality due to inconsistencies in the datasets e.g. dataset include replicated instances and contradictory instances. “Existence of outlier in the real data degrades trained model performance” [23].Most data sets are highly skewed toward a specific class of instances. In such cases, due to imbalanced data set the efficiency of defect predictors is badly distress.

Efficient artificial intelligence (AI) systems can be designed by integrating AI and software engineering (SE) disciplines together. Advantages of AI applications can be seen in situations where complex decisions are needed to be taken. AI knowledge can be integrated into computer science field to deliver more efficient and correct systems to its customers. Software fault proneness prediction process can benefit from such knowledge. An active research is being happening for this purpose. This research also puts an effort to use AI specifically evolutionary algorithms in predicting fault prone modules. This is the junction of two entirely different research fields, fault prediction and evolutionary algorithms. Considering the significance of early fault prediction process and the maturity of evolutionary algorithmic

Corresponding author: Tamim Ahmed Khan, Department of Computer and Software Engineering, Bahria University, Islamabad.
Email address: tamim@bui.edu.pk

methods, we believe the time has come for the software quality and AI researchers to join forces in assisting software quality engineers and testers in developing the quality software systems. Software defect prediction is the process of predicting software component as defective or non-defective based on historical data or expert opinion when require data is missing. Fault proneness can be measured using software metrics, which provide a quantitative sketch of software components. Currently, software fault prediction model performance suffers from the poor quality of software measurement data and difficulty in selection of appropriate software metrics there is a need to develop a model that overcomes these problems.

Another important issue that needs to be addressed for making fault proneness prediction process more effective is to deal with imbalanced data sets. Usually data in the real world is imbalanced [26], [29]. Class imbalance means that data set contains a large no of instances for a particular class then other classes exit in that data set. The class imbalance problem is insidious and ever-present in the data set used for fault proneness prediction due to the fact that number of faulty modules instances is less than number of non-faulty modules instances. The class imbalance problem typically occurs in a classification problem. Due to imbalance classes most of data sets are highly skewed toward a specific class of instances. In such cases, because of imbalanced data set the efficiency of defect predictors is badly distress. Therefore, there is a need to balance the data set in order to improve the efficiency of prediction model. Outlier is an observation that appears to be deviate from other instances of sample in which it occurs i.e. an observation that is inconsistent with the remainder of data sets. Presence of outliers in the datasets used for predicting software defects often impact the performance of defect prediction models.

Our research is motivated by the aim of reducing testing efforts and minimizing the cost of developing high quality software product. Software defect prediction is the process of predicting software component as defective or non-defective based on historical data or expert opinion when required data is missing. Fault proneness can be measured using software metrics, which provide a quantitative sketch of software components. We build a fault proneness prediction model that has high probability of fault detection, accuracy, precision and balance where as low probability of false alarms. We are using ensemble classifier. In order to obtain optimal results of fault prediction it is very important to use data that contains as much information as possible related to the software product. In this research study, static code attributes are used to build fault proneness predictor, since static code attributes are last deliverable of the software development process before actual product is delivered [35], [36]. Both method level and class level modules are used for fault proneness prediction [9], [33], [32]. We used method level modules for defect prediction.

In this research study data set from (http://promisedata.org). We have improved our previous defect prediction model based on ensemble classifier and optimize results using GA [23]. In that model we had used accuracy, Probability of detection, Probability of false alarms and balance as evaluation measure. Our previous model deals with two issues i.e. outlier detection and feature selection. In order to deal with class imbalance we have improved our previous model to deal with class imbalance. For balancing the data sets we have employed SMOTE. In this research study we have also incorporated precision as an evaluation measure. We have repeated experiments with previous model as well in order to examine the precision rate of that approach. By applying SMOTE precision rate increases significantly as compared to the first model whereas accuracy, Pd balance and Pf remains maintained. We finally compare the results with previous model. The paper is organized as follows: section 2 present overview of related work; Section 3 gives details about the empirical data collection, while research methodology has been elaborated in section 4 and section 5 includes results of the study. Finally paper has been concluded in section 6.

2. RELATED WORK

Software industry has been trying to find out ways for developing software product within time and according to the needs of customers. A large number of defect prediction models have been proposed so far. Some defect prediction models used class level metrics while other use method level metrics. T. M. Khoshgoftaar et al proposed a process that includes a technique for selecting an appropriate attributes and a technique for dealing with imbalanced class. They concluded that featured based sample data performs significantly better then featured selection based on original data. They concluded that selection of appropriate metrics is very important [2]. L. Pelayo and S. Dick used SMOTE technique for sampling minority class examples. Results have shown that by applying SMOTE resampling, an improvement of 23% in average geometric mean of classification accuracy [3]. P.S.Sandhu et al used Genetic algorithm in order to develop prediction model and for the validation of proposed model they used JEdit[7]. Z. Li and M. Reformat used SimBoost machine learning method to handle the software fault prediction problem with highly imbalanced datasets. The results of experiment have shown that their proposed methods reasonably reduce the effect of imbalanced datasets; however the prediction for balanced data set was not accurate. In order to deal with this issue they proposed fuzzy label for classification [4]. S.Lessmann et al investigated several classification models over 10 public domain software data sets from NASA MDP repository. Their experimental results specify that the significance of particular classification algorithm may possibly be less supposed as there could be no noteworthy difference detected in the performance of top 17 classifiers [23].

I.Gondra applies Artificial Neural Network (ANN) for predicting software defects. For selection and comparison of matrices they compared SVM with ANN and findings show that SVM performs better [8]. M. Mie and T.S.Quah

presented the application of neural network for fault prediction including object oriented faults. They have compared neural network model with multiple regression model. They used three industrial real time systems for conducting research study. According to their experimental results neural network perform better then multiple regression models [11]. P.Singh compared the performance of defect prediction model by using static attribute of embedded software. He applied three machine learning algorithm for defect prediction. Results have shown that J48 and oneR perform better then Naïve Bayes learner [12]. B.Turhan and A. Bener proposed a model for defect prediction using multivariate approaches in combination with Bayesian methods. They employ 8 different data sets from NASA Metric Data (MDP) repository. They concluded that it is better to investigate more about feature extraction [13].

H.Wang et al investigated ensembles of feature selection methods for predicting defective modules. In their research study 18 different filter based ranking techniques has been applied and examined 17 different ensembles of rankers. Experimental results have shown that ensemble of very few rankers were effective than many or all rankers [14]. Q. Sung et al proposed a general framework for defect prediction that supports impartial and deep comparison between defect prediction systems. They point of view was that if learning scheme would not be assessed properly may lead to wrong results [15]. Y. Weimin and L. Longshu presented a rough set model for software defect prediction. They applied rough set model on data sets of NASA Metric Data Program (MDP) [16].

Miguel E.R.Bezera et al proposed a novel approach based on constructive RFB neural network for predicting probability of errors in faulty modules. They have compared their results with KNN as well as variants of KNN that are S-POC-NN and R-POC-NN [17]. A.T. Misirh et al proposed an ensemble of classifier for embedded system domains. They used ensemble of classifiers to solve the issues of optimization of testing resources. They concluded that ensemble of classifiers perform significantly better than individual classifier [18].P.C. Pendharkar [22] proposes a defect prediction model learning problem using hybrid ES–PNN and SA–PNN solution procedures. The results of his experiments signify that hybrid procedures do better than standard machine learning techniques. C.Catal et al. investigated the impact of dataset size, metrics sets, and feature selection techniques on software fault prediction process [21]. P.S.Sandhu et al explored the effectiveness of software metrics requirement and code metrics. They also explored the effectiveness of combination of both types of metrics in order to identify defective modules. They used genetic algorithm technique for classification. Results have shown that combination of both requirement and code metrics is best prediction model for detecting software modules as compared to code metrics that are commonly used for defect prediction [19]. J.Zheng build faults prediction model based on boosting neural networks and three cost sensitive boosting algorithms were empirically studied on mission critical projects of NASA. He used NECM as an evaluation measure in order to evaluate the performance of prediction models. Results have shown that threshold moving based algorithm achieved better prediction results and is easier to implement [20].

3. Empirical Data Collection

Based on concept of ensemble classifiers, we propose the fault proneness model. In order to train our proposed classifier, we acquired data from previous software projects to estimate the performance of the proposed model. Another reason of using previous projects is to determine the confidence level for the utilization of model in practice. The objective of the research study is to find the relationship between the characteristics of software system and their fault proneness. We have observed that certain properties of software modules can be captured from static code attributes.

One more important reason for choosing static code attributes as a data is the fact that we have proposed prediction model to be used prior to the testing phase and the static code attributes are the most suitable data for fault predictor in this situation [24]. Also as static code attributes are the last deliverable of the software development process before development of the actual product; there is a minor chance of change in requirements regarding the functionality concerning these attributes. The data sets used for the purpose of this research is available at Promise Data Repository (http://promisedata.org or http://www.tunedit.org).

4. METHODOLOGY

Prediction of software defect is an important technique that suggested by many researchers [1], [2], [5], [8], [10], [12], [13] to calculate estimated the possible defects modules (i.e. methodology, file or class) system software. Defect prediction can also be helpful in service-oriented system where regression testing is cost and time intensive [38]. Software error proneness aimed at defect liability in module. Ensembles of classifiers offer promise in increasing general the accuracy of the classification [54]. Different classifiers have different knowledge about the problem, so it is an undeniable fact that combination of trained classifiers results in the integration of various features which lead to the far better outcome than in case of individual classifiers. It was the fact that the errors of the classifier be rectified with the classifier used in the ensemble.

Software fault proneness prediction model uses software metrics and erroneous data collected from prior software products to predict the defect status of the software components. Earlier research has shown that this kind of model can predict fault proneness in software modules with significant accuracy [9], [33], [34]. One of the common characteristic in most of the fault proneness prediction approaches is that they use software metrics as an independent variable and the class to which a particular module belongs is set as dependent variable. For fault proneness prediction independent and

dependent variable are specified and after that, it is a matter of finding a pattern that is common in the independent variable so that this pattern can be recognized as one of the dependent variable. In this research study, we propose the model using the concept of classifier ensemble. Figure 1 given below represents the improved fault prediction model. In the proposed model we have employed SMOTE before applying CFS and DBSCAN on the data sets. The reason for applying SMOTE is to balance the data sets, as the faulty instances are in minority and in order to train the classifiers on faulty instance there is a need to increases the samples of faulty class. Pre-processing data sets by applying CFS and DBSCAN classifiers are trained. The data was divided in 30:70.

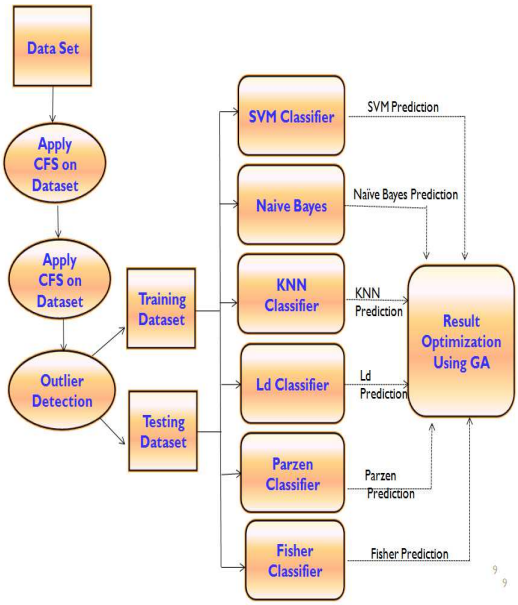

Figure1: Improved Fault Prediction Model

5. EXPERIMENTS AND RESULTS

Results have shown that by applying pre- processors to the data sets significantly improve the performance of prediction model. In our previous research study we have deal with feature selection and outlier detection where as in current approach we have address class imbalance issue as well. SMOTE has been employed to balance the data sets. As, this is a fact that number of faulty modules in a data set are in minority as compared to number of non-faulty data sets. Therefore when train classifier with imbalanced data set, the classifier tends to ignore the small classes concentrating on majority classes. The classifiers tend to generate high predictive accuracy over the majority class, but poor predictive accuracy over the minority class.

We have observed that dealing with class imbalance issue greatly improve the precision rate of prediction model whereas maintain Pf, balance and accuracy. However from the experimental results we have also find that probability of detection has been reduced by applying SMOTE. As in our previous research study Pd was 9% greater than existing approach on average. On the other hand precision rate of prediction model has been increased by 36% on average. Precision is measuring number of defective software modules correctly predict as defective among the modules classified as defective. Extension of our approach to Agile methods can also be beneficial from impact as well as cost benefit point of view [39].

Precision is low if the number of defective modules predicted correctly is small in number or large number of defective modules is predicted as non-defective. So it is a significant evaluation measure for critical system where it is more important to correctly predict which software component is defective so that more testing has been done in order to make the software product error free. If a software module that is actually defective but because of poor precision rate predicted as non-defective would be less tested and therefore there is a chance that error may occur when software is deployed in real space. Therefore precision is a vital evaluation measure for the critical real time software. We have also compared our result with [18] and [31] in order to endorse the efficiency of proposed approach. Figure-2 and figue-3 illustrate the comparison of our technique with other techniques. Results have shown that our improved approach outperforms the existing technique.

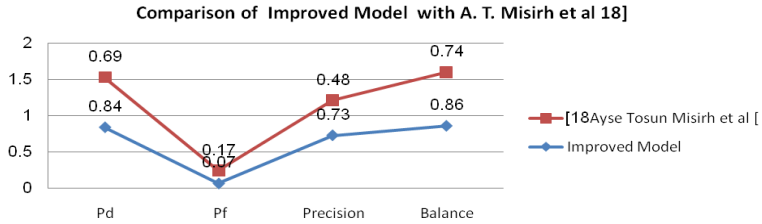

Figure-2: Comparison with A T Misirh et al[18]

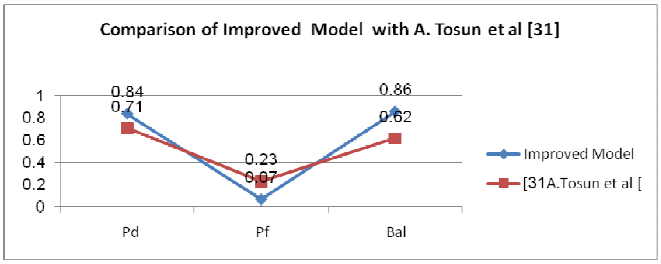

Figure-3: Comparison with A Tosun et al[31]

On the other hand probability of detection is number of correctly classified defective modules among the modules that are predicted as faulty. Probability of detection is an important measure for the commercial software's where we want to reduce the cost of software development. High Pd means less number of software modules are incorrectly labelled as defective so testing resources has been used optimally utilize that would results in reducing software cost. As 40% of software development budget has been spend on testing phase. Form experimental results we have find that in order to improve the precision rate class imbalance issue must be addressed. Figure-4 depicts the comparison of improved approach with our previous proposed approach.

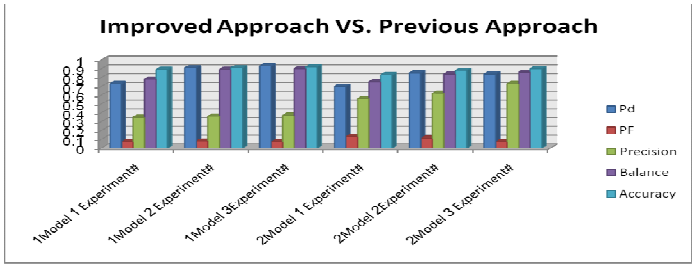

Figure-4: Comparison of the proposed and available techniques

6. CONCLUSION

We study how to improve our previous defect prediction approach. Results has shown that by addressing the issue of class imbalance the precision rate of defect predictor greatly improved as class imbalance significantly affect the performance of defect predictors. We have also observed that the probability of prediction would be affected a bit when using balanced datasets. Therefore, for the critical system this approach performs well.

The improved model offers a new dimension for software fault proneness prediction research. This problem can be solved using different other classifier based algorithms and combinations of both machine learning and statistical algorithms.

REFERENCES

1. Ching-Pao Chang, Chih-Ping Chu, "Defect prevention in software processes: An action-based approach", Journal of Systems and Software, Volume 80, Issue 4, April 2007, Pages 559–570.

2. T.M. Khoshgoftarr et al, Attribute Selection and Imbalanced Data: Problems in software Defect Prediction. In 22nd International Conference on Tools with Artificial Intelligence (2010).

3. L.Pelayo, S. Dick, "Applying novel resampling strategies to software defect prediction." Fuzzy InformationProcessing Society, 2007. NAFIPS'07.Annual Meeting of the North American.IEEE, 2007.

4. Zhan Li; Reformat, M.; , "A practical method for the software fault-prediction," Information Reuse and Integration, 2007. IRI 2007. IEEE International Conference on , vol., no., pp.659-666, 13-15 Aug. 2007.

5. Singh, P.; Verma, S.; , "An Investigation of the Effect of Discretization on Defect Prediction Using Static Measures," Advances in Computing, Control, & Telecommunication Technologies, 2009. ACT '09. International Conference on , vol., no., pp.837-839, 28-29 Dec. 2009.

6. Huanjing Wang; Khoshgoftaar, T.M.; Napolitano, A,"A Comparative Study of Ensemble Feature Selection Techniques for Software Defect Prediction," Machine Learning and Applications (ICMLA), 2010 Ninth International Conference on , vol., no., pp.135-140, 12-14 Dec. 2010.

7. P.S.Sandhu et al, A Genetic Algorithm Based Classification Approach for Finding Fault Prone Classes. In Proceedings of World Academy of Science Engineering and Technology (2009).

8. I. Gondra, Applying machine learning to software fault-proneness prediction. The Journal of Systems and Software (2008).

9. C.Catal , B. Diri , "A systematic review of software fault prediction studies". Expert Systems with Applications (2009).

10. S. Lessmann et al; "Benchmarking Classification Models for Software Defect Prediction: A Proposed Framework and Novel Findings". IEEE Transactions on Software Engineering, Vol. 34, No. 4, July/August 2008.

11. M Mie et al , "Application Of Neural Network For Predicting Software Development Faults Using Object-Oriented Design Metrics". Proceedings of the 9th International Conference on Neural Information Processing (2000).

12. P. Singh, "comparing the effectiveness of machine learning algorithms for defect prediction". International Journal of Information Technology and Knowledge Management July-December 2009, Vol 2, No. 2, pp. 481-483.

13. B. Turhan; A. Bener, "A Multivariate Analysis of Static Code Attributes for Defect Prediction".

14. H.Wang; T.M. Khoshgoftaar; A. Napolitano, "A Comparative Study of Ensemble Feature Selection Techniques forSoftware Defect Prediction". Ninth International Conference on Machine Learning and Applications 2010.

15. Q.Song et al, "A General Software Defect-Proneness Prediction Framework". IEEE transactions on software engineering, vol. 37, No.X (2011).

16. Y. Weimin ;L.Longshu , "A Rough Set Model for Software Defect Prediction". 2008 International Conference on Intelligent Computation Technology and Automation.

17. M.R.E. Bezerra et al, "A Constructive RBF Neural Network for Estimating the Probability of Defects in Software Modules". Proceedings of International Joint Conference on Neural Networks, Orlando, Florida, USA, August 12-17, 2007.

18. A. T. Mısırh et al, "An industrial case study of classifier ensembles for locating software defects", Springer Science and Business Media, LLC (2011).

19. P.S.Sandhu et al , "A Study on Early Prediction of Fault Proneness in Software Modules using Genetic Algorithm". World Academy of Science, Engineering and Technology 72 2010

20. J. Zheng, Cost sensitive boosting neural networks for software defect prediction. Journal Expert Systems and Applications (2010).

21. C catal et al , "Investigating the effect of dataset size, metrics sets, and feature selection techniques on software fault prediction problem". In the Journal Information Sciences (2009).

22. P.C. Pendharkar, "Exhaustive and heuristic search approaches for learning a software defect prediction model". In Journal of Engineering Applications of Artificial Intelligence, (2010).

23. B. Hamid et al ; " Anticipating Software Fault Proneness using Classifier Ensemble: An Optimize Approach". In Proceeding (780) Software Engineering / 781: Control Applications – 2012.

24. Gray , D , Bowes , D , Davey , N , Sun , Y & Christianson , B 2010 , ' Software defect prediction using static code metrics underestimates defect-proneness ' . in : IEEE International Joint Conference on Neural Networks (IJCNN) . IEEE , pp. 1-7.

25. D. Rodr´ıguez et al, Searching for Rules to find Defective Modules in Unbalanced Data Sets. In Proceedings of 1st International Symposium on Search Based Software Engineering (2009).

26. GuQiongn et al ,"Data Mining on Imbalanced Data Sets". In International Conference on Advanced Computer Theory and Engineering (2008).

27. N.Chawla, "Data Mining and Knowledge Discovery Handbook, Springer US 853-867,2005.

28. Gray, Det al, "Software defect prediction using static code metrics underestimates defect-proneness," Neural Networks (IJCNN), The 2010 International Joint Conference on , vol., no., pp.1,7, 18-23 July 2010.

29. S. Kotsiantis et al, “Handling imbalanced datasets: A review”. In GESTS International Transactions on Computer Science and Engineering, Vol.30, (2006).

30. E . Cantu-Paz.“Feature Subset Selection, Class Separability, and Genetic Algorithms”.In Genetic and Evolutionary Computation Conference, Seattle, WA, June 26-30, (2004).

31. A.Tosun et al , Reducing False Alarms in Software Defect Prediction by Decision Threshold Optimization, Proc 3rd International Symposium on Empirical Software Engineering and Measurement IEEE Computer Society Washington, DC, USA ,2009.

32. Koru, A. Günes, and Hongfang Liu. "An Investigation of the Effect of Module Size on Defect Prediction Using Static Measures." (2005).

33. Mizuno, Osamu, and Hideaki Hata. "Prediction of Fault-prone Modules Using A Text Filtering Based Metric."International Journal of Software Engineering and Its Applications Vol. 4, No. 1, January 2010 .

34. Yu, Liguo, and Alok Mishra. "Experience in predicting fault-prone software modules using complexity metrics."Quality Technology & Quantitative Management (ISSN 1684-3703), to appear (2012).

35. Menzies, Tim, Jeremy Greenwald, and Art Frank. "Data mining static code attributes to learn defect predictors." Software Engineering, IEEE Transactions on 33.1 (2007): 2-1[38]3.

36. Menzies, Tim, et al. "Defect prediction from static code features: current results, limitations, new approaches." Automated Software Engineering 17.4 (2010): 375-407.

37. A.G. Kour, H.Liu, “Building Effective Defect-Prediction Models in Practice”. In Journal of IEEE Software November 2005 Vol 22, No.6, pp. 23-29.

38. Tamim Ahmed Khan, Umair Maqsood, Waqar Ahmed, Sadia Ashraf, Faisal Aftab, On Model Based Regression Testing of Web Services, Journal of Applied Environmental and Biological Sciences, Volume 1 (9), Article ID J. Appl. Environ. Biol. Sci.-679-S.

39. Tina Roustapisheh1, Hasan Dehgan Dehnavi2, Shahnaz Nayebzadeh3, A comparison of the Impacts of Lean and Agile Strategies on Improvement of Corporate Performance (Case Study: Shiraz City), Journal of Applied Environmental and Biological Sciences J. Appl. Environ. Biol. Sci., 4(7)25-31, 2014.