A Review on Software Defect Prediction

H. S. Shukla
Department of Computer Science,
D.D.U. Gorakhpur University, Gorakhpur, India

Deepak Kumar Verma
Department of Computer Science,
D.D.U. Gorakhpur University, Gorakhpur, India

Abstract— **Today software has become the key element in every environment. Quality of a software product is correlated with the number of defects as well as it is limited by time and cost. Software defects are expensive in terms of quality and cost. Software defect prediction is the process of tracing defective components in software prior to the deployment of the product. Occurrence of defects is inevitable, but we should try to limit these defects to minimum count. Defect prediction leads to reduced development time, cost, reduced rework effort, increased customer satisfaction and more reliable software. Therefore, defect prediction practices are important to achieve software quality and to learn from past mistakes. Here in this paper we have a literature survey of last two decades and investigated about recent advancement in the area of defect prediction.**

Keywords: Defect Prediction, Software Quality, Defect Detection.

I. Introduction

A software defect is an error, flaw, bug, mistake, failure, or fault in a computer program or software that may generate an inaccurate or unexpected result. A project team always tries to produce a quality software product with zero or little defects. Quality of software can not be achieved by identifying the bugs or errors only at the testing phase. Bugs are automatically injected at every phase of software development life cycle. So the detection of the bugs should be at every phase in place of only at the testing phase. High risk components within the software project should be caught as soon as possible, in order to enhance software quality. Software defects always increases the cost and time in completing a software product with expected quality. Moreover, identifying and rectifying defects is one of the most time consuming and expensive software processes. It is not practically possible to eliminate each and every defect but reducing the magnitude of defects and their adverse effect on the projects is achievable. Software defect prediction is the process of locating defective modules in software. To produce high quality software, the final product should have as few defects as possible. Early detection of software defects could lead to reduced development costs and rework effort and more reliable software. So, the study of the defect prediction is important to improve software quality. Many organizations want to predict the number of defects (faults) in software systems, before they are deployed to gauge the likely delivered quality and maintenance effort.

Manuscript received Dec, 2015.

***Prof. H. S. Shukla**, Department of computer Science, D.D.U. Gorakhpur University, Gorakhpur(U.P.), India; Mobile No-9451678201*

***Deepak Kumar Verma**, Department of computer Science, D.D.U. Gorakhpur University, Gorakhpur(U.P.), India; Mobile No-9452834077*

II. Review Work

In this section we did the literature survey of software defect prediction models and techniques. Below we have given a brief review of the work done by many workers in the above filed of defect forecast with the objective of finding out future strategies in this field:

Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006)[1] used machine learning technique for finding out the number of defects viz. called Regression via Classification (RvC). A comparative experimental study of many machine learning algorithms was carried out in order to evaluate this approach. Pekka Forselious collected the data, whose applications were maintained by Finland bank. The framework developed provided the potential to detect faults that were otherwise not detectable.

Norman Fenton et.al.(1999) [2], have described a probabilistic model for software defect prediction. The aim here is to design a model which is a combination of diverse forms that may be often casual, with available evidence in development of software so that the work can be done in more natural and efficient manner than it was previously done. Here a critical review of numerous software metrics and statistical models and the state-of-the art has been carried out. In most of the models used for prediction of defect size and complexity metrics is used.

Other reside on the testing data, quality development process and/or a multivariate style is followed. To authenticate this approach, Graphical probability models (also known as Bayesian Belief Networks) has been made use of. In order to develop the probability model, subjective judgement of professionals, project managers who are experienced has been utilised in order to predict the model for error. This has been used throughout the development life cycle of the project.This model can not only be used for assessing ongoing projects, but also for exploring the possible effects of a range of software process improvement activities. If costs can be associated with process improvements, and benefits assessed for the predicted improvement in software quality, then the model can be used to support sound decision making for SPI (Software Process Improvement).

Ahmet Okutan, et.al.(2012)[3], proposed a novel method using Bayesian networks to explore the relationships among software metrics and defect proneness. Nine data sets from Promise data repository has been used and show that RFC, LOC, and LOCQ are more effective on defect proneness. Also proposal for two more metrics, i.e. NOD for the number of developers and LOCQ for the source code quality has been given. Finally in the end, marginal defect probability of the software, its effective metrices and their relationships has been discussed.

Mrinal Singh Rawat et. al.(2012)[4], identified causative factors which in turn suggest the remedies to improve software quality and productivity. They showed how the various defect prediction models are implemented resulting in reduced magnitude of defects. They presented the use of various machine learning techniques for the software fault prediction problem. The unfussiness, ease in model calibration, user acceptance and prediction accuracy of these quality estimation techniques demonstrate its practical and applicative magnetism. These modeling systems can be used to achieve timely fault predictions for software components presently under development, providing valuable insights into their quality. The software quality assurance team can then utilize the predictions to use available resources for obtaining cost effective reliability enhancements.

Supreet Kaur, et.al. (2012)[5],did performance analysis of Density-Based Spatial Clustering of Applications with Noise. It was used for error forecasting in OOSS and C++ language based software parts. It used metric approach for forecasting. Firstly, in Nkc3, KNOWN AS Java based dataset, 39 metrics were used, which were later reduced to eight by calculating the worth of the subset of attributes.

Xiao-dong Mu et. al.,(2012)[6], in their work to improve the accuracy of software defect prediction, a co-evolutionary algorithm based on the competitive organization is put forward for software defect prediction. During this algorithm, firstly, competition mechanism is introduced to organization co-evolutionary algorithm. Then, three evolution operators which are reduced operator, allied operators and disturbed operators are developed for evolution of population. And competition is considered for calculate the fitness function. When the algorithm applied into software defect prediction, it improves the accuracy of software prediction through increases the diversity of population.

N Fenton, et. al. (2008)[7], Used Baysian networks for forecasting of reliability and defectiveness of software. It used casual process factors and qualitative and quantitative measure, thus catering to the limitations of traditional software limitations. The use of dynamic discritization method results in better prediction model for software defect.

Jie Xu, et. al. (2010)[8], various statistical techniques and machine learning methods were used to variefy the validity of software defect prediction models.. Here neuro fuzzy approach was used. Data from ISBSG were taken to carry out the work.

Manu Banga, (2013) [9], here a new computational intelligence sequential hybrid architectures involving Genetic Programming (GP) and Group Method of Data Handling (GMDH) viz. GPGMDH have been discussed. Besides GP and GMDH, a host of techniques on the ISBSG dataset has been tested. The proposed GP- GMDH and GMDH-GP hybrids outperformed all other stand-alone and hybrid techniques. It is concluded that the GPGMDH or GMDH-GP model is the best model among all other techniques for software cost estimation.

Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014)[10] for the prediction of software defects used artificial neural network in order to better the generalization capability of the algorithm. Further support vector machine technique was used along with the learning algorithm and evolutionary technique. Thus this led to the maximization classification margin and prevented overfitting problem. This algorithm was tested with eleven machine learning models from NASA datasets. The conclusion drawn was that it provided better accuracy and precision than other models.

Kamaljit Kaur (2012)[11] led to the identification of reusable software modules in OOPS using neural network technique. Used metrics for structural analysis. These became the input for neural network. The training data using these metric values were used and were further tested using MAE, RMSE. It was found that the present model could very well improve the accuracy of the defect prediction.

Mrs. Agasta Adline, Ramachandran. M (2014)[12] Predicting the fault-proneness of program modules when the fault labels for modules are unavailable is a challenging task frequently raised in the software industry. They attempted to predict the fault–proneness of a program modules when fault labels for modules are not present. Supervised techniques like Genetic algorithm based software fault prediction approach for classification has been proposed.

Karpagavadivu.K, et.al. (2012)[13] analyzed the performance of various techniques used in software fault prediction. And also described some algorithms and its uses. They found that the aim of the fault prone module prediction using data mining is to improve the quality of software development process. By using this technique, software manager effectively allocate resources. The overall error rates of all techniques are compared and the advantages of all methods were analyzed.

Ahmet Okutan and Olcay Taner Yıldız, (2013)[14] proposed a new kernel method to predict the number of defects in the software modules (classes or files). The proposed method is based on a pre-computed kernel matrix which is based on the similarities among the modules of the software system. Novel kernel method with existing kernels in the literature (linear and RBF kernels) has been compared and show that it achieves comparable results. Furthermore, the proposed defect prediction method is also comparable with some existing famous defect prediction methods in the literature i.e. linear regression and IBK. It was seen that prior to test phase or maintenance, developers can use the proposed method to easily predict the most defective modules in the software system and focus on them primarily rather than testing each and every module in the system. This can decrease the testing effort and the total project cost automatically.

Yajnaseni Dash, Sanjay Kumar Dubey, (2012)[15] surveyed different research methods for the prediction of OO metric using neural network techniques. This technique was found to be best suited for prediction in case of object oriented metrics. Neural network used minimum calculation work as compared to other artificial intelligence techniques. It has better representation capability and is capable of performimg complicated functions.

Ms. Puneet Jai Kaur, Ms. Pallavi, (2013)[16] used different data mining techniques for software error prediction, like association mining, classification and clustering techniques. This has helped the software engineers in developing better models. In case where defect labels are not present, unsupervised techniques can be used for model development.

Xiaoxing Yang, et.al. (2014)[17] Used the rank performance optimization technique for software forecasting model development. For this rank to learning approach was used. The model was developed on previous work and was later studied for improving the performance of the model. . The work includes two aspects: one is a novel application of the learning-to-rank approach to real-world data sets for software defect prediction, and the other is a comprehensive evaluation and comparison of the learning-to-rank method against other algorithms that have been used for predicting the order of software modules according to the predicted number of defects. This study shows that the effect of optimization of the model performance using rank to learning approach truly improve the prediction accuracy.

Yi Liu et al., (2010) investigated the problem of software quality classification modeling using the history of metric dataset obtained from single software project. The classification modeling got from a single dataset is generally not adequate to build strong and an accurate model. To tackle the issue, software quality classification modeling was done using multiple datasets sourced from different software projects. Previous study has demonstrated that using multiple datasets for validation can achieve robust genetic programming -based models. The effectiveness of the modeling using multiple datasets is extensively studied in this paper. Moreover, a 30 novel general purpose based classifier consisting of training, multiple-dataset validation, and voting phases, is proposed. The datasets used for experimentation were obtained from NASA software projects. The performance by the proposed classifier was compared with the results of seventeen other data mining techniques. The comparison shows that the proposed approach is more effective and accurate with the use of multiple datasets.

Song et al.,(2006) proposed prediction of defect associations and defect correction effort based on association rule mining methods. Test resources are more effectively allocated for detecting software defects. The proposed method was applied to more than 200 projects. The experiment results show the accuracy achieved is high for both defect association prediction and defect correction effort predictions. The result of the proposed method was also compared with PART, C4.5 AND Naïve Bayes methods. The comparison shows that the proposed method accuracy was higher by at least 23 percent. Lessmann et al., 2008 investigated the performance of classification algorithm. To compare the software defect prediction, experiments were conducted using 10 public domain datasets from NASA Metric Data repository, using 22 classifiers. The general impression is that the predictive accuracy metric based classification is useful. The results also indicated that the importance attached to particular classification algorithms was not significant as generally 31 assumed. The results showed that there was no significant difference among the top 17 classifiers.

Munson et al.,(1992) investigated statistical based methods like discriminating analysis for the detection of fault prone programs. In this paper it was proposed to implement principal components to reduce multicollinear complexity metrics to uncorrelated measures on orthogonal complexity domains. The transformed data was used to classify the programs. Eleven software metrics were computed from the programs and data prepared for the classification engine. The misclassification rate was 10 percent showing a high degree of classification.

Tom M. Mitchell (1997) attempted to classify data usingNaïve Bayesian algorithm. Naïve Bayes is one of the popularly used learning algorithms in data mining and machine learning. It is popular because of its efficient and effective inductive learning algorithms. Classification based on Naïve Bayes algorithms gives very competitive performance due to its conditional independence assumption.

Menzies et al.,(2004) proposed Naïve Bayes learners for studying the defect detectors from static code measures. Comparison 32 of Naïve Bayes learners and entrophy-based decision tree learner was done to show the effectiveness of Naïve Bayes learners. The study concludes that accuracy was not an effective way to assess those detectors. When using Naïve Bayes learners on heavily stratified data, 200-300 examples are enough to learn adequate detectors.

Riquelme et al., (2009) used the promise repository to obtain the software metrics program dataset and proposed a genetic algorithm search for rules characterizing subgroups with a high probability of being defective. The genetic algorithm handles the problem of unbalanced datasets efficiently especially when the unbalanced sets consists of more non-defective samples than the defective samples.

Catal et al., (2007) modeled Artificial Immune System based on the Human immune system for defect prediction. The proposed classifier imitates the behavior of the antigen and the antibody during an attack by pathogens into the human biological system. The evolution of the immune system to new attacks is modeled to solve the software defect prediction problem.

Achcar et al., (1991) approach to the software reliability prediction extended the Bayesian approach by using Poisson distribution to propose a novel software reliability model. The Bayesian inference models used Metropolis-within-Gibbs algorithms for the Moranda's model. Model selection was based on the predictive density.

Xin Jin et al., (2006) attempted to provide software reliability for Software Engineering Management and made a list of metrics and implemented them in a common dataset. This research work tries to improve all the measures made by them by incorporating additional metrics with a combined effort. Good results were got from experimenting with the artificial immune recognition system of classifiers.

Andersson et al., (2007) proposed a novel method for analysis of fault distributions in software systems. The proposed method used replicated quantitative analysis which was mathematically modeled and proved in detection of fault distributed across complex software system.

Emam et al., (2001) based on the previous work on predicting faulty modules using object oriented design metrics proposed enhancement in the computation of design metrics. The proposed method of feature extraction performed better than the previous models; however the disadvantages of the proposed method were also highlighted.

Cartwright et al., (2000) investigated an industrial object-oriented (OO) system made of 133,000 lines of C++ using empirical methods. The data system was a subsystem of a telecommunication product. The study showed that the OO constructs such as inheritance, polymorphism are not really useful. The study found that the classes in inheritance structures were three times more defect prone than the classes that are not in inheritance structures. Prediction systems were constructed using a number of states and events per class. Though the prediction systems have only local significance, the need of suites of metrics in OO technology is not required; thus measurement technology becomes more accessible.

Kim et al., (2008) proposed change classification method for predicting dormant software bugs. Change classification was based on machine learning classifiers, which helps in determining the similarity of change to previous buggy changes or clean changes. The presences of bugs were predicted using the change classification. The classifier is trained using the features from revision history of the software, these when applied classifies the changes in the software as buggy or clean. The results showed 78% accuracy. The change classification was superior as it had small prediction granularity and semantic information of source code was not required for classification. Change classification works on a broad array of programming languages.

Liu et al., (2010) proposed a novel genetic programming based on search approach for software quality modeling with multiple software project repositories. The training on multiple projects provides a cross project perspective on software quality modeling which can effectively sum up the quality trends of the development organization. This approach includes three strategies using Baseline classifier, Validation classifier and Validation-and-Voting classifier. Case study of software metrics and defect data using 7 systems showed that the Validation-and-Voting classifiers are better software quality model. The paper also presents another case study consisting of 17 different machine learners using majority-voting approach for predicting fault proneness class of program modules.

Ebru Ardil et al., (2009) investigated the modules of feed forward neural network. The feed forward neural networks were the first and the simplest type of artificial neural network. Faults are mainly found in the modules of the neural network, the study investigates the most severely affected modules in comparison with other modules.

Jianhong, et al (2010) explored five Neural Network Based techniques and comparative analysis is performed for the modeling of severity of faults present in function based software systems. The NASA's public domain defect dataset is used for the modeling. The comparison of different algorithms is made on the basis of Mean Absolute Error, Root Mean Square Error and Accuracy Values. It is concluded that out of the five neural network based techniques Resilient Backpropagation algorithm based Neural Network is the best for modeling of the software components into different level of severity of the faults. Hence, the proposed algorithm can be used to identify modules that have major faults and require immediate attention.

Yuan, et al (2011) presented a method to evaluate the software reliability using Fuzzy-Neural network. In order to improve the accuracy of the evaluation, this paper established a reliability prediction model based on adaptive-network based fuzzy inference system (ANFIS). The model use the reliability data(defect counts of every thousand code lines) of one software project as input data, and use the prediction of reliability as output data, training Adaptive-Fuzzy neural network, get the membership function of defect counts of every thousand code lines.

III. Defect detection and prevention approaches

Important decisions needed to be made during the development of software products. Conceivably the most vital of these is the choice of time of releasing the software. The cost of making a wrong decision can be potential decision for the reputation of a product or its provider. However, such decisions are often made casually, and not on the basis of more objective and responsible criteria. Hence software task and quality developers must deal with a mixture of unsure factors like workers, equipments, improvement methods and testing strategy to realize the deliverance of a excellent product to budget and on time. These undecided factors influence the opening, recognition and improvement of defects at all stage in the growth life cycle from early requirements to product release. Hence, to achieve software excellence during development, exceptional importance needs to be tackled to the following three activities:

- ➢ Defect prevention
- ➢ Defect detection
- ➢ Defect correction

- **Empirical defect prediction technique:** This technique predicts the number of defects per size (defect density). Defect density determines the number of defects per thousand lines of codes based on historical data.
- **Defect discovery technique:** This technique describes the projections based on based on time or phases of defect density found "in process" onto a theoretical discovery curve. It predicts defect density by time period enabling the estimation of defects to be found in the test. This technique is found in the SWEEP and STEER models.
- **Defect prevention technique:** This technique focuses on root cause analysis of most frequently occurring defects. Sample of defect reports are selected for in-depth casual analysis and actions can be taken to make process change or improve training to eliminate the root cause of defects and prevent their recurrence.
- **Orthogonal defect classification technique:** This technique is based on classification and analysis of defects to identify project status based on comparison of current defects with historical patterns, identify areas for process improvement based on analysis of defect types, Triggers, Impact and Source.
- **Statistical process control technique:** This technique use the control charts to determine whether inspection performance was consistent with prior process performance in terms of selected attributes.

- **Coqualmo technique:** This is a defect prediction model for the requirements, design and coding phases based on sources of introduction and discovery techniques used. The strength of this technique is that it predicts the defects for three phases.

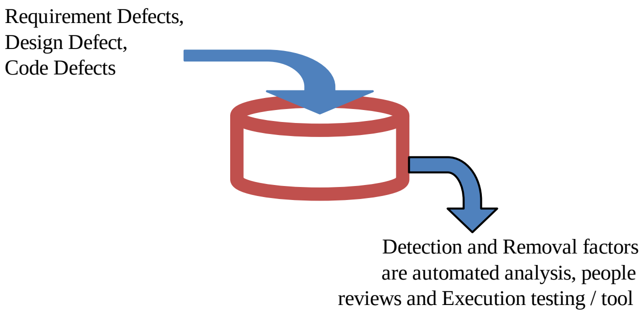
Requirement Defects,
Design Defect,
Code Defects
Detection and Removal factors are automated analysis, people reviews and Execution testing / tool

Fig.1:Coqualmo Technique

- **Fault proneness technique:** This technique focuses on analysis of work product attributes to plan for allocation of defect detection resources as inspection and testing.

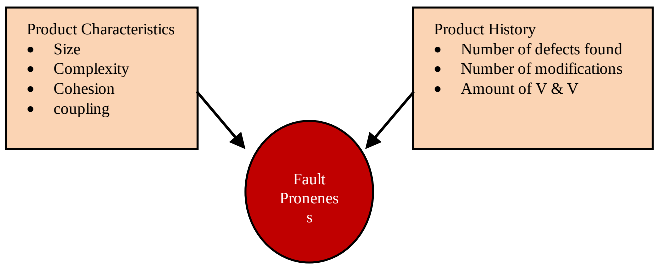

Fig.2: Fault Proneness Techniques

- **Capture Recapture technique:** This technique focuses on analysis of pattern of defects detected within an artifact by independent defect detection activities. This technique can be used as soon as data are available.

Software defect containing the quality issue can be understood in many ways but the most common one is the variation in the specific results or expectations which will ultimately lead to the failure of the operation. A defect might originate in one development stage and be detected in the same or a later stage. For this Verification and Validation process is most widely used. For instance, a missing interface in a design specification could propagate to the coding stage, resulting in missing functionality in the code. We might detect this design defect during a design inspection, code inspection, unit test, function test, or system test. Because defect detection focuses on abstraction levels, we consider that *primary* defect detection activities are at the same level of abstraction and *secondary* defect detection activities are at a different level. For example, for design defects, design inspection and functional testing are primary activities, and code inspection and unit testing are *secondary* defect detection activities. The figure3 shows the systematic classification of pre deployment and post deployment defect detection techniques:

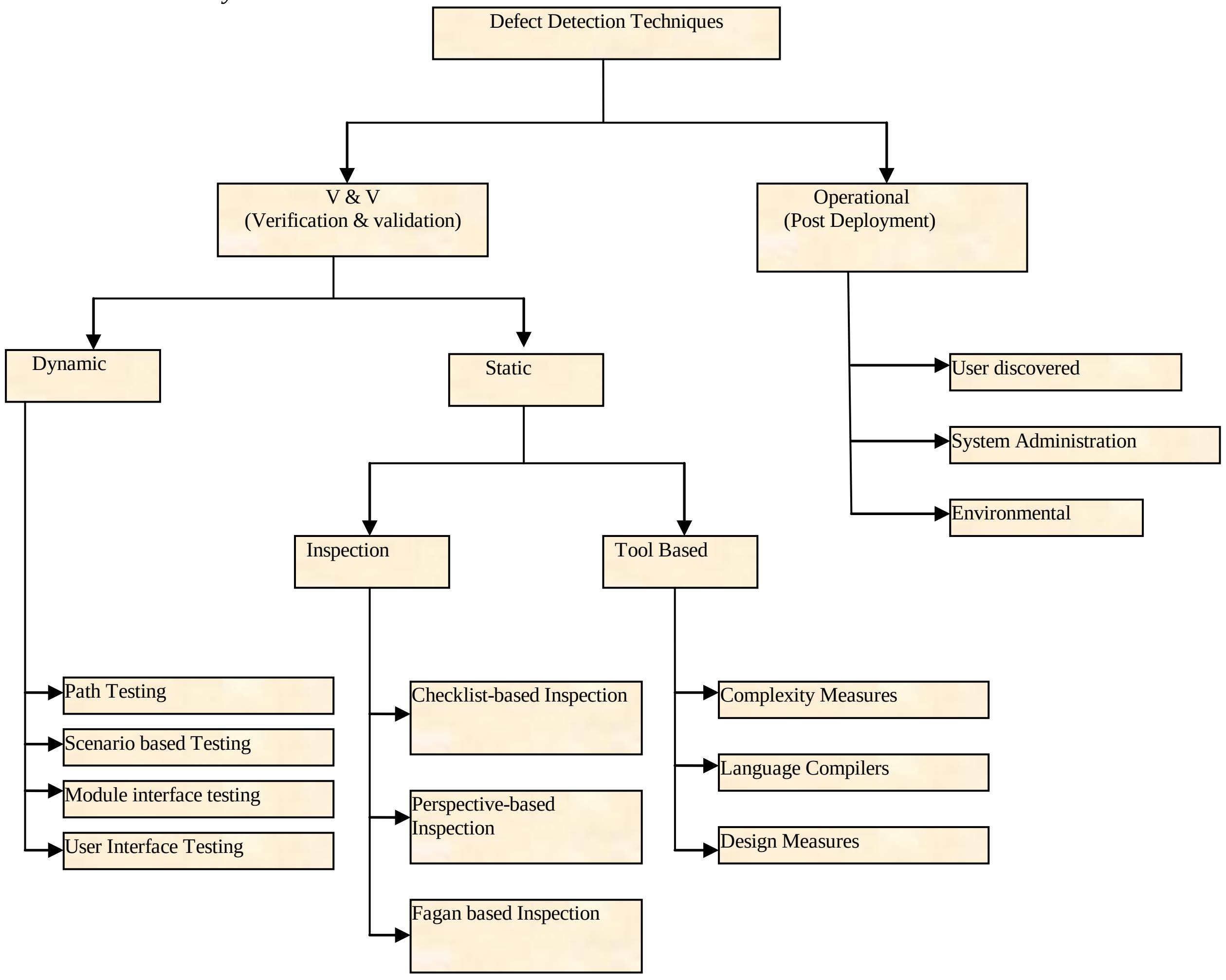

Fig.3: Classification of Defect Detection Techniques

IV. Results and Discussion

Defect prediction can offer an added probability to the event team to retest the modules or files that the faultiness chance is high. By spending longer on the defective modules and no time on the non-defective ones, the resources of the project would be used better and as a result, the upkeep part of the project are easier for each the customers and also the project owners. Whereas making a critique of the software defect prediction studies, [18] argue that though there are several studies in the literature, defect prediction drawback problem from solution. There are some wrong assumptions regarding however defects are outlined or observed and this causes misleading results. Their claim will be understood better once we notice that some outline defects as observed deficiencies whereas some others define them as residual ones. We have a tendency to explore the publications related to defect prediction we see that in early studies static code features were used a lot of. However later on, it absolutely was understood that beside the impact of static code metrics on defect prediction, different measures like process metrics are effective and may be investigated as an example, [18] argue that static code measures alone don't perform to be ready to predict software defects accurately. To support this concept one argues that, if a software is flawed this may be associated with one in all the following:

- The specification of the project could also be wrong either as a result of contradictory requirements or missing features. it's going to be too complicated to realize or not very well documented.

- The style may be poor, it may not think about all requirements or it's going to reflect some requirements incorrectly.
- Developers don't seem to be competent enough for the project.
- There may be a project management drawback and also the software life cycle methodologies won't be followed alright.
- The software might not be tested enough therefore some defects won't be fixed during the period.

Not any of the above factors are associated with code metrics and all of them could very well have an effect on defect proneness. So, the question is that factors or metrics are effective on defectiveness and the way can we measure their effect.

Each of these uncertain factors influences the detection and correction of defects at all stages in the development life cycle from initial requirements to product delivery. Software efforts tend to pay more attention to the following three basic issues:

1. To forecast the error in abundance in the software.
2. Time estimation of the system as related to its reliability.
3. To test the design and test process impact on the number of defects and their densities.

V. CONCLUSION

Defect prediction techniques vary in the types of data they require, some require little data and other requires more. Some use work product characteristics and others require defect data only. All the techniques have strengths and weaknesses depending on the quality of the inputs used for prediction. The problem occurs during the selection of defect detection method. The choice of defect detection method depends on factors such as the artifacts, the types of defects they contain, who is doing the detection, how it is done, for what purpose, and in which activities. Factors also include which criteria govern the evaluation. These factors show that many variations must be taken into account. When we search the evidence for the pros and cons of using some defect detection method, we must choose specific levels of these factors to guide the appraisal of empirical evidence. Defect prediction techniques are very useful in producing quality software. With the help of the defect prediction techniques we can improve the quality and reliability of the software. Also the defect prediction leads to high quality software, reduced cost, reduced maintenance, more customer satisfaction. Further investigation need to be carried with emphasis on preprocessing of data and classifiers specifically designed for defect prediction in software modules.

REFERENCES

[1] Bibi S., Tsoumakas G., Stamelos I., Vlahavas I.(2006), "Software Defect Prediction Using Regression via Classification", IEEE International Conference on Computer Systems and Applications, pp.330 – 336.

[2] Norman Fenton, Paul Krause and Martin Neil, (1999), “A Probabilistic Model for Software Defect Prediction”, For submission to IEEE Transactions in Software Engineering.

[3] Ahmet Okutan, Olcay Taner Yıldız,(2012) “Software defect prediction using Bayesian networks”, Empirical Software Eng (2014) 19:154–181 © Springer Science+Business Media, LLC.

[4] Mrinal Singh Rawat, Sanjay Kumar Dubey,(2012) “Software Defect Prediction Models for Quality Improvement: A Literature Study”, IJCSI International Journal of Computer Science Issues, Vol. 9, Issue 5, No 2, pp 288-296.

[5] Supreet Kaur, and Dinesh Kumar, “Software Fault Prediction in Object Oriented Software Systems Using Density Based Clustering Approach”, International Journal of Research in Engineering and Technology (IJRET) Vol. 1 No. 2 March, (2012)ISSN: 2277-4378

[6] Xiao-dong Mu, Rui-hua Chang, Li Zhang, “Software Defect Prediction Based on Competitive Organization Co-Evolutionary Algorithm”, Journal of Convergence Information Technology(JCIT) Volume7, Number5, (2012).

[7] N. Fenton and M. Neil (2008) “Using Bayesian networks to predict software defects and reliability”, Proc. IMechE Vol. 222 Part O: J. Risk and Reliability, pp 702-7.

[8] Jie Xu, ²Danny Ho and ¹Luiz Fernando Capretz, “An Empirical Study On The Procedure Drive Software Quality Estimation Models”, International journal of computer science & information Technology (IJCSIT) Vol.2, No.4, (2010).

[9] Manu Banga, “Computational Hybrids Towards Software Defect Predictions”, International Journal of Scientific Engineering and Technology Volume 2 Issue 5, pp : 311-316, (2013)

[10] Mohamad Mahdi Askari and Vahid Khatibi Bardsiri (2014), “Software Defect Prediction using a High Performance Neural Network”, International Journal of Software Engineering and Its Applications Vol. 8, No. 12 (2014), pp. 177-188.

[11] Kamaljit Kaur (2012), “Analysis of resilient back-propogation for improving software process control” International Journal of Information Technology and Knowledge Management July-December 2012, Volume 5, No. 2, pp. 377-379.

[12] Mrs.Agasta Adline, Ramachandran. M(2014), “Predicting the Software Fault Using the Method of Genetic Algorithm”, International Journal of Advanced Research in Electrical, Electronics and Instrumentation Engineering, Vol. 3, Special Issue 2,, pp 390-398.

[13] Karpagavadivu.K, et.al. (2012), “A Survey of Different Software Fault Prediction Using Data Mining Techniques Methods”, International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 1, Issue 8, pp 1-3.

[14] Ahmet Okutan1 and Olcay Taner Yıldız, (2013), “A Novel Regression Method for Software Defect Prediction with Kernel Methods”, ICPMRA 2013 - International Conference on Pattern Recognition Applications and Methods, pp 216-221.

[15] Yajnaseni Dash, Sanjay Kumar Dubey, (2012), “ Quality Prediction in Object Oriented System by Using ANN: A Brief Survey”, International Journal of Advanced Research in Computer Science and Software Engineering, Volume 2, Issue 2,, pp.1-6.

[16] Ms. Puneet Jai Kaur, Ms.Pallavi, (2013), “Data Mining Techniques for Software Defect Prediction”, International Journal of Software and Web Sciences (IJSWS),International Journal of Software and Web Sciences 3(1), pp. 54-57.

[17] Xiaoxing Yang, et.al. (2014), IEEE TRANSACTIONS ON RELIABILITY, This article has been accepted for inclusion in a future issue of this journal.

[18] Norman Fenton, Paul Krause and Martin Neil, (1999), “A Probabilistic Model for Software Defect Prediction”, For submission to IEEE Transactions in Software Engineering.

About the authors

Professor H. S. Shukla is working as Head, Department of Computer Science & Dean, Faculty of Science, Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur-273001. Prof. Shukla has published more than ten books and above fifty research papers in various National and International Journals/Conferences

Mr. Deepak Kumar Verma is MCA, M.Phil(CS), UGC-NET and working as Research Scholar (Pursuing Ph.D.) in the Department of Computer Science, Deen Dayal Upadhyaya Gorakhpur University, Gorakhpur-273001. The area of research is Software Engineering, Database System. Mr. Verma has published ten papers in various National and International Journals and Conferences.