A Survey on Quality Prediction of Software Systems

Atul Bisht, Amanpreet Singh Dhanoa, Amanpreet Singh Dhillon, Gagandeep Singh

Abstract—In this study, a survey on quality prediction of software systems is performed. Here, the quality is accessed using on the basis of fault proneness of the systems. The work of various researcher is discussed in brief.

Keywords—Software Quality, Software Metrics, Function Based, Object Oriented.

I. INTRODUCTION

HIGHLY reliably software is becoming an essential ingredient in many systems. Public safety and the fabric of modern life depend on software-intensive systems. We can ill afford for important systems to fail due to inadequate software reliability. Software reliability engineering is one of the most important aspect of software quality [1]. A software fault prediction is a proven technique in achieving high software reliability. Prediction of fault-prone modules provides one way to support software quality engineering through improved scheduling and project control. Quality of software is increasingly important and testing related issues are becoming crucial for software. Although there is diversity in the definition of software quality, it is widely accepted that a project with many defects lacks quality. Methodologies and techniques for predicting the testing effort, monitoring process costs, and measuring results can help in increasing efficiency of software testing. Being able to measure the fault-proneness of software can be a key step towards steering the software testing and improving the effectiveness of the whole process. The interest of the software community in program testing continues to grow – as does the demand for complex, and predictively reliable programs. It is no longer acceptable to postpone the assurance of software quality until prior to a product’s release. As the cost of removing bugs increases exponent means earlier the bugs detected lesser is the cost. Recent research in the field of computer program reliability has been directed towards the identification of software modules that are likely to be fault–prone, based on product and/or process–related metrics, prior to the testing phase, so that early identification of fault–prone modules in the life–cycle can help in channeling program testing and verification efforts in the productive direction. Hence, the quality prediction has become important.

Software metrics represent quantitative description of program attributes and the critical role they play in predicting the quality of the software has been emphasized by Perlis et al [3]. That is, there is a direct relationship between some complexity metrics and the number of changes attributed to faults later found in test and validation [4]. Many researchers have sought to develop a predictive relationship between complexity metrics and faults. Crawford et al [5] suggest that multiple variable models are necessary to find metrics that are important in addition to program size. Consequently, investigating the relationship between the number of faults in programs and the software complexity metrics attracts researchers’ interesting. There are several different techniques have been proposed to develop predictive software metrics for the classification of software program modules into fault–prone and non fault–prone categories. Hence, a metric based approach can be investigated for prediction of software quality by identification of fault prone modules.

When the artificial neural network is applied to model software reliability, the invalid software time in the software mistake reports is adopted as the input of software quality prediction model [6-8], and the software quality metrics is used as the input of the neural networks [9-11].

Recently, the object-oriented software metrics has been adopted as the inputs of the neural networks in order to predict object-oriented software quality [12-16]. By applying the software quality metrics as the inputs of neural networks, Khoshgoftaar et al [17-18] proposed some software quality prediction models using neural network of BP (back propagation) and compared these models with the one using nonparametric discriminant, and they found that the software quality prediction models using neural network of BP obtained good prediction accuracy. Hu and Zhong [19] applied the learning vector quantization network to predict the software quality and proposed a software module risk model. With consideration of software fault severity, Zhao and Zhong et al [20] presented a software fault-proneness prediction model by support vector machine and the Chidamber-Kemerer (C&K) object-oriented metrics. In most of the literature of fault-proneness prediction the performance of the techniques are compared by Accuracy Percentage, Probability of Detection and Probability of False Alarms.

Atul Bisht, Amanpreet Singh Dhanoa and Gagandeep Singh are working as faculty at CSE/IT Deptt., Rayat-Bahra Institute of Engineering & Bio-Technology, Mohali, India
Amanpreet Singh Dhillon is working as faculty at CSE Deptt., GGSCMT, Kharar, India.
II. RELATED WORK

The following work is related with the prediction of faults in software systems:

Saida et al [21] surveyed that the basic premise behind the development of object-oriented metrics is that they can serve, as early predictors of classes that contain faults or that are costly to maintain. In their paper they have shown that size can have an important confounding effect on the validity of object-oriented metrics [21].

A critical review of the literature is given by Fenton & Neil [22] and they also made heroic contributions to the subject. In their study, most of the wide range of prediction models used size and complexity metrics to predict defects. Others are based on testing data, the “quality” of the development process, or take a multivariate approach. They also argued for research into a theory of “software decomposition” in order to test hypotheses about defect introduction and help construct a better science of software engineering. The comparison of Fault-Proneness Estimation Models and conclusion that software quality has become one of the most important requirements in the development of systems and fault-proneness estimation could play a key role in quality control of software products. The main objective was to find a compromise between the fault-proneness estimation rate and the size of the estimation model in terms of number of metrics used in the model itself [23].

With the existence of a correlation between a reasonable set of static metrics and software fault-proneness. Static metrics, e.g., the McCabe's cyclomatic number or the Halstead's Software Science, statically computed on the source code, try to quantify software complexity. Dynamic metrics, e.g., structural and data flow coverage measure the thoroughness of testing as the amount of elements of the program covered by test executions. Such metrics only partially reflect the many aspects that influence the software fault-proneness, and thus provide limited support for tuning the testing process [24].

The conclusion to remain competitive in the dynamic world of software development, organizations must optimize the usage of their limited resources to deliver quality products on time and within budget. The proposed fault prediction model used in his study was based on supervised learning using Multilayer Perceptron Neural Network and the results were analyzed in terms of classification correctness and based on the results of classification, faulty classes were further analyzed and classified according to the particular type of fault [25].

Yan Ma [26] suggested that accurate prediction of fault prone modules in software development process enables effective discovery and identification of the defects. Thomas Zimmermann and Nachiappan Nagappan [27] in their paper suggested that in the software development, the resources for quality assurance are limited by time and by cost.

Bindu Goel et al [31] as suggested in their book published in 2008 that the predictions can be used to target improvement efforts to those modules that are likely to be faulty during the operation. The basic hypothesis of software quality prediction is that a module currently under development has defects if a module with the similar product or process metrics in an earlier project (or release) developed in the same environment had defects [32]. Therefore, the information available early within the current project or from the previous project can be used in making predictions. The empirical study detailing software maintenance for web based java applications can be performed to aid in understanding and predicting the software maintenance category and effort [33]. Software maintenance organizations are no exception. Visual approach [34] can be used to uncover the relationship between evolving software and the way it is affected by software bugs. By visually putting the two aspects close to each other, we can characterize the evolution of software artifacts. Software maintenance is central to the mission of many organizations. Thus, it is natural for managers to characterize and measure those aspects of products and processes that seem to affect cost, schedule, quality, and functionality of a software maintenance delivery [35].

Statistical, machine learning, and mixed techniques are widely used in the literature to predict software defects. Khoshgoftaar used zero-inflated Poisson regression to predict the fault-proneness of software systems with a large number of zero response variables. He showed that zero-inflated Poisson regression is better than Poisson regression for software quality modeling.

Munson and Khoshgoftaar [36] also investigated the application of multivariate analysis to regression and showed that reducing the number of “independent” factors (attribute set) does not significantly affect the accuracy of software quality prediction. Menzies, Ammar, Nikora, and Stefano [37] compared Decision Trees, Naïve Bayes, and 1-rule classifier on the NASA software defect data. A clear trend was not observed and different predictors scored better on different data sets. However, their proposed ROCKY classifier outscored all the above predictor models.

Emam, Benlarbi, Goel, and Rai [38] compared different case-based reasoning classifiers and concluded that there is no added advantage in varying the combination of parameters (including varying nearest neighbor and using different weight functions) of the classifier to make the prediction accuracy better.

Salah Bouktif et al [39] presented how the general problem of combining quality experts, modeled as Bayesian classifiers, can be tackled via a simulated annealing algorithm customization. The general approach was applied to build an expert predicting object-oriented software stability, a facet of software quality. The findings demonstrate that, on available data, composed expert predictive accuracy outperforms the best available expert and it compares favorably with the expert build via a customized genetic algorithm.

Ping Guo et al. [40] showed Expectation-Maximum likelihood (EM) algorithm to build the quality model. By only
employing software size and complexity metrics, this technique can be used to develop a model for predicting software quality even without the prior knowledge of the number of faults in the modules. The technique was successful in classifying software into fault-prone and non-fault-prone modules with a relatively low error rate, providing a reliable indicator for software quality prediction.

III. CONCLUSION

Based on the survey, this necessitates the need to develop a real-time assessment technique that classifies these dynamically generated systems as being faulty/fault-free. A variety of software fault predictions techniques have been proposed, but none has proven to be consistently accurate. These techniques include statistical method, machine learning methods, parametric models and mixed algorithms. Therefore, there is still a need to find the best techniques for Quality prediction of the software systems by finding the fault proneness.

REFERENCES


