Software Defect Prediction Based on GUHA Data Mining Procedure and Multi-Objective Pareto Efficient Rule Selection

Bharavi Mishra, Department of Computer Engineering, Indian Institute of Technology (BHU), Varanasi, India
K.K. Shukla, Department of Computer Engineering, Indian Institute of Technology (BHU), Varanasi, India

ABSTRACT

Software defect prediction, if is effective, enables the developers to distribute their testing efforts efficiently and let them focus on defect prone modules. It would be very resource consuming to test all the modules while the defect lies in fraction of modules. Information about fault-proneness of classes and methods can be used to develop new strategies which can help mitigate the overall development cost and increase the customer satisfaction. Several machine learning strategies have been used in recent past to identify defective modules. These models are built using publicly available historical software defect data sets. Most of the proposed techniques are not able to deal with the class imbalance problem efficiently. Therefore, it is necessary to develop a prediction model which consists of small simple and comprehensible rules. Considering these facts, in this paper, the authors propose a novel defect prediction approach named GUHA based Classification Association Rule Mining algorithm (G-CARM) where “GUHA” stands for General Unary Hypothesis Automaton. G-CARM approach is primarily based on Classification Association Rule Mining, and deploys a two stage process involving attribute discretization, and rule generation using GUHA. GUHA is oldest yet very powerful method of pattern mining. The basic idea of GUHA procedure is to mine the interesting attribute patterns that indicate defect proneness. The new method has been compared against five other models reported in recent literature viz. Naive Bayes, Support Vector Machine, RIPPER, J48 and Nearest Neighbour classifier by using several measures, including AUC and probability of detection. The experimental results indicate that the prediction performance of G-CARM approach is better than other prediction approaches. The authors’ approach achieved 76% mean recall and 83% mean precision for defective modules and 93% mean recall and 83% mean precision for non-defective modules on CM1, KC1, KC2 and Eclipse data sets. Further defect rule generation process often generates a large number of rules which require considerable efforts while using these rules as a defect predictor; hence, a rule sub-set selection process is also proposed to select best set of rules according to the requirements. Evolution criteria for defect prediction like sensitivity, specificity, precision often compete against each other. It is therefore, important to use multi-objective optimization algorithms for selecting prediction rules. In this paper the authors report prediction rules that are Pareto efficient in the sense that no further improvements in the rule set is possible without sacrificing some performance criteria. Non-Dominated Sorting Genetic Algorithm has been used to find Pareto front and defect prediction rules.

Keywords: Data Mining, Defect Patterns, Fault Prediction, General Unary Hypothesis Automaton (GUHA), Pareto Optimality

DOI: 10.4018/ijssci.2014040101
1. INTRODUCTION

Nobody could possibly deny the omnipresence of software in the 21st century with pivotal role in areas of engineering, science and human activities. Complex systems are heavily dependent on their reliability. A single exploited software related problem such as bug, defect or failure can cause severe damage to an organization and can also bring down the customer satisfaction. Therefore, defects in code continue to be a major software quality problem. Due to which, quality assurance activity such as software testing become crucial fundamental task for software development. However, testing is a very expensive task and consumes considerable amount of software development effort.

To mitigate the testing costs, a good strategy is essential to distribute the testing efforts only to the defect prone modules. In addition, a guideline to design low defect prone modules is also required to ease the future development activities. These issues can be undertaken by building the defect prediction model using historical data sets which can be used in defect prediction for future projects.

Being able to detect the defective module can be a fruitful step towards the steering software testing, which can efficiently improve the whole process of development. It has long been recognized that software defect prediction is well known and proven technique to produce high quality software. Timely prediction of defective modules directly helps in quality assurance activity in cost effective manner. Early prediction of defects in software development life cycle can be used to improve the quality of process as well as products. Defect prediction approach mainly focuses to identify the defective modules to improve the quality of software. Availability of public data repository (Promise Data Repository, 2010) and public domain models (Weka, 2010) can ease the task of model generation and test. In literature variety of defect prediction techniques have been proposed for defect prediction which includes machine learning methods, parametric models and mixed approaches (Guo et al., 2004; Pai & Dugan, 2007; Elish & Elish, 2008; Khoshgoftaar et al., 1996; Azar et al., 2002; Selby & Porter, 1998; Emam et al., 2001; Zhan & Reformat, 2007). In the article by Thwin (2005), quality of the software products has been evaluated using Neural Networks (NN) and object-oriented metrics. Regression techniques have been used by Bibi et al. (2008) and Graves et al. (2000) on software change history in order to evaluate the defect proneness of software modules. A value based software testing framework is proposed by Du Zang (2011).

The clustering has been applied Hand et al (2001) on the data specified by the experts using the complexity metrics of software products to cluster the defective and non defective modules. Application of static code attribute in defect prediction is tested by Menzis et al (2007) using Naïve Bayes (Witten & Frank, 2005), Oner (Holte, 1993) and J48 (Quinlan, 1992) with lognum transformation and achieved 71% probability of detection and 25% false alarm rate. Based on their findings they concluded that static code attributes are good quality indicators and can be used in software defect prediction. Random Forest (RF), proposed by Guo et al. (2004), has been used in defect proneness prediction using the NASA MDP defect data sets and concluded that performance of RF is better than other statistical learning algorithms in terms of accuracy. Pai and Dugan (2007) used same data sets to explore the importance of Bayesian Network (BN) in software defect prediction. Support Vector Machine (SVM) has been applied by Elish & Elish (2008) on four NASA projects and achieved 90% mean recall for non-defective modules. Some other techniques such as discriminent analysis; Khoshgoftaar et al (1996), genetic algorithm (Azar et al., 2002), classification tree (Selby & Porter, 1998), case based reasoning(Emam et al., 2001) and Ant Colony Optimization (ACO) (Zhan & Reformat, 2007) are also used successfully in software fault-proneness prediction.
Relevant and Non-Redundant Amino Acid Sequence Selection for Protein Functional Site Identification
Chandra Das and Pradipta Maji (2010). International Journal of Software Science and Computational Intelligence (pp. 19-43).
www.igi-global.com/article/relevant-non-redundant-amino-acid/43896?camid=4v1a