Identification of Error Prone Classes for Fault Prediction Using Object Oriented Metrics

Puneet Mittal[1], Satwinder Singh[1], and K.S. Kahlon[2]

[1] BBSBEC, Fatehgarh Sahib, Punjab, India
[2] GNDU, Amritsar, Punjab, India

Abstract. Various studies have found that software metrics can predict class error proneness. However their study is focused on the relationship between class error proneness and software metrics during the development phase of software projects not in system's post-release evolution. This study is focused on the three releases of Javassist- open source java based software. This paper describes how we calculated the object-oriented metrics to illustrate error-proneness detection. Using Findbugs we collected errors in the post-release system and applied logistic regression to find that some metrics can predict the class error proneness in post release evolution of system. We also calculated model's accuracy by applying one model on other version's data.

Keywords: Object oriented metrics, Class error-proneness, Javassist, Findbugs, JColumbus, Together tool.

1 Introduction

It is the dream of every developer to develop error-free software. But inspite of undergoing software testing, walkthroughs and inspections; post maintenance of software is required. Few errors still remain in software that can make the life of software developer hell. It is very difficult to do changes after the software has been released. But if we know the probable areas where error can be located, the problem can be solved and the software testers can be helped to locate the errors easily and effectively. Software metrics are one way to measure the software and can be used for locating the errors. One goal of software metrics is to identify and ensure the essential parameters that affect software development. Software metrics provide quantitative basis for development and validation of models of software development process. Metrics can be used to improve software productivity and quality.

In this paper, we describe how we calculated the object-oriented metrics for error-proneness detection from source code of open source software, Javassist[12]. We employed statistical methods (i.e. logistic regression) for predicting error proneness of code.

A. Abraham et al. (Eds.): ACC 2011, Part II, CCIS 191, pp. 58–68, 2011.
© Springer-Verlag Berlin Heidelberg 2011

2 Literature Survey

Various software metrics have been proposed by various researchers in different paradigms. Various studies have sought to analyze the connection between object-oriented metrics and code quality ([16], [9], [2]).

Chidamber et al. [5] developed and implemented a new set of software metrics for OO designs. These metrics were based on measurement theory and also reflect the viewpoints of experienced OO software developers. He gave set of six OO metrics (WMC, DIT, RFC, LCOM, NOC, and CBO) where WMC, NOC and DIT metrics reflect class hierarchy; CBO and RFC metric reflect class coupling and LCOM reflects cohesion. **Basili et al. [2]** collected data about faults found in object oriented classes. Based on these data, they verified how much fault-proneness is influenced by internal (e.g., size, cohesion) and external (e.g., coupling) design characteristics of OO classes. From their results, five out of the six CK OO metrics appear to be useful to predict class fault-proneness during the high- and low-level design phases of the life-cycle. **Chidamber et al. [6]** investigated the relationship between the CK metrics and various quality factors: software productivity, rework effort, and design effort. The study also showed that the WMC, RFC, CBO metrics were highly correlated. Therefore, **Chidamber et al**. did not include these three variables in the regression analysis to avoid generating coefficient estimates that would be difficult to interpret. The study concluded that there were associations between the high CBO metric value and lower productivity, more rework, and greater design effort. In another study, **Wilkie and Kitchenham [18]** validated the relationship between the CBO metric and change ripple effect in a commercial multimedia conferencing system. The study showed that the CBO metric identified the most change-prone classes, but not the classes that were most exposed to change ripple effect. **Cartwright and Shepperd [4]** also investigated the relationship between a subset of CK metrics in a real-time system. The study showed that the parts of the system that used inheritance were three times more error prone than the parts that did not use inheritance. **Subramanyam and Krishnan [16]** validated the WMC, CBO, and DIT metrics as predictors of the error counts in a class in a business-to-consumer commerce system. Their results indicated that the CK metrics can predict error counts. They examined the effect of the size along with the WMC, CBO, and DIT values on the faults by using multivariate regression analysis. They concluded that the size was a good predictor in both languages, but WMC and CBO could be validated only for C++. **Alshayeb and Li [1]** conducted a study on the relationship between some OO metrics and the changes in the source code in two client–server systems and three Java Development Kit (JDK) releases. Three of the CK metrics (WMC, DIT, and LCOM) and three of the Li metrics (NLM, CTA, and CTM) were validated. They found that the OO metrics were effective to predict design effort and source lines of code added, changed, and deleted in short-cycled agile process (client–server systems); however, the metrics were ineffective predictors of those variables in long-cycled framework evolution process. **Olague et al. [14]** indicated that the CK and QMOOD OO class