Chapter 78
Investigating the Effect of Sensitivity and Severity Analysis on Fault Proneness in Open Source Software

D. Jeya Mala
Department of Computer Applications, Thiagarajar College of Engineering, Madurai, India

ABSTRACT

Fault prone components in open source software leads to huge loss and inadvertent effects if not properly identified and rigorously tested. Most of the reported studies in the literature have applied design metrics alone, to identify such critical components. But in reality, some of the components' criticality level can be identified only by means of dynamic code analysis; as some of the components seem to be normal but still have higher level of impact on the other components. This leads to an insight on the need of a rigorous analysis based on how sensitive a component is and how severe will be the impact of it on other components in the system. To achieve this, an efficient mechanism of evaluating the criticality index of each component by means of sensitivity and severity analysis using the static design metrics and dynamic source code metrics has been proposed. Then, testing is conducted rigorously on these components using both unit testing and pair-wise integration testing.

INTRODUCTION

Recent studies have indicated that, most of the faults in the software are due to a few components in the overall software (El-Emam et al., 2001, Janes et.al. 2006, Mathur A.P. 2008, Nagappan et.al. 2006). If these components are identified prior to testing, they can be rigorously tested by optimally allocating the resources needed for testing (Garousi et al., 2006).

In the case of Object Oriented Systems, design metrics and measures play a crucial role in predicting the critical components from the models. El-Emam et al. (2001) have conducted a case studies based analysis and concluded that; a component's criticality level cannot be predicted only based on the design

DOI: 10.4018/978-1-7998-3016-0.ch078

metrics but also by means of prototypes and metrics of development process. Hence, to analyze the components criticality level, one has to analyze the components sensitivity and severity not only based on design metrics but also on dynamic code metrics.

The application of Object Oriented (OO) metrics have been used by several researchers in the past (Abreu, (1994), Benlarbi and Melo, (1999), Briand et.al. (2000), Cartwright and Shepperd (2000), Khoshgoftaar et al. (2002), Shin et.al. (2011)) in constructing prediction models. However, they have used the design oriented metrics only for the prediction model. From the literature, it has been observed that, empirical validations of applying OO metrics to open source software have been done extensively (Gyimothy, Ferenc & Siket, 2005).

The major observations derived as part of the literature survey showed the problems associated with the existing approaches. Some of them are: proposal of a fault prediction model using design metrics alone; evaluation based on basic design and code metrics only; the application of risk and reusability based analysis in fault-prone components identification and lack of real time validation of the proposed approach using fault injection based impact analysis.

Also, it has been observed that, only because a component has more Lines of Code (LOC), No. of Attributes (NOA), No. of Methods (NOM), Cohesion between Methods (CBM), No. of Static Fields (NOSF), No. of Static Methods (NOSM) and No. of Classes (NOCL), one cannot conclude that the component has high probability of fault-proneness. But it should be noted that, at times a very small component with very less functionality decides the entire product's functionality due to its impact over the other dependent components.

Many of the existing works have applied some of the design metrics such as Coupling between Objects (CBO), Depth of Inheritance Tree (DIT), No. of Children (NOC), and Lack of Cohesion between Methods (LCOM), Class Coupling (CC) and Measure of Aggregation (MOA) to identify the fault prone components. Based on our statistical analysis, it has been identified that, DIT metric cannot be used to find the impact of a base class over the derived classes as it will not reveal the level of reusability. Similarly, as the LCOM metric provides the inverse effect on the complexity of a component, it cannot be used to predict the fault-proneness of a component. Also, it has been observed that, some of the impact analysis based derived metrics from basic OO metrics such as CBO, NOC and CC can be used as potential indicators of fault-prone components (Ruchika and Ankita, 2012).

Concerning the above, the objective of this research work is to propose fault-prone components identification and testing framework to address the limitations in the existing approaches. The focus is now on identification of other types of analysis with various other important metrics to solve the said problem effectively.

In the light of the above consideration, this research work proposes a novel framework that performs an impact analysis on the components, based on their sensitivity and severity by taking into consideration the metrics related not only to design but also the static and dynamic code metrics to identify potential fault-prone components. Based on this impact analysis, a list of fault-prone components is generated and is validated against the list of potential fault-prone components identified using fault injection based impact analysis.

Hence, the proposed framework can be used as a quality model to ensure that all the components have been covered or tested at least once with rigorous testing being applied to critical or faulty components. The results of the empirical validation indicate that the proposed work has high accuracy in identifying the fault prone components and can be used as a prediction model to predict the quality of the software prior to the release of the software. As the outcome of this research work, a prototype tool