CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

In the process of software development, varied techniques like structure-oriented, procedure-oriented, and object-oriented paradigms are implemented to achieve simplicity, modularity, efficiency, integrity of data, and reduction of complexity. The software development can be more efficient and less complex only when the user-defined programs are combined into packages of predefined functions. So the knowledge of the packages of the predefined functions is necessary in order to implement these functions in the software development. A developer needs to learn thoroughly about the packages, classes and methods which can be used in the software development. Software industry is migrating towards open source development environment as it is cost efficient, secured and also provides the source code. On the contrary to commercial software's, open source system provides many inbuilt functions so as to develop an application effectively with less complexity under free open source license agreement.

1.2 PROBLEM DEFINITION

The developers in the open source environment do not strictly adhere to software development standards as any developer can contribute to open source environment. So there is a possibility of intrusion of less efficient and error prone software in open source environment. This requires a mechanism to detect the quality and complexity of open source software. Although there are numerous metrics in the research arena, there are no proper guidelines as to which type of metrics can be applied to the open source environment to detect the complexity of open source software.

Manual work involved in measuring the complexities and error proneness of any open source software is time consuming and tedious. This necessitates an automated tool to minimize the drawbacks.

1.3 OBJECTIVES

The objectives of this research work are as follows:

- To validate the available metric suites in the open source environment.
- To provide guidelines for the appropriate selection of metric suite for computing the complexities of open source softwares.
- To develop an automated mechanism for gathering metric values and error proneness.

1.4 MOTIVATION FOR THE RESEARCH WORK

The following are the research works of the past that stimulated the researcher to delve into the current research.

The validation of the metric, as to whether it supports the attributes which it tries to measure, became intrinsic in the use of metrics to measure any software Hector et al (2007). The metrics are needed to be validated to discover its functionality and support of the theory of attributes in the open source environment.

Jie et al (2008) have validated object-oriented design metrics for defects estimation utilizing empirical analysis. The Chidamber and Kemerer metrics suite were utilized to assess the amount of defects in the programs. The approach involves statistical analysis and neuro-fuzzy techniques. The results pointed out that reliable defect estimation can be achieved by Source Lines of Code (SLOC), Weighted Methods per Class (WMC), Coupling between Object Classes (CBO) and Response for a Class (RFC) metrics. Particularly SLOC achieved the most prominent effect on the number of defects Pavinder and Hardeep (2005).

Ramanath et al (2003) have given experimental proof that a subset of the Chidamber and Kemerer suite which are object oriented design complexity metrics plays an important role in identifying software defects. Experimental results on industry data belonging to software developed in two prevalent object oriented development programming languages showed that the metrics were considerably connected with defects even after controlling the software size. Moreover, effects of the metrics on defects varied over different samples from the two programming languages. Important inferences for designing high-quality object oriented software were provided by these results (Henderson 1996) and Briand et al (2001).

Tibor et al (2005) have demonstrated a procedure for detecting the fault-proneness of the source code of Mozilla which is an open source web and e-mail suite, by illustrating the computing procedure of Chidamber and Kemerer object-oriented metrics. The usefulness of the metrics for fault-proneness prediction was verified by utilizing regression and machine learning methods to compare the obtained values with the amount of bugs present in its bug database known as Bugzilla by Tibor Gyimothy et al (2005). The variation in the predicted fault-proneness during the development cycle of the software system was identified by comparing the metrics of different versions of Mozilla.

1.5 METRIC SUITES VALIDATION USING FAULT PRONENESS

Object-oriented design has emerged as a dominant method in software industry and many new metrics have been proposed for quality prediction of object-oriented programs, but the significance of these metrics is not yet confirmed. Software process control can be improved and high software reliability can be achieved if faults are predicted early in the software life cycle. Testing quality related issues of software has become critical with the increasing importance of the quality of software. Many authors have suggested theoretical validation followed by empirical evaluation using proven statistical and experimental techniques for evaluating in the field of usefulness and relevance of any new metrics.

In this research work an empirical validation of software quality metric suites on open source software for fault-proneness prediction in object oriented system has been presented. The three metrics used here are Chidamber and Kemerer (CK) metrics, Robert C. Martin metric suite and McCabe’s metric suite. Using these metrics suite, the defects present in different versions of Rhino software have been analyzed to predict the software quality by making use of the fault proneness concept. From the results and empirical analysis, it is clear that the different metric suites have different efficiency in faults prediction. With the aid of this empirical analysis, it is suggested that the software professionals need to find out those metric suites that can predict faults while developing the metric-quality software products using the object oriented approach IEEE (1990).

Chidamber and Kemerer metrics, Robert C. Martin Metric Suite and McCabe's Metric Suite are the metrics that are utilized in this research work. Here, the ability of these three metrics suites to predict the quality for different versions of Rhino is analyzed with an open-source implementation of JavaScript written in Java (Kemerer 1987).The selected software metrics for the object-oriented systems are empirically analyzed. In our research, three OO metrics that suites their capability to predict software quality with the help of the metric measures is studied.

A typical empirical validation of object-oriented metrics is presented by investigating the relationship between each metric and the outcome of interest. The selected three OO metrics suites for their ability to predict software quality is empirically validated. The Rhino software has been analyzed to predict the software quality with different versions in terms of fault proneness, with these metrics suites (Britoe 1994). The empirical analysis aids software professionals to find out these metric suites that can predict faults while developing the metric-quality software products using the OO approach.

1.6 ADAPTIVE FAULT TOLERANCE IN OPEN SOURCE SOFTWARE

This research work implements an effective methodology in order to make the open source software useful by means of an automated fault tolerant technique to deduct and report the error found in the software, avoiding manual effort.

In this research work, Java-based open source software is examined. In this kind of software, the downloaded software may be a Java ARchieve (JAR) file, or a class program or it may be of some packages.

After downloading the software, the proposed method, processes for fault tolerance.

An algorithm has been proposed to explain the process carried out while compiling the open source software. The proposed work simplifies the user's requirements by making use of the automated software to tolerate the fault. This research work is implemented on Java-based open source Software Hanny et al (2007). In future, it may be implemented on all kinds of Software by adding little more features of rectifying the possible errors automatically which will reduce the user's work.

1.7 DEDUCING AND MAPPING THE CLASS PATH OF JAR FILE

In the research work, a methodology is proposed to compile the package automatically and to set up the proper class path for implementing the methods in JAR files.

The summary of the work is as follows: The user first selects the software needed to develop the program. After that, the proposed methodology has to compile the software to identify the errors. From the deduced errors, the error rate is to be found out.

The aim of the research work is to develop an automated search engine to deduce the errors occurring in the JAR file mapping and to correct the mapping automatically by detecting the correct path of the JAR file.

1.8 SCOPE OF THE RESEARCH WORK

The aim of the thesis is to validate the software empirically in an object oriented system. In order to use open source software, the reliability and efficiency of the software and error-proneness are checked. If the software contains error, the error rate is calculated. The software is evaluated as useful or not based on error rate.

If the software is deducted to be error-free, then the mapping path is identified. If the mapping path of the JAR file is not correct, then the execution of the software will not be proper. The user may not have the ability to identify the JAR file and to map it to the path. Hence, the JAR file is identified automatically and the path is mapped properly.

1.9 ORGANIZATION OF THE THESIS WORK

The research has been organized as follows: An overview of the software metric suites is discussed in chapter one. In chapter two basic definitions and concepts are discussed. In chapter three is devoted to literature survey. Following these chapters, in chapter four, Reliability framework OSS is described.In chapter five metric suites validation using fault proneness is discussed. In chapter six adaptive fault tolerance mechanism in open source software is elaborately presented and in chapter seven the methodology for mapping the class path in JAR files is elaborated. The last chapter presents the conclusion of the work and the future research directions.