Available online at www.sciencedirect.com

ScienceDirect

Procedia Computer Science 82 (2016) 107 – 114

Procedia Computer Science

Symposium on Data Mining Applications, SDMA2016, 30 March 2016, Riyadh, Saudi Arabia

Defectiveness Evolution in Open Source Software Systems

Yasir Javed[a,b]*, Mamdouh Alenezi[a]

[a].Prince Sultan University, Riyadh,KSA
[b]FIT, UNIMAS, SARAWAK, Malaysia

Abstract

One of the essential objectives of the software engineering is to develop techniques and tools for high-quality software solutions that are stable and maintainable. Software managers and developers use several measures to measure and improve the quality of a software solution throughout the development process. These measures assess the quality of different software attributes, such as product size, cohesion, coupling, and complexity. Researchers and practitioners use software metrics to understand and improve software solutions and the processes used to develop them. Determining the relationship between software metrics aids in clarifying practical issues with regard to the relationship between the quality of internal and external software attributes. We conducted an empirical study on two open source systems (JEDIT and ANT) to study the defectiveness Evolution in Open Source Software Systems. The result reveals that a good designed software has lesser defects and have high cohesion. Moreover the study also revealed that defects are higher in initial versions and most corrected errors are from major classes in initial version. Removal of defects also reveals that a good software is consistently improved and feed backs are important part of open source systems.

Peer-review under responsibility of the Organizing Committee of SDMA2016
Keywords: Software Evolution; cohesion ; defectiveness evolution

1. Introduction

Software systems have become an essential component of any critical infrastructure. Software systems usually evolve constantly, which requires constant development and maintenance. As software evolves, its design and code qualities determine how costly is to develop and maintain that software. Software evolution is the vigorous activities

* Yasir Javed. Tel.: +966 11 494 8287; fax: +966 11 494 8317.
E-mail address: yjaved@psu.edu.sa

Peer-review under responsibility of the Organizing Committee of SDMA2016
doi:10.1016/j.procs.2016.04.015

of software systems while they are improved and maintained over their lifespan[10]. Software systems change and evolve throughout their life cycle to accommodate new features and to improve their quality. Software needs to evolve in order to survive for a lengthy period. The changes that software undergo lie within corrective, preventive, adaptive and perfective maintenance that lead to software evolution.

The availability of open source systems data allows us to explore different kinds of relationships. Internal characteristics and external characteristics can be investigated using several data mining techniques. The resulted insights will shed light on different decisions and endeavors while the system is being evolved over time. One of the main goals of software engineering research is to provide evidence to support and facilitate in making correct decisions during the development of the software[5]. Reaching these decisions always depends on how the data are analyzed and which information is extracted from the data during the analysis.

To understand how software quality changes as software evolve, we use both internal and external quality metrics as used by Neamtiu et al[3]. The attributes of software quality can be categorized into two main types: internal and external. Internal quality attributes can be measured using only the knowledge of the software artifacts, such as the source code, whereas the measurement of external quality attributes requires the knowledge of other factors, such as testability and maintainability. The attributes of software quality, such as defect density and failure rate, are external measures of the software product and its development process. External quality means how users' are perceiving and accepting the software. To quantify this, we use the defect density. Internal quality metrics assess internal quality, coupling, cohesion, and complexity. The more complex the software the more difficult to is to change/extend[3].

Software metrics[4] are measures utilized to evaluate the process or product quality. These metrics helps project managers to know about the progress of software and assess the quality of the various artifacts produced during development. The software analysts can check whether the requirements are verifiable or not. Software metrics are required to capture various software attributes at different phases of the software development. Software metrics can be utilized to adequately measure various phases of the software development life cycle. Software product metrics represent several aspects of the source code. They reveal a lot of design and complexity problems in the source code. Several empirical studies have established the notion that certain source code characteristics like code size, complexity, and coupling, programming language, and programming style hugely influence software maintenance efforts, costs, and effectiveness[1]. These characteristics include code size, complexity, and coupling, programming language, and programming style[2].

We conducted an empirical study on two open source systems Jedit and Ant. A theory of software evolution must be based on empirical results, verifiable and repeatable by the constant development at each phase whereas thinking (feedback) must be included from testers or end users[11].

2. Datasets of the Investigated Systems

We ran our empirical study on two open-source applications written in Java. We used several criteria to select the systems: 1) well-known systems that are used very widely; 2) sizable systems that yield realistic data; 2) actively maintained systems; 4) systems with publically available data, which is crucial in empirical studies. Apache Ant is a Java library and command-line tool whose mission is to drive processes described in build files as targets and extension points dependent upon each other. The main known usage of Ant is the build of Java applications. Ant supplies a number of built-in tasks allowing to compile, assemble, test and run Java applications. jEdit is a mature programmer's text editor supported by hundreds (including the time-developing plugins) of person-years of development. It is written in Java and runs on any operating system that supports Java, including Windows, Linux, Mac OS X, and BSD. The POI project consists of APIs that are used to manipulate various file formats based on Microsoft's OLE 2 Compound Document format, and the Office OpenXML format, which uses pure Java. Table 1 shows some characteristics about the dataset.

Table 1. Characteristics about the dataset

System	Version	LOC	Defect Density	# of Classes
ANT	1.3	37699	0.000749411	125
	1.4	54195	0.009223589	178
	1.5	87047	0.000519024	293
	1.6	113246	0.001395763	351