Evaluating Maintainability of Open Source Software:
A Case Study

Feras Hanandeh, Hashemite University, Jordan
Ahmad A. Saifan, Department of Computer Information Systems, Yarmouk University, Jordan
Mohammed Akour, Department of Computer Information Systems, Yarmouk University, Jordan
Noor Khamis Al-Hussein, Department of Computer Information Systems, Yarmouk University, Jordan
Khadijah Zayed Shatnawi, Department of Computer Information Systems, Yarmouk University, Jordan

ABSTRACT
Maintainability is one of the most important quality attribute that affect the quality of software. There are four factors that affect the maintainability of software which are: analyzability, changeability, stability, and testability. Open source software (OSS) developed by collaborative work done by volunteers through around the world with different management styles. Open source code is updated and modified all the time from the first release. Therefore, there is a need to measure the quality and specifically the maintainability of such code. This paper discusses the maintainability for the three domains of the open source software. The domains are: education, business and game. Moreover, to observe the most effective metrics that directly affects the maintainability of software. Analysis of the results demonstrates that OSS in the education domain is the most maintainable code and cl_stat (number of executable statements) metric has the highest degree of influence on the calculation of maintenance in all three domains.

KEYWORDS
Chat System, Maintainability, Open Source Software, OSS

INTRODUCTION
Software maintenance is a primary phase of the software development life cycle. Several studies have reported that this phase is the most effort and time consuming of all. For example, in one study it was reported that the software maintenance phase takes around seventy percent of total resources and 40%-60% of the total software lifecycle efforts (Tahir & Ahmad, 2010). Having maintainable software decreases maintenance cost and effort. Software maintainability can be defined as “the degree to which an application is understood, repaired or enhanced” (Cast, 2017).

Four factors affect the maintainability of software: (1) analyzability, which measures the ability to identify the fault or failure within the software, (2) changeability which is the capability to modify software products, (3) stability which refers to the ability to avoid unexpected effects from changing
the software product, and (4) testability which is the capability to test and validate the modified software product (Ghosh & Dubey, 2012).

A number of software metrics can be used to measure maintainability. Software metrics is a predictor to assess and predict software quality; however, although maintenance is a critical task, it has been poorly managed due to inadequate measurement, so we need precise criteria for measuring software maintenance (Rizvi & Khan, 2010). Finding a tool that provides accurate and relevant results is not easy and it is a big issue. Moreover, some tools are dedicated to special development languages and other tools have an interface that requires extensive training (Albeladi et al., 2014). Indeed, some tools are able to measure specific metrics, and they vary according to the different sizes of systems they evaluate (Lincke et al., 2008).

Many researchers depend on open source software, because it is free in terms of licensing, especially in industrial organizations. Moreover, it is constantly being updated by different developers. Therefore, there is a need to measure the quality of such codes and to ensure the maintainability of open source software (Johari & Kaur, 2012).

Open source software (OSS) is developed by collaborative work done by volunteers worldwide using different management styles. Issues with OSS’s include: lack of attention to user interface design that will cause it to be used less; lack of documentation which is a serious issue, with many OSS’s being poorly documented since there is no contractual responsibility (Levesque, 2004). An OSS is sometimes more secure than proprietary software, but there are some drawbacks in its security and many E.Y.E reviews have led to vulnerabilities being discovered (Open Source, 2017). There are also some usability problems in OSS’s but they are not significant (Nichols & Twidale, 2003). So, the quality of open source software needs to be studied deeply and maintainability is one of the major factors used to assess their quality.

Empirical studies refer to OSS’s as being more maintainable than proprietary software. Design metrics literature indicates that after a long period of maintenance an OSS will be harder to maintain and a deterioration in maintenance is to be expected over time (Ayalew & Mguni, 2013).

Many studies have been conducted on evaluating the maintainability of OSS. Some research considers whether the maintainability of OSS’s is different from that of closed software. On the other hand, other research studies show the variance of maintainability measurement with different versions for the same OSS, and some examines the impact of design metrics on maintainability.

In this paper, we will answer the following questions: Do the domains of the OSS have a direct effect on its maintainability? What are the most available metrics that directly affect the maintainability of software? What tool can we trust to provide accurate measurement to maintainability?

**BACKGROUND**

**Software Metrics and Tools**

In this paper, the Chidamber and Kemerer (C&K) of object oriented metrics have been used. These metrics are as described in Gulia and Chhillar (2011):

- **WMC (weighted method per class):** “Total complexity for all class methods”;
- **DIT (depth of inheritance tree):** “Defines maximum length from node to root from tree”;
- **NOC (number of children):** “Number of immediate subclasses”;  
- **CBO (coupling between objects):** “Two classes are coupled when method declared in one class uses method or variables instance of other class”;
- **RFC (response for class):** “Number of methods that can be invoked in response to a message sent to an object of class”;
- **LOCM (lack of cohesion method):** “Number of different methods within class that reference given instance variable”.


Strategy of Good Software Governance: FLOSS in the State of Turkey
www.igi-global.com/chapter/strategy-of-good-software-governance/197111?camid=4v1a

Transmission Line Routing Using Open Source Software Q-GIS
www.igi-global.com/article/transmission-line-routing-using-open-source-software-q-gis/203649?camid=4v1a