**A METRIC BASED APPROACH FOR EVALUATING INTERFACE COMPLEXITY OF COMMON OPEN SOURCE SOFTWARE (OSS)**

**CHAPTER ONE**

**INTRODUCTION**

**1.1 BACKGROUND OF STUDY**

Open-source software (OSS) is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose. (Laurent, 2008) This is in contrast with propriety software which cannot be inspected and modified by anyone. In the last ten years, open source software (OSS) has attracted the attention of not only the practitioner, but also the business and the research communities. OSS has proven to produce software of high quality, functionality and wide development. Open-source software may be developed in a collaborative public manner. Open-source software is a prominent example of open collaboration In other words, an open-source software (OSS) is software that is distributed with source code that may be read or modified by users. OSS has become very popular and there are OSS applications for a variety of different uses such as office automation, web design, content management, operating systems, and communications. Some of the most popular software packages such as Mozilla Firefox, the Linux Operating System and Apache web server software, are examples of OSS. The key difference between OSS and proprietary software is its license. As copyright material, software is almost always licensed. The license indicates how the software may be used. The OSS community generally agrees that open-source software should meet the following criteria:

1. The program must be freely distributed
2. Source code must be included with the program
3. Anyone must be able to modify the source code
4. Modified versions of the source code may be redistributed

(Levine, 2013).

As well, an open-source software license must not require the exclusion of, or interfere with, the operation of other software. Unlike traditional software distributed in an unchangeable compiled format, open-source software is delivered with both compiled and non-compiled formats, allowing open code modification. In traditional software licenses, this privilege would be reserved for copyright holders. Not all software developers favor the use of open-source software, but many have accepted it because it allows for quicker repair of software issues and may ultimately lead to higher quality applications. When considering the advantages of open source software you should consider the open source product itself. Open source products vary in quality.

The software evaluation is performed by the different members of IT team. It is important to measure the efficiency of the software. Test efficiency is to measure the effectiveness and success of the evaluation process. Metrics are used for measurement, comparison or to track performance or production. Metrics helps to compare the performance of software at various levels. Metrics helps in good decision making or improvements in the project. Metrics should be accurate, timely and actionable. Metrics are derived from the earlier data. These must be understandable, economical and must be useful at the various levels of the development of the project.

(Pinker et al, 2009).

Software metrics is a standard of measure that contains many activities which involve some degree of measurement. In other words, A software metric is a measure of software characteristics which are measurable or countable. Software metrics are valuable for many reasons, including measuring software performance, planning work items, measuring productivity, and many other uses.

Within the software development process, many metrics are all connected. Software metrics are similar to the four functions of management: Planning, Organization, Control, or Improvement. Examples of metric for open system software.

1. Total Number of Contributions
2. Average Domain Experience of Contributors
3. Average Time for a Completion of a version of the software.
4. Bugs Track per Version
5. Patch Accept Ratio
6. Total Number of Weekly Downloads

All these factors can be considered to determine the performance of a open system software.

Controlling and minimizing software complexity is one of the most important objective of each software development paradigm because it affects all other software quality attributes like reusability, reliability, testability, maintainability etc Theinterface of any software is a important part of a software, For software components, one of the major components of a software is the interface. Interface complexity measures are the estimates of the complexity of interfaces. Interface defines provided services of a component and acts as a basis for its use and implementation. It acts as one of the major definitive source for component understanding and may be the only available source. An interface consists of a set of operations, which act as access points for interaction with the outside computing environment. Integration metrics are the measures of efforts required in the integration process of components and semantic measures estimate the complexity of relationship of components to application. (Usha et al, 2011)

Software interfaces (programming interfaces) are the languages, codes and messages that programs use to communicate with each other and to the hardware. Examples are the Windows, Mac and Linux operating systems, SMTP email, IP network protocols and the software drivers that activate the peripheral devices. A software interface allow you to access certain functionality in a system or a library without caring to the way it is implemented on the system or library side while also ensuring that if someone updates the system or the library the interface will ensure that your application will still work the same way even if there is a totally different specification on the other side.

**1.2 STATEMENT OF PROBLEM**

Recent studies have largely investigated the detection of some design anomalies. They proposed a large set of metrics that help in detecting those anomalies and in predicting the quality of software design. While those studies and the proposed metrics are valuable, they do not address the particularities of open source software interfaces complexity. Interfaces define the contracts that spell out how software modules and logic units interact with each other and the user of the software. Interface complexity measures are the estimates of the complexity of interfaces (Shuchita, 2011). In this work, we would propose a metric based approach to evaluate OSS interface complexity. We identify and describe those design defects.

**1.3 MOTIVATION OF STUDY**

Controlling and minimizing open source software (OSS) complexity is one of the most important objective of each software development paradigm because it affects all other software quality attributes like reusability, reliability, testability, maintainability etc. For this purpose, a number of OSS complexity measures have been reported to quantify different aspects of complexity. As the development of component-based software is rising, more and more complexity metrics are being developed for the same purpose. (Shuchita, 2011).

Open source software complexity has been an important research topic for several decades and the increasing pervasiveness of software systems suggests that the topic will remain relevant in the future. This line of work has traditionally focused on assessing the complexity of a particular unit of software such as functions, modules or components and examining its impact in the context of software maintenance activities, interfaces of software and software quality. Numerous code complexity measures have been proposed however, empirical examinations of the relationship between complexity and quality have produced disappointing results. Having understood the approach behind metric-based techniques, its is know that the evaluation of the interface of OSS can be evaluated.

**1.4 AIM AND OBJECTIVES**

**1.4.1 Aim**

The aim of this work is to use a metric based approach for evaluating interface complexity of common open source software (OSS)

**1.4.2 Objectives**

To be able to get to the aim of this project work. We would consider some following objectives.

1. Identify the key features (Metrics) that determines a good OSS Interface.
2. Evaluate the software with the metric identified.

**1.5 METHODOLOGY**

To achieve the aim of this work, we would consider the use of metric base approach to solve the problem in question.

1. Review of related work so as to get the back ground knowledge of OSS and Interfaces
2. Identify the metrics to be used for the evaluation of the software interface.
3. Evaluation of the system.

**1.6 SCOPE OF STUDY**

This work concentrates on open source software alone and interface among its components.

**1.7 DEFINITION OF TERMS**

1. **Software:** is a set of instructions, data or programs used to operate computers and execute specific tasks.
2. **Open System Software:** Open-source software is a type of computer software in which source code is released under a license in which the copyright holder grants users the rights to study, change, and distribute the software to anyone and for any purpose.
3. **Interface:** In computing, an **interface** is a shared boundary across which two or more separate components of a computer system exchange information. The exchange can be between software, computer hardware, peripheral devices, humans, and combinations of these.
4. **Hardware Interface:** An architecture used to interconnect two devices together. It includes the design of the plug and socket, the type, number and purpose of the wires and the electrical signals that are passed across them.
5. **Software Interface:** A **software interface** is used to allow either two pieces of **software** to communicate with each other (**software**-**software interface**), or to allow **software** to communicate with a hardware device (**software**-hardware **interface**).