**Introduction/ Literature review of relevant research**

1. **Open Source Software**

Open Source Software (OSS) is relatively a new and alternative idea of software development. The term “open source software” was first coined in 1998 at a meeting of the OSS movement pioneers at Palo Alto [1]. A software product is OSS when its distribution license fulfills the “freedoms” of OSS. To be more precisely, a software can be said that it is OSS when the receiver of the software can:

• use it at wish

• copy and redistribute it

• modify it and distribute the modified versions

The majority of OSS projects use development practices, models and methods which very far away from those recommended by the “classical” software engineering, for example software development life-cycle models like waterfall and the spiral. The main idea behind the development model of OSS is simple. A single person or a group of people have an idea about a software product or want particular software to address a particular need, so they write a first release of that software to satisfy their needs, their “personal itch” [11]. They put that software on-line, along with its source code, and ask for other people to contribute sending either bug fixes or functional improvement for their software.

Over the past decade, open-source development processes [11] have emerged as an effective approach to reduce cycle-time and decrease design, implementation, and quality assurance (QA) costs for certain types of software, particularly infrastructure software, such as operating systems, compilers and language processing tools, editors, and distribution middleware [3]. Widely used examples of successful open-source infrastructure software include:

• UNIX/POSIX operating systems,

• Web servers, e.g., Apache and JAWS

• Compilers, e.g., GNU C/C++

• Editors, e.g., GNU emacs

**2. Quality of Software**

As any other product, for example, a building or a car, has a level of quality, software products have some kind of quality as well. By quality we mean whether or not the product conforms to a set of standards posed by someone, either by the manufacturer or by the customer. For example: Does the software do what the user wants it to do? Or is it well-designed, well coded, well tested, error free, so that it will function properly? [6].

In order to assess software quality, it is usual to construct models which combine different aspects of quality to produce a final result about the level of a software product quality. There exist several models of software quality, which suggest various ways to bring together different quality attributes. Each one of these models tries to aggregate these several attributes of quality in order to give an overall view of the quality of software. One of such models is the ISO 9126 model [5].

The ISO 9126 quality model was suggested in 1991 and is a hierarchical model consisting of six major attributes contributing to software quality. The attributes and what they represent are:

• Functionality

• Reliability

• Usability

• Maintainability

• Portability

• Efficiency

As with other models, these attributes do not necessarily lead to direct measurement, but to other indirect measurements, which can be further divided into other quality factors. Some of the above attributes, which can be measured using direct measurement, are usually expressed as some kind of equation, which has as inputs various source code metrics such as the McCabe's cyclomatic complexity [9] or Halstead Effort Metrics [4]. An example of such equation is the Maintainability Index proposed by Carnegie Mellon's Software Engineering Institute. An extensive discussion and reference to software metrics can be found in [2].

**3. Problem to be solved**

# There are a lot of researches on the quality of traditionally developed software or Closed Source Software (CSS). OSS are said to be possibly of comparable quality with CSS. However, there are very few studies investigating common software quality factors for various types of Open Source projects [12]. Open Source Projects have their own characteristics which might affect their quality:

a) Different levels of programmers. Open Source Project is joined by different levels of people who are interested in the open source project. In Closed Source Project, programmers are well organized in one organization.

b) Frequent beta releases. A cornerstone of open-source is short feedback loops between users and core developers, which typically result in frequent “beta” releases, e.g., several times a month. Although this schedule satisfies the end-users who receive quick patches for bugs they found in earlier betas, it can be frustrating to other end-users who want more stable, less frequent software releases.

c) Platforms-independence. Another cornerstone of open-source software is its platform-independence, which stems from its roots in the open systems—rather than proprietary systems—community. Support for platform-independence, however, can yield the task of keeping an open-source source software base operational despite continuous changes to the underlying operating system and compiler platforms.

d) Support for many compile-time and run-time configurations. The availability of the software in open-source projects encourages core developers to increase the number of options for configuring and subsetting the software at compile-time and run-time. Although this flexibility enhances the software’s applicability for a broad range of use-cases, it can also greatly magnify QA costs due to the combinatorial number of code paths that must be regression tested.

As the OSS paradigm makes progress within these organizations any potential software procurer is tasked with some important questions which, currently, cannot be answered with any real assurance:

* Many OSS projects are very similar. How do we choose between them? Which is the most appropriate system for the company’s IT infrastructure?
* How can we distinguish the “good” and “bad” projects?
* How can we reason about the quality of a software product in order to trust its future development?

Unfortunately these organizations often have nothing more than word-of-mouth on which to base their judgments of OSS products. With 1097071 projects currently hosted on SourceForge it is understandable that products of excellent quality may be overlooked. It is possible to supplement the word-of-mouth tradition with more formal metrics to evaluate the OSS.

**Objective(s) or Activities**

### The objectives of this project include:

# Investigate suitable metrics to measure important software quality factors and choose those suitable for OSS

# Select a range of representative Open Source projects for measurement

# Evaluate the software of these projects according to the selected quality factors

# Using statistic tools to analyse and present the result

**References**

(1) Feller, J., Fitzgerald, B., 2000. Framework analysis of the open source software development paradigm. Proceedings of 21st annual international conference on information systems

(2) Fenton N.E., & Pfleeger, S.L.,1997. Software metrics: a rigorous and practical approach. 2nd ed. London: International Thomson Computer Press.

(3) Golden, B., 2005. Succeeding with Open Source. Addison-Wesley.

(4) Halstead, M., 1975. Elements of Software Science. Holland: Elsevier

(5) International Organization for Standardization, 1991. Information technology-Software product evaluation: Quality characteristics and guidelines for their use. ISO/IEC IS 9126. Geneva: ISO

(6) Kitchenham, B., & Pfleeger, S.L., 1996. Software quality: The elusive target. IEEE Software, 13, 1, 12-21

(7) Liu, D. and Xu, S., 2007. New Quality Metrics for Object-Oriented Programs, Proceedings of the 8th ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing (SNPD 2007), pp. 870-875, July 30 - August 1, Qingdao, China.

(8) Liu, D., Xu, S., Brockmeyer, M., 2008. Investigation on Academic Research Software Development, 2008 IEEE International Conference on Computer Science and Software Engineering (CSSE 2008), December 12~14, 2008, Wuhan, China.

(9) McCabe, T.,1976. A complexity measure. IEEE Transactions on Software Engineering, 2, 4, 308-320

(10) Omondo tool, 2009. www.omondo.com

(11) O’Reilly, T., 1998. The Open-Source Revolution, O’Reilly.

(12) Stamelos, L. Angelis, A. Oikonomou, G.L. Bleris, 2002. Code Quality Analysis in Open Source Software Development. Information Systems Journal 12, 43-60.