Software Quality in Open Source Software Ecosystems

Pankaj Kamthan
Concordia University, Canada

INTRODUCTION

The steady rise of open source software (OSS) (Raymond, 1999) over the last few decades has made a noticeable impact on many sectors of society where software has a role to play. As reflected from the frequency of media articles, traffic on mailing lists, and growing research literature, OSS has garnered much support in the software community. Indeed, from the early days of GNU software, to X Window System, to Linux and its utilities, and more recently the Apache Software Project, OSS has changed the way software is developed and used.

As the deployment of OSS increases, the issue of its quality with respect to its stakeholders arises. We contend that the open source community collectively bears responsibility of producing "high-quality" OSS. Lack of quality raises various risks for organizations adopting OSS (Golden, 2004). This article discusses the manifestation of quality in open source software development (OSSD) from a traditional software engineering standpoint.

The organization is as follows. We first outline the background and related work necessary for the discussion that follows, and state our position. This is followed by a detailed treatment of key software engineering practices that directly or indirectly impact the quality of OSS. Next, challenges and directions for future research are outlined and, finally, concluding remarks are given.

BACKGROUND

The concept of open source can mean different things in different contexts (Gacek & Arief, 2004). For the purposes of this chapter, we will use the term "open source" to imply software whose source is available without cost to the user, imposes minimal non-restrictive licensing conditions, and is itself based upon non-proprietary technologies. Software that does not fall into this category is termed as non-OSS. For example, commercial software is one class of non-OSS.

Software engineering (Ghezzi, Jazayeri, & Mandrioli, 2003) advocates a disciplined and systematic approach to the development of high-quality software within budget, schedule, and other organizational constraints. Although OSS itself has a long and rich history, it is only in recent years that a software engineering viewpoint towards it has been taken (Vixie, 1999). There are different ways in which OSS can be used in software engineering education (Kamthan, 2007).

In software engineering, there is much emphasis on quality in all aspects of software: project, process, product, and occasionally even people. In this article, we address these aspects at a high-level either directly or indirectly in the context of OSSD. By the term product, we mean any artifact created during the process, including models, process documents, and source code.

Significance of Quality in OSS

Although comprehensive studies are still lacking (Aberdour, 2007), there are many OSS that seem to exhibit "high" quality, particularly those that have been in use for a while.

Still, issue of OSS quality has been acknowledged as an issue. The concerns of maintainability, performance, portability, reliability, security, and usability have been addressed in the past (Halloran & Scherlis, 2002; Michlmayr, Hunt, & Probert, 2005; Nichols & Twidale, 2005; Schmidt & Porter, 2001; Seidel & Niedermeier, 2003; Spinellis, 2006). To address usability concerns has led to the launch of OpenUsability.org that provides guidance for improving usability of open source projects that are submitted to it.

However, the approach to quality assurance and assessment in OSSD is not systematic (Porter et al., 2006) and therefore the results do not seem to be repeatable. For example, there is little evidence of any quality model (Fenton & Pfleeger, 1997) being followed. In OSS, peer reviews are used as a technique for an informal

evaluation whereas formal inspections are apparently non-existent. Comprehensive collections of test cases, test suites or test harnesses are rare, and broad testing is even rarer. More importantly, participation is voluntary and monitoring is almost non-existent. The linear relation of the number of bugs found to improvement of quality that has been proposed (Raymond, 1999) and endorsed (Aberdour, 2007; Verma, 2006) is not necessarily accurate (Glass, 2003).

A software engineering perspective towards quality in OSS is necessary for a variety of reasons: OSS may be adopted and used in critical areas of an organization and so need to be carefully examined with respect to non-OSS alternatives, OSS installed in an organization may need to be maintained over time and therefore need to be well-understood by maintenance engineers, and current OSS practices could be of interest from an academic (teaching, learning, or research) standpoint. Lack of quality (or perceptions thereof) can adversely affect the adoption of OSS in organizations (Graham, Mansingh, & Grant, 2006).

ELEMENTS OF QUALITY IN A SOFTWARE ENGINEERING CONTEXT AND ITS OPEN SOURCE PERSPECTIVE

There are different views of software quality (Wong, 2006), including social (organizational), manufacturing (production), economical, technical, and user perspectives. Since these views are not necessarily mutually exclusive, we take a *heterogeneous* approach to quality.

This section looks at six broadly classified and interrelated aspects that are related to quality that are both common and essential in most software engineering organizational contexts. They are: quality of project management; standards and quality; quality of team organization and dynamics; quality of process, workflows, and collaborative activities; modeling, specification, and documentation for quality; and quality evaluation via measurement. It examines how well they are realized (or not) in an OSSD environment, and challenges and obstacles in doing so.

Quality of Project Management

Managing a software project is important for its eventual success. We shall limit our discussion to ethical conduct, and time and configuration management.

OSS is carried out on an "honor system" and is not bound by organizational or professional code of ethics. Specifically, there are little or no repercussions for not following up on work or on schedule, or stalling the project altogether. This flexibility may be attractive to some in a "casual" context but does not scale well in a professional or educational setting.

The reason for abandoning an OSS project as can be observed on development and distribution environments like SourceForge are often not given or made public. In general, there is no specific price (exhibited in a loss of pay or demotion) for not performing up to the expectations or not working to your full potential.

The development of OSS is not bound by time lines that are associated with any cost. Therefore, there is little sense of urgency in OSS projects.

The distributed nature of contribution as well as the desire of the developers to be able to disseminate "up-to-the-minute" code has led to a usually strong support for configuration management (Asklund & Bendix, 2001) in OSS development. These could include version control, bug tracking, or build management. For example, posting nightly builds for tryout is quite common in an OSS environment and the same can be expected in a medium-to-large industrial environment.

Standards and Quality

There are a variety of reasons for introducing and adhering to standards in software engineering. Standards provide a common ground for a team, streamline efforts, and when applied well, are known to contribute towards quality improvement (Schneidewind & Fenton, 1996). The use of standards can also improve interoperability.

There is little evidence to support the use of standards for process documents (such as those from ANSI, IEEE, and/or ISO/IEC) in OSS. However, the OSS approach serves as a platform for trying out new technologies and developing "proof-of-concept" implementations, and any use of standards is limited to that context.

Quality of Team Organization and Dynamics

There are differences between the social structures of a team in a software engineering environment versus participants in the OSSD. In general, software engineers working on a software project in a professional or learning context are collocated while those in OSS