Applied Mathematical Sciences, Vol. 6, 2012, no. 62, 3051 - 3056

The Impact of Release Management and Quality Improvement in Open Source Software Project Management

N. Arulkumar [1] and S. Chandra Kumramangalam

Anna University, Coimbatore, India

Abstract

The open source practices have been remarkable successful in recent years, there are a number of product quality challenges facing the open source development model. Regarding to the fact that rare open source projects have achieved success with end product of high level quality.

There is significant room for further quality improvement and one area that deserves special attention is release management. This research will identify problems with current release practices, verify possible advantages of an increasingly popular release model, and develop interventions to improve release management in free software projects. The research also aims to answer the fundamental question as to how volunteer projects can deliver predictable and high quality software.

Keywords: Release management, Open source, FOSS

Introduction

Open source software is a part of the software ecosystem that affords software developers and users an alternative style of software development and distribution. It coexists in that environment along with a broad spectrum of other development and distribution methods, including public domain software, freeware, shareware, proprietary commercial software, and even vaporware. Open source software is found in development tools, utility code, operating systems, and applications [1]. Finding a precise definition of "open source" software can be tricky. A recent Google search produced no less than 10 definitions from various sources, each of which gives a somewhat different perspective and slant. As discussed, different members of the open source

[1] nava_200@sify.com

community differ in their ideas and goals for open source software development; however, one helpful and concise definition is: Open Source Software is software for which the underlying programming code is available to the users so that they may read it, make changes to it, and build new versions of the software incorporating their changes [2]. There are many types of Open Source Software, mainly differing in the licensing term under which (altered) copies of the source code may (or must) be redistributed. In comparison with commercial software, open source software differs in a number of ways. Commercial software is most often distributed only in binary, executable form (closed source software), and its developers reserve to themselves the ability to know the source code, to modify the software, to distribute the software, and to authorize others to do those things. It is not unusual for commercial software developers to refer to their software source code as the "crown jewels" of the company and to jealously guard it against disclosure to others. On the other hand, "open source software" is distinguished from commercial software by the availability of source code to everyone who receives a license to use the software and, in many cases, Free and open source software has had a major impact on the computer, industry since the late 1990s and has changed the way software is perceived, developed and deployed in many areas [3]. Free and open source software, or FOSS, is typically developed in a collaborative fashion and the majority of contributors are volunteers.

Even though this collaborative form of development has produced a significant body of software, the development process is often described as unstructured and unorganized. This dissertation studies the FOSS phenomenon from a quality perspective and investigates where improvements to the development process are possible. In particular, the focus is on release management since this is concerned with the delivery of a high quality product to end-users. This research has identified considerable interest amongst the FOSS community in a novel release management strategy, time based releases. In contrast to traditional development which is feature-driven, time based releases use time rather than features as the criterion for the creation of a new release. Releases are made after a specific interval, and new features that have been completed and sufficiently tested since the last release are included in the new version. This dissertation explores why, and under which circumstances, the time based release strategy is a viable alternative to feature-driven development and discusses factors that influence a successful implementation of this release strategy. It is argued that this release strategy acts as a coordination mechanism in large volunteer projects that are geographically dispersed. The time based release strategy allows a more controlled development and release process in projects which have little control of their contributors and therefore contributes to the quality of the output.

Statement of the Problem

This research aims to identify existing quality related problems in free software projects and to use the outcomes as a starting point for the development of quality improvement strategies. In exploratory interviews with a number of free software and open source developers, release management has been found to be a problematic area; this research will therefore focus on this topic in particular as one aspect of quality management. There are several reasons as to why release management in distributed, volunteer free software projects may often be associated with problems. First of all, many of those who maintain software projects are programmers, who do not necessarily have the coordination and management skills required for release management [4]. Secondly, extra commitment from project participants is necessary during a release so that deadlines are met, but volunteers may not be able to put in more work than usual [5]. Finally, there is often a dichotomy between the requirements of users and developers. Since developers mainly use development releases, they might not see the need for well tested and stable releases aimed at less technical and corporate users. Inadequate release management can lead to a number of problems, such as software which is out of date, breaks compatibility, or does not meet the quality standards or the requirements of users. This research aims to identify such problems and good practices in open source projects, and develop further processes and techniques to improve release management in free software and open source projects. There will be a close interaction with the free software community to ensure that the outcomes of this research will be used by projects to improve release management. Release management will then be proposed. These will subsequently be tested in live projects using action research.

Proposed Research Methods

This research will test whether time-based releases are actually associated with such advantages. The circumstances under which time-based releases should be chosen over more conventional release strategies will be studied. Following this, the question of how a project can successfully move to a predictable time-based release will be addressed number of open source projects that follow good release practices will be observed; interventions that can improve the quality problems that have been identified in the interviews are of particular interest since they are largely issues which have yet to be solved in the FOSS community. Good solutions for some of the quality problems have yet to be developed Broadly speaking, the research can be subdivided into the following six phases which in turn employ particular research methods.

A. Identification of Processes and Problems

In this phase, current release processes and strategies will be investigated. Problems related to these processes will also be identified. This phase will mostly employ interviews, along with surveys, in order to obtain an in-depth input from a wide variety of sources, such as developers and end-users of free software and open source.

B. Investigation of Time-Based Release Strategy and Testing of Hypotheses

Based on the results from the first phase, hypotheses will be generated that will subsequently be tested. At the time of writing, there are a number of preliminary hypotheses that will be clarified and refined further before the research moves to phase two. This phase is characterized by a positivist approach in which hypotheses are tested in quasi-experiments. Comparable projects employing time and feature-based releases will be compared using empirical data. For these studies, a number of methods will be used to mine and analyze data. For example, a tool has been developed to reconstruct the status of bug reports on any given date, thereby allowing the investigation of a project's evolution over time. Furthermore, existing tools to analyze the development process and evolution based on data from version control systems, such as CVS, will be used.

C. Development of Interventions

In the third phase, case studies will be performed to study good practices. Based on these studies, interventions will be developed to improve release management. They will subsequently be tested in live projects. On the assumption that the results of phase two demonstrates that time-based releases are indeed a viable strategy offering certain advantages over other strategies, this phase will also consider the migration of a project to time-based releases. This work is based on action research involving live projects, such as Debian, which faces considerable problems with its releases and is searching for solutions. There will also be quasi experiments to test the effectiveness and the impact on the quality of different release practices [6].

D. Configuration Management

Many FOSS projects offer a high level of customization. While this gives users much flexibility, it also creates problems with testing. It is very difficult or impossible for the lead developer to test all combinations so only the most popular configurations tend to be tested. It is quite common that, when a new release is made, users report that the new version broke their configuration.

E. Problems with Coordination and Communication

In some projects, there are Problems with coordination and communi-

cation which can have a negative impact on project quality. Sometimes it is not clear who is responsible for a particular area and therefore bugs cannot be communicated properly. There may also be duplication of effort and a lack of coordination related to the removal of critical bugs.

F. Company Involvement

While most FOSS projects are carried out by volunteers, companies are increasingly getting involved in the development of FOSS projects, especially in large and successful projects. An interesting question is how FOSS projects can best benefit from company involvement. There are some constraints under which FOSS projects work that are not applicable to companies. For example, not many FOSS projects have good user documentation, and few projects have automatic regression test suites. These are areas where companies might be able to contribute in a way most volunteers could not.

Conclusion

This research focuses on release management as one aspect of quality management and quality improvement in volunteer free software and open source projects. Release management is a problematic area in open source development in which significant improvements are possible. Research in this area that is carried out in close collaboration with the free software community has the potential to make a substantial contribution. In addition to identifying good release management practices, this research will investigate whether a group of volunteers can make predictable and high quality releases. This addresses fundamental questions regarding the open source development model and qualifies whether consistent levels of quality and predictable schedules are possible in distributed, volunteer projects.

REFERENCES

1. T. J. Halloran and W. L. Scherlis, *High quality and open source software practices*, in Proceedings of the 2nd Workshop on Open Source Software Engineering. Orlando, FL, USA: ICSE, (2002).

2. J. R. Erenkrantz, Release management within open source projects, Fuzzy Sets and Systems, in 3rd Workshop on Open Source Software Engineering. ICSE, (2003) 51-55.

3. D. M. German, Mining CVS repositories, the softChange experience, in International Workshop on Mining Software Repositories. ICSE, (2004), 17-21.

4. A. Senyard and M. Michlmayr, How to have a successful free software

project, in Proceedings of the 11th Asia-Pacific Software Engineering Conference. Busan, Korea: IEEE Computer Society, (2004), 84-91.

5. M. Michlmayr, Managing volunteer activity in free software projects, in Proceedings of the 2004 USENIX Annual Technical Conference, FREENIX Track, Boston, USA, 2004, pp. 93-102.Chanas S., Kuchta D., A concept of the optimal solution of the transportation problem with fuzzy cost coefficients, Fuzzy Sets and Systems 82 (1996) 299-305.

6. M. Michlmayr and B. M. Hill, Quality and the reliance on individuals in free software projects, in Proceedings of the 3rd Workshop on Open Source Software Engineering. Portland, OR, USA: ICSE, (2003), 105-109.

Received: January, 2012