Release Management in Free Software Projects: Practices and Problems

Martin Michlmayr, Francis Hunt, and David Probert

Centre for Technology Management
University of Cambridge
Cambridge, CB2 1RX, UK
martin@michlmayr.org

Abstract. Release management plays an important role in every software project since it is concerned with the delivery of a high quality product to end-users. This paper explores release practices employed by volunteer free software projects and shows problems that occur. A challenge that has been identified is the difficulty of coordinating a distributed team of volunteers in order to align their work for a release.

Key words: Release management, coordination, volunteers

1 Introduction

Release management is an important part of quality management since it is concerned with the delivery of high quality software to users [2]. Free and open source software (FOSS) is characterized by a highly iterative development model in which new development releases are typically made available very frequently. Despite the frequency of development releases in some projects, there are often problems with stable releases for end-users. The following two examples should illustrate this problem:

- Debian: in recent years, the project has faced increasingly delayed and unpredictable releases. Most notably, the release process of Debian 3.1 was characterized by major delays. Initially announced for December 1, 2003, the software was finally released in June 2005 – a delay of one and a half years. By the time the new version was released, the previous stable release was largely considered out of date and did not run on modern hardware.
- GNU tools: despite their popularity and importance, development has been slow in recent years and there is a long interval between releases. Version 1.13 of `tar` came out in August 1999, followed by version 1.14 at the end of 2004. The compression utility `gzip` saw a new version in December 2006, more than a decade after the last stable release in 1993. As a consequence of these long delays between stable releases, several vendors started shipping pre-releases.

These examples show that release management is a real matter of concern. Despite the importance of this aspect of software production, little attention has been given to release management in FOSS projects [1].

This paper performs an exploratory study in order to get a better picture of actual practices and problems associated with release management in FOSS projects. The topic has been studied from a quality perspective and follows a similar approach to a previous paper which investigated quality practices and problems in FOSS projects [3].

2 Study

2.1 Methodology

For this study, interviews with twenty experienced developers from different FOSS projects were carried out. The interviews were conducted at a conference over the course of three days. Interviewees were either core developers or release managers of FOSS projects. The range of FOSS projects in which developers participated was very wide, ranging from small to very large and complex projects, and included projects of all types, such as desktop and server software and development tools. This great variety gives a good coverage of practices found in the FOSS community.

2.2 Types of Release Management

The study has revealed that the general term 'release management' is used to refer to three different types of releases. These types differ quite significantly regarding the audience they address and the effort required to deliver the release. The three types are:

- Development releases aimed at developers interested in working on the project or experienced users who need cutting edge technology.
- Major user releases based on a stabilized development tree. These releases deliver significant new features and functionality as well as bug fixes to end-users and are generally well tested.
- Minor releases as updates to existing user releases, for example to address security issues or critical defects.

Since developers are experts, development releases do not have to be polished and are therefore relatively easy to prepare. Minor updates to stable releases also require little work since they usually only consist of one or two fixes for critical or security bugs. On the other hand, a new major user release requires significant effort: the software needs to be thoroughly tested, documentation has to be written and the software needs to be packaged up. Since the main challenges are associated with the preparation of major new user releases, the focus will therefore be on releases aimed at end-users.

In terms of a project's release strategy, projects can be classified according to the following two release strategies:

- Feature based strategy: the basic premise of this strategy is to perform a new release when a specific set of criteria has been fulfilled and certain goals attained, most typically a number of features which developers perceive as important. This strategy is in line with traditional software development which is feature driven.
- Time based strategy: in this strategy a specific date is set for the release well in advance and a schedule created so people can plan accordingly. Prior to the release, there is a cut-off date on which all features are evaluated to decide whether they can be included in the release or have to be postponed.

2.3 Skills of the Release Manager

The role of the release manager is diverse and demanding because they have to interact with a large number of different people, understand technical issues but also know how to plan and coordinate. The following important skills have been identified:

- Community building: showing people that their input is useful. Release managers also need respect in the community in order to perform their work.
- Strong vision: showing developers in which direction the project should be moving.
- Discipline: saying 'no'. Release manager have to focus on overall goal and can not make everyone happy.
- Judgement: gauging the risk and possible impact of a particular change.
- Attention to detail: walking through every line of code that has changed.
- Good communication: writing release notes, asking for feedback, interacting with users.
- Management skills: talking to people, organizing, planning, making sure that different tasks are performed.

It is interesting to note that release managers in small and large projects play a vastly different role even though they essentially have the same responsibility, namely getting a high quality release out. In a small project, the release manager usually has an administrative role which involves the preparation of the release in different formats that can be distributed, the creation of release notes and the actual distribution of the software. In large projects, on the other hand, there is a big emphasis on coordination that needs to be performed by the release manager. They have to make sure that different parts of the software are ready at the same time and that all developers are aligned towards the common goal of creating a stable release.

2.4 Tools and Practices

Despite the important role release management plays in the delivery of quality software to users, there is little knowledge as to how FOSS projects perform releases. This section covers tools and practices employed during release management.

The study has revealed that there are few dedicated tools used in release management. However, many FOSS projects have tightly integrated their development tools with their whole development process, including release management. In particular, the use of version control and bug tracking systems serves important functions during release management as they give a good overview of the status of the project.

In addition to the use of tools, there are specific practices which are related to release management:

- Freezing: development is locked down in order to focus on the removal of defects and publication of the release.
- Scheduling: relatively few projects make use of a schedule but it is a vital component in those projects which employ a time based release strategy.
- Establishing milestones: some projects have loosely defined milestones but given the volunteer nature of most projects there is no guarantee that they will actually be achieved.
- Setting deadlines: many projects set deadlines but they are not always effective because the release manager has no control over volunteer participants.
- Building on different architectures: as part of a project's testing plan, it is beneficial to build the software on a number of different hardware platforms because each of them may exhibit bugs not visible on another platform.
- User testing: one of the main benefits from preparing a release is the feedback potentially obtained from a wide range of users. Even those projects which heavily deploy automatic test suites think that the most significant insights usually come from actual users. It is therefore important to make snapshots easily available.
- Following a release check list: a number of projects use a check list to make sure that all steps that are necessary to make a new release are followed.
- Holding a post-release review: surprisingly few projects have a formal post-release review but there are often informal discussions on the mailing list of the project, in particular when there were problems with the release.

2.5 Problems

As discussed earlier, the process of preparing a new stable release for end-users is quite elaborate and complex since the software needs to be sufficiently tested, documented and packaged for release. The release management process often faces certain problems, the most common of which are as follows:

- Major features not ready: planning in volunteer teams is very hard and it happens regularly that major features which are on the critical path are not ready. These blockers need to be resolved so the release process can continue.
- Interdependencies: with the growing complexity of software, there are increasing levels of interdependencies between software. For example, a piece of software may use libraries developed by another project or incorporate certain software components created elsewhere. This creates a dependency and can lead to problems with a project's release when those other components are not ready.
- Supporting old releases: there is generally a lack of resources in the FOSS community and in many projects old releases receive little support. Some vendors which distribute a given release may step in and offer basic support but it may still be problematic for other users of the software.
- Little interest in doing user releases: even though this study has emphasized the importance of user releases, many developers show little interest in making releases. Since developers by definition generally use the development version they often do not understand the need for a user release or do not see when a user release is massively out of date.
- Vendors shipping development releases: when projects do not publish new releases, some vendors start shipping development releases because they contain features or fixes which their customers need. This situation is problematic because it can lead to fragmentation and because development releases are generally not as well tested as user releases.
- Long periods without testing: some projects apply many changes with relatively little testing. At the time of the release, many issues are discovered and this leads to major delays.
- Problem of coordination: development in FOSS projects is done in a massively parallel way in which individual developers independently work on features they are interested in. Towards the release, all of this development needs to be aligned and these parallel streams have to stabilize at the same time. This can require substantial amounts of coordination.

3 Discussion

This study has shed light on practices and problems related to release management in FOSS projects, an important area which has so far been relatively unexplored. A major observation is that the preparation of user releases is associated with considerable problems, in particular in large projects. Small projects often face human resource problems but large projects deal with a more fundamental issue, namely that of coordinating a loosely defined team of virtual volunteers towards the common goal of making a release. These volunteers typically work independently in parallel development streams which require little coordination with other members of the project. However, during the preparation for the next release, these parallel streams need to be integrated and

that requires coordination. Even more problematic is that each independent development stream has to be completed at the same time even though there is usually no schedule which provides guidance. When freezes are announced out of the blue, everyone wants to get their features and fixes in and the high number of changes pushes back the release date. Each delay is seen as a further opportunity to make more changes, thereby causing even more delays.

The present study has identified such problems in a number of projects but there is also evidence that some projects have found ways to deal with these problems. There is considerable interest among projects in the time based release strategy, in which time rather than features is used as orientation for the release and a schedule is followed. This release strategy is used in a growing number of projects, such as GNOME, OpenOffice.org and X.org. Time based releases promise solutions to cope with the complexity that occurs when a large team of distributed volunteers needs to be coordinated toward a common goal. However, further work is needed to study this release strategy in detail.

4 Conclusions

This paper has shed light on an important and as yet fairly unexplored area of FOSS development. The main finding of this study is that projects, in particular those with many developers, face severe challenges during the preparations for a release. These challenges are related to the coordination of a team which is not only large but also geographically dispersed and mainly consists of volunteer participants. While FOSS projects often rely on self-assignment of tasks with little coordination, all developers need to align their activities towards a common goal during release preparations. The time based release strategy has been suggested as a mechanism to coordinate large volunteer teams but further work is needed to explore this release strategy.

5 Acknowledgements

This work has been funded in part by Google and the EPSRC. Thanks to Fieldwave Ltd covering the expenses to attend the conference where the interviews for this study were conducted.

References

1. J. R. Erenkrantz. Release management within open source projects. In *3rd Workshop on Open Source Software Engineering*, pages 51–55. ICSE, 2003.
2. K. D. Levin and O. Yadid. Optimal release time of improved versions of software packages. *Information and Software Technology*, 32(1):65–70, 1990.
3. M. Michlmayr, F. Hunt, and D. Probert. Quality practices and problems in free software projects. In M. Scotto and G. Succi, editors, *Proceedings of the First International Conference on Open Source Systems*, pages 24–28, Genova, 2005.