Time-Based Release Management in Free and Open Source (FOSS) Projects

Martin Michlmayr, Centre for Technology Management, University of Cambridge, Cambridge, UK

Brian Fitzgerald, Lero – Irish Software Engineering Research Centre, University of Limerick, Limerick, Ireland

ABSTRACT

As the Free and Open Source (FOSS) concept has matured, its commercial significance has also increased, and issues such as quality and sustainability have moved to the fore. In this study, the authors focus on time-based release management in large volunteer FOSS projects, and reveal how they address quality and sustainability issues. They discuss the differences between release management in the traditional software context and contrast it with FOSS settings. Based on detailed case studies of a number of prominent FOSS projects, they describe the move to time-based release management and identify the factors and criteria necessary for a successful transition. The authors also consider the implications for software development more generally in the current dynamic Internet-enabled environment.

Keywords: Empirical Research, Open Source Software, Project Management, Software Engineering, Time-Based Release Management

INTRODUCTION

“Release early and release often.”

Despite Raymond’s (1999) provocative characterisation of release management in Free and Open Source Software (FOSS), the topic has been the subject of little research in the interim, exceptions being studies by Erenkrantz (2003) and Michlmayr et al. (2007). Furthermore, even in the traditional software literature there have been relatively few studies of release management, most commonly in conference proceedings (e.g. Dayani-Fard et al., 2005; Du & Ruhe, 2005; Erdogmus, 1999; Li et al., 2003; Ruhe & Greer, 2003; Sassenburg & Bergout, 2006) and a smaller number of journal papers on the topic (e.g. Greer & Ruhe, 2004; Levin & Yadid, 1990).

As the FOSS concept has matured, its commercial significance and economic potential has also increased, and issues such as quality and sustainability have become increasingly important (Fitzgerald, 2006). Indeed there is
evidence that a significant inhibitor to FOSS adoption arises from the perception of a lack of guaranteed quality in FOSS products (Tawileh et al., 2006). As a consequence, FOSS projects need to mitigate risk from such specific issues as lack of deadlines (Garzarelli & Galoppini, 2003), reliance on volunteers (Robbins, 2002; Michlmayr & Hill, 2003) and ad-hoc coordination and management processes (Bergquist & Ljungberg, 2004; Zhao & Erlbaum, 2003). A number of FOSS projects appear to have addressed the above issues through formalising their release management process. The latter is an important part of a project’s approach to quality assurance since developers stop adding new features during the preparation for a release and instead focus on the identification and removal of defects. The feedback obtained after a release also provides information as to which parts of the software might need more attention.

Despite the increased company involvement in FOSS, a comprehensive study found that more than two-thirds of FOSS developers comprise individual volunteers (Ghosh, 2006). Consequently, our study focuses on volunteer FOSS projects as these will require more formalised processes to mitigate perceptions of risk in adoption. Furthermore, many of the unique and most significant benefits of FOSS arise in large projects. For example, large projects are more likely to result in communities forming around them which have sufficient levels of participants with diverse skills to ensure a rapid development trajectory and prompt defect removal (Mockus et al., 2002; Raymond, 1998).

Given the above, our overall research objective was to investigate release management in large volunteer-oriented FOSS projects.

The paper is laid out as follows: First, we discuss the topic of release management in traditional software contexts and then discuss its role in the specific context of FOSS projects. Then we present our two-phase research approach for this study. Afterwards we analyses and discusses the findings of the study. Finally, we discuss our conclusions and identify implications for research and practice.

SOFTWARE RELEASE MANAGEMENT

Software maintenance is the sub-field concerned with evolution and maintenance of software after its initial development and release. Levin and Yadid (1990) criticise traditional models of software development which only focus on the initial release and ignore subsequent releases. A continuous release strategy is important for several reasons: it delivers both fixes and new functionality to users (Levin & Yadid, 1990). Also, it staves off obsolescence by ensuring the value of the software is maintained (Baetjer, 1997). As the development environment has become more dynamic and fast-paced, the so-called Internet-time (Baskerville et al., 2002), incremental releases have become more common (Greer & Ruhe, 2004). However, in proprietary release management, this is a complex issue which requires delicate balancing as early introduction of a new release may erode the market-share and revenue generating potential of the existing release.

In FOSS projects such commercially-driven balancing has not been significant to the same extent (Robbins, 2002), and there are other significant differences also. Table 1 provides a summary which contrasts the differences between release management in traditional proprietary development and FOSS projects.

Overall, however, release management has been under-researched in relation to FOSS. While Erenkrantz (2003) has identified characteristics related to release authority and actual work during a release, we know little about actually moving from the development phase to preparation of a release. Fundamentally, it is not obvious how a team of loosely-connected globally distributed volunteers can work...
The Impact of Open Source Development on the Social Construction of Intellectual Property
[www.igi-global.com/chapter/impact-open-source-development-social/18729?camid=4v1a](www.igi-global.com/chapter/impact-open-source-development-social/18729?camid=4v1a)