Chapter 2.29 Bug Fixing Practices within Free/Libre Open Source Software Development Teams[1]

Kevin Crowston
Syracuse University, USA

Barbara Scozzi
Politecnico di Bari, Italy

ABSTRACT

Free/Libre open source software (FLOSS, e.g., Linux or Apache) is primarily developed by distributed teams. Developers contribute from around the world and coordinate their activity almost exclusively by means of email and bulletin boards, yet some how profit from the advantages and evade the challenges of distributed software development. In this article we investigate the structure and the coordination practices adopted by development teams during the bug-fixing process, which is considered one of main areas of FLOSS project success. In particular, based on a codification of the messages recorded in the bug tracking system of four projects, we identify the accomplished tasks, the adopted coordination mechanisms, and the role undertaken by both the FLOSS development team and the FLOSS community. We conclude with suggestions for further research.

INTRODUCTION

In this article, we investigate the coordination practices for software bug fixing in Free/Libre open source software (FLOSS) development teams. Key to our interest is that most FLOSS software is developed by distributed teams, that is, geographically dispersed groups of individuals working together over time towards a common goal (Ahuja et al., 1997, p. 165; Weisband, 2002). FLOSS developers contribute from around the world, meet face to face infrequently, if at all, and coordinate their activity primarily by means of computer mediated communications (Raymond, 1998; Wayner, 2000). As a result, distributed teams employ processes that span traditional boundaries of place and ownership. Since such teams are increasingly commonly used in a diversity of settings, it is important to understand how team members can effectively coordinate their work.

The research literature on distributed work and on software development specifically emphasizes the difficulties of distributed software development, but the case of FLOSS development presents an intriguing counter-example, at least in part: a number of projects have been outstandingly successful. What is perhaps most surprising is that FLOSS development teams seem not to use many traditional coordination mechanisms such as formal planning, system level design, schedules and defined development processes (Mockus et al., 2002, p. 310). As well, many (though by no means all) programmers contribute to projects as volunteers, without working for a common organization and/or being paid.

The contribution of this article is to document the process of coordination in effective FLOSS teams for a particularly important process, namely bug fixing. These practices are analyzed by adopting a process theory, that is, we investigate which tasks are accomplished, how and by whom they are assigned, coordinated, and performed. In particular, we selected four FLOSS projects, inductively coded the steps involved in fixing various bugs as recorded in the projects' bug tracking systems and applied coordination theory to identify tasks and coordination mechanisms carried out within the bug-fixing process.

Studying coordination of FLOSS processes is important for several reasons. First, FLOSS development is an important phenomenon deserving of study for itself. FLOSS is an increasingly important commercial issue involving all kind of software firms. Million of users depend on systems such as Linux and the Internet (heavily dependent on FLOSS software tools) but as Scacchi notes "little is known about how people in these communities coordinate software development across different settings, or about what software processes, work practices, and organizational contexts are necessary to their success" (Scacchi, 2002, p. 1; Scacchi, 2005). Understanding the reasons that some projects are effective while others are not is a further motivation for studying the FLOSS development processes. Second, studying how distributed software developers coordinate their efforts to ensure, at least in some cases, high-performance outcomes has both theoretical and managerial implications. It can help understanding coordination practices adopted in social collectives that are not governed, at least apparently, by a formal organizational structure and are characterized by many other discontinuities that is, lack of coherence in some aspects of the work setting: organization, function, membership, language, culture, etc. (Watson-Manheim et al., 2002). As to the managerial implications, distributed teams of all sorts are increasingly used in many organizations. The study could be useful to managers that are considering the adoption of this organizational form not only in the field of software development.

The remainder of the article is organized as follows. In Section 2 we discuss the theoretical background of the study. In Section 3 we stress the relevance of process theory so explaining why we adopted such a theoretical approach. We then describe coordination theory and use it to describe the bug-fixing process as carried out in traditional organizations. The research methodology adopted to study the bug-fixing process is described in Section 4. In Section 5 and 6 we describe and discuss the study's results. Finally, in Section 7 we draw some conclusions and propose future research directions.

BACKGROUND

In this section we provide an overview of the literature on software development in distributed environment and the FLOSS phenomenon.