Chapter 5
Are Developers Fixing Their Own Bugs?
Tracing Bug-Fixing and Bug-Seeding Committers

Daniel Izquierdo-Cortazar
Universidad Rey Juan Carlos, Spain

Andrea Capiluppi
University of East London, UK

Jesus M. Gonzalez-Barahona
Universidad Rey Juan Carlos, Spain

ABSTRACT

The process of fixing software bugs plays a key role in the maintenance activities of a software project. Ideally, code ownership and responsibility should be enforced among developers working on the same artifacts, so that those introducing buggy code could also contribute to its fix. However, especially in FLOSS projects, this mechanism is not clearly understood: in particular, it is not known whether those contributors fixing a bug are the same introducing and seeding it in the first place. This paper analyzes the comm-central FLOSS project, which hosts part of the Thunderbird, SeaMonkey, Lightning extensions and Sunbird projects from the Mozilla community. The analysis is focused at the level of lines of code and it uses the information stored in the source code management system. The results of this study show that in 80% of the cases, the bug-fixing activity involves source code modified by at most two developers. It also emerges that the developers fixing the bug are only responsible for 3.5% of the previous modifications to the lines affected; this implies that the other developers making changes to those lines could have made that fix. In most of the cases the bug fixing process in comm-central is not carried out by the same developers than those who seeded the buggy code.

DOI: 10.4018/978-1-4666-2937-0.ch005

1. INTRODUCTION

One of the most recognised advantages of the Free/Libre/Open Source Software (FLOSS) development model is its reliance on an open process: anyone is welcome to contribute; the majority of developers can focus on modularised, limited sections within a very large and complex system; and few core developers are generally experts in several areas of the source code, in a well accepted layered model (the "onion model" Mockus et al., 2002). These layers have been connected to actual responsibilities; core developers should focus on the main, more important features, while experimental versions should be implemented and tested by contributors on the development fringes (Goldman & Gabriel, 2004). Also, the layers of such model have been related to a shift in productivity: a recurring finding within FLOSS empirical research has shown that most of the development work is achieved by a small amount of developers, in a typical Pareto distribution (Koch, 2009).

The combinations of all the findings above have various, and not completely understood, effects. In some cases, a strong *territoriality* will emerge among developers "owning" certain parts of the code, and becoming more and more proficient in those (German, 2004; Robles et al., 2006). In other cases, the very nature of the FLOSS development implies that contributors join and then leave without necessarily halting the project (Robles & González-Barahona, 2006), but resulting in abandoned code and orphaned lines (Izquierdo-Cortazar et al., 2009).

Finally, certain developers will need to be active in maintenance activities: *corrective* maintenance fixing bugs in various parts of the code, for instance when source code is first introduced by developers with a low knowledge of the project (junior developers); *perfective* maintenance, for instance when new improved features are needed but the original developers have left the project and abandoned their contributions (Adams et al., 2009); *adaptive* maintenance, for instance when adaptations are needed, but the source code has been contributed in a programming language different from the main one supported by the project, so the current developers do not have enough skills in that language. Although in specific FLOSS communities there is the shared expectation that the original contributor will support his/her modules (especially in highly modular FLOSS projects, as Moodle or Drupal, Capiluppi et al., 2010), the volatility of contributors and the process of bug-fixing need to be clarified with respect of who introduced a certain bug, and who contributed the code to fix it. Examining and determining the proportion of errors that are fixed by different developers than those who introduced the error could provide a first approach to better understand the bug-fixing process in the specific FLOSS communities being studied.

In order to tackle this problem, the present study analyses the code base contained within the *comm-central* project (http://hg.mozilla.org/comm-central), a Mercurial Software Configuration Management (SCM) repository of Mozilla components (Thunderbird, SeaMonkey, the Lightning extension and Sunbird). Given the number and ID of each fixed bug, this research evaluates which changes have been performed, and by who, in the process of fixing the specific bug. The objective of this research is to evaluate patterns of bug-fixing activities within this FLOSS community, in order to detect, if any, the most recurrent and relevant scenarios among developers fixing bugs and those seeding the problem in the first place.

This paper makes two main contributions:

1. **Identifying Bug-Fixing and Bug-Seeding Committers:** The detection of those commits that have fixed a bug is crucial to determine the previous changes that took place to *seed* that bug. Using the source code lines that were handled by committers and tracing their history back make possible to know who previously handled those lines. Thus, it is possible to trace the changes in