2002

Exploring Strengths and Limits on Open Source Software Engineering Processes: A Research Agenda

Kevin Crowston
Syracuse University, School of Information Studies

Barbara Scozzi
Politecnico di Bari, Dipartimento di Ingegneria meccanica e Gestionale

Exploring the Strengths and Limits of Open Source Software Engineering Processes: A Research Agenda

Kevin Crowston
Syracuse University
School of Information Studies
4-206 Centre for Science and Technology
Syracuse, NY 13244–4100 USA
+1 (315) 443–1676
crowston@syr.edu

Barbara Scozzi
Politecnico di Bari
Dipartimento di Ingegneria meccanica e Gestionale
Viale Iapigia, 182
70126 Bari—Italy
+39 (080) 59 62 722
bscozzi@dimeg.poliba.it

1. INTRODUCTION

Many researchers have investigated the nature and characteristics of open source software (OSS) projects and their developer communities. In this position paper, after examining some success factors, we discuss potential limits on the replicability and portability of OSS engineering processes. Based on this analysis, we propose a research agenda to better understand the current nature of the processes and thus the strengths and the limitations.

2. THREE FACTORS IN THE SUCCESS OF OSS PROJECTS

The success of OSS projects has been mostly attributed to the speed of development and the reliability, portability, and scalability of the resulting software [1-6]. In turn, these qualities are attributed to three main issues, namely the fact that developers are usually also users of the software, the public availability of the source code, and the fact that developers are members of a community of developers.

First, OSS projects often originate from a personal need [7, 8]. Such needs attract the attention of other user-developers and inspire them to contribute to the project. This approach to software offers some real benefits in the design process.

Since developers are users of the software, they understand the requirements in a deep way. As a result, the ambiguity that often characterizes the identification of user needs or requests for improvement in the traditional software development process is eliminated: programmers know their own needs [9].

2nd Workshop on Open Source Software engineering, May 25, 2002, Orlando, Florida.

Second, in OSS projects, the source code is open to inspection by and contributions from any interested individual. Therefore, users can also be developers. If they find bugs, they can fix them themselves rather than having to wait for the developers to do so; if a specialized feature is needed, it can be added, even if it is not one that the developers feel is cost-justified. As a result, OSS bugs can be fixed and features evolved more quickly.

Finally, developers are part of a community. The OSS community represents a nexus of exchanges in which people report bugs expecting that other members will fix them. Similarly those who fix bugs expect other developers to contribute to other parts of the project [10]. Reputation is another important aspect—the community is in fact frequently described as being based on peer recognition and in some cases on a "cult of the personality". In particular, peer recognition is a value for the community that can sometimes lead to employment opportunities or access to venture capital [11]. In such an environment, developers may be motivated to do the best work they can, rather than anonymously finishing code so it can be shipped.

The main implication of the three characteristics described above is that OSS software engineering processes have evolved to develop software that meets developers' needs [12]. On the other hand, OSS, with its reliance on self-interested developers, may be less well suited for developing applications that address problems that developers tend not to face. We see very good OSS tools for software development and good end-user tools for issues faced by developers (e.g., email, word processing), for example, but would expect to see few OSS applications for problems developers rarely face (e.g., accounting, textual analysis).

There is some empirical support for this limitation to the OSS software engineering process. In our analysis of projects supported by SourceForge (http://sourceforge.net/) [13], for example, we found fewer projects for business and specialized topics. Furthermore, these projects tended to be in earlier stages of development and less used. Therefore, for the OSS model to work for a broad class of applications, projects need mechanisms to address the potential divide between developers and non-developers.

3. PROPOSED RESEARCH AGENDA

Based on the analysis above, we propose a research agenda for exploring the strengths and limits of OSS engineering processes. Since we are interested in projects where there is a sharp divide between users and developers, we must first clearly identify the population of such projects. Are there many successful OSS applications that are used primarily or exclusively by non-developers? What kinds?

For these project, we are interested first in requirements analysis. How are requirements developed for OSS projects on non-developer topics, where developers do not have a deep knowledge of the domain?

A possible source for requirements is direct communications from users. We are interested in how often feature requests are submitted. What is the process for handling feature requests? What happens when feature requests require substantial changes to the system design? Is there a role for user testing? How is it carried out in OSS? Can OSS software engineering processes support the development of novel user interface metaphors for such applications?

Also, we are interested in the nature of the bug fixing process. What kind of bugs are reported (e.g., architectural vs. non-architectural)? Which is the nature of bug reports? For example, what proportion of bug reports include code fixes or patches vs. just symptoms? What is the process for handling bug report (i.e., what is the sequence of activities, who actually perform them, and how are dependencies managed)? How do projects handle symptom reports? How are bug fixes from diverse sources integrated and tested?

Finally, we are interested in the role the support community (e.g., people involved in writing support documentation) play in projects developed for non-developers. Their role is considered not relevant in most OSS projects, but it can reveal fundamental in developing software that will not be used by developers.

As data for these studies, we hope to use available archives created during the process of software development. For example, many projects maintain archives of bug reports and disposition, which could be used to address some of these questions.

For other questions, we may carry out detailed case studies of particular projects.

4. REFERENCES

[1] Prasad, G.C. A hard look at Linux's claimed strengths. http://www.osopinion.com/opinions/GaneshCPrasad/GaneshCPrasad2-2html#LinuxStrenghts, 1999.

[2] Valloppillil, V. Halloween I: Open Source Software. http://www.opensource.org/halloween/halloween1.html, 1998.

[3] Valloppillil, V., and Cohen, J. Halloween II: Linux OS Competitive Analysis. http://www.opensource.org/halloween/halloween2.html, 1998.

[4] Hallen J., Hammarqvist, A., Juhlin, F., and Chrigstrom, A. Linux in the workplace. IEEE Software, 16 (1999), 52–57.

[5] Pfaff, B. Society and open source: Why open source software is better for society than proprietary closed source software. http://www.msu.edu/user/pfaffben/writings/anp/oss-is-better.html, 1998

[6] Leibovitch, E. The business case for Linux. IEEE Software, 16 (1999), 40–44.

[7] Moody, G. Rebel code—Inside Linux and the open source movement. Perseus Publishing, Cambridge, MA, 2001.

[8] Vixie, P. Software engineering, in Open sources: Voices from the open source revolution, C. Di Bona, S. Ockman, and M. Stone, O'Reilly, Eds. San Francisco, 1999.

[9] Kraut, R. E., and Streeter L. A. Coordination in software development. Communications of the ACM, 38 (1995), 69–81.

[10] Moon, J. Y., and Sproull L. Essence of distributed work: The case of Linux kernel. First Monday, 5, (2000).

[11] Markus, M. L., Manville, B., and Agres, E. C. What makes a virtual organization work?. Sloan Management Review, 42 (2000), 13–26.

[12] Ousterhout, J., Free Software needs profit. Communications of the ACM, 42 (1999), 44–45.

[13] Crowston, K., and Scozzi, B. Open source software projects as virtual organizations: Competency rallying for software development. IEE Proceedings Software, In press.