Collaborative Development within Open Source Communities

Javier Soriano
Universidad Politécnica de Madrid, Spain

Sonia Frutos
Universidad Politécnica de Madrid, Spain

Miguel Jiménez
Universidad Politécnica de Madrid, Spain

INTRODUCTION

Open source communities are one of the most successful--and least appreciated--examples of high-performance collaboration and community building on the Internet today. Open source communities began as loosely organized, ad-hoc communities of contributors from all over the world who shared an interest in meeting a common need. However, the organization of these communities has proven to be very flexible and capable of carrying out all kind of developments, ranging from minor projects to huge programs such as Apache (Höhn, & Herr, 2004; Mockus, Fielding, & Herbsleb, 2005).

Other collaboration-intensive communities could benefit enormously by learning what open source communities are and how they work. In fact, their motivation and objectives are not confined to software development projects. They are increasingly taking shape around non-software-related collaborative activities (Shah, 2005). Moreover, open source has come to stand for much more than software whose source code can be freely modified and redistributed subject to just a few restrictions imposed by the terms of its distribution license. Information, documentation, and other "sources" generally related to innovation, and knowledge building and sharing processes, tend to come under the open source umbrella.

A full comprehension of open source communities requires an in-depth understanding of the underlying organizational process (e.g., the software development process for software development projects). Some of the patterns underlying these organizational processes are not confined to software development and are common to other successful communities as well. One of the main goals of current research into open source communities is to identify these patterns (Kim, 2003) and develop a pattern language that can be used to describe, build, and improve other types of successful communities.

Finally, there is a trend toward two traditionally different development styles derived from opposing assumptions about the nature of development tasks—the model of most of the commercial world vs. the model of the open source world—converging. People would be astute to try to import some of the open source community model's virtues into a commercial context and will find it worthwhile taking a look at the conditions necessary for creative work.

Bearing these premises in mind, this article begins by defining and characterizing the term "open source community." It then tackles the issue of how these communities work (i.e., what the *patterns of collaboration* within successful open source communities are) and describes how these patterns could be applied in other types of communities apart from software related communities, and vice versa. This is intended to further the understanding of the open source model and its implications outside the realm of software development. In examining these questions, the article discusses existing, relevant research, and presents original case studies of working open source communities. These case studies hit at how collaboration works within successful projects.

OPEN SOURCE COMMUNITIES FUNDAMENTALS

We can define an *open source community* as a loosely organized, ad-hoc community of contributors from all over the world. These contributors share an interest in meeting a common need, ranging from minor projects to huge developments, which they do through a high-performance collaborative development environment, allowing the organizational scheme and processes to emerge over time. The term derives from the notion of *community* (i.e., an amalgamation of people with related interests), where intent, belief, resources, preferences, needs, goals, and a multitude of other conditions may be present and common, affecting the degree of adhesion within the group. Communities may meet to share information, to participate in shared projects, or to complete group tasks. What most characterizes a community is the pursuit of a common productive goal and sharing interaction in many ways.

Essentially born out of a desire for increased general access to source and binary code, open source communities have been bound to computer networks, and have evolved at the same pace as the Internet. Inexpensive access to Internet resources and source or binary code has allowed programmers to collaborate irrespective of where they are, and is one of the major factors in the growth of the number and size of communities.

The open source collaborative development carried out by open source communities has led to the use and spread of more and more sophisticated collaborative development environments (CDEs), virtual spaces where all the stakeholders of a software project, possibly distributed in time and space, can negotiate, brainstorm, discuss, share knowledge and resources, and generally labor together to carry out some task in the context of a software development process (Booch & Brown, 2003). CDEs serve as the meeting point not only for the developers of the community, but also for the users, who play an important role in open source software development.

The philosophy behind open source communities is founded on peer-to-peer collaboration and delegating tasks for other developers to provide input at will. They are based on meritocracy, where it is the more active or capable contributors who act as coordinators or leaders, since there are no project managers as in a software development company. Therefore, the contributors to a community are motivated not only by the desire to produce a certain functionality or the idea that the source code and applications should be open, but also by the recognition of their achievements by the community or the intellectual stimulus.

The collaborators in an open source community are referred to as hackers, namely people with strong computer skills who try to reach a goal by enhancing existing code or resources. This idea of enhancing source code or programs is closely knit with the beginning of the open source movement, because open source communities grew up around an existing program or solution, either by enhancing free software or by an open software release of an application (Raymond, 1999). In any case, the community starts working with an open release that has to be attractive and promising enough to encourage new developers to join the community.

Finally, the influence of the user's opinions in the software development process is a key feature in open source communities, in which the barriers between developers and users are generally quite low (Kim, 2003) and the communication channels between both groups are easily accessed. Moreover, users can easily become developers due to the inherent organization to open source communities, which enhances the community knowledge about the users' needs.

PATTERNS OF COLLABORATION FOR SUCCESSFUL OPEN SOURCE COMMUNITIES

Some of the patterns of collaboration underlying the organizational processes performed by open source communities are not specific to software development. These patterns represent knowledge blocks for building and improving successful communities as suggested by Alexander (1979). The following are prominent examples of patterns that have been identified in the most successful communities such as those analyzed in Kim (2003):

- **Evolve the community:** Bearing in mind the ideas explained in Raymond (1999), we conclude that designing an organizational structure for what might be rather than what is is likely to hinder not boost the project. Most successful communities allow an organizational scheme and processes to emerge over time rather than attempting to impose any structure. This will help minimize the risk of creating unnecessary organizational overheads with no immediate benefits. Community processes are lightweight, and tend to emerge in response to changing conditions.
- **Co-evolution:** Co-evolution is a term coined by Engelbart (1992) to describe how tools and their

users symbiotically influence each other's evolution. We use it in this context to describe how users and other community roles (and products) influence each other's evolution. The roles that users play in the software development process are not yet well understood. They can play a much more significant role than simply reporting bugs or evangelizing the projects. Users and developers are usually part of the same community. They interact mutually and many users become active members of the community by answering other users' questions or even becoming developers. An important research goal is to identify what those roles are and how they affect the overall software development process and community dynamics (Hippler, 2005).

- **Forking:** Forking has been coined to describe how small groups of people form new communities around already existing open source projects, how they lead those projects in new directions, and how the knowledge moves from one to another. How do the macro processes of the open source community as a whole affect the process within individual projects? (Brown Duguid, 2000) have proposed that Silicon Valley is particularly conducive to innovation because there are networks of knowledge transfer that transcend organizational boundaries. Open source communities seem to exhibit similar traits.
- **Lead by example:** The more active and visible the leader is within the community, the more active the community will become (Kim, 2003). Leading by example is especially crucial for open source communities, because a lot of their participants are volunteers.

PIONEER OPEN SOURCE-BASED COMMUNITIES

The software development industry has clearly undergone a change of paradigm due to the eruption of the open source phenomenon (Ghosh, 2002; MIT F/OSR, 2006). The features distinguishing open source from proprietary software go beyond the merely technical points and stretch to philosophical viewpoints, new economic rules, and different market models (Wynants & Cornelis, 2005). It also brings with it new development models, whose potential for success is well tried and tested and which differ from the classical methodologies on several points. The chief feature of this new approach is that development is network focused enabling people who are geographically far apart to collaborate, using Internet to communicate with each other and coordinate their activities to form what are known as open source communities. The ultimate goal is to promote both development and the use of open source software, and one way to do this is to provide tools and resources to enable communication, cooperation, and coordination between developers and users.

There are prominent examples of open source communities that have been paradigmatic because they have achieved such magnificent results. They serve as an example and case study of the best practices and characteristics that should be adopted by any newborn community, regardless of whether it is based on software or any type of source code.

These communities grew up around existing software that was either already open source or commercial software released as open software. The differences between the two models are very important during the community building process. However, they are not decisive characteristics for the success of the project, since both project types have proven to be valid. Other community features, such as the community government or the contribution policy, are also very important for encouraging contributions and, therefore, for the successful outcome of the project.

Apache is being developed by the Apache Software Foundation (ASF). ASF began in 1995 as a combined effort to coordinate existing fixes to the httpd program by the National Center for Supercomputing Applications, its own board of governors and a limited group of committers. All of them are elected from ASF members on their merits. See Mockus et al. (2005) for a detailed case study of the Apache project.

GNU/Linux is the largest open source development project in terms of number of developers (Hippler, 2005), and has from 7 to 21 million users worldwide with a 200% annual growth rate (Lerner, & Tirole, 2005). The project was started up by Torvalds, who began to build a free Posix-compatible operating system in 1991, being helped by the GNU project, which had been started by Stallman, along with the Free Software Movement (Kuhn, 2001). It has received thousands of contributions since then and has become an open source alternative to commercial proprietary operating systems like Microsoft Windows. A great many business opportunities have been created around GNU/Linux, and thousands of companies have made a profit by enhancing and retailing their own distributions, providing technical support, training, etc.

Mozilla was born out of a commercial product, Netscape, whose source code was released by the company under the same name to ensure the product's survival. The Mozilla Foundation was originally funded by AOL until it became self-supporting. The need for economic support during the early stages of a community's foundation is a common issue among communities with commercial roots. The process of gaining commitment access is based on merits and the candidate's worth, as well as on support from another committer. This point encourages the relations between developers and draws the community together. See Mockus et al. (2005) for a detailed case study of the Mozilla project.

MySQL is a paradigmatic example of the "dual licensing" business model. The program is distributed under both a commercial license for proprietary use and GPL license for free distribution. In this case, the commercial company, MySQL AB, was founded by the developers themselves and was followed by the community. MySQL has also demonstrated the economic viability of open source as a business model.

It is also worth mentioning a group of communities working on projects that are not necessarily software based. They promote the dissemination of knowledge as collaborations, ideas, courses, manuals, discussions, and almost any form of information that can be distributed on the web. The spirit, ideology and the idea of improving something by making it available to everyone is the same as in open source software and can benefit from the same type of tools, although it has other needs. This is the case of the *Berlios* project, funded by the German government to support the building of information and documentation management communities for projects related to open source software, and the communities around several projects hosted at OurProject.org, whose goal is to exchange ideas and jointly solve problems.

CONCLUSION

This article has presented the concept of open source community as one of the most successful and least understood examples of high-performance collaboration and community building on the Internet today. We have stressed that other types of collaboration-intensive communities could benefit enormously from an understanding of what open source communities are, how they work and what patterns of collaboration underlie their organizational processes. In examining these questions, the article has discussed existing, relevant research and has presented working open source communities, stressing how collaboration works within their successful projects.

REFERENCES

Alexander, C. (1979). *Timeless way of building*. New York: Oxford University Press.

Booch, G., & Brown, A. W. (2003). Collaborative development environments. *Advances in Computers, 59*. Academic Press.

Brown, J. S., & Duguid, P. (2000). Mysteries of the region: Knowledge dynamics in Silicon Valley. In C. Lee, W. F. Miller, M. Hancock & H. S. Rowen (Eds.), *The Silicon Valley edge*. Stanford, CA: Stanford University Press.

Engelbart, D. C. (1992). *Toward high-performance organizations: A strategic role for groupware*. Bootstrap Institute. Retrieved from http://www.bootstrap.org/augdocs/augment-132811.htm

Ghosh, R. A. (2002). *Free/libre and open source software: Survey and study final report*. Berlin, Germany: International Institute of Infonomics, University of Maastricht, The Netherlands and Berlecon Research GmbH. Retrieved from http://www.infonomics.nl/FLOSS/report/

Hippler, E. Von (2005). *Democratizing innovation*. Cambridge, MA: MIT Press.

Höhn, S., & Herr, G. (2004). *Open source: Open for business*. Leading Edge Forum. USA: Computer Science Corp.

Kim, E. E. (2003). *An introduction to open source communities*. Blue Oxen Associates, MIT's Free / Open Source Research Community. Retrieved from http://opensource.mit.edu/papers/blueoxen.pdf

Kuhn, B. M. (2001). *The GNU general public license protects software freedoms*. (Press Release). Boston, MA: Free Software Foundation. Retrieved from http://www.gnu.org/press/2001-05-04-GPL.pdf

Lerner, J., & Tirole, J. (2005). Economic perspectives on open source. In J. Feller et al. (Eds.), *Perspectives on free and open source software* (pp. 47-78). Cambridge, MA: MIT Press.

MIT's Free Open Source Research Community. Retrieved from http://opensource.mit.edu/

Mockus, A., Fielding, R. T., & Herbsleb, J. (2005). Two case studies of open source software development: Apache and Mozilla. In J. Feller et al. (Eds.), *Perspectives on free and open source software* (pp 163-210). Cambridge, MA: MIT Press.

Raymond, E. S. (1999). *The cathedral and the bazaar: Musings on linux and open source from an accidental revolutionary.* Sebastopol, CA: O'Reilly and Associates.

Shah, S. K. (2005). Open beyond software. In C. Cooper, & Stone, M. (Ed.), *Open sources 2.0.* Sebastopol, CA: O'Reilly Media.

Wynants, M., & Cornelis, J. (2005). *How open is the future? Economic, social, and cultural scenarios inspired by free and open source software.* Brussels, Belgium: VUB Brussels University Press.

KEY TERMS

Co-Evolution: This term describes how tools and their users symbiotically influence each other's evolution. Used in the community context to describe how users and other community roles (and products) influence each other's evolution.

Collaborative Development Environment: A virtual space wherein all the stakeholders of a project, even if separated by time or distance, may negotiate, communicate, coordinate, brainstorm, discuss, share knowledge, and liaise to carry out some task, most often to create an executable deliverable and its supporting artifacts, holistically integrating multiple collaborative tools and resources.

Community: An amalgamation of people with related interests. Intent, belief, resources, preferences, needs, goals, and a multitude of other conditions may be present and common, affecting the degree of adhesion within the group. Communities may meet to share information, to participate in shared projects, or to complete group tasks. What most characterizes a community is the pursuit of a common productive goal and sharing interaction in many ways.

Hacker: In the computing community, a skilled, experienced or even a wizard software developer. Also a person who creates and modifies computer software or hardware. Used in the media and popularly to mean computer and network security expert, this term has nothing to do with the *hackers* that contribute to the OSCs.

Meritocracy: A system of government based on rule by ability (merit) rather than by wealth, race, or other determinants of social position. Meritocratic governments, organizations, and communities stress talent, formal education, and competence, rather than existing differences such as social class, ethnicity, or sex.

Open Source Community: A loosely organized, ad-hoc community of contributors from all over the world who share an interest in meeting a common need, ranging from minor projects to huge developments, and carry it out using a high-performance collaborative development environment, allowing the organizational scheme and processes to emerge over time. The concept represents one of the most successful examples of high-performance collaboration and community building on the Internet.

Open Source: Describes practices in production and development that promote access to the end product's sources and allow for the concurrent use of different agendas and approaches in production. Some consider it as a philosophy, and others consider it as a pragmatic methodology. Open source has come to represent much more than software whose source code may be freely modified and redistributed with few restrictions imposed by the terms of its distribution license. Information, documentation, and other "sources" generally related to innovation and knowledge build and share processes, tend to fall under the open source umbrella.