Evaluating Software Engineering Processes in Commercial and Community Open Source Projects

Anthony I. Wasserman
Carnegie Mellon West
Moffett Field, CA 94035 USA
tonyw@west.cmu.edu

Eugenio Capra
Politecnico di Milano
Department of Electronics and Information
20133 Milan, Italy
capra@elet.polimi.it

Abstract

We describe a current study for obtaining information about management of commercial and community open source projects. We have interviewed and surveyed leaders of more than 75 open source projects. Questions focused on the styles of leadership and communication, with a technical focus on testing and quality assurance processes. This paper describes the methods used to identify and contact subjects, as well as the questions posed and some results.

1. Introduction

From an informal look at a cross-section of open source projects on a repository such as SourceForge (http://www.sf.net), one can see that there is a wide variation in the extent to which these projects follow generally accepted software engineering practices. Smaller projects, in particular, tend to be informally managed, while larger projects with a large user community achieve a high level of quality that results from a well-managed project with careful planning, development, and review.

Formal studies of widely used open source projects, such as those conducted by Reasoning Systems on Apache, Tomcat, and MySQL (downloadable from [1]), show that the quality of those projects compares favorably with that of widely used commercial (closed source) software products (see also [2]).

In this paper, we present some observations resulting from a series of interviews and surveys that we have done with the leaders of more than 75 software projects, ranging from closed products and services through commercial open source and community-based open source projects.

2. The Survey

A key goal of our project was to gain a better understanding of how commercial open source companies (e.g., MySQL AB, SugarCRM, and others) compared to closed source commercial products and to community-driven open source projects in how they manage the software development process. We had the informal sense beforehand that established software vendors (such as Oracle or IBM Rational) had well defined product development processes, and that these processes were largely followed, since the entire development team shared the goal of creating a commercially successful product and contributing to the earnings of the company. Similarly, we had the sense that community open source projects were more loosely organized, with a group of geographically dispersed volunteers being less likely and less willing to adhere to systematic development methodologies (see also [3] and [4]). We anticipated that commercial open source companies would have some characteristics of each; since many of them are startups, we expected that they would focus on building their businesses, perhaps forgoing a well-defined software process. While many of our assumptions proved to be correct, we were occasionally surprised by the results.

We prepared a questionnaire that addressed four areas of the development process:

1) managerial style;
2) collaboration and communication;
3) overall QA process, and;
4) testing techniques and tools

The questionnaire was prepared after some preliminary interviews used to focus the research, and was partially inspired by Fogel [5]. The Appendix contains a slightly edited version of the questionnaire.

Where feasible, we met project leaders in person and went through the questions. In other situations, we asked people to complete an online survey with these questions. Most responses from commercial open source companies were obtained through face-to-face interviews, while most responses from community open source projects came from the online survey. For the community projects, we selected well established projects with an active community, which helped us to obtain a good response rate. For projects hosted on SourceForge, we were also able to consider the number of downloads as a measure of activity (see [6]).

In general, we found it quite easy to make contact with the key people in the various companies and projects. While some of these contacts came through personal networking, the nature of open source projects and communities makes it possible to find and contact project leaders much more easily than with traditional software vendors. They perceived that the results of our survey would be valuable to them, which made them willing to spend 30-60 minutes answering the questions.

Table 1 presents some statistics about our sample.

Table 1 – Statistics about the sample.

Parameter	Commercial	Community
Number of projects [n.]	15	61
Number of personal interviews conducted	17	8
Average size [KLOC]	871	829
Average number of active developers	38	95

3. Preliminary Findings

While our survey and interview activities are not yet complete, we have been able to make some observations and draw some conclusions from the material we have gathered, which we present in the following subsections.

3.1. A company is a company, not a project

While the notion of "commercial open source" is gaining momentum, with MySQL AB among the pioneers of this concept, it's important to observe that there is a company behind each commercial open source project. As with any other commercial business, these companies are intended to provide a return to their owners. That goal affects the business models that they choose, as well as the services that they provide.

Commercial open source products offer numerous advantages for both commercial customers and those who wish to use the software at no charge under a non-commercial license. For the latter group, an open source version of the software is freely available, typically through a download. While the company may also offer a more advanced version of the product in commercial form only, the open source version is usually quite similar to the commercial version, simply because it is inefficient for the company to maintain more than one code base.

Basic support for a commercial open source product is typically through online documentation and discussion boards. Anyone can purchase commercial support for the open source project directly from the company; as with support services for closed source software products, such support typically includes telephone and email support, as well as notification of new releases. In addition, there are published books and third-party support and training for many open source products. Many corporate IT organizations require a support contract for any critical software components that they use, so a commercial open source company can build a successful business based on selling support services.

The paying customers of commercial open source companies expect that that the vendor will maintain the software and will develop a roadmap describing planned enhancements for successive releases of the software, exactly as is done in traditional software companies. Similarly, the paying customers expect that the company will have quality assurance and testing processes in place for their software, so that the resulting products are of good quality. These aspects of commercial open source often create a strong contrast with some community open source projects, which lack a business orientation. Commercial open source projects have an open source *distribution* model, but not an open source *development* model. According to the first results of our survey, on average 92% of the code is developed internally, and this percentage goes up to 100% in some cases. Moreover, in most cases these external developers do not have commit right, as the company has to keep control over its product.

These requirements imposed by the paying customers of a commercial open source product bring benefits to all users, not just those who are paying for

the associated services. For example, everyone benefits from the company's testing and QA efforts. According to our survey, in almost 40% of commercial open source projects, testing is planned even before implementation starts. In addition, 65% of them adopt regression testing tools, whereas those activities happen in only 10% and 30% of community projects, respectively. Also, release plans and schedules are more reliable, since the work is almost entirely done by people hired by the company.

Now that there are some successful commercial open source companies, it seems likely that there will be more companies that follow this path and that sources of commercial support will emerge for popular community-based open source projects. Indeed, this has already occurred for open source projects, such as Apache's HTTP server and PostgreSQL. In these situations, anyone can freely use the open source project and can acquire support for the software from a company that is in the business of supporting that project.

3.2 Companies support community open source projects

While there are thousands of community open source projects listed on various repositories, such as SourceForge and GForge, only a very small percentage of them have an active community that is dispersed well beyond the groups represented by the project leaders. Such projects gain widespread use by being included in various Linux distributions, e.g., OpenSUSE or Fedora, or in integrated stacks, such as those offered (and supported) by companies such as SpikeSource and OpenLogic.

In addition to broad distribution, a successful community open source project displays other similar qualities. First, there is a strong leadership group that serves as "champion" for the project, with limited turnover within this group. We found that approximately 65% of the community projects in our sample have a structured organization and governance rules, while the remaining 35% are not managed at all, and issues are only informally discussed in mailing lists. Second, the project follows numerous "best practices" for software development, including issue tracking, configuration management, subdivision of large projects, and effective collaboration among team members. 90% of community projects in our sample adopt a version control system and a 80% adopt a bug tracking system, 80% use mailing lists to communicate, and about 55% use IRC channels. It is interesting to note that more than a third of the projects have regular physical meetings of active developers. In some cases, this happens because many of the contributors to the project work for the same company and work on the project as their primary job responsibility. Almost 50% of the code of the community open source applications in our sample was developed by people hired by companies or institutions, but in some cases this value can go up to 95%. As examples of this situation, Sun Microsystems supports large teams to work on the OpenOffice.org suite, the NetBeans programming environment, the JavaDB database system, and the Open Solaris operating system, among others. Similarly, IBM has been a patron to the Eclipse programming environment and related framework, as well as Derby, Geronimo, and numerous other projects. Even companies such as Oracle, Microsoft, and BEA long associated with proprietary software, have substantial participation in various open source projects. While Sun and IBM are the leaders in paying their employees to contribute to open source projects, they are far from alone.

This observation may come as a bit of a surprise to people who are unfamiliar with open source software. Many of these people, including some of those responsible for corporate software acquisition, have the impression of free and open source software being written by people without professional software development experience. While that is true of many projects on the various open source repositories, it is rarely true of the most widely used projects, even those without official corporate backing.

Beyond corporate contribution to individual open source projects, many companies provide financial support to non-profit open source activities. (Some companies do both.) Apache and Eclipse, for example, have non-profit foundations that are largely funded by corporate contributions. The same is true of industry groups such as the Linux Foundation. Google supports many open source projects through the Summer of Code initiative, which pays students to work with open source projects. Finally, many academic researchers in open source receive corporate support for their work.

One might ask why for-profit corporations would donate their people, their funds, and other resources to open source projects where there is no obvious immediate source of revenue. We identified several common reasons. First, this form of contribution to the community often accelerates the development of technology by getting more people to contribute to a project. Next, a company can have a significant influence in community acceptance (and standardization) of software components if they are open to all. Sun Microsystems, for example, has made Solaris and Java open source, and has opened StarOffice into Open Office.org with the goal of

making it a stronger competitor to the proprietary Microsoft Office.

Next, corporate participation in open source projects gives companies a good mechanism for cooperating with other companies, particularly in the development of standards (formal and *ad hoc*). Acceptance of such standards often allows a company to build added value products and services.

Third, companies can get very early feedback from the community associated with an open source project. That feedback, including informal testing, contributes resources to the project and reduces the cost of developing the software exclusively with internal resources.

Fourth, companies can leverage open source projects as a broad marketing platform. Although the software itself is free, they can sell support, training, consultancy and personalization services.

Finally, many companies find that participation in the open source community is beneficial to their image, with the added benefit of giving them the ability to recruit top open source developers to work for them. The company can see the quality of a developer's code, as well as observe that person's participation as a member of a project, thereby providing additional insight into how that person might fit into the company.

4. Summary

These findings are still preliminary, and we are planning to validate them through additional interviews and surveys. Furthermore, we are examining some additional areas, including the types of communication and collaboration within commercial and community open source projects, and hope to extend our findings to these areas.

In the future, we also expect to look more closely at some of the business issues for commercial open source projects and for company participation in community open source projects. Closely related to this issue is the question of how community open source projects can best receive widespread consideration and adoption.

5. Acknowledgments

We are grateful to the project managers who provided data to us and who were willing to discuss with us the issues we are investigating. We also thank Professor Chiara Francalanci and Francesco Merlo (Politecnico di Milano) for their support and advice, as well as the anonymous reviewers who made valuable suggestions.

6. References

[1] www.reasoning.com/downloads.html

[2] "Study: Open source produces best results", downloaded from www.builderau.com.au/program/unix/soa/Study_Open_source_produces_best_results/0,339024638,320272271,00.htm, January 2007.

[3] S. Slaughter, J. Roberts, and I. Hann, "Communication Networks in an Open Source Software Project", Proc. of 2nd Conference on Open Source Systems, Italy, Jun 2006.

[4] S. Slaughter, J. Roberts, and I. Hann, "Motivations, Participation and Performance in Open Source Software Development", Management Science, (forthcoming).

[5] K. Fogel, "Producing Open Source Software", O'Reilly, Sebastopol (CA), 2006.

[6] J. Howison, and K. Crowston, "The Perils and Pitfalls of Mining SourceForge", *Proc. of 1st International Workshop on Mining Software Repositories (MSR 2004)*, 2004, pp. 7-12

Appendix: The Questionnaire

(Note that we use semicolons instead of new lines in this Appendix as a space-saving measure.)

Project/ Company

Size of the application (SLOC)

Number of hired committers

Number of external committers

How many external developers (belonging to Open Source community) have commit right to your project?

What percentage of your project is committed by independent and non-hired developers?

How is commit right achieved? How is code reviewed?

Explain how internal and external developers achieve the right to commit to your application.

What is the managerial style of your project? (select one)
Democracy/ Meritocracy: non structured organization; Meritocracy, but with a structured governance organization; Benevolent dictatorship; Mainly led by a company

Which communication tools do you use for your project? (select all that apply)

Forum; Blog; Wiki; Mailing list; Bug tracking; CVS/ Subversion; New feature request; Real time chat/ Instant messaging; Phone conferences; Physical meeting; Shared tasks list; Video chat

How often do you have lunch/ dinner with the developers of your project?
Often (at least once a month); Once in a while; Never

How much are you satisfied with the quality of your code? (select one)
5) Very satisfied 4) Satisfied 3) Quite satisfied 2) Something can be improved 1) Several things can be improved

Please indicate your feelings about the number of bugs and the general quality of the code.

Average number of bugs per new KLOC

How much are you satisfied with the responsiveness of your community? (select one)
5) Very satisfied: all urgent bugs fixed within days 4) Satisfied 3) Quite satisfied 2) It can be sligthly improved: we have been focusing on other issues 1) Not really satisfied: we have been focusing on other issues

Please consider the time on average required to fix bugs INTERNALLY, not the time elapsed to release a new public version of the application.

How often do you release new versions or service packs?

Do you wait until a certain amount of new bugs are detected and fixed or do you do that on a fixed schedule?

Do you backport bug fixing to previous releases?
Yes, always; Yes, but with some limitations; No

How much does the project culture value predictability?
Deadlines are usually respected; Deadlines are usually met with up to 1 week of delay; Deadlines are usually met with more than 1 week of delay; Deadlines are not respected in most cases/ We do not set deadlines

How much are you satisfied of the overall productivity of your community (LOC/ day)?
5) Very satisfied 4) Satisfied 3) Quite satisfied 2) It can be slightly improved 1) It can be improved

Do you have a Software Quality Assurance plan? (select one)
Yes, written and approved by team members; Yes, but it is not written; No

How much is quality important in the culture of your project? What do you do to help members of the project to improve the quality of their job?

Which percentage of total development man days is dedicated to testing?

Which testing practices do you adopt on a regular basis? (select all that apply)
Defect tracking;Unit testing; Integration testing; System testing; Outside beta testing; Daily "smoke test"; Regression testing tools; Test suites/ scenarios; User Interface Prototypes; Code coverage/ source code tracing tools; Other

Which tools do you use? (select all that apply)
Abbot; Cactus; Clover; Eclipse plugins; EMMA; FindBugs; GJTester; iTracker; Java Development Tools; Java Tool Suite; JBench; JCove; JLint; JProbe; JTest; JUnit; Koalog Code Coverage; OpenSTA; OptimizeIt; PMD; soapUI; TCAT; Other In house tools

What other testing techniques and tools do you use?

Please list any technique or tool that you use and that is not mentioned in the previous questions.

When does testing or test planning begin? (select one)
Before implementation starts; Just after implementation has started; During the implementation; Just before the application is released; After bugs are reported; Never

Who is responsible for testing?
Every developer tests his code; Peer review/ Technical review sessions; People other than developers are charged with testing and quality assurance; Users.