Software Engineering Lessons from Open Source Projects

Position Paper for the 1st Workshop on Open Source Software, ICSE-2001

Jai Asundi
Department of Engineering and Public Policy
Carnegie Mellon University
asundi@andrew.cmu.edu

Abstract

The Open Source form of software development has captured the attention of academics and software practitioners alike. Though, this 'phenomenon' has been touted by some to be how all software will eventually be developed, many are critical about how far this form of organization will be successful. It is very likely that both the traditional form as well as the Open Source form of organization for software projects may co-exist. The position taken in this paper is that commercial projects can learn important lessons from Open Source projects. This paper tries to capture the elements of Open Source projects that can be applied to traditional commercial projects and the means by which further information can be gathered.

1. Introduction and Overview

Open Source(OS) software has captured the attention of the entire software community. Many perceive it to be a movement against a common enemy (read Microsoft and other large software houses). However, the real battle is for the increasing need for high quality and reliable software in spite of the increasing complexity of the applications. Though it is highly unlikely that the OS form of software development will completely replace the traditional commercial software development practice, there are important lessons that can be borrowed and applied from each other.

Very little systematic work has been done to analyze and measure the OS form of software development. There are case-studies explaining the software architecture of OS products[2][5] and how architecture recovery can be applied[3]. In terms of studying the organization of OS projects preliminary work [1][8] has shown how publicly available archived information can be used to study and test hypotheses in OS projects.

In this paper I shall try to highlight the important lessons that can be gathered from OS projects that could have interesting implications for commercial projects.

2. Lessons from Open Source projects

Organization of personnel

The Apache project[1] has shown us that 85% of the enhancements to the software were done by the top 15 people[8]. These top 15 are more or less the members of the core-group. The numbers are not very different for other OS projects[4]. We can thus see the parallels for a commercial project where the core design team would be expected to do most of the changes. What we can learn further is how the various tasks are distributed amongst this team. Since OS projects are self formed and responsibilities are assumed rather than assigned, based on patch submit patterns we can infer about how developers like to perceive their roles in a software project. Mirroring this structure in a commercial project could lead to better management of the product.

Informal Communication

Studies in distributed software development have shown that *informal* communication amongst personnel is vital for the coordination of the effort[6]. A glance at the mail archives for various OS projects shows us that there is extensive informal communication between the lead-developers, other developers and users. Some projects also run chat sites once a week to discuss issues. This in spite of the fact that the developers are from differing cultures and in different time zones. Commercial projects could increase their level interaction amongst the developers (either collocated or distributed) by setting up discussion boards and run chat sites to discuss technical issues so that there is no confusion regarding accents or tone of voice. Also, other interested developers could "tune-in", thus reducing the confusion due to learning about issues second hand or through a much delayed, formally stated memo.

Improved customer support

Many users of software products will agree that most commercial software organizations fail miserably when it comes to supporting their software product. A study[7] shows that OS projects has a good record with user support due to a larger number of people willing to offer information about the product. A large fraction of the responses seem to be simple and easily answered - done usually by a very small fraction of people. Thus, the lesson for commercial projects is that increased user-user as well as user-developer interaction could lead to improved support as well as better and more meaningful bug reports.

1.http://www.apache.org/

3. Differences from Commercial Projects

There are a number of acknowledged differences between OS projects and commercial projects that make the two difficult to compare.

Existing code with design completed

Typically, most OS projects begin with a piece of code and an already working piece of software. The design is more or less frozen and only under unique circumstances is the design redone or revisited.

Design is done by single person or two collocated persons

In commercial projects, the design is usually done by a group of people and in some cases people who are not even at the same location. This leads to larger problems of coordination at the most important phase of the development cycle.

4. Conclusions and Future Work

In spite of the difference in organization of OS projects, important lessons can be borrowed and applied to commercial software projects. The mailing list archives contain important information that can be culled to benefit traditional software projects. Improved communication channels and methods of increasing interaction between developers and users are few of the preliminary lessons that can be borrowed from OS projects. The next step would be gaining information for the identification of personnel for various responsibilities, allocation of tasks and structuring a software project.

Other future work will include the use of the mail archives to test various hypotheses in OS projects. This would include a study of the code review process as well as further analysis of the “many eyeballs” hypothesis.

5. References

[1] John Bley and Ashish Arora, How Many Eyeballs Are Enough?: Peer Review in Open Source Development, Workstudy Report, Carnegie Mellon University, 2000.

[2] Ivan Bowman, Richard Holt and Neil Brewster, Linux as a Case Study: Its Extracted Software Architecture, ICSE '99: International Conference on Software Engineering, Los Angeles, May 1999.

[3] Ivan Bowman and Richard Holt, Software Architecture Recovery Using Conway's Law, Proc. of CASCON'98, December 1998, pages 123-133.

[4] Rishab Ghosh and Vipul Prakash, The Orbiten Free Software Survey:May 2000, http://www.orbiten.org/,ofss/01.html, (accessed March 10th, 2001).

[5] Ahmed E. Hassan and Richard C. Holt, A Reference Architecture for Web Servers, WCRE 2000: Working Conference on Reverse Engineering, Brisbane, Australia, Nov 6, 2000.

[6] James Herbsleb and Rebecca Grinter, Splitting the Organization and Integrating the Code: Conway’s Law Revisited, Proceeding of the 21st International Conference on Software Engineering, 1999, Los Angeles, CA, 1999

[7] Karim Lakhani and Eric von Hippel, How Open Source software works: “Free” user-to-user assistance, MIT Sloan-School of Management Working Paper #4117, May, 2000

[8] Audris Mockus, Roy Fielding and James Herbsleb, A Case Study of Open Source Software Development: The Apache Server, Proceedings of the 22nd International Conference on Software Engineering, 2000, Limerick, Ireland, June 2000