Shared Assumption Concerning Technical Determination in Apache Web Server Developer Community

Juho Lindman
Helsinki School of Economics, Information Systems Science,
Runeberginkatu 22-24, 00101 Helsinki, juho.lindman@hse.fi,
WWW home page: http://www.hse.fi

Abstract. Our main finding is that OSS community seems to coordinate its activities by relying on technical determination. First, we review previous literature to understand OSS community coordination mechanisms. Then we empirically review OSS Apache Web Server community by using qualitative case study methods. Our data consist of developer list's email-discussions. Finally, we speculate that coordination rests on community's members' shared assumption concerning technical determination.

Keywords: Open Source Software, OSS, Coordination

1 Introduction

The Open Source Software (OSS) is a promising software development and distribution method. It is agreed that OSS can deliver business benefits [1]. There is, however, a gap in the previous research concerning on how community coordinates its efforts. If a company wants to adapt OSS processes, it has to understand what coordination mechanisms are used. Furthermore, those mechanisms are probably related to the community's shared understanding concerning OSS.

In this paper we outline mechanisms that are described in literature and pose question of how these mechanisms function in a software development community. Then we analyze a case community to find out can we find such mechanisms. Our research question is: is there a shared assumption concerning technical determination that enables coordination mechanisms?

2 Previous research

In OSS, there was originally a promoted understanding that if companies and communities would work together, everyone would be better off [2]. Recently, the

relation between OSS communities and companies has been called into question. Dahlander&Magnusson [3] describe these kinds of relationships and divide them into 1) symbiotic, 2) commensalistic, and 3) parasitic depending on who is gaining from the relationship. If companies want to adapt OSS processes, they first need to understand how the communities function and how their coordination mechanisms are rooted in OSS cultural dynamics. That is only the first step in implementation, but the required organizational and financial changes fall outside of this study [4-5].

Coordination is required in software development to create consistent software [6]. This is especially true for OSS, since source code is easier to access. Ad hoc approaches might work for small projects, but more mature projects need ways to guarantee internal and external consistency of the software [7]. Traditionally coordination is defined as coordination activities toward common goal [7]. Malone and Crowston [9] redefine coordination as "managing dependencies between activities". The latter seems to suit better for dispersed development activity, since it requires knowledge of dependencies, not knowledge of the activities [9].

Egyedi et al. [10] propose coordination mechanisms that coordinate the OSS development processes. They stress the importance of the 1) committee standardization, but focus more on: 2) bandwagon coordination, 3) regulatory coordination, 4) operational coordination, and 5) coordination by authority. Committee standardization means technical specifications for products in response to technical complexity [11]. Bandwagon mechanism means that popular OSS projects attract developers and set example. Regulatory mechanism means licenses, contracts and participatory agreements. Operational coordination mechanism means tools used in development. Coordination by authority means gatekeepers and controlling distribution contents.

Coordination mechanisms are built into OSS communities. OSS communities require high level of technical expertise [2] and build upon technical savvy "geek culture" [12]. An OSS development community can be viewed from organisation cultural perspective [13;15]. This culture consists of shared understanding how things are accomplished inside a certain developer community. OSS communities' self-understanding is affected by influential writings of Eric Raymond [2] and Richard Stallman [15].

Raymond [2] described a motivational and coordinative entity of OSS: a plausible promise. Plausible promise is required to start OSS development. This promise will then motivate, but also direct the development activity. The promise includes not only goal, but also a process of how to get there. This correspondence between the end-product and the organisation, Conway's law, was originally proposed nearly forty years ago [16]. The idea is that the software architecture mirrors the structure of the organisation that created it. Parnas [17] developed the idea further proposing that software modules should be regarded as job assignments not sub-programs. Ovaska [9] proposes that software architecture should be viewed as assignment communication tool used for coordinating interdependencies. This is in line with previous findings of Mockus [18]: interdependencies are controlled by limiting their number by using standard interfaces.

3 Method

We have outlined literature concerning coordination mechanisms and their relation to a technical subculture. In following sections we select one community and view how the coordination mechanisms function. Our research method is qualitative case study [19]. We do not seek statistical, but analytical generalizations. Therefore we need to focus only on one case community during quite limited time.

4 Data

We chose Case Apache Web Server (httpd) as our case community, because of its maturity. It has produced OSS code of quality comparable to the commercial actors of its domain and is currently the most used web server software. Apache has been researched before [10;18;20] since it is the first OSS "killer application". Originally the community was launched in 1995 based on the work of NSCA on httpd-program.

Our main data source is the Apache Web Server developer mailing-list (dev@httpd.apache.org) that can be found archived in the internet at http://mail-archives.apache.org/mod_mbox/httpd-dev. The data includes all emails (768) sent to the list between 1.1.2006-28.2.2006. We have also monitored other Apache foundations relevant email-lists that have something to do with httpd development. We have monitored the relevant Newsgroups archives and went trough the broad material offered in the portal web pages. The choice of data assumes that community's coordination happens on public email-lists. This choice seems justified as Apache promotes openness in its own action [21].

5 Case

We were interested in coordination mechanisms and their assumptions. We started out with Egyedi's [10] proposed five coordination mechanisms. Most of the emails discuss the technical specifications of software. The focus of this study is not about the functionalities and their quality, but on how coordination takes place. The discussion in the lists was very technical. Normally someone started a thread by posing a technical problem or question. Then the people on the list tried to tackle the issue and usually asked some additional information about the problem. Usually, a solution was offered after some discussion. Sometimes the question revealed that some additions needed to be made to the documentation or software. The overall development process seemed to follow the process proposed by Sharma [22].

From the five mechanisms most common was technical specification. This activity was not very formal, but followed problem-answer model. This mechanism was used more than all the other mechanisms combined. We also found signs of

bandwagon mechanism, operational coordination mechanism, regulatory coordination and coordination by authority to function in case community.

Bandwagon mechanism was quite clear: the community self-image and popularity was reinforced in the official communication, but also in development email-discussion. The community was self-conscious and also able to attract interest and new members. Regulatory coordination was used. Higher involvement members were required to sign contracts. There were also positions of power, or hats [22] that could in some circumstances regulate action. We did not find in practice any clear instances of this kind of exercise of power in our data. The only seemingly accepted superiority was of technical nature. Operational coordination mechanisms were used. Concurrent Versioning Systems (CVS) coordinate activity as do status report emails that were automatically sent to the mailing list. Coordination mechanisms by authority were used. A release manager is chosen before major releases solve the open issues or to postpone them to the next version. Another authority coordination mechanism was that of gatekeepers. It takes a lot of technical skills to contribute to main modules of httpd. This gives those individuals that are capable authority over those that are not, based on meritocracy [22]. This authority is tied to individual, not organization.

We also found coordination mechanisms that do not fit very well to the proposed research framework. Those were: bug report control, voting and intentional consensus building. Bug fixing is a control mechanism, because it gives very important input and sets the agenda question to be answered. Bug reporting was made quite difficult because it required significant time and know-how even to submit a bug report. Furthermore, it seemed that given bug reports were not of very high priority [23]. This combined to the meritocracy will lead to a situation where outside input is quite low. Voting was another coordination mechanism. Voting can also be seen as a way to force decisions. In this case, it seemed that it was used rather as a way to test consensus. In the community, there was a very strong belief that voting should not be used as a coercive tool, but rather to see who has different opinion and why [24]. Third coordination mechanism that does not fall under previous categories was conscious community building. It seems that the active developer community is quite small. The rules of engagement seem to be written in the netiquette: rules of not only about form of communication, but also of content. At the same time, this mechanism enhanced coordination inside the email-dependant community by limiting the tone and content of messages. Another issue of community building was that the architecture was made in a way that the core httpd-module was not altered, but developments were rather redirected to more peripheral modules of the software. This creates consistency, but also limits the community size and offers possibility to re-allocate resources.

6 Results and discussion

Our research question was: Is there a shared assumption concerning technical determination that enables coordination mechanisms? All the five coordination mechanisms proposed by Egyedi [10] were used: committee standardisation, bandwagon coordination, operational coordination, regulatory coordination and coordination by authority. We also found three other possible coordination mechanisms: bug report control, voting and intentional consensus building.

All of these mechanisms seem to assume certain technical basis of how OSS communities function. We call this shared assumption technical determination. It means a belief that posed problems have technical solution, which will be found when the best developers focus on the problem - if not in the next version, then later. This also means that the software is never ready, but always developing.

This assumption enables the coordination mechanisms as follows: standardisation is required to reduce technical complexity by technical specifications Thus it is a technical solution. Bandwagon mechanism draws curious users and ambitious developers to successful communities. It rests on the belief in open source software is a "geek culture" of meritocratic technicians that are able to show their skills by developing software. Operational coordination mechanisms are technical solutions to problems related to learning, joint problem solving and division of work. Regulatory coordination is an agreement on how and which developer to credit for the contribution to the community. Coordination based on authority is required to smooth the releases of the continuously developing software and to make software available to larger population. Bug report coordination assumes that software is always developing and perceived difficulty of bug reporting hints that bug reports should be of high technical quality. Voting coordination tests consensus of the developers and strengthens the meritocracy by allowing merited programmers to vote. Community building coordination mechanism also strengthens the meritocracy and also assumes that the development effort of the software will continue for a long time.

7 Summary

We have outlined relevant literature of coordination, coordination mechanisms and organisational culture of a subgroup. Then we have selected a case community Apache Web Server and identified five mechanisms previously described in the literature and identified three other possible coordinating mechanisms. We have speculated that all these mechanisms rest on shared assumption of technical determination. These findings call for more research in the area of technical problem solving nature of OSS development.

References

1. B. Fitzgerald, The Transformation of Open Source Software, *MIS Quarterly,* 30 (4), 587-598 (2006).
2. E. Raymond, *The Cathedral and the Bazaar: Musings on Linux and Open Source by an Accidental Revolutionary* (O'Reilly Sebastopol, 1999).
3. L. Dahlander & M. Magnusson, Relationship between open source software companies and communities: Observations from Nordic firms, *Research Policy,* 34, 481-493 (2005).
4. J. Feller& B. Fitzgerald, *Understanding Open Source Software Development* (Addison Wesley, Boston, 2002).
5. M. Fink, *The Business and Economics of Linux and Open Source* (PHPTR, Prentice Hall, 2003).
6. E. Kraut & L. Streeter, Coordination in Software Development, *Communications of the ACM*, 38 (3), 69-81 (1995).
7. B. Curtis., H. Krasner, N. Iscoe, A Field Study of the software design process for large systems, *Communications of the ACM,* 31 (11), 1286-1287, (1988).
8. T.W. Malone, & K. Crowston, The interdisciplinary study of coordination, *Computing Surveys,* 26 (1), 87-119, (1994).
9. P. Ovaska, M. Rossi & P. Marttiin, Architecture as a coordination tool in multi-site software development, *Software Process: Improvement and Practice* , 8 (4), 233 – 247, (2003).
10. T. Egyedi & R.W. Van de Joode, Standardization and Other Coordination Mechanisms in OSS, *International Journal of IT Standards & Standardisation research*, 2(2), 1-17, (2004).
11. S.K. Schmidt & R. Werle, *Co-ordinating Technology. Studies in the International Standardization of Telecommunications* (MIT Press, Cambridge Mass, 1998).
12. R. Pavlicek, *Embracing Insanity: Open Source Software Development* (SAMS, Indianapolis, 2000).
13. J. Martin, *Organizational Culture: Mapping the Terrain*, (Sage, Thousand Oaks, 2002).
14. H. Schein, *Organizational culture and leadership* (San Francisco, Jossy-Bass, 1992).
15. S. Williams, *Free as in Freedom: Richard Stallman's Crusade for Free Software.* (O'Reilly, Sebastopol, 2002).
16. M.E. Conway, How do committees invent? *Datamation,* 14 (4), 28-31, (1968).
17. D. L. Parnas, On the criteria to be used in decomposing systems into modules, *Communications of the ACM*, 15 (12), 1053-1058, (1972).
18. A. Mockus, R. Fielding, & J. Herbsleb, Two Case Studies on Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11 (3), 309-346, (2002).
19. R. K. Yin, *Case Study Research, Design and Methods* 2nd ed., (Sage, Newbury Park, 1994).
20. R. Fielding, Shared leadership in the Apache project. *Communications of the ACM.* 42 (4), 42-43, (1999).
21. Apache Web Server, (2006a), http://www.apache.org/ foundation/how-it-works.html.
22. S. Sharma, V. Sugumaran, & B. Rajagopalan, A Framework for Creating Hybrid Open Source Software Communities, *Informations Systems Journal,* vol. 12 (1), pp. 7-25, (2002).
23. Apache Web Server, (2006b), http://issues.apache.org/bugwritinghelp.html
24. Apache Web Server, (2006c), http://www.apache.org/foundation/voting.html