2005

Control in Open Source Software Development

Bo Xu
Texas Tech University, bo.xu@ttu.edu

Yan Xu
Texas Tech University, yan.xu@ttu.edu

Zhangxi Lin
Texas Tech University, zlin@ba.ttu.edu

Control in Open Source Software Development

Bo Xu
Rawls College of Business
Texas Tech University
bo.xu@ttu.edu

Yan Xu
Department of Computer Science
Texas Tech University
yan.xu@ttu.edu

Zhangxi Lin
Rawls College of Business
Texas Tech University
zlin@ba.ttu.edu

ABSTRACT

Open source software is a revolution in software development, and has great impact on the software industry today. It is significantly different from the traditional software development in organization, process and management. Control mechanisms are important to the success of collaborative work, especially for open source software development which is community-based and loosely structured in nature. This paper investigates the control modes in open source community based on control theories and finds that portfolios of outcome, clan and self-controls are used in open source community, and the control modes change with the evolution of an open source community.

Keywords

Open source software, control, community

INTRODUCTION

Open source software is a revolution in software development and represents a new mode of software distribution. Computer software is traditionally sold only as a finished product, or precompiled binary, with a license. Software users do not have access to the source code, bug fixes and support are completely dependent on the organization that sells the software. By contrast, open source software is software that is licensed to guarantee free access to the source code, so users who are technical inclined can fix bugs or make modifications by themselves (Bretthauer 2002). The past decade has seen a marked expansion in the open source software movement.

The open source initiative starts from the ideology that software should be free and open. The concept of copyleft is the core to open source software. To copyleft a program, the programmer, besides copyrighting the program to himself, also signs a General Public License (GPL) granting everyone the right to use, modify and distribute the program on the condition that the license also grants similar rights over the modifications he or she has made. Under this arrangement, everyone has free access to the program but it is protected from becoming someone's private intellectual property (Mustonen 2003). The widespread diffusion of Internet access in the early 1990s led to a dramatic acceleration of open source activity. The Internet-based community provides an environment for developers to collaborate online. Open source software development is community-based with open membership; Worker incentives and motivations shift from those of employees to those of volunteers, and unlike in a firm setting, there isn't an authority relationship to regulate the behavior of community members; The development team members are organizationally and geographically dispersed, linked through the Internet. Finally, the software-creation platform is based on a many-to-many, computer mediated communication technology (Lee and Cole 2003). These four characteristics make open source software different in organization structure from traditional software development.

Control mechanisms have been studied much in organization, and it has been demonstrated that control systems play an important role in the governance and management of information systems project (Henderson and Lee 1992; Kirsch 1996; Kirsch 1997). Control effectiveness is critical to the success of information systems development internally within organization (Nidumolu and Subramani 2003) and externally between alliances (Choudhury and Sabherwal 2003). The open

source software development is based on virtual community pf voluntary participants which are geographically dispersed. The motivations for open source project participation include economic incentives, job prospects, enjoyment, learning purpose, cooperation needs, open source ideology, and personal needs (Markus et al. 2000; Hars and Ou 2002; Hippel and Krogh 2003; Hann et al. 2004). Each member in an open source community may have different types of motivation for participation, so how to align their activities with the goal of the project is essential to the viability and final outcome of the project. Control system is an important component in organization and information systems development project to facilitate coordination, align member activities and guarantee the progress toward the common goals, they also make the systems development process more reliable, predictable and efficient. The open source community has showed us many successful software, among which the most well-known examples are Linux, Apache and Mozilla. However, few research has been conducted to understand how control is exerted in open source community, which may influence the project performance. To fill this gap, this paper investigates the control mechanism common in open source communities, and its dynamics with the evolution of an open source project.

THEORETICAL BACKGROUND

Information systems development is not just a technical process, but also a social process with multiple stakeholders. Exercising control is a powerful approach that project managers use to ensure progress by fusing together the complementary roles and capabilities of project participants, motivating individuals to work in accordance with organizational goals and objectives (Henderson and Lee 1992). According to Kirsch (1996, 1997), both formal and informal modes of control are used in information systems development projects. The formal control includes behavior and outcome control. In behavior control, specific rules and procedures are articulated, and controllers observe the behaviors of the controllees, who are rewarded based on the degree to which they follow the procedures (Eisenhardt 1985; Mahmood 1987). To implement outcome control, desired outcomes or goals are articulated, and controllees are rewarded for meeting the goals (Eisenhardt 1985; Snell 1992). Informal control is based on social and people strategies, and mainly includes clan and self controls. Clan control is implemented by promulgating common values, beliefs, and philosophy within a clan, which is defined as a group of individuals who are dependent on one another and who share a set of common goals (Ouchi 1980). For self control, an individual sets his own goals for a particular task, and then proceeds to self-monitor, self reward and self sanction, so it is a function of individual objectives, standards and intrinsic motivation (Manz et al. 1987).

According to Kirsch (1996, 1997), different portfolios of control modes are used in different development situations. Behavior control is implemented when appropriate behaviors are known or when controllee behaviors are observable to the controller; outcome control is implemented when outcomes are measurable and the controller is able to assess whether the target was reached; clan control is implemented when neither outcomes are measurable nor appropriate behaviors are known; the organizational and individual antecedents of self-control include task complexity, ambiguous performance evaluation, lack of rules and procedures for completing a task, and individual abilities. There have been some empirical studies about the use of control modes in traditional software development.

CONTROL IN OPEN SOURCE SOFTWARE COMMUNITY

Open Source Software vs. Traditional Software Development

Open source software development is significantly different from traditional software development in both organization and process. The traditional software development is conducted within formal organization, the relationship between developers and organizations are formally maintained through contract; while open source software development takes Internet-based community as its organization form, the developers are volunteers who are self-motivated, so it is loose-structured and self-organized (Lee and Cole 2003). The most generic form of the traditional software development life cycle (SDLC) includes the stages of planning, analysis, design and implementation. However, in the open source software development life cycle, the planning, analysis and design phases are largely conducted by the initial project founder, and are not part of the general development life cycle (Feller and Fitzgerald 2002). The open source software development life cycle is located primarily within the implementation phase of the traditional SDLC, and follows a spiral model. Jorgensen (2001) identified the main phases in the open source software development life cycle as code, review, pre-commit test, development release, parallel debugging and production release.

Roles in Open Source Community

Members in an open source software community play different roles. (1) the project leaders in charge of the project and maintain the versions, they are usually the initiators of the project; (2) the core developers perform most of the work in software development and maintenance, their tasks include creating patches, adding features, and fixing bugs; (3) the peripheral developers contribute codes, report bugs and suggest features; (4) the pure end users only utilize the open source software, but do not contribute to the development. They are important to the software popularity.

In an open source community, the project leaders or administrators play the most important role in control and coordination. Their ability as controllers greatly affects the project performance and the viability of the community.

Control Modes in Open Source Community

Among the four major types of control (behavior, outcome, clan, self), behavior control is infeasible in open source software development since open source software development is an incremental innovation process which is based on participants' autonomy and initiatives, and there is no pre-defined procedures in the process. Nidumolu and Subramani (2003) empirically demonstrated that centralized behavior controls would decrease the software development process performance for the innovative projects which depend more on the developers' initiatives and improvisation. Outcome control is used in the control of project leaders over peripheral developers through the version control mechanism. The peripheral developers contribute patches of code to the project, the project leader reviews and decides whether the code can be adopted and added to the source code of software. In this process the project leader evaluates the code against his criteria for quality.

Self-control plays an important part in open source community, because the open source software development is non-routine, complex and innovative. Developers' initiatives and motivations are critical in the process. The control of project leader over the core developers is mainly through self-control. The developers are self-motivated and self-rewarded by both intrinsic motivations like enjoyment and self-enhancement, and external incentives like gaining and maintenance of reputation (Sagers 2004). However, the project leader can influence the exercising of self-control through membership selection and job assignment. He may select the potential developers who are most motivated and fit the project best as the core developers, and assign specific job task to the developer who has the ability and potential to accomplish it well.

Clan control is also effective in open source community. It has been studied that the ideology and culture is important for the performance of an open source community. The shared goals and values among developers greatly affect their involvement in the project, and their effort put into the software development. The violation of community philosophy is punished by collective sanction, and good performance is rewarded by praise and reputation gaining in the community (Sagers 2004). Project leader may promote the exercising of clan control through member socialization and promulgation of common values and norms to help the developers understand what behaviors are acceptable or valued in the community. Clan control is mainly effective in the control of project leader over core developers and the control between core developers.

So, based on the model of Kirsch (1996, 1997), three control modes are expected to be used in open source software development. The portfolios of outcome, clan and self controls may be various in different projects, but they are important to project success.

Dynamics of Control Modes in Open Source Community

Kirsch (2004) indicated that the portfolios of control modes may change dynamically with the different phases of an information system development project. The evolution of an open source community makes it have different characteristics in the phases. The control modes are believed to change with the evolution of community.

In the initial phase, the open source project starts by the idea or personal interest of a given individual or a small group. The people discuss and share the ideas, set the goal, and initialize the development process. In this phase, the size of the open source community is small, usually less than ten people. All individuals work closely and make the contributions to the project. With the project is carrying on, some of them may leave, but people who stay could become the key individuals in the later phase of software development. The self-control mode is mainly used here to manage the project. Each individual has his or her own goal and process for performing the task. These goals or processes may or may not be well defined and documented.

In the developing phase, the community is growing fast. More and more individuals join the community. Some roles are created and assigned to individuals, such as managers/coordinators, developers, reviewers, contributors, and users. In some big projects, the organizational structure is decentralized. A central manager or coordinator coordinates the overall development, while the second level mangers or coordinators are in charge of the specific subsystems. In each subsystem, developers, reviewers, and contributors make their contributions to the project by coding, testing, bug reporting and proposing new ideas. In this phase, self-control also plays an important role, but clan control becomes effective because community norms are developed and individuals in the community are more and more dependent on one another and share a set of common goals. The acceptable process and behavior is formed and evolved over time. From the initial phase to the developing phase, the control mechanism is changed from self-control to combination of self and clan controls.

In the maturity phase, the open source community is relatively stable. The number of individuals is either stable with minor change or increased regularly. Some successful examples in this phase are the Linux, the Apache Web server, and the Perl community. The organization of the open source community is well defined with an applicable development process. The outcome is much more predictable, and the process is more controllable. Managerial formal control becomes possible and is expected to be effective. So in this phase, the outcome control is used combined with the self and clan control.

CONCLUSION AND FUTURE RESEARCH

In this paper, we study and summarize the control modes in open source software development based on control theories. The portfolios of outcome, clan and self-control play critical role in the performance of open source community, and they change dynamically with the evolution of an open source community. Understanding of control mechanism has both theoretical and practical implications for open source software development management. Some case studies will be conducted to empirically confirm the findings in the paper.

REFERENCES

1. Bretthauer, D. (2002) "Open source software: A history", *Information Technology and Libraries*, March 2002.
2. Choudhury, V. and Sabherwal, R. (2003) "Portfolios of control in outsourced software development projects", *Information Systems Research*, September 2003.
3. Eisenhardt, K. M. (1985) "Control: organizational and economic approaches", *Management Science*, 31(2), 1985.
4. Feller, J. and Fitzgerald, B. (2002) *Understanding Open Source Software Development*, Addison-Wesley, 2002.
5. Hann, I., Roberts, J. and Slaughter, S. (2004) "Why developers participate in open source software projects: An empirical investigation", *Proceedings of International Conference of Information Systems*, December 2004, Washington, D.C.
6. Hars, A. and Ou, S. (2002) "Working for free? Motivations for participation in open source projects", *International Journal of Electronic Commerce*, Spring 2002, vol. 6, No. 3.
7. Henderson, J. C. and Lee, S. (1992) "Managing IS design teams: A control theories perspective", *Management Science*, June 1992.
8. Hippel, E. and Krogh, G. (2003) "Open source software and the "private-collective" innovation model: issues for organization science", *Organization Science*, Mar/Apr 2003, vol.14, no.2.
9. Jorgensen, N. (2001) "Putting it all in the trunk: incremental software development in the FreeBSD Open Source Project", *Information Systems Journal*, 11(4), 2001.
10. Kirsch, L. J. (1996) "The management of complex tasks in organizations: controlling the systems development process", *Organization Science*, January-February 1996.
11. Kirsch, L. J. (1997) "Portfolios of control modes and IS project management", *Information Systems Research*, September 1997.

12. Kirsch, L. J. (2004) **"**Deploying Common Systems Globally: The Dynamics of Control", *Information Systems Research*, December 2004, vol.15, issue 4.
13. Lee, G.K. and Cole R.E. (2003) "From a firm-based to a community-based model of knowledge creation: the case of the Linux kernel development", *Organization Science*, Nov/Dec 2003, vol.14, issue 6.
14. Mahmood, M. A. (1987) "System development methods – A comparative investigation", MIS Quarterly, 7(3), 1987.
15. Manz, C. C., Mossholder, K. W. and Luthans, F. (1987) "An integrated perspective of self-control in organizations", *Administration & Society*, 19(1), 1987.
16. Markus, M. L., Manville, B. and Agres, C. E. (2000) "What makes a virtual organization work? ", *Sloan Management Review*, Fall 2000.
17. Mustonen, M. (2003) "Copyleft – the economics of Linux and other open source software", *Information Economics and Policy*, 15, 2003.
18. Nidumolu, S. R. and Subramani, M. R. (2003) "The matrix of control: Combining process and structure approaches to managing software development", *Journal of Management Information Systems*, Winter 2003.
19. Ouchi, W. G. (1980) "Markets, bureaucracies, and clans", *Administrative Science Quarterly*, 25(1), 1980.
20. Sagers, G. (2004) "The influence of network governance factors on success in open source software development projects", *Proceedings of International Conference of Information Systems*, December 2004, Washington, D. C.
21. Snell, S. A. (1992) "Control theory in strategic human resource management: The mediating effect of administrative information", *Academy of Management Journal*, 35(2), 1992.