A Study of Open Source Software Development from Control Perspective

Bo Xu
School of Management
Fudan University
670, Guoshun Road
Shanghai, 200433, China
Phone: (86) 21-25011229, Fax: (86)21-65644783
bxu@fudan.edu.cn

Zhangxi Lin
College of Business Administration
Texas Tech University
Lubbock, TX 79409, USA
Phone: (1) 806-742-1926, Fax: (1)806-742-3193
zhangxi.lin@ttu.edu

Yan Xu
Computer Science and Information Technology Department
Del Mar College
Corpus Christi, TX 78413, USA
Phone: 361-906-2276
yxu@delmar.edu

1 Corresponding author
2 This research was supported by the National Social Science Foundation of China (No. 09CTQ023).
A Study of Open Source Software Development from Control Perspective

ABSTRACT

Open source software (OSS) has achieved great success and exerted significant impacts on the software industry in past decade. OSS development takes online community as its organizational form, and the developers voluntarily work for the project. In the project execution process, control aligns individual behaviors towards the organizational goals via the Internet, and becomes critical to the success of OSS projects. This paper investigates the control modes in OSS project communities, and studies their effects on the project performance. Based on the web survey and archival data from OSS projects, it is revealed that three types of control modes, i.e. outcome, clanship, and self-control, are effective in an OSS project community. The study contributes to a better understanding of OSS project organizations and processes, and provides advices for OSS development.

KEYWORDS: open source software, software development, control, project management
INTRODUCTION

The past decade has seen a marked expansion in the open source software (OSS) movement. The open source initiative sprung from the idea that software should be free and open. OSS contrasts with the traditional software distribution model, in which computer software is sold only with a license to use precompiled binary code without giving users the access to the source code. OSS, on the other hand, is licensed to guarantee free access to the source code, often under a license that sets conditions for modification, reuse, and re-distribution (Bretthauer, 2002). The concept of copyleft is the core to OSS. 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 (Lerner and Tirole, 2002).

Open source software is the result of Web-based collaboration. Once started, an OSS project is usually accomplished by a community of participants that are geographically dispersed and communicate through the Internet (Lee and Cole, 2003), which makes OSS different from traditional software development in both organizing and process (Feller and Fitzgerald, 2002). Today, numerous open source projects are categorized into three types: (1) community projects, which are completely online community based, involving voluntary software developers; (2) non-profit organization projects, which have matured to the level where they can get funding towards a more formal organization but still maintain some features of community projects (e.g. Apache Software Foundation), in which developers can be either paid workers or volunteers; and, (3) commercial projects sponsored by companies like IBM, HP, SUN, etc., in which major
contributors are paid developers from the companies (Fitzgerald, 2006). Currently, most of the open source projects belong to the community projects category, and most of the successful software products (e.g. Linux, Apache) used to be community projects in their initial stages, although some of them took on the non-profit organization or commercial project model after they became very popular. Thus, in this paper we choose community projects as the target for research, and refer open source software to as those developed by online communities of volunteers.

Open source project participation is developers’ voluntary actions. The motivations for project participation include reputation gaining, job prospects, enjoyment, learning purpose, cooperation needs, open source ideology, and personal software needs (Hars and Ou, 2002; Roberts, Hann, and Slaughter, 2006; von Hippel and von Krogh, 2003). And each member in an open source project community may have different motivations for participation (Wu, Gerlach, and Young, 2007). However, it has been demonstrated that although OSS development is a process of voluntary activities, the developers’ behaviors can be affected by the project environment, such as the values, beliefs, and norms in project community (Stewart and Gosain, 2006), interpersonal relationship between developers (Xu, Jones, and Shao, 2009), and satisfaction of developers’ psychological needs (Agerfalk and Fitzgerald, 2008). Thus, the development activities of an OSS project can be regulated to some extent although the project takes online community as its organizational form.

In traditional software development teams, how to control members’ behaviors to align them with the goals of the project is critical to project success (Henderson and Lee, 1992; Kirsch, 1996; Kirsch, 1997). It has been demonstrated that control mechanisms play an important role in the governance and management of software development internally within an organization.
Both formal and informal control modes are used in traditional organizations and software development teams (Kirsch, 1996; Ouchi, 1980). The formal controls depend on the formal rules, procedures and evaluations, while informal controls depend on the factors like cultures, values, beliefs and members’ self-regulation. OSS development is significantly different from traditional software development in both organizational form and process. Previous research indicated that in OSS project communities there exists some types of governance that take effects as controls in traditional organizations, and they are necessary to project success (Hagan, Watson, and Barron, 2007). For example, Demil and Lecocq (2006) defined the governance structure in open source projects as neither market nor hierarchy, and termed it as “bazaar governance”. However, to provide guidance for OSS development it is essential to understand the controls in open source community in more depth. For this purpose, this paper investigates the control mechanisms common in open source projects, and empirically tests how they affect the outcome of projects.

The rest of the paper is organized as follows: First, the theories and literature of management controls are reviewed; Second, the control modes in open source projects are discussed based on the theories; Third, an empirical test is conducted to demonstrate the effects of the control modes on open source project performance; Finally, a discussion on the findings and implications concludes the paper.

THEORETICAL BACKGROUND AND LITERATURE REVIEW

Control in organizations has long been a topic of interests for researchers and practitioners alike, who generally recognize that control mechanisms are needed to help and ensure organizations achieve their goals. In project management, controls are conducted for
various purposes, like the control processes for scope, schedule, cost, risk, and quality. While control can be viewed from various perspectives, it is studied here in a behavioral sense, i.e. to ensure that individuals working on organizational projects act in accordance with an agreed-upon strategy to achieve desired objectives (Jaworski, 1988; Merchant, 1988). From this angle, software development is not only a technical process, but also a social process with multiple stakeholders whose behaviors are influential to the performance of software development projects (Guinan, Cooprider, and Faraj, 1998). Exercising control is a powerful approach to ensuring project progress by fusing together the complementary roles and capabilities of project participants and motivating them to work in accordance with organizational goals and objectives (Henderson and Lee, 1992).

The control mechanisms have been studied in either organizational or software development context. In the organizational context, Ouchi (1977, 1978, 1980) categorized the organizational control mechanisms into market, bureaucratic, and clan, and found that the adoption of control mechanisms is related to the organizational structure and task situation. Market mechanism is efficient when performance evaluation is not ambiguous, but the goals are incongruent; Bureaucratic mechanism is efficient when both evaluation ambiguity and goals incongruence are moderately high; Clan mechanism is the most effective if evaluation is ambiguous, but goal congruity is high (Ouchi, 1980).

Henderson and Lee (1992) indicated that control is important to the management of software development. According to Kirsch (1996, 1997), both formal and informal modes of controls are used in software development projects. The formal controls include behavior and outcome controls. 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, controllees are rewarded for meeting the desired outcomes or goals (Eisenhardt, 1985; Snell, 1992). Informal control is rooted in 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. Thus, it is a function of individual objectives, standards, and intrinsic motivations (Manz, Mossholder, and Luthans, 1987).

According to Kirsch (1996, 1997), different portfolios of control modes are used in different situations in traditional software development. Behavior and outcome controls are implemented if the appropriate procedures are known, outcomes are measurable, and the project manager is able to assess whether the target has been reached; clan and self controls are often implemented, when the task is complex, there is lack of rules and procedures for completing the task, and the performance evaluation is ambiguous (Kirsch, Sambamurthy, Ko, and Purvis, 2002). Choudhury and Sabherwal (2003) investigated the control mechanisms in alliance-based software development like outsourcing, and found that both formal and informal modes are used in outsourced software development projects. There are also some studies on the dynamics of control modes to know how control evolves with the maturity of an organization (Cardinal, Sitkin, and Long, 2004) or in the different phases of a development project (Kirsch, 2004).

As described above, multiple control modes are exerted in traditional organizations, which are termed as “Cathedral” in Raymond (2001). These controls are performed by the leaders (formal controls), by the members themselves (self control), or by the collectives (clan
control). OSS development takes a totally new online community-based organizational form, which is described as “Bazaar”. How controls apply to the open source context still remains unclear.

**CONTROLS IN OPEN SOURCE SOFTWARE DEVELOPMENT**

Open source software development is significantly different from traditional software development in both organization and process (Ljungberg, 2000). An open source project begins when an individual or a group of individuals contribute to an initial functional prototype of the software (Raymond, 2001). Then a project community grows as more and more persons are attracted into the project as developers. They work collaboratively to continue developing the software (Crowston and Howison, 2005; Long and Siau, 2007).

The software evolves incrementally through rapid development iterations produced by the community. In the development process, developers offer most of the contributions and fulfill most of the development tasks. After they submit a code patch, the source code is tested and reviewed by the peripheral users, who may request for bug fixes, patches, supports, and new features. The requests are usually treated by developers as recognized work tasks (Stewart and Gosain, 2006). Thus, the participation of developers is critical to the success of open source projects. The controls in open source projects are mainly related to the controls over the activities of developers.

Since the success of an OSS project depends on the voluntary contribution from developers, how to align the developers with the objectives of the project is critical to the final outcome of the project. In this section, the existence and effects of various control modes are discussed for the context of open source project.
**Formal Controls**

There are two types of formal controls: behavior control and outcome control. Behavior control, or process control, is exercised when managers attempt to influence the process of a given job. It therefore centers on evaluating an individual in terms of the means or procedures that are thought to lead to a given outcome (Ouchi, 1980). For example, on an assembly line the manager closely monitors whether the workers are following the established procedure. Outcome control, in contrast, is exercised when a given individual is evaluated in terms of the results relative to setting standards of performance (Ouchi, 1980). University professors are evaluated based on their teaching evaluation and the number and quality of their academic publications. This is a typical outcome control, because the university may only monitor their results of teaching and research, and they have much autonomy on how to reach the results.

**Behavior Control**

Among the four major types of controls (behavior, outcome, clan, self), behavior control is infeasible in OSS development since OSS development is an incremental innovation process which is based on participants’ autonomy and initiatives, and there is no pre-defined procedures in the process. In addition, because a typical OSS project community is composed of geographically dispersed developers communicating through the Internet, it is impossible to closely monitor the developers’ behaviors. Behavior control may also have negative impact on technological innovation (Cardinal, 2001). Nidumolu and Subramani (2003) empirically demonstrated that centralized behavior controls would decrease the software development performance for the innovative projects that depend more on the developers’ initiatives and
improvisation. In open source context, the project leaders may assign tasks to developers, but do not regulate how they perform the tasks. In addition, many volunteers participate in OSS projects because they enjoy the sense of freedom and autonomy, control on their behaviors may have negative impacts on their motivations and initiatives. Therefore, we can conclude that behavior control is not appropriate for OSS development.

Outcome Control

Outcome control is a control model in which controllees are rewarded for meeting the desired goals or performance standards (Eisenhardt, 1985). Cardinal (2001) indicated that outcome control may increase the likelihood of enhancement innovation by encouraging participants to put more efforts into the innovative project. In OSS development, a developer’s performance can be reviewed based on the volume and quality of his contributions. And there is still rewarding or punishing on the developers’ performance through some mechanism, for example, voting by the project members. In a matured open source community, some relatively formal authority could be developed, and project leaders may conduct the control or governance based on the shared conception of the community (O’Mahony and Ferraro, 2007). Fielding (1999) indicated that there is a hierarchy of ranks in some successful open source projects. For example, the Apache project community has a ranking system that is composed of the levels of developer, committer, project management committee member, and Apache Software Foundation member, in order of increasing status; and good performance of a developer can be recognized by promotion to higher rank in the project community. In many less popular open source projects, there exists a similar mechanism, in which developers are periodically evaluated and better performance are recognized by advancement within the ranking system. Thus, in an OSS project,
outcome control is enacted to some extent. The level of outcome control is expected to impact the project performance.

Hypothesis 1: Outcome control is positively related to the performance of open source software development.

**Informal Controls**

Informal controls differ from formal controls in that they are based on social or people strategies (Eisenhardt, 1985; Jaworski, 1988). Informal controls include the modes of clan control and self-control. They are often used when the formal controls are infeasible or insufficient (Kirsch, 1996).

**Clan Control**

Clan control is a mode of informal control that relies on values, beliefs, organizational culture, shared norms, or interpersonal relationships, to regulate human behaviors and facilitate the reaching of organizational goals (Eisenhardt, 1985). Ouchi (1980) described a clan as a group of individuals who are dependent on one another. OSS development is a collective innovative process, and clanship control is considered effective in such a context. Prior studies found that common values, norms, and beliefs do exist among developers in open source projects, and they have great influences on developers’ participative activities (Xu et al., 2009).

Stewart and Gosain (2006) defined open source ideology as a set of shared values, norms, and beliefs that are recognized by the open source communities, and found that the shared values, norms, and beliefs have significant impact on the developers’ efforts, and consequently have influence on the project effectiveness. In addition, trusting relationships are developed among
members in many open source projects (Bergquist and Ljungberg, 2001; Gallivan, 2001), and
have significant influence on developers’ involvement and participative behaviors (Xu et al.,
2009). Clanship is characterized by the shared values and beliefs, and collegial relationships
among members in a collective (Jaworski, Stathakopoulos, and Krishnan, 1993). OSS developers
are volunteers, they are not formally affiliated, and do not have legal or contractual relationship
to the project. There is no much formal control over the voluntary developers. Thus, to make the
project community a clanship is critical to the project performance. In open source projects,
common values and beliefs can be recognized and promulgated through communications among
developers; and relationships can be developed among developers in the process of collaboration.
The level of clan control is expected to have significant effects on the project performance.

Hypothesis 2: Clan control is positively related to the performance of open source
software development.

Self-control

Self-control is a mode of informal control in which an individual sets his own goals, self-
monitors goal achievement, and rewards or sanctions himself accordingly (Manz et al., 1987).
Whereas clan control is a function of organizational culture and peer influence in a group, self-
control stems from individual objectives and standards (Jaworski, 1988). The community-based
nature of OSS development predetermines that external force or intervention is limited. Although
there is some outcome control and clanship control in open source projects as discussed above,
the success of software development still relies on developers’ self control or self management.
In addition, many developers put efforts into software development because they enjoy the way
of open source coding, from which they obtain the sense of autonomy, competence and freedom
(Roberts et al., 2006). Over-regulation may have negative impact on developers’ motivations (Nidumolu and Subramani, 2003). Thus, to promote developers’ initiatives and inspire their creativity, OSS development highly depends on developers’ self control. To conduct effective and successful self control, it is essential to have individual goals aligned with the project goals. In OSS development, self control can be realized by involving developers in setting project goals. If the project goals are determined through the developers’ consensus, then the developers may self-regulate their activities against the agreed goals consciously. The level of self control is expected to significantly influence the project performance.

The relationship between self and clan control is subtle as described by Kirsch (1996). They complement each other. Self-control leads to more active participation and creativity, while clan control is important to safeguard and promote cooperation in the project.

Hypothesis 3: Self-control is positively related to the performance of open source software development.

Therefore, based on the control theories, three control modes are expected to take effects in OSS development. The portfolios of outcome, clan and self controls may be diversified in different projects and in different stages of a project, but they are important to project success.

CONTROL MODES AND OPEN SOURCE PROJECT PERFORMANCE

To test the relationships between the control modes and open source project performance, an empirical study was conducted with data collected from open source projects. The methods and findings are described in the following parts of this section.

Sampling Procedure
Many studies of OSS have focused on case studies of the largest, most well-known projects such as Linux and Apache. These projects are interesting and important, but they may not be representative of the majority of OSS projects, which are in the “initial” stage, and attain much lower levels of participation and prominence. In addition, this study is to investigate the control aspects of open source project communities. Thus, only those projects that are purely community-based are included in the study. Corporate-related products, like Red Hat Linux, are not targets for the study. Data of open source projects were collected from Sourceforge.net, one of the world’s largest websites with free virtual community hosting services for OSS projects. By February 2009, there were more than 230,000 registered projects, and more than 2 million registered users on it. Sourceforge provides these projects with a standard technology toolset, thereby reducing the variance in project effectiveness that may be due to differences in technology used to support workflow, code distribution and versioning. We chose the category of communications on Sourceforge to control the differences across different product categories. We selected those projects that had some activities in the prior month before data collection in terms of the contributions of participants to the code repository, changes in mailing list or discussion forum, or requests for bug fixes, supports, patches or features. This was done to ensure the sample include ongoing projects that had not been abandoned by developers. Since the research is related to the social control factors in virtual communities, only the OSS projects with at least four members (including project managers and other developers) were eligible for the study.

472 projects met the criteria and were selected as the subjects for study. These projects covered a wide range of topics, such as BBS, chat, and ICQ. They were in various development phases, including planning, pre-alpha, alpha, beta, production, and maturity.
Measurements

Meeting budgets and requirements may not be relevant in OSS (Scacchi, 2002). Thus, the more appropriate measurement for OSS project success may be through the ongoing productivity of the project. The Sourceforge site tracks the number of requests for bug fixes, patches, supports, and new features on each project as well as the number of such requests that have an uncompleted status. The requests are treated as identified work tasks in the OSS project. The extent to which identified work tasks are completed (task completion) reflects the effectiveness of the project in feature addition, code modification, and bug fixing. It was used in prior studies of OSS development such as Stewart and Gosain (2006) to measure the project effectiveness. In this study, we used task completion as the measurement for project performance, and calculate task completion as the percentage of tasks completed, i.e. task completion = (total requests – requests open)/total requests x 100.

The control modes are assessed using multiple item measures through the perception of project members. They are measured based on well-established scales from information systems, management science, and organization science. Minor modifications are made to the scales to fit the research context better. The measurements use a 1-7 likert scale. The measurement for outcome control is adapted from Kirsch et al. (2002) and Jaworski et al. (1993). The scale asks several questions about how much the developers’ positions or status in the project community are based on their merits or performance. It is difficult for project members to directly perceive and report the level of common goals, values, and norms that characterize clan control in the project. Thus, in this study we use the measurement for clan control adapted from Jaworski and MacInnis (1989) and Jaworski et al. (1993). The scale includes several questions about
interactions, communications, and cooperation among project members, and is more perceivable and critical to the reaching of clanship. For self control, the controllee sets the goals, self-monitors, and self-evaluates accordingly. Thus, self control is characterized by controllee as the source of project goals. In open source software development, self control occurs when the project goals are set by the developers rather than totally decided by the project manager. Then the developers may act consciously to meet the goals without external interference. In this study, the measurement for self-control is adapted from Kirsch (1996) and Kirsch et al. (2002). The scale includes questions about the developers’ role in setting of project goals and procedures. The questionnaire is shown in the Appendix.

The form and level of control modes, and developers’ motivations seem to vary with the age and stage of open source projects. In addition, project community size may also have influence on project performance, which is measured with task completion. Thus, project tenure (number of days from project registration), development stage (planning, pre-alpha, alpha, beta, production, and maturity), and community size (number of developers in the project) are treated as control variables when we study the effects of the control modes on project performance.

Data Collection

The study is at the community or project level. To measure the community-level constructs, we used the project members as informants, and the level of controls in the projects were measured through their perception. There were totally 3,214 members in the selected projects on Sourceforge. Among them 826 were project managers. Data were collected using the web survey. The questionnaire was posted on the researcher’s website. Members of the selected projects were contacted through emails. The email address of every member was obtained from
the project web pages on Sourceforge, and email was sent to each of the 3,214 members. The content of the email was to explain the purpose of the research and invite them participating in the survey. In order to get a higher response rate, one week later, a reminder email was sent to request that they answer the survey if they had not done so. 253 emails were bounced back, and remaining 2,961 emails were believed to have reached the project members. 423 responses to the survey were received by the end of data collection. Considering the fact that project managers and other developers may have different perception or feelings toward the projects, only projects with at least one response from managers and one response from other developers were kept in the sample, which resulted in the final data set of 265 responses from 93 different projects. The responses covered OSS projects with various development phases, topics, intended audiences, and license types.

The response rate was 14.3%. Non-response bias was estimated using the extrapolation method (Armstrong and Overton, 1977). The extrapolation method is based on the assumption that subjects who respond less readily are more likely non-respondents. There were two waves of responses in this study, the first wave followed the first round of invitation emails, and the second wave followed the reminder emails. Responses from the two different waves were compared, and no significant difference was found between the early and late responses. This indicates that the respondents in the study can represent the whole sample, and the conclusions from the responses can be generalized to the population.

To measure project performance, numbers of total and open requests for each project in the sample were obtained from Sourceforge website; and the control variables, such as value for project tenure, development stage, and community size, were also collected from the website.

The statistical characteristics of the sample projects are shown in Table 1.
Data Analysis and Results

An exploratory factor analysis was conducted first. Four items with low loadings were dropped based on the results of the exploratory factor analysis.

Reliability and Validity

The constructs were assessed for reliability using Cronbach’s $\alpha$ (Cronbach, 1951). Nunnally (1978) suggested that a value of at least 0.70 indicate the adequate reliability. The Cronbach’s $\alpha$ of constructs are shown in Table 2.

The questions were tested for validity using confirmatory factor analysis with the principal component analysis and the varimax rotation. Convergent validity was assessed by checking loadings to see if terms within the same construct correlate highly amongst themselves. The results of confirmatory factor analysis are shown in Table 2.

The Cronbach’s $\alpha$ values here range from 0.81 to 0.89, which is above the cut-off value of 0.70. Thus the reliability of constructs is confirmed. Item loadings should be greater than 0.70 for convergent validity (Chin, 1998). The item loadings range from 0.75 to 0.87, which suggests that the factor loadings are sufficiently large and the items loaded as intended for every construct. In addition, the cross-loading matrix in Table 3 shows that each indicator loads much higher on the construct of interest than on any other factors. Thus, the convergent and discriminant validity of constructs is suggested.
The final constructs and items are shown in the Appendix.

Hypotheses Tests

Four factors were studied, project performance, outcome control, clan control, and self control. Three factors, project tenure, development stage, and community size, are included as control variables. The unit of analysis for this study is the project (n=93). The hypotheses are concerned with the relationship between control variables and performance for open source projects. Measures for the controls of each project are computed by averaging the scores of respondents from the project and computing the project scores. Equal weight is given for each respondent and each role. Prior to averaging respondents’ scores, interrater agreement was assessed using the multiple-item estimator for within-group interrater reliability as proposed by James, Demaree, and Wolf (1984). The results indicate generally a strong agreement of ratings referring to the same project, which means there is no discrepancy between the managers and developers for rating the controls in the open source projects.

The descriptive statistics of the project level factors are shown in Table 4.

[insert Table 4 here]

To test the relationship between control modes and open source project performance, multiple linear regression analysis was performed, with project performance as dependent variable, the control modes as independent variables, and project tenure, development stage, and community size as control variables. The results are reported in Table 5.

[insert Table 5 here]

The regression results demonstrate that three types of control (outcome, clan and self control) have positive effects on the performance of open source project. And they account for
43% of the total variance in project performance, which is assessed through the percentage of task completion in the project. The effects of outcome control, clan control, and self control are significant at (P ≤ 0.05) level.

**DISCUSSIONS**

OSS development takes the online community as the organizational form, and developers are volunteers who are not formally affiliated to the project. How to control the activities of developers in such a loosely-structured organization is important to software development and concerned by the leaders and managers of open source projects. By analyzing the data from OSS projects, this study finds that the portfolios of outcome, clan control, and self control play a critical role in the performance of OSS projects. The levels of outcome, clan control, and self control have significant impact on the OSS project performance.

This study contributes to research of OSS by providing an in-depth understanding of OSS development from the control perspective. Previously, there have been some studies to investigate factors related to open source project success. For example, Stewart, Ammeter, and Maruping (2006) indicated that license restrictiveness and organizational sponsorship have impacts on success of open source projects. Similarly, Subramaniam, Sen, and Nelson (2009) found that restrictive licenses have an adverse impact on OSS project success. Grewal, Lilien, and Mallapragada (2006) indicated that network embeddedness, which is the linkages of project managers and developers with other projects, is important to open source project success. Open source is adopted by some companies as a new software development organization form (Fitzgerald, 2006). Agerfalk and Fitzgerald (2008) discussed outsourcing to open source community from a psychological contract perspective. Each of these studies tried to understand
the influential factors for open source project success from certain perspectives. However, they
did not provide a comprehensive view on how to regulate developer activities for project
management. Among the factors related to OSS development success, how to manage
volunteers’ participation behaviors is essential. Currently, research has been focused on the
motivations of open source developers’ participation, and found that participation is motivated
and influenced by multiple factors, including extrinsic and intrinsic motives (Roberts et al.,
2006), and social relational factors in the community (Bagozzi and Dholakia, 2006; Oh and Jeon,
2007). These studies made progresses in understanding the motivational factors. However, they
did not reveal how to enhance or promote developers’ motivations from the project management
perspective. This paper, for the first time, incorporates the control concept and theory into OSS
research, and indicates that the developers’ activities can be controlled or “managed” to some
extent to fit into the project goals, although OSS development is based on online communities.
This provides implications for OSS researchers to conduct more study from management
perspectives to better know how to regulate participants’ activities for the success of OSS
development.

This study shows that the controls in OSS development are different from those in
traditional software development teams. In traditional software development, both formal and
informal controls are incurred, and managers may choose to apply the formal controls, including
process and outcome controls, when they have enough knowledge and skills, there is established
procedure, and the results are measurable. Informal controls, like clan control and self control,
are used mainly as complements for the formal controls in case the formal controls are infeasible
or inefficient. However, in OSS development controls take effects mainly in informal way.
Regulation of developer behaviors mainly depends on the clanship and developers’ self control.
Although there is outcome control to some extent, it is not as formal as in traditional teams, and just based on some common agreement among developers. In addition, in traditional teams the controls are mainly exerted by the leaders. However, in OSS projects the leadership of project managers is not so formal and the authority is quite limited. The controls are exerted mainly by the collective of developers through the means of voting, discussion, and peer influence.

The results have practical implications for project management in open source context. The role of outcome control implies that ranking mechanism is still important. OSS development requires initiatives and investment of time and energy. Ranking based on performance may stimulate participation and reduce the chance of free riding. In open source projects, monetary rewards may not be a concern of developers. Therefore, ranking may be an important factor to promote contribution since developers need to accomplish the sense of achievement through higher status in the projects. The role of clan control implies that the clanship or community quality is critical for project success. An effective OSS project leader is expected to be a good facilitator to build a high-quality community by promulgating the common values and norms among developers, facilitating communications and relationships building among developers, so as to promote clanship control in the project. The role of self-control implies the importance of developers’ self-regulation. For self-control to be effective, it is necessary for project managers to set goals for the project collectively with other developers, or to socialize developers to the project goals, thus, the developers will behave according to the goals consciously.

Open source software is typically of the open content model, which refers to the online collaborative knowledge creation by volunteers. Today, the open content model has been applied to many other contexts in addition to software, such as open source music composition (e.g. jMusic), online encyclopedia (e.g. wikipedia), and eR&D (e.g. Procter & Gamble’s “Connect &
Develop” R&D model). Thus, findings from this research may not only provide advice for open source software development, but also have implications for other open content systems as mentioned above.

The study has some limitations. First, as stated at the beginning of the paper, this research is conducted for community projects, which are online community-based with volunteers as developers. Thus, the research findings may not be appropriate for the non-profit organization and commercial projects. Second, the dataset is based on projects from Sourceforge.net. Currently, some large projects have their own websites and are not included. Thus the results may not be applicable to those projects not listed on Sourceforge.net. In addition, only projects that had activity within the month prior to data collection are considered for study in the research. This may cause certain biases since completed projects are not included in the study. However, the projects selected in this study are believed to be representative of community OSS projects, since they cover a wide range of topics, license types, and development phases. Third, this study demonstrates that the three modes of control are important in open source software development. However, it does not explain how they can affect the performance of open source software projects. Previous studies indicated that volunteers’ participation and contribution in open source software development are motivated by their enjoyment (Hars and Ou, 2002; Xu et al., 2009), sense of achievement (Roberts et al., 2006), identification (Hars and Ou, 2002; Xu and Jones, 2010), and obligation (Xu and Jones, 2010). To further understand how and why the control modes affect open source software development, it is necessary to study the linkage between the control modes and the motivational factors, for example, how the control modes satisfy developers’ sense of achievement, enhance their enjoyment, and promote their identification and
obligation with the projects, and consequently affect the performance of open source software projects.

CONCLUSIONS

In this paper we investigate the control modes in OSS projects, and study their effects on OSS project performance. The research findings show that outcome, clan control, and self control play important roles in OSS development. The research enriches the theory and literature of OSS development. It also has practical implications for OSS development and project management. According to prior studies of control mechanisms, control modes may change dynamically in the lifecycle of software development. The dynamics or evolution of control modes in open source context has not been studied, and it is expected to be investigated in the follow-up research. The current research is based on community open source projects. It could be extended to projects supported by non-profit organizations and conducted by commercial firms in future research, to see what control mechanisms are effective in other types of open source projects. Finally, some motivational factors can be included as mediators in future study to provide a better understanding on how the control modes affect open source project performance. The potential mediators may be the motivational factors such as sense of achievement, enjoyment, identification, and obligation. Relationship between the control modes and motivational factors will be studied in future research to know the paths that control modes influence the performance of open source software development.

Appendix: Measures and Scales
Outcome Control
Outcome1: To what extent developers are awarded higher level positions dependent on their performance in the project.
Outcome2: To what degree developers’ positions in the project are tied to their performance.
Outcome3: To what extent developers’ status in the project depends on the amount and quality of their contribution.

Clan Control
Clan1: The project encourages cooperation between developers.
Clan2: Developers in the project are familiar with each other.
Clan3: The project fosters an environment where developers respect each other’s work.
Clan4: The project encourages discussions between developers.

Self Control
Self1: The setting of the project goals is pretty much under developers’ control.
Self2: For this project, to what extent the developers are allowed a high degree of influence in determination of project goals.
Self3: Developers have significant influence in setting specific procedures for project activities.

REFERENCES


<table>
<thead>
<tr>
<th>Development Status</th>
<th>Number (N=93)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Planning</td>
<td>2</td>
<td>2.2%</td>
</tr>
<tr>
<td>Pre-alpha</td>
<td>3</td>
<td>3.2%</td>
</tr>
<tr>
<td>Alpha</td>
<td>8</td>
<td>8.6%</td>
</tr>
<tr>
<td>Beta</td>
<td>33</td>
<td>35.5%</td>
</tr>
<tr>
<td>Production/Stable</td>
<td>45</td>
<td>48.4%</td>
</tr>
<tr>
<td>Maturity</td>
<td>2</td>
<td>2.2%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Project Age</th>
<th>Number (N=93)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;1 year</td>
<td>8</td>
<td>8.8%</td>
</tr>
</tbody>
</table>
1-2 years  26  28.1%
2-3 years  19  20.2%
3-4 years  12  13.2%
4-5 years  14  14.9%
5-6 years  11  11.4%
>6 years  3  3.5%

<table>
<thead>
<tr>
<th>License Type</th>
<th>Number (N=93)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>Apache Software License</td>
<td>2</td>
<td>2.2%</td>
</tr>
<tr>
<td>Artistic License</td>
<td>2</td>
<td>2.2%</td>
</tr>
<tr>
<td>BSD License</td>
<td>8</td>
<td>8.6%</td>
</tr>
<tr>
<td>GNU General Public License (GPL)</td>
<td>49</td>
<td>52.7%</td>
</tr>
<tr>
<td>GNU Library or Lesser General Public License</td>
<td>16</td>
<td>17.2%</td>
</tr>
<tr>
<td>MIT License</td>
<td>3</td>
<td>3.2%</td>
</tr>
<tr>
<td>Common Public License</td>
<td>6</td>
<td>6.5%</td>
</tr>
<tr>
<td>Others</td>
<td>7</td>
<td>7.5%</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Community Size</th>
<th>Number (N=93)</th>
<th>Percentage</th>
</tr>
</thead>
<tbody>
<tr>
<td>&lt;=10</td>
<td>28</td>
<td>30.1%</td>
</tr>
<tr>
<td>11-20</td>
<td>51</td>
<td>54.8%</td>
</tr>
<tr>
<td>21-30</td>
<td>11</td>
<td>11.8%</td>
</tr>
<tr>
<td>31-40</td>
<td>3</td>
<td>3.2%</td>
</tr>
</tbody>
</table>

Table 2 Item Loadings and Construct Reliability

<table>
<thead>
<tr>
<th>Construct</th>
<th>Item</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Standard Loading</th>
<th>Cronbach’s α</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome Control</td>
<td>Outcome1</td>
<td>5.18</td>
<td>1.31</td>
<td>0.80</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Outcome2</td>
<td>5.21</td>
<td>1.23</td>
<td>0.84</td>
<td>0.89</td>
</tr>
<tr>
<td></td>
<td>Outcome3</td>
<td>5.34</td>
<td>1.17</td>
<td>0.87</td>
<td></td>
</tr>
<tr>
<td>Clan Control</td>
<td>Clan1</td>
<td>4.87</td>
<td>1.52</td>
<td>0.75</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clan2</td>
<td>4.93</td>
<td>1.42</td>
<td>0.75</td>
<td>0.81</td>
</tr>
<tr>
<td></td>
<td>Clan3</td>
<td>4.63</td>
<td>1.32</td>
<td>0.81</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Clan4</td>
<td>4.94</td>
<td>1.41</td>
<td>0.77</td>
<td></td>
</tr>
<tr>
<td>Self Control</td>
<td>Self1</td>
<td>5.41</td>
<td>1.14</td>
<td>0.80</td>
<td>0.85</td>
</tr>
<tr>
<td></td>
<td>Self2</td>
<td>5.18</td>
<td>1.18</td>
<td>0.81</td>
<td></td>
</tr>
</tbody>
</table>
### Table 3 Cross-loadings of Constructs

<table>
<thead>
<tr>
<th>Question</th>
<th>Factor</th>
<th>1</th>
<th>2</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome1</td>
<td></td>
<td>0.80</td>
<td>0.21</td>
<td>0.25</td>
</tr>
<tr>
<td>Outcome2</td>
<td></td>
<td>0.84</td>
<td>0.17</td>
<td>0.36</td>
</tr>
<tr>
<td>Outcome3</td>
<td></td>
<td>0.87</td>
<td>0.15</td>
<td>0.30</td>
</tr>
<tr>
<td>Clan1</td>
<td></td>
<td>0.17</td>
<td>0.75</td>
<td>0.22</td>
</tr>
<tr>
<td>Clan2</td>
<td></td>
<td>0.19</td>
<td>0.75</td>
<td>0.26</td>
</tr>
<tr>
<td>Clan3</td>
<td></td>
<td>0.05</td>
<td>0.81</td>
<td>0.07</td>
</tr>
<tr>
<td>Clan4</td>
<td></td>
<td>0.14</td>
<td>0.77</td>
<td>0.09</td>
</tr>
<tr>
<td>Self1</td>
<td></td>
<td>0.26</td>
<td>0.11</td>
<td>0.80</td>
</tr>
<tr>
<td>Self2</td>
<td></td>
<td>0.23</td>
<td>0.12</td>
<td>0.81</td>
</tr>
<tr>
<td>Self3</td>
<td></td>
<td>0.20</td>
<td>0.16</td>
<td>0.81</td>
</tr>
</tbody>
</table>

### Table 4 Descriptive Statistics of Project Factors

<table>
<thead>
<tr>
<th>Factor</th>
<th>Mean</th>
<th>Standard Deviation</th>
<th>Median</th>
<th>Max. Value</th>
<th>Min. Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outcome Control</td>
<td>5.36</td>
<td>0.76</td>
<td>5.34</td>
<td>6.84</td>
<td>3.5</td>
</tr>
<tr>
<td>Clan Control</td>
<td>4.35</td>
<td>0.86</td>
<td>4.38</td>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>Self Control</td>
<td>5.39</td>
<td>0.62</td>
<td>5.5</td>
<td>6.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Project Performance</td>
<td>79.21</td>
<td>10.09</td>
<td>81.18</td>
<td>96.73</td>
<td>43.93</td>
</tr>
<tr>
<td>Factor</td>
<td>DF</td>
<td>Parameter Estimate</td>
<td>Standard Error</td>
<td>T Value</td>
<td>Pr &gt;</td>
</tr>
<tr>
<td>--------------------</td>
<td>----</td>
<td>--------------------</td>
<td>----------------</td>
<td>---------</td>
<td>------</td>
</tr>
<tr>
<td>Outcome Control</td>
<td>1</td>
<td>3.95</td>
<td>1.52</td>
<td>2.60</td>
<td>0.011*</td>
</tr>
<tr>
<td>Clan Control</td>
<td>1</td>
<td>2.59</td>
<td>1.06</td>
<td>2.44</td>
<td>0.017*</td>
</tr>
<tr>
<td>Self Control</td>
<td>1</td>
<td>4.60</td>
<td>1.80</td>
<td>2.56</td>
<td>0.012*</td>
</tr>
<tr>
<td>Project Tenure</td>
<td>1</td>
<td>0.0006</td>
<td>0.0014</td>
<td>0.42</td>
<td>0.679</td>
</tr>
<tr>
<td>Development Stage</td>
<td>1</td>
<td>0.16</td>
<td>0.89</td>
<td>0.18</td>
<td>0.854</td>
</tr>
<tr>
<td>Community Size</td>
<td>1</td>
<td>0.015</td>
<td>0.018</td>
<td>0.83</td>
<td>0.411</td>
</tr>
</tbody>
</table>

* P < 5%, $R^2 = 0.43$

**Authors’ Biography**

Bo Xu is an assistant professor at School of Management, Fudan University in Shanghai, China. He received Ph.D. degree in Management Information Systems from Texas Tech University, USA. His research interests include electronic commerce, open source software, and online community. He has published papers in *Decision Support Systems, Information & Management, Database for Advances in Information Systems, Journal of Global Information Technology Management*, and some major information systems conferences.

Zhangxi Lin is an associate professor at the Rawls College of Business Administration, Texas Tech University. His research interests include electronic commerce, data communications, information economics, and IT strategy. He has published papers in *Information Systems*. 

<table>
<thead>
<tr>
<th>Project Tenure</th>
<th>1730</th>
<th>590</th>
<th>1812</th>
<th>2578</th>
<th>533</th>
</tr>
</thead>
<tbody>
<tr>
<td>Development Stage</td>
<td>4.31</td>
<td>0.94</td>
<td>5</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>Community Size</td>
<td>18.8</td>
<td>8.72</td>
<td>19.7</td>
<td>73</td>
<td>4</td>
</tr>
</tbody>
</table>
Yan Xu is an instructor of Computer Science and Information Technology at Del Mar College, Texas. He received master degree of Computer Science from Texas Tech University in 2002. His research interests include controls in open source software, self-healing software component and architecture, software security modeling, network security, and software product line. Mr. Xu has published technical papers in the related areas.