AN EXPLORATORY STUDY OF OPEN SOURCE PROJECTS FROM A PROJECT MANAGEMENT PERSPECTIVE

Jungpil Hahn
Krannert School of Management
Purdue University
West Lafayette, IN 47907
jphahn@krannert.purdue.edu

Chen Zhang
Krannert School of Management
Purdue University
West Lafayette, IN 47907
zhang153@krannert.purdue.edu

ABSTRACT

As open source software (OSS) development is gaining tremendous popularity, effective OSS project management is becoming increasingly crucial to the success of OSS projects. However, few studies have examined open source from project management perspective. Furthermore, the congruence between project characteristics and project management has not been recognized in the existing OSS literature despite the vast variety of open source projects in existence. In this paper we identify two fundamental types of OSS projects based on the intended audience of projects, namely user-targeted projects and developer-targeted projects. Then we try to identify the aspects of project management that have an impact on OSS project performance for different types of projects. Our empirical analysis of a large sample of open source projects indicates that human resource staffing and release management are crucial to both types of projects. In addition, compensation management is particularly important to user-targeted projects whereas communication and coordination is particularly essential to developer-targeted projects.

1. INTRODUCTION

During the past few years there has been a surge of interest in open source software development among practitioners, governments, businesses, and researchers in academia. Open source software is receiving more and more attention from industry and government. Many small and mid-sized companies turn to open source software when creating IT infrastructures due to low cost and high flexibility (Greenemeier 2005). Governments in Japan, Europe and other developing countries encourage government agencies to increase the use of Linux and other open source software.

A number of open source projects are being widely adopted, ranging from user applications (such as Emacs and Open Office) and programming languages (such as Perl and php) to Internet infrastructure. The most famous and successful open source projects such as Apache web server, Linux operating system and sendmail program have demonstrated the attractiveness of open source development as a new way of producing software.

Despite the huge successes of some open source projects, it is the reality that many projects fail to take off and became abandoned. One of the main reasons for these failed projects may be lack of effective project management. However, given that OSS is a relatively new way of software production, there has been little work done examining the management of open source projects. Furthermore, despite the huge variety of OSS projects, the need for a fit between project characteristics and project management has not been adequately addressed in the existing open source literature.

In this paper we propose a theoretical model stressing the fit between project characteristics and project management. We claim that open source project management needs to focus on different aspects for user-focused projects and developer-focused projects.

We also conduct an empirical analysis on a large sample of OSS projects to provide support for our claim. A number of case studies on OSS mainly focused the most successful projects; therefore they may be insufficient in capturing the common nature of open source projects in general. Some recent studies have attempted to offer a more complete view of OSS by empirically investigating OSS projects on a larger-scale (Crowston et al. 2003; Crowston et al. 2004; Koch 2004; Lerner and Tirole 2005; Xu et al. 2005). This study extends this stream of research. More importantly, the results of the study can provide practitioners especially open source project managers with some actionable insights and guidelines.

We attempt to answer the following research questions:

- What factors that are under the control of project manager impact the performance of OSS project?
- Are the project management factors impacting the performance of OSS projects different for user-focused projects and developer-focused projects?

The remainder of the paper is organized as follows. The next section reviews the relevant literature and develops a conceptual framework. Section 3 describes the research methodology. Detailed results are presented in section 4. And finally we conclude in section 5.

2. CONCEPTUAL FRAMEWORK

Many studies have focused on the fit relationship between context and design in organizational working units (Argote 1982; Drazin and Van de Ven 1985; Fry and Smith 1987; Van de Ven and Ferry 1985). These scholars suggest that the level of fit between context and design impacts the performance achieved by the organizational unit. Borrowing from the contingency theory we argue that management of an open source project must match or fit the

project characteristics and that the level of congruence between project characteristics and management affects the performance of the project. In this section we provide a review of the relevant literature in open source software development and project management and then propose our conceptual model.

2.1. Open Source Software Development

OSS has posed interesting questions for researchers in many fields. The two issues that are particularly relevant to this study are developer motivation and community structure.

A number of researchers have addressed what motivates programmers to contribute code in the lack of monetary compensation and how to attract and retain developers. Among the possible explanations for developers' participation in OSS projects are career concern incentive and ego gratification incentive (Lerner and Tirole 2002). Hars and Qu (2002) identify both internal motivations such as altruism and external motivations such as direct compensation.

Nakakoji et. al (2002) classify the eight roles that a member of the OSS community can take, namely passive user, reader, bug reporter, bug fixer, peripheral developer, active developer, core member, and project leader. They point out the importance of keeping a balanced structure that contains the different roles in the community. Although passive users do not contribute directly to the software development, they do so indirectly by attracting more active members who obtain ego gratification by serving a large number of users. Xu et. al (2005) categorize OSS community members into four overlapping groups - project leaders, core developers, co-developers, and active users. In this paper, we only distinguish between core developers, developers and users in general. Core developers refer to the people in the developer team of the project. They are typically managed by project managers. Developers refer to those who do not belong to the

developer team but have the technical capability to understand the code and to contribute code and fix bugs, frequently or infrequently. Users refer to those end users without the technical capability to understand the source code. They typically use the software, discover bugs, and submit user requests without understanding how the software is actually coded.

2.2. Project Management

The Project Management Institute identifies nine areas of knowledge that a project manager needs. These knowledge areas are integration management, scope management, time management, cost management, quality management, human resource management, communications management, risk management, and procurement management (Duncan 1996).

Of the nine areas, time, human resource, and communications management are especially relevant in the open source software development context. We further identify some key aspects of open source software management as follows.

Human Resource Staffing. In the open source community developers usually choose which project to join based on their personal interests or needs and then submit a request to the project manager. The project manager then decides whether to allow the developer to become a member of the development team. Sometimes the project manager may need to recruit more actively by submitting project volunteer postings to the entire community. Given the fact that there are millions of developers scattered around the world, project managers may be overwhelmed with a large pool of candidates interested in joining the project. Furthermore, most of the community members are totally strangers to each other, which greatly increases project managers' uncertainty in the candidates' qualifications and personalities. Consequently, acquiring the right quantity of personnel with the right quality becomes a challenging task to project managers.

Compensation Management. Unlike proprietary software development, open source software development usually provides no monetary compensation to developers. However, an interesting phenomenon is beginning to appear in the open source community - more and more projects and individual developers choose to accept monetary support from members. Most donations are used to compensate the time and the effort of the developers so that they could keep up their contribution to the projects. How should project managers decide whether to opt to this compensation scheme is worth further examining.

Communication and Coordination. The project manager of an open source project acts not only as the key coordinator in the project team itself but also as the liaison with the outer community of the project, including developers who occasionally contribute code and end users who form the majority of the community. Tools such as mailing lists, discussion forums, and bug tracking systems provide an important communication and coordination infrastructure for distributed software development.

Release Management. In OSS release management becomes even more critical and challenging than in proprietary software development. The entire community is dynamically changing; users and developers freely join and leave projects constantly. An effective release management can help attract and retain members to the project. Raymond (2000) identifies early and frequent releases as a critical part of the Linux development model. He argues that releasing early and releasing often can keep the users constantly stimulated and rewarded, which then leads to a high-quality product.

2.3. Conceptual Framework

Bezroukov (1999) argues that open source projects tend to be more successful in domains that are interesting to developers themselves. Based on his arguments we can infer that developer-targeted projects tend to be more successful than user-targeted projects. However, after examining the active open source projects listed on SourceForge during the week of October 30, 2005 we noticed that all top 10 projects have targeted at end users and 4 of them have end users as the only target audience.

To further explore this apparent anomaly, we categorize open source software projects into two types based on target audience of projects - developer-targeted projects and user-targeted projects. Developer-targeted projects aim to develop a tool or a software component that can be customized and reused by other programmers. These projects enable developers to extend the existing software by standing on the shoulders of the previous developers through reusing the source code. On the other hand, user-targeted projects aim to develop a system that primarily provides utilities to end users in various fields such as education and government. Their main objective is to provide end users with functionalities that are either difficult or expensive to obtain through proprietary software. End users, most of whom lack the capability to understand and modify the source code, mainly adopt the software as a black box without digging into the inside or customizing the software.

The difference in target audience of the OSS project then leads to a difference in the people interested in the project, or in other words, the characteristics of the project community.

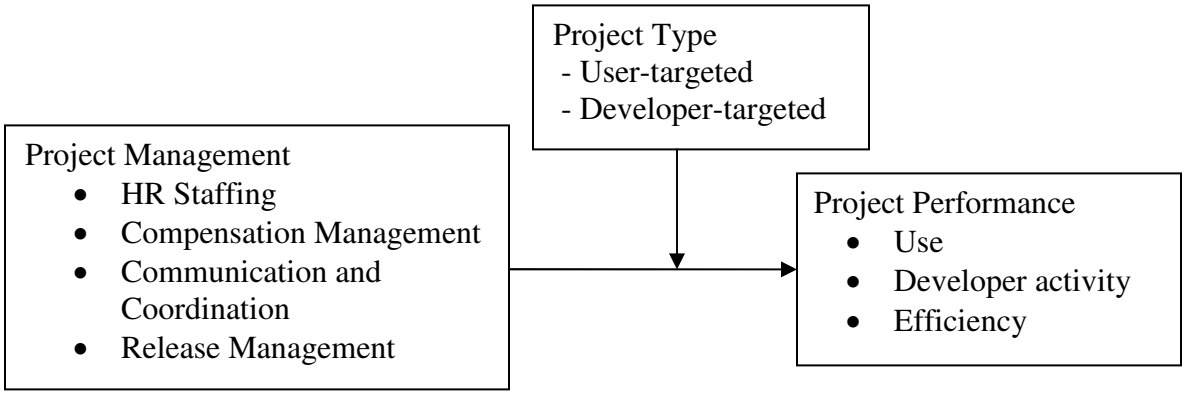
First of all, the user communities formed for the two types of projects differ in *technical expertise*. User-targeted projects tend to attract people with relatively little programming skills. Even though some programmers may also join the community for various reasons, majority of

the community members are still end users. In contrast, developer-targeted projects mainly attract programmers who have relatively higher level of expertise in programming.

Secondly, the *motivations* for people to become interested in user-targeted projects differ from those in developer-targeted projects. People are attracted to user-targeted projects mainly because of the utility provided by these projects. On the contrary, people become interested in developer-targeted projects mainly due to high performance visibility among peers (Lerner and Tirole 2002), ego gratification brought by the intellectual challenges in the source code, and intention to improve programming skills.

Therefore, we propose that fundamental differences exist between user-targeted projects and developer-targeted projects and that project type plays a moderating role on the relationship between project management and performance. In other words, in order for project management to impact the performance of the project, it has to be aligned with project type. Project managers need to place various levels of emphasis on HR staffing, incentive management, communication and coordination, and release management based on the intended audience of projects and characteristics of project community. Figure 1 presents our conceptual framework.

Figure 1 - Conceptual Framework



Next we perform an empirical examination of some OSS projects to confirm the impact of the fit between project type and project management on project performance.

3. METHODOLOGY

This section presents a preliminary exploration of the open source projects from project management perspective to gain some insights into the factors impacting the performance of user-targeted projects and those impacting the performance of developer-targeted projects.

3.1. Data Collection

Our data was collected from open source software projects hosted on SourceForge.net. As the largest repository of open source applications on the Internet, SourceForge.net currently provides free hosting to more than 100,000 projects and more than 1,100,000 subscribers. It also offers a variety of services to hosted projects, including site hosting, mailing lists, bug tracking, message boards, file archiving, and other project management tools. SourceForge has been an attractive source of data for many researchers studying open source software mainly due to the abundance of publicly accessible data (Howison and Crowston 2004).

Data collection was conducted between October 17, 2005 and October 22, 2005. Because this study is our first attempt to examine the management of open source projects, we tried to limit the sample size and the data collection time but still be able to gain some insights into our research question. Therefore we chose to analyze the projects in the database category only since this category contains a considerably degree of variety in terms of target audience of projects and does not contain a disproportionate percentage of user-targeted or developer-targeted projects when compared to the other categories (Lerner and Tirole 2005). Within this

category we further controlled for *Programming Language*, *Operating System* and *Intended Audience*. When a project is first started, the developer usually specifies in what language or languages the software will be written. And the choice of programming language is rarely changed as the project progresses. Past literature in software engineering suggests that programming language has an explicit impact on programming productivity and program size (Jones 1986). Hence we controlled for the programming language of the project by allowing projects written in Java[1] in our data set. Secondly, operating system of the project impacts the complexity of the software and the development effort required. Like *Programming Language*, once *Operating System* is decided for a project, it rarely changes and is relatively stable. We ensured that all the projects in our data set are designed to be operating system independent, which is also consistent with selecting Java for *Programming Language*. Thirdly, we also included *Intended Audience* as a control variable. In the early stage of the project, the project administrator usually specifies the intended audience of the software (i.e. whether the software is targeted at end users, developers, system administrators, etc.), which closely corresponds to project type as described in our research model. Since the focus of this paper is to study two types of project, namely user-targeted projects and developer-targeted projects, a project is eligible to be included in the sample if it has either end users or developers as *Intended Audience*. In order to avoid ambiguity and to more clearly differentiate between user-intended projects and developer-intended projects, we excluded those projects that are targeted at both end users and developers. However, this could introduce sampling bias in favor of developer-targeted projects if most of the projects targeted at both groups truly focus on developers only.

In the final step of sampling, we dropped the inactive projects and the discontinued projects from the sample because we are only interested in the projects that are under on-going

[1] We do not distinguish between projects written in Java only and those written in multiple languages including Java.

development. The final sample size is 673 projects. Next we developed a web crawler to download and parse the HTML files regarding project summary, developer information, activity statistics, mailing lists, file releases, as well as bug and request tracking systems. To avoid disruptions to the servers of SourceForge we spread the data collection process over a period of six days.

3.2. Measures

This section explains the variables in the exploratory analysis that are based on the constructs as described in our theoretical framework.

Project Performance. There has been a lack of widely accepted success measures for open source projects. We measure the performance of OSS projects on the following three metrics.

Downloads. It is the total number of downloads of the software for all releases. This variable serves as an approximate measure of system use, which is used as an indicator of information systems success in a number of studies (Crowston et al. 2003; Crowston et al. 2004).

Activity. The major drawback of using *Downloads* as a measure of performance is that it can be measured only for the projects that have released software. Projects without any releases have total downloads of 0. Hence, to be able to measure the performance of all projects, we use *Activity* as a second success metric. *Activity* is the activity percentile computed by SourceForge based on the traffic, communication, and development statistics of each project[2] for the week prior to the data collection. Adopting *Activity* as a performance measure is also consistent with Crowston et al. (2004) who suggest that the level of activity of developers in writing code and submitting bug reports may be another indicator of project success.

[2] The activity percentile for a project is the percentile of that project (0-100%) against all other projects with a non-zero result for ranking. SourceForge.net applies a single ranking formula based on the statistics data for all hosted projects.

Efficiency. Even though *Activity* allows evaluating the performance of all projects, it still highly favors those projects with releases over those without releases. To further overcome this drawback and to be able to better evaluate the performance of projects in the earlier stages that have not had any releases yet, we introduced the third performance measure - *Efficiency*. The rational for this metric is that because an open source software project can be viewed as an economic production process, one common way to evaluate the performance of the process is to compute the relative efficiency of decision making units when multiple outputs and multiple inputs make the comparison difficult.

Project Type. We distinguish between developer-focused projects and user-focused projects by using the variable *ProjectType*. We code the project type as a dummy variable that takes the value of 0 for projects whose *Intended Audience* is end user and the value of 1 for those whose *Intended Audience* is developer.

HR Staffing. Based on the discussion in Section 2, we include the following variables that capture the human resource staffing decisions.

Developers. It represents the number of developers working on the project when the data was collected.

Managers. It represents the number of project administrators. For many single-developer projects in our dataset that do not designate a project administrator, we set the number of managers to be 1 since the developer has to be responsible for every aspect of the project and implicitly takes on the role of project manager.

DeveloperExperience. As discussed in section 2, human resource recruitment not only concerns the quantity of developers and managers but also relates to the quality of the personnel. Developers' quality can be represented with their technical skills as well as their experience in

developing open source softwares. The fact that only a small percentage of members on SourceForge make their technical profile viewable to the public limits our ability to assess developers' programming skills in this study. Therefore we use *DeveloperExperience*, approximated by the average number of projects on SourceForge that each developer of the project has participated in since 1999, to capture the quality of the personnel in the development team.

Compensation Management. The incentives for programmers to participate in an open source project include ego gratification, career concern, and so on whereas programmers develop proprietary software primarily for monetary compensation. However, an increasing number of open source projects have opted to receive monetary donations from users. Although some developers and projects choose to allocate part or all of the incoming donations to SourceForge or other organizations, most recipients of the donations rely on the monetary supports to fund development time and other key resources that are necessary for the continuation of the projects. Therefore, we choose to use the variable *AcceptingDonations* to capture whether a project is using monetary compensation as part of its incentive mechanism. It takes the value of 1 if the project is accepting donations and 0 otherwise.

Communication and Coordination. Communication and coordination in projects hosted on SourceForge are mainly accomplished through the use of mailing lists and discussion forums.

We include the following variables to capture whether a project sets up a mailing list with specific purpose. *GeneralList* represents whether the project uses a general-purpose mailing list to communicate with developers and users. *UserList* indicates whether the project sets up a list that mainly discusses user related issues. *DeveloperList* denotes whether the project maintains a mailing list that is devoted to discussions of issues that are of particular interest to developers

only. *NewsList* captures whether a mailing list is used to broadcast project-related news. *CVSList* corresponds to the setup of a list for sending CVS commits and other CVS notifications. All these variables are binary variables that take the value of 1 if the corresponding mailing list is set up for the project and the value of 0 otherwise. We coded these variables based on descriptions and naming of mailing lists. For example, most of user-oriented lists can be identified by their names (i.e. project-name-users).

We also include the following variables to capture the overall interaction between users and developers that can be observed from the discussion forums.

TrackerUsed. This is a binary variable, with 1 indicating that at least one tracker item (i.e. bug report or feature request) has been opened since the project's registration and 0 indicating otherwise.

TrackerRatio. It is the ratio of the number of tracker items (including bug reports and feature requests) closed to the number of tracker items opened since the registration of the project. It has values ranging between 0 and 1.

Release Management. We employ the variable *ReleaseSpeed* to capture one key aspect of release management in open source software development. *ReleaseSpeed* denotes how many releases on average the project has within a period of one month.

Control Variable. We employ one control variable - *DevelopmentStatus* - to capture the development stage of a project, which is typically determined by the developer in charge of the project on SourceForge. *DevelopmentStatus* takes values from 1 to 6 representing development stages of Planning, Pre-Alpha, Alpha, Beta, Production/Stable, and mature respectively. The larger the value of *DevelopmentStatus*, the more mature the project is.

The variables used in the analysis are summarized in Table 1.

Table 1 - Summary of Variables

Variable	Definition
DEPENDENT VARIABLES	
Project Performance	
Downloads	Total number of software downloads on SourceForge.net.
Activity	Activity level of the project calculated by SourceForge's ranking formula.
Efficiency	Relative efficiency score calculated by Data Envelopment Analysis.
INDEPENDENT VARIABLES	
Project Type	
ProjectType	Equals 0 if the project is user-targeted and 1 if the project is developer-targeted.
HR Staffing	
Developers	Number of developers working on the project.
Managers	Number of project administrators.
DeveloperExperience	Average experience of developers in open source software development.
Compensation Management	
AcceptingDonations	1 if the project is accepting donations, 0 otherwise.
Communication and Coordination	
GeneralList	1 if the project uses a general-purpose mailing list, 0 otherwise.
UserList	1 if the project uses a user-oriented mailing list, 0 otherwise.
DeveloperList	1 if the project uses a developer-oriented mailing list, 0 otherwise.
NewsList	1 if the project uses a news-posting mailing list, 0 otherwise.
CVSList	1 if the project uses a CVS-oriented mailing list, 0 otherwise.
TrackerUsed	1 if at least one tracker item has been opened since the project started, 0 otherwise.
TrackerRatio	Ratio of the number of tracker items closed to the number of tracker items opened.
Release Management	
ReleaseSpeed	The number of releases the project has on average within a one-month period.
CONTROL VARIABLE	
DevelopmentStatus	It ranges from 1 to 6 representing development stages of Planning, Pre-Alpha, Alpha, Beta, Production/Stable, and mature respectively.

3.3. Analysis

Data were analyzed in two stages. First we adopted data envelopment analysis (DEA) to determine the relative efficiency of each project, which is one of the project performance variables in the exploratory analysis. In the second stage we used a linear regression model to identify the antecedents to the success of user-targeted projects and developer-targeted projects.

3.3.1. Data Envelopment Analysis

As an important way to evaluate efficiency, data envelopment analysis is a linear programming formulation for frontier analysis that defines a nonparametric relationship between multiple outputs and multiple inputs by building an efficiency frontier (Charnes et al. 1978). The efficient decision making units have an efficiency score of one whereas the inefficient units have an efficiency score less than one but greater than zero.

In the DEA model we use two output variables – *Development Status* and *Size*. *Development Status,* which ranges from 1 to 6 with 1 being the planning stage and 6 being the mature stage, is determined by the person in charge of the project. *Size* is the total number of bytes of all files released for the project. There are two input variables – *Adjusted Number of Developers* and *Project Age*. *Adjusted Number of Developers* is the number of developers (including project managers, developers, translators, etc.) adjusted by the number of projects each developer is working on based on the intuition that the more projects a developer is participating in, the less time he or she can spend on each project. *Project Age* represents the number of days since the registration of the project. Furthermore, we prefer an output-oriented DEA model over an input-oriented model because in the OSS context project managers have difficulty controlling how much time each developer spends on the project and thus tend to be more interested in w the outputs. Lastly, we adopted the BCC model allowing variable returns to scale (Banker et al. 1984) after empirically determining that the production function actually exhibits variable returns to scale (Banker and Slaughter 1997).

3.3.2. Regression Analysis

Past studies related to the concept of fit have used subgroup analysis, in which the sample is split into groups based on the contextual variable, to test the strength of the fit between strategies and environments (Venkatraman 1989). Therefore we adopted subgroup analysis by breaking the sample into two groups representing user-targeted projects and developer-targeted projects and then performed regression analysis for each subgroup.

An initial diagnosis of the regression model reveals that the error variances are not constant across observations. In order to control for possible heteroskedasticity in the sample we estimate the following generalized least squares (GLS) regression models using a heteroskedasticity-consistent covariance matrix (White 1980):

$$y_{user\text{-}focused} = \alpha + \beta_1 Developers + \beta_2 Managers + \beta_3 DeveloperExperience + \beta_4 AcceptingDonations + \sum_i \gamma_i CommunicationCoordination + \beta_5 ReleaseSpeed + \varepsilon$$

$$y_{developer\text{-}focused} = \alpha + \beta_1 Developers + \beta_2 Managers + \beta_3 DeveloperExperience + \beta_4 AcceptingDonations + \sum_i \gamma_i CommunicationCoordination + \beta_5 ReleaseSpeed + \varepsilon$$

where y is one of the two dependent variables representing project performance (i.e., *Efficiency*, and *Downloads*) and γ_i's (i = 1 to 7) represent *GeneralList*, *UserList*, *DevelopterList*, *NewsList*, *CVSList*, *TrackerUsed* and *TrackerRatio* respectively.

Unlike *Efficiency* and *Downloads*, *Activity* captures only a snapshot of a project's performance based on the statistics obtained for the week prior to our data collection. It does not reflect the cumulative performance over the entire duration of the project. Thus, it may be dependent on the development stage of the project. Projects at more mature stages tend to have

higher activity level than those at more preliminary stages. Therefore we include an additional predictor variable - *Development Status* - in the regression model for *Activity*:

$$Activity_{user\text{-}focused} = \alpha + \beta_1 Developers + \beta_2 Managers + \beta_3 DeveloperExperience + \beta_4 AcceptingDonations + \sum_i \gamma_i CommunicationCoordination + \beta_5 ReleaseSpeed + \beta_6 DevelopmentStatus + \varepsilon$$

$$Activity_{developer\text{-}focused} = \alpha + \beta_1 Developers + \beta_2 Managers + \beta_3 DeveloperExperience + \beta_4 AcceptingDonations + \sum_i \gamma_i CommunicationCoordination + \beta_5 ReleaseSpeed + \beta_6 DevelopmentStatus + \varepsilon$$

4. Results

4.1. DEA Results

Efficiency ranges from 0.167 to 1.000 with the mean of 0.535 and standard deviation of 0.250. Out of the 673 projects in the sample, 20 projects lie on the efficiency frontier with an efficiency score of 1.000. User-targeted projects have a mean of 0.461 and developer-targeted projects have a mean of 0.5521 in *Efficiency*.

4.2. Descriptive Results

We present descriptive statistics and pairwise correlations in Table 2 and Table 3 respectively. The correlation table shows that the highest correlation between the independent variables is between *TrackerUsed* and *TrackerRatio* ($\rho = 0.769$, $p < 0.001$). But the other correlations are below 0.37, indicating that our model does not exhibit major multicollinearity problem (Hair et al. 1995). We also check for variance inflation factors (VIF) for the independent variables especially for *TrackerUsed* and *TrackerRatio* in the regression model. All VIFs are within acceptable ranges (Neter et al. 1996).

Table 2 – Descriptive Statistics

Variable	N	Mean	Std. Dev	Min	Max
Efficiency	673	0.53	0.250	0.16	1.00
Downloads	673	5254.00	48004.000	0.00	1147162.00
$Downloads_{ReleaseSpeed>0}$	427	8241.20	60083.387	0.00	1147162.00
Activity	673	66.67	26.664	1.44	99.96
ProjectType	673	0.81	0.392	0.00	1.00
Developers	673	2.67	3.968	1.00	45.00
Managers	673	0.79	0.433	0.00	2.00
DeveloperExperience	673	1.54	2.204	0.00	17.00
AcceptingDonations	673	0.14	0.342	0.00	1.00
GeneralList	673	0.05	0.228	0.00	1.00
UserList	673	0.11	0.307	0.00	1.00
DeveloperList	673	0.16	0.367	0.00	1.00
NewsList	673	0.06	0.245	0.00	1.00
CVSList	673	0.04	0.207	0.00	1.00
TrackerUsed	673	0.33	0.469	0.00	1.00
TrackerRatio	673	0.18	0.335	0.00	1.00
ReleaseSpeed	673	0.12	0.285	0.00	4.31

Table 3 – Correlations

Variable	Efficiency	Downloads	Activity	Project Type	Developers	Managers	Developer Experience
Efficiency							
Downloads	0.140***						
Activity	0.557***	0.120***					
ProjectType	0.142***	0.038	0.079**				
Developers	0.077**	0.409***	0.176***	0.017			
Managers	-0.025	-0.059	-0.029	-0.053	-0.099**		
Developer Experience	0.196***	0.013	0.201***	0.035	-0.072*	0.013	
Accepting Donations	0.141***	0.009	0.216***	0.002	-0.001	0.037	0.020
GeneralList	0.047	0.010	0.052	0.050	0.220***	-0.006	0.007
UserList	0.151***	0.062	0.191***	0.030	0.099**	0.017	0.044
DeveloperList	0.032	0.142***	0.113***	0.097**	0.247***	-0.092**	0.024
NewsList	0.070*	0.063	0.068*	0.048	0.037	-0.031	0.016
CVSList	0.073*	0.260***	0.128***	0.067*	0.372***	-0.031	-0.046
TrackerUsed	0.368***	0.144***	0.432***	0.085**	0.276***	-0.087**	0.077**
TrackerRatio	0.357***	0.181***	0.394***	0.076**	0.258***	-0.064*	0.065*
ReleaseSpeed	0.369***	0.405***	0.379***	0.081**	0.215***	-0.028	0.049

Variable	Accepting Donations	General List	UserList	Developer List	NewsList	CVSList	Tracker Empty	Tracker Ratio
Efficiency								
Downloads								
Activity								
ProjectType								
Developers								
Managers								
Developer Experience								
Accepting Donations								
GeneralList	0.038							
UserList	0.076**	-0.062						
DeveloperList	-0.031	0.019	0.272***					
NewsList	0.057	0.044	0.227***	0.200***				
CVSList	0.062	0.074*	0.113***	0.219***	0.061			
TrackerUsed	0.114***	0.110***	0.183***	0.170***	0.064*	0.172***		
TrackerRatio	0.173***	0.091***	0.173***	0.165***	0.090**	0.159***	0.769***	
ReleaseSpeed	0.195***	-0.002	0.155***	0.050	0.014	0.203***	0.301***	0.369***

Significance levels: *** 0.01, ** 0.05, * 0.1

4.3. Regression Results

The results of the regression for user-targeted projects and developer-targeted projects are summarized in Table 4. All six regression models show good level of fit. Overall the models exhibit better fit for user-targeted projects than for developer-targeted projects. For user-focused

projects the model achieves 31.76% adjusted R-square for *Efficiency*, 49.78% for *Downloads*, and 50.03% for *Activity*. For developer-focused projects the model has 24.85% adjusted R-square for *Efficiency*, 34.15% for *Downloads*, and 39.54% for *Activity*.

The results for user-targeted projects show that *ReleaseSpeed* has a significantly positive impact on all three performance measures (β_5 $^{\text{Efficiency}} = 0.81$, $p < 0.01$; β_5 $^{\text{Downloads}} = 29899.30$, $p < 0.05$; β_5 $^{\text{Activity}} = 53.01$, $p < 0.01$). *Developers*, *DeveloperExperience*, *AcceptingDonations* and *UserList* have significant impacts on two performance measures. *Developers* significantly and negatively impacts *Efficiency* but positively impacts *Downloads* (β_1 $^{\text{Efficiency}} = -0.01$, $p < 0.01$; β_1 $^{\text{Downloads}} = 1080.88$, $p < 0.10$). The parameter estimates for *DeveloperExperience* and *AcceptingDonations* are significant and positive when regressed on *Efficiency* and *Activity* (β_3 $^{\text{Efficiency}} = 0.02$, $p < 0.05$; β_3 $^{\text{Activity}} = 2.41$, $p < 0.01$; β_4 $^{\text{Efficiency}} = 0.10$, $p < 0.05$; β_4 $^{\text{Activity}} = 10.75$, $p < 0.01$). *UserList* has a significant and positive effect on both *Downloads* and *Activity* (γ_2 $^{\text{Downloads}} = 8379.98$, $p < 0.05$; γ_2 $^{\text{Activity}} = 7.71$, $p < 0.10$). In addition, variables *DeveloperList, NewsList*, *CVSList* and *TrackerUsed* have significant impacts on one performance measure (γ_3 $^{\text{Downloads}} = -3793.15$, $p < 0.10$; γ_4 $^{\text{Efficiency}} = -0.11$, $p < 0.01$; γ_5 $^{\text{Efficiency}} = -0.18$, $p < 0.01$; γ_6 $^{\text{Activity}} = 13.39$, $p < 0.01$).

The results for developer-targeted projects indicate that no factor has a significant impact on all three performance metrics. However, factors including *DeveloperExperience*, *UserList*, *NewsList*, *TrackerUsed* and *ReleaseSpeed* significantly and positively affect two performance measures. *DeveloperExperience*, *UserList*, *TrackerUsed* and *ReleaseSpeed* positively impact both *Efficiency* and *Activity* (β_3 $^{\text{Efficiency}} = 0.01$, $p < 0.01$; β_3 $^{\text{Activity}} = 1.21$, $p < 0.01$; γ_2 $^{\text{Efficiency}} = 0.07$, $p < 0.05$; γ_2 $^{\text{Activity}} = 4.13$, $p < 0.10$; γ_6 $^{\text{Efficiency}} = 0.12$, $p < 0.01$; γ_6 $^{\text{Activity}} = 9.25$, $p < 0.01$; β_5 $^{\text{Efficiency}} = 0.29$, $p < 0.01$; β_5 $^{\text{Activity}} = 19.21$, $p < 0.01$). *NewsList* affects *Efficiency* and *Downloads*

positively ($\gamma_4^{\text{Efficiency}} = 0.06$, $p < 0.10$; $\gamma_4^{\text{Downloads}} = 11556.55$, $p < 0.10$). In addition, variables *Developers* and *AcceptingDonations* have positive and significant impacts on *Downloads* and *Activity* respectively ($\beta_1^{\text{Downloads}} = 4241.56$, $p < 0.10$; $\beta_4^{\text{Activity}} = 7.82$, $p < 0.01$).

In summary, the aspects of project management that impact project performance exhibit both similarities and differences for user-targeted projects and developer-targeted projects. Release management and HR staffing have a significant impact on both types of projects. Hence regardless of the target audience of the project, controlling the speed of software releases and recruiting project core developers should be managed with care. However, compensation management is particularly important to user-targeted projects whereas communication and coordination is particularly essential to developer-targeted projects.

Additionally, we derive some interesting observations based on the results. First, the significant and positive impact of *ReleaseSpeed* on almost all three performance metrics for both types of projects confirms Raymond's argument that open source software projects need to release frequently (Raymond 2000). Second, *DeveloperExperience* seems to positively affect *Efficiency* and *Activity* for all projects, indicating that a project with developers who have worked on open source projects before has better performance than a project consisting of developers unfamiliar with open source development. Third, although it is not surprising to observe that *AcceptingDonations* positively impacts the performance of user-focused projects based on the discussion in section 2, it is interesting to see that this factor also influences the activity level of developer-focused projects. User-oriented projects that are open to donations can offer monetary incentive to developers to encourage their participation. In developer-oriented projects developers may feel grateful for donators' appreciation and hence become more actively involved in the projects. Fourth, having more developers on the project seems to

increase the total number of downloads of the project, which is consistent with some prior empirical studies on open source (Krishnamurthy 2002). However, for user-targeted projects the project manager may need to make a trade-off between efficiency of software production process and total downloads of the software because having more developers attracts more people to try the product but actually decreases project efficiency.

Table 4 - Regression Results for User-Targeted Projects and Developer-Targeted Projects

Independent Variables	Dependent Variables (Project Performance)					
	User-Targeted Projects			Developer-Targeted Projects		
	Efficiency	Downloads	Activity	Efficiency	Downloads	Activity
Intercept	0.36***	-5152.97**	36.20***	0.46***	-12616.20*	33.81***
Developers	-0.01***	1080.88*	0.11	-0.00	4241.56*	0.26
Managers	0.01	2093.88	-0.97	-0.01	-1369.68	0.75
DeveloperExperience	0.02**	10.33	2.41***	0.01***	298.41	1.21***
AcceptingDonations	0.10**	-1013.60	10.75***	0.01	-8443.59	7.82***
GeneralList	0.00	3391.84	-10.92	0.05	-16187.20	0.69
UserList	0.03	8379.98**	7.71*	0.07**	-6283.88	4.13*
DeveloperList	-0.07	-3793.15*	10.81	-0.03	4782.84	1.41
NewsList	-0.11***	-2432.15	-2.24	0.06*	11556.55*	-0.36
CVSList	-0.18***	1132.18	-1.60	-0.01	8567.54	-0.49
TrackerUsed	0.03	-3077.22	13.39***	0.12***	-6510.87	9.25***
TrackerRatio	0.11	2468.88	-5.19	0.04	-3381.48	1.00
ReleaseSpeed	0.81***	29899.30**	53.01***	0.29***	90424.31	19.21***
DevelopmentStatus	N/A	N/A	5.27***	N/A	N/A	7.04***
R-Square	38.26%	54.56%	55.18%	26.52%	35.61%	40.99%
Adjusted R-Square	31.76%	49.78%	50.03%	24.85%	34.15%	39.54%
F Statistic	5.89***	11.41***	10.70***	15.94***	24.43***	28.26***
Sample Size (N)	127			543 [3]		
Significance levels: *** 0.01, ** 0.05, * 0.1						

5. CONCLUSIONS AND IMPLICATIONS

This paper proposes a theoretical framework that stresses the fit between project characteristic and project management. We argue that in order for a project to be successful project management needs to be aligned with project characteristics in terms of target audience. We also conduct empirical analysis on 673 OSS projects to gain some insights into the factors

[3] Three outliers were identified and removed from the analysis.

and aspects of project management that impact project performance for user-targeted projects and developer-targeted projects.

This paper makes both theoretical and practical contributions to the open source literature. Theoretically, it proposes a model focusing on the fit between project type and project management, an issue that has never been addressed in the open source area. We emphasize that there exist fundamental differences between user-focused projects and developer-focused projects and that they should be managed accordingly. Practically, the factors included in this study are under the control of project managers or organizations starting an OSS project. Thus, the results can provide practitioners with some actionable insights as to how to manage different types of OSS projects. Specifically, for user-focused projects administrators need to pay extra attention to release management (i.e. how fast to release software), HR staffing (i.e. how many developers to recruit or allow into the core developing team and their level of experience in open source), and compensation management (i.e. whether to accept donations from users). Administrators of developer-focused projects need to carefully manage release management, HR staffing (i.e. how much open source experience developers should have), as well as communication and coordination (i.e. use of mailing lists to communicate with users of software and to broadcast project news, use of bug tracking and feature request tracking systems).

In addition, many prior studies on open source development focus on successful, mature, or large projects (Crowston et al. 2004; Godfrey and Tu 2000; Krishnamurthy 2002; Mockus et al. 2000; Nakakoji et al. 2002). We included both successful and unsuccessful, mature and less mature, large and small projects in our analysis to gain a holistic view on the performance of open source projects. Furthermore, total number of downloads has been frequently used as a performance measure in open source, which limits the scope of studies to those projects with

released software. We used data envelopment analysis to evaluate the efficiency of the software production process, which is one of the performance metrics in this paper. Thus we are able to study not only the projects with released products but also those without any releases.

This study is not without limitations. First of all, we limited the sample to those projects in database category that are written in Java and operating system independent. We plan to collect a more complete set of data and improve the generalizability of the results. Secondly, given that open source software development involves numerous aspects that can be studies from multiple perspectives, we may have left out some important variables that impact project performance. For example, choice of license type has been found to have performance implications in some studies (Lerner and Tirole 2005). Furthermore, we use project as unit of analysis and do not incorporate product life cycle into our model. It would be interesting to examine the impact of the fit between project type and management on performance at different stages of the product life cycle. Thirdly, Crowston et al. (2003) propose recognition, project influence, user involvement, and porting as success measures for OSS projects based on a model of IS success and a survey of open source developers. We plan to obtain some objective measures from project developers through a survey and incorporate them into the analysis.

Reference:

Argote, L. "Input Uncertainty and Organizational Coordination in Hospital Emergency Units," *Administrative Science Quarterly* (27) 1982, pp 420-434.

Banker, R.D., Charnes, A., and Cooper, W.W. "Some Models for Estimating Technical and Scale Inefficiencies in Data Envelopment Analysis," *Management Science* (30:9) 1984, pp 1078–1092.

Banker, R.D., and Slaughter, S. "Field Study of Scale Economies in Software Maintenance," *Management Science* (43:12) 1997, pp 1709-1725.

Bezroukov, N. "Open Source Software Development as a Special Type of Academic Research (Critique of Vulgar Raymondism)," *First Monday* (4:10) 1999.

Charnes, A., Cooper, W.W., and Rhodes, E.L. "Measuring the Efficiency of Decision Making Units," *European Journal of Operational Research* (2:6) 1978, pp 429 – 444.

Crowston, K., Annabi, H., and Howison, J. "Defining Open Source Software Project Success," Proceedings of ICIS 2003, Seattle, WA, 2003.

Crowston, K., Annabi, H., Howison, J., and Masango, C. "Towards A Portfolio of FLOSS Project Success Measures," The 4th Workshop on Open Source Software Engineering, International Conference on Software Enginnering (ICSE 2004), Edinburgh, Scotland, 2004.

Drazin, R., and Van de Ven, A.H. "Alternative Forms of Fit in Contingency Theory," *Administrative Science Quarterly* (30) 1985, pp 514-539.

Duncan, W. *A Guide to the Project Management Body of Knowledge* Project Management Institute Standards Committee, 1996.

Fry, L.W., and Smith, D.A. "Congruence, Contingency and Theory Building," *Academy of Management Review* (12:1) 1987, pp 117-132.

Godfrey, M., and Tu, Q. "Evolution in Open Source Software: A Case Study," Proceedings of 2000 International Conference on Software Maintenance San Jose, CA, 2000.

Greenemeier, L. "Open Doors To Innovation -- Small and midsize companies are creating IT infrastructures based on open-source software to reduce licensing fees and increase flexibility," in: *InformationWeek*, CMP Media LLC 2005.

Hair, J.F., Anderson, R.E., Tatham, R.L., and Black, W.C. *Multivariate Data Analysis with Readings*, (4th ed.) Prentice Hall, Englewood Cliffs, NJ, 1995.

Hars, A., and Qu, S. "Working for Free? Motivations for Participating in Open-Source Projects," *International Journal of Electronic Commerce* (6:3), Spring2002 2002, p 25.

Howison, J., and Crowston, K. "The Perils and Pitfalls of Mining Sourceforge," In Proceedings of Mining Software Repositories Workshop, International Conference on Software Enginnering (ICSE 2004), Edinburgh, Scotland, 2004.

Jones, C. *Programming Productivity* McGraw-Hill, New York, 1986.

Koch, S. "Profiling an Open Source Project Ecology and Its Programmers," *Electronic Markets* (14:2), Jun2004 2004, pp 77-88.

Krishnamurthy, S. "Cave or Community? An Empirical Examination of 100 Mature Open Source Projects," *First Monday* (7:6), June 2002.

Lerner, J., and Tirole, J. "Some Simple Economics of Open Source," *Journal of Industrial Economics* (50:2) 2002, pp 197-234.

Lerner, J., and Tirole, J. "The Scope of Open Source Licensing," *Journal of Law Economics & Organization* (21:1), Apr 2005, p 20.

Mockus, A., Fielding, R., and Herbsleb, J. "A Case Study of Open Source Software Development: The Apache Server," Proceedings of 2000 International Conference on Software Engineering (ICSE2000) Limerick, Ireland, 2000, pp. 263-272.

Nakakoji, K., Yamamoto, Y., Nishinaka, Y., Kishida, K., and Ye, Y. "Evolution Patterns of Open-Source Software Systems and Communities," Proceedings of 2002 International Workshop on Principles of Software Evolution, 2002, pp. 76-85.

Neter, J., Kutner, M.H., Nachtsheim, C.J., and Wasserman, W. *Applied Linear Statistical Models*, (4th ed.) Irwin, Chicago, IL, 1996.

Raymond, E.S. "The Cathedral and the Bazaar," 2000.

Van de Ven, A.H., and Ferry, D.L. "The Concept of Fit in Contingency Theory," in: *Research in Organizational Behavior,* B.M. Staw and L.L. Cummings (eds.), JAI Press, Greenwich, CT, 1985, pp. 333-365.

Venkatraman, N. "The Concept of Fit in Strategy Research: Toward Verbal and Statistical Correspondence," *Academy of Management Review* (14:3) 1989, pp 423-444.

White, H. "A Heteroskedasticity-Consistent Covariance Matrix Estimator and a Direct Test for Heteroskedasticity," *Econometrica* (48:4), May 1980, pp 817-838.

Xu, J., Gao, Y., Christley, S., and madey, G. "A Topological Analysis of the Open Source Software Development Community," Proceedings of 38th Hawaii International Conference on System Sciences (HICSS 05), Hawaii, USA, 2005.

Both open source software projects in their size and their programmers' effort differ significantly, and the evolution of projects' size over time seems in part to contradict the laws of software evolution proposed for commercial systems. Both the inequality of effort distribution between programmers and an increasing number of developers in a project do not lead to a decrease in productivity, opposing Brooks's Law. Effort estimation based on the COCOMO model for commercial organizations shows a large amount of effort expended for the projects, while a more general Nordenâ€“Rayleigh modeling shows a distinctly smaller expenditure.Â An exploratory study of open source projects from a project management perspective. Jungpil Hahn Krannert. In this roundup of open source project management tools, we look at software that helps support Scrum, Kanban, and other agile methods.Â For the project managers running the show, Gitlab's interface is intuitive, consistent, and robust. A project manager doesn't ever have to leave Gitlab. All tasks, from assigning tasks and planning sprints to accepting merge requests and deleting branches, can be done from within the Gitlab UI. As Gitlab continues to develop, there's every reason to believe that even more features for project management will be added. The Gitlab developers are happy to take feature requests and are quick to respond to feedback. Research on open source software has focused mainly on the motivations of open source programmers and the organization of open source projects (Kogut and Metiu, 2001 and Lerner and Tirole, 2002). Some researchers portray open source as an extension of the earlier open systems movement (West and Dedrick, 2001). While there has been some research on open-systems software adoption by corporate MIS organizations (Chau and Tam, 1997) the issue of open source adoption has received little attention. We use a series of interviews with MIS managers to develop a grounded theory of open source platform adoption.Â Dedrick, Jason and West, Joel, "An Exploratory Study into Open Source Platform Adoption" (2004). Faculty Publications. Paper 1. This project has carried out an exploratory study into the use of ERP systems, within Hawke's Bay New Zealand. ERP systems make up a major investment and undertaking by those companies. Therefore, research and lessons learned in this area are very important. In addition to a significant initial literature review, this project has conducted a survey on the local users' experience with Microsoft Dynamics NAV (a popular ERP brand). As a result, this study will contribute new and relevant information to the literature on business information systems and to ERP systems, in particular. 1. Project Ai... The project management framework provides structure and direction to a project. However, unlike project management methodologies it is neither too detailed nor too rigid. Frameworks guide projects to their goal while being flexible enough to adapt to evolving conditions.Â Leaves room to include other practices and tools. Cannot be embedded with other practices and tools. Traditional project management (PMBOK) is a framework. PRINCE2 is a well-known project management methodology. While these two project management approaches are inherently different, when it comes to applying those definitions, there is a general disagreement in the project management community.