THE ROLE OF OWNERSHIP AND SOCIAL IDENTITY IN PREDICTING DEVELOPER TURNOVER IN OPEN SOURCE SOFTWARE PROJECTS

by

Pratyush Nidhi Sharma

B.E. Computer Science, VTU, India, 2005

M.S. Computer Science, SUNY at Buffalo, 2008

Submitted to the Graduate Faculty of

Joseph M Katz Graduate School of Business in partial fulfillment

of the requirements for the degree of

Doctor of Philosophy

UNIVERSITY OF PITTSBURGH

JOSEPH M KATZ GRADUATE SCHOOL OF BUSINESS

This dissertation was presented

by

Pratyush Nidhi Sharma

It was defended on

July 22nd, 2013

and approved by

Brian Butler, PhD, Associate Professor, University of Maryland

John Hulland, PhD, Professor, University of Georgia

Kevin Kim, PhD, Associate Professor

Sherae Daniel, PhD, Assistant Professor

Thesis Director: Dennis Galletta, PhD, Professor

THE ROLE OF OWNERSHIP AND SOCIAL IDENTITY IN PREDICTING DEVELOPER TURNOVER IN OPEN SOURCE SOFTWARE PROJECTS

Pratyush Nidhi Sharma, PhD

University of Pittsburgh, 2013

Open source software (OSS) development methodology that promises to produce reliable, flexible, and high quality software code, at minimal cost, by harnessing the power of distributed peer review and transparency of process and has become increasingly popular in the past few years. For-profit companies have increasingly adopted the OSS paradigm to produce quality software at low cost. A vast majority of OSS projects depend on voluntary contributions by developers to sustain their development. In this context, turnover of developers has been considered a critical issue hindering the success of projects. This dissertation develops two studies addressing the issue. The first study is a methodological pilot and lays the foundation of this research by focusing on modeling turnover behavior of core open source contributors using a logistic hierarchical linear modeling approach. It argues that argue that taking both the developer and the project level factors into account will lead to a richer understanding of the issue of turnover in open source projects. The second study provides a conceptual integration of developer and project level factors using the Ownership, Role theory and Social Identity literatures, and proposes testable hypotheses, methods and findings. The implications of this research are likely to benefit OSS managers in understanding the developer and project level factors associated with developer turnover and the contexts in which they interact.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

PREFACE

We wish to express our deepest gratitude to the committee members, Dr. Dennis Galletta, Dr. Brian Butler, Dr. John Hulland, Dr. Kevin Kim and Dr. Sherae Daniel, for their invaluable guidance throughout. Special gratitude is reserved for Dr. Rachel Chung, Dr. Richard Moreland and Ms. Carrie Woods. Without the help of this exceptionally kind group of people, this thesis would not have been possible. Finally, we wish to thank the Joseph M. Katz Graduate School of Business and the University of Pittsburgh for providing a stimulating and supportive atmosphere.

1.0 INTRODUCTION

Open Source is a software development methodology that promises to produce reliable, flexible, and high quality software code, at minimal cost, by harnessing the power of distributed peer review and transparency of process, thereby preventing predatory vendor lock-in[1]. In addition to the access to source code, Open Source Software (OSS) must meet the criteria of, among others, free redistribution, ability to modify the source code and create modified works[2.] The OSS phenomenon has been of great interest to the field of Information Systems since it has emerged as a viable and successful alternative to the conventional forms of software production that have traditionally been the focus of the field.

The existing research in OSS can be broadly organized in three streams. The first stream of research has focused on the motivations of OSS participants to contribute, especially since participants may not receive any financial compensation for their efforts. This research has established that participants are motivated due to a variety of reasons including the need for software, enjoyment, learning, altruism, ideological commitment, peer recognition and reputation building (e.g. Feller et al. 2005; Lakhani and Wolf 2003; Shah 2006). A second stream of research has studied the coordination mechanisms, decision making and management practices

[1] http://opensource.org/

[2] http://opensource.org/docs/OSD

prevalent in OSS (e.g. O’Mahony and Ferraro 2007; Mockus et al. 2003; Crowston et. al. 2005). Finally, the third stream of existing research has focused on the quality and success of the end-product (i.e. software) and the socio-technical factors affecting it (e.g. Crowston et al. 2006; Chengalur-Smith et al. 2010; Setia et. al. 2010; Grewal, et al. 2006, Singh 2010, Singh, et al. 2011; Daniel et. al. 2012).

Open Source Software communities are often cited as prime examples of modern, online, community based forms of production that have been recognized by organizational theorists as alternatives to the traditional market and hierarchical forms of organization and production (O’Mahony and Ferraro, 2007). Such communities depend on voluntary contributions to create software either for public or private benefit (von Hippel and von Krogh, 2003), shun bureaucracy and authoritarian forms of governance (Rothschild-Whitt, 1979), and encourage democratic participation of members involved in production (Rothschild and Russell, 1986). A critical issue in this context therefore, is how to organize the production of software by directing individuals’ efforts toward a common goal without any contractual or hierarchical reinforcement (O’Mahony and Ferraro, 2007). Furthermore, it is not trivial to align individual motives and goals with a common objective (March and Simon, 1958), and hence achieve efficiency in software production. Such democratic forms of organization face difficulty in decision making and coordinating and most importantly, in sustaining member activities. Thus, it is essential to create and nurture a consensual basis of authority to facilitate the management and direction of development and sustain developer interest and commitment (O’Mahony and Ferraro, 2007). This issue assumes an even critical importance in the light of the fact that most OSS projects fail to develop due to a lack of a critical mass of developers needed for sustained development (Chengalur-Smith et. al., 2010).

Indeed, developer turnover in Open source software (OSS) projects is a non-trivial issue because of the frequency with which it occurs and its negative impact on project performance. Turnover is specified as voluntary job termination (Sheridan 1985) or more simply as an employee leaving a current job (Fields et al. 2005). Robles and Gonzales-Barahona (2006) analyzed the evolution of some OSS projects (e.g., GIMP, Mozilla) over 7 years and found that these projects suffered from yearly turnover in core development teams and had to rely heavily on regeneration. Similar results on turnover were reported by von Krogh et al. (2003) in their analysis of the Freenet project.

Turnover has been recognized as a critical issue in organization research due to its adverse effects on firms' productivity levels (e.g. Ton and Huckman 2008; Shaw 2011). Turnover is also a critical problem in software development projects because it can lead to schedule overruns (Collofello et al. 1998) and regenerating teams is a complicated issue (Reel 1999). Regeneration is challenging in OSS development because of the "contribution barrier" where newcomers face difficulty in acclimatizing themselves with the complex architecture of the project (Crowston et al. 2004; von Krogh et al. 2003). Once participants overcome this contribution barrier, it is in the best interest of the project to retain them. Therefore, there is intense competition for participants among OSS projects (Ahokas and Laurila 2004; Krishnamurthy 2005).

Past research has studied how OSS communities evolve organization structures for better effort management and coordination (O'Mahony and Ferraro, 2007; Mockus et al. 2002; von Krogh et al., 2003). This body of research has consistently pointed to the gradual introduction of ownership (access to rights) to developers based on meritocracy and expertise. Developers generally start contributing as peripheral members and are accorded progressively higher set of rights, duties and ownership of code modules based on their demonstration of commitment and expertise[3]. Google Code for example, allows developers to be owners (co-owners), committers and contributors[4]. Thus, owners may be "sole owners" or "part owners" depending upon the ownership structure within the project. The owners enjoy the highest levels of rights and have the ability to control the project structure, code and workforce. Committers' rights are restricted to making changes to the code, but not controlling the overall structure of the code or the workforce. Contributors on the other hand, may only comment and point out issues; however their rights may be upgraded and they may be allowed to make code commits. Such an organization allows the module owners to oversee the efficient development of code and grant rights and duties to developers to make changes to the code.

However, the impact of the organization structure of OSS communities on member retention is an often a neglected area of work in OSS research. The implicit assumption in the existing OSS studies is that ownership provides prestige to the owners and makes them more committed, but what has not been studied is under what conditions do we expect to see such positive effects of developer ownership levels on their attitudes such as turnover. Expecting a simple positive main effect of ownership level on retention, however, may be too simplistic due to the complex nature of the ownership construct and the novelty of the OSS context, as we outline below.

---

[3] Different projects may follow different strategies to accord ownership rights. Apache project for example, follows an emergent ownership structure where some participants emerged as "de facto" owners of modules through their continued code development and commitment to a module. Mozilla on the other hand, follows an "enforced" ownership strategy where every change to the code is reviewed by the module owner (Mockus et al. 2002).

[4] http://code.google.com/p/support/wiki/Permissions

Sociology, Psychology and Organizational Behavior literature have studied the issue of formal and psychological ownership as an organizational arrangement and employee participation. Since the 1970s, there has been continued interest in employee ownership, anchored in the hope that it would promote favorable worker attitudes, strengthen industrial democracy and enhance firm performance (Pierce and Rodgers, 2004). On the one hand, researchers following the macro-tradition in this area have tended to focus on the positive effects of employee ownership on firm level outcomes such as productivity. On the other hand, researchers following the micro-tradition have treated the individual as the level of analysis and have sought to observe a simple main effect of level of ownership (equity or share stakes) and their level of performance (Pierce and Rodgers, 2004). These models suggest that this ownership-performance relationship is an outgrowth of, among other things, economic incentives (Conte and Svejnar, 1990) and favorable employee-owner attitudes (Long, 1980). However, in contrast to positive effect models, there are models that have reported negative effects mainly due to increased monitoring costs and the free-rider effect (Blasi et al. 1996; Conte and Svejnar, 1990).

In addition, the ownership literature does not offer concrete evidence and the process involved that can help differentiate among the effect of sole ownership and co-ownership on employee performance. On the one hand, it may be argued that the effect of sole ownership on developer retention rates may be greater than the effect of co-ownership because sole ownership allows for a greater degree of control, authority, and sense of responsibility. Along similar lines, Wagner and Rosen (1985) argued that as the actual amount of ownership increases, the incentives associated with making the project may also increase. On the other hand, Long (1978) argued that when ownership leads to an integration of an employee in the organization, performance (and hence retention rates) should increase as a result of peer pressure, motivation and cooperative behaviors. This suggests that the effect of co-ownership on retention rates occur possibly due to different mechanisms (e.g. peer pressure) than sole ownership. Therefore, it is unclear whether we may expect greater effects of sole ownership on retention rates as compared to co-ownership (or vice-versa) and under which conditions. Overall, Pierce and Rodgers (2004) note that the effects of ownership on employee attitudes are mixed and depend to a great extent on contextual effects.

In the OSS context, a critical component of the traditional perspective on ownership-performance relationship (economic incentive) is attenuated or missing, since most developers work voluntarily on projects without any financial compensation. Additionally, the traditional ownership models have assumed that an employee works for a given company. However, in the OSS context developers are not restricted to maintain their association to only a single project at any given time. Hence developers may simultaneously be involved in multiple projects and may have different ownership levels in each. Therefore, assuming homogeneity among developers based on their ownership levels in the project may provide an inaccurate picture. In addition, each project may have different characteristics that may impact developers' willingness to maintain the association with the project. For example, projects may differ on the ratio of number of owners and the number of developers. Given these issues, we argue that OSS research needs to study the effect of ownership levels on developer turnover under the varying contextual surroundings of developer level and project level characteristics.

The existing research on OSS developer motivation has tended to focus on the explanation of developer activity levels using either the individual perspective (Hars et al. 2003; Hertel et al., 2003) or the project perspective (Stewart and Gosain 2006; Stewart et al. 2006).

Empirically, however, since OSS participants are embedded in (often multiple) projects it is important to relate the characteristics of individuals and the characteristics of projects in which they function. Previous studies have neglected this important distinction (however, see Setia et. al., 2010 for an exception).

Disaggregating all project level variables in an individual level analysis may lead to the violation of the assumption of independence of observations, since all developers will have the same value on each of the project variables. On the other hand, aggregating developer level variables to a project level analysis may lead to unused within group information (Raudenbush and Bryk 2002). Including project level variables in a developer level model is likely to create aggregation bias, which can underestimate the effects of variables that are estimated at the inappropriate level. While including aggregated values of developer level variables in a project level model may fail to fully capture the effects of certain variables (Rumberger 1995). None of the research studies have attempted to model turnover behavior in OSS in a comprehensive fashion taking into account both the developer level and project level factors.

In addition, OSS developers do not work in vacuum; rather they work in a very dynamic environment and may fluidly move across multiple projects, contribute and play different roles in them simultaneously. Therefore, it is reasonable to expect that both the ownership levels and the project level characteristics may determine developers' interactions with projects.

Past research has convincingly established the adverse impacts of developer turnover on the performance of software projects (Hall et al. 2008, Abdel-Hamid, 1992). However, we are interested in exploring the inverse relationship, i.e. whether the performance of a software project itself may contribute to developer turnover. If this is so, then this may create a feedback mechanism through which the prospect of sustaining the project may quickly spiral down. In Psychology literature, Social Identity Theory (SIT) posits that people tend to classify themselves and others into various social categories such as group membership, religious affiliation etc (Tajfel and Turner, 1985). Such a social classification allows an individual to define him or herself in the social environment. Thus, social identification is the perception of oneness or belongingness to some human aggregate and results in the individual perceiving the fate of a group as their own. SIT maintains that the need for a positive identity among individuals produces strong reactions when that identity is threatened, such as in the case of group failure. When the social identity is unsatisfactory, individuals will either strive to leave their existing group and join some more positively distinct group or make their group more positively distinct by contributing (Tajfel and Turner, 1985). Thus, SIT proposes a theoretical mechanism to test the effect of project (group) performance on the mean turnover rate. Therefore, using the Ownership literature and the Social Identity Theory (Tajfel and Turner, 1985), we hope to address the following broad research questions:

- How does a developer's ownership level influence turnover from a given open source project, given that a developer may appear in multiple projects at once?
- What other developer level factors moderate the relationship between developer's ownership level and turnover from a given project, given that a developer may appear in multiple projects at once?
- Does project level success or failure moderate the relationship between developer's ownership level and turnover from a given project, given that a developer may appear in multiple projects at once?

The dissertation is organized as follows. In the next section, we develop a methodological pilot study that focuses on modeling turnover behavior of core open source contributors using a logistic hierarchical linear modeling approach. Here, we argue that taking both the developer and the project level factors into account will lead to a richer understanding of the issue of turnover in open source projects. This study allowed us to explore whether there exist significant variation in turnover among OSS projects and whether this variation may be explained using developer and project level characteristics. We note the implications and deficiencies in this pilot study and propose that further enhancements leading to a *conceptual* integration of developer and project level factors in modeling turnover would lead to a richer understanding rather than just an *empirical* integration.

In the second (main) study we explore such a conceptual integration using the Ownership and Social Identity Theory literatures, propose testable hypotheses, and describe the empirical methodology and results. This study also compares some alternative ways to measure turnover. Finally, we conclude with the weaknesses, and overall research and practical implications of this thesis.

2.0 STUDY 1: EXAMINING TURNOVER IN OPEN SOURCE SOFTWARE PROJECTS USING LOGISTIC HIERARCHICAL LINEAR MODELING APPROACH[5]

This pilot study develops a model of turnover behavior in OSS. The analysis focuses on two levels: the developer level, which examines factors that may affect developers' decisions to become inactive, and the project level, which examines the factors that may influence the rates of turnover among projects. Specifically, we hope to address the following research questions:

- Do the open source projects vary in their mean turnover rates (intercepts)?
- What developer level factors influence turnover from open source projects?
- What project level factors influence the mean turnover rates among projects?

Answering these questions allows us to motivate our larger goal of studying turnover in OSS. In what follows, we present a brief theoretical background of the developer and project level factors that may affect mean turnover rates among projects. This is followed by the methodology section where we outline the empirical methods used in the study and present the preliminary results. Finally we conclude by noting the limitations of this work and suggesting the steps we intend to take in the future to further improve this study.

[5] This pilot study has appeared as the following book chapter: Sharma, P.N., Hulland, J. and Daniel, S. "Examining Turnover in Open Source Software Projects Using Logistic Hierarchical Linear Modeling Approach," in I. Hammouda et. al. (Eds): Open Source Systems: Long Term Sustainability, IFIP Advances in Information and Communication Technology (OSS 2012), v. 378, pp. 331-337, Springer Berlin Heidelberg, 2012.

2.1 THEORETICAL BACKGROUND

Research on participation in OSS projects has mainly focused on two different levels of analysis. The first, focusing on individual level addresses the general question: What factors motivate developers to contribute to projects? A second strand of research focuses on the project level factors. The focus of this research has been: What project level characteristics explain activity levels in projects? Past studies have tended to focus on one level or the other, but in order to understand why there is such widespread turnover in OSS projects, it is necessary to consider both perspectives (Rumberger 1995). In what follows, we describe the developer level and project level factors that may influence turnover behavior in OSS projects.

2.1.1 Developer Level Factors

Central to our discussion of the developer perspective is the notion of contribution barrier (von Krogh et al. 2003). If the developer does not have the required knowledge and skills to contribute to the project then the effort level required by him/her increases. In their study of Freenet, von Krogh et al. (2003) found that among other skills the knowledge of the programming language also erected barriers for beginning participants. They confronted the need to learn the language before they could contribute, thereby increasing the effort required. The knowledge of software architecture and the development processes also raise this barrier (Crowston et al. 2004). If the cost of effort required to contribute to the project is excessively high as perceived by the participant, s/he may not be motivated to continue contributing to the project. As per Crowston and Fagnot (2008), the higher the domain knowledge of the developer, the lesser is the effort needed to contribute and higher the motivation. If an employee (or an OSS participant) does not have the requisite resources to meet the demands of the organization (or an OSS project) his/her performance will suffer and s/he is more likely to quit (Kristof-Brown et al. 2005). Developers with longer tenure are more likely to have scaled contribution barrier because they have had more time to develop programming skills and to acclimatize themselves with the complex architecture of the project and its requirements. Therefore we expect that developers with longer tenure are more likely to remain active in projects.

Furthermore, the higher the number of projects the developer is contributing to, the more is the effort required to scale the contribution barriers erected by different projects. However, it may also allow them develop broader and more portable skills. Even so, working on multiple projects at one time places more demands on developers in terms of their attention and the time spent contributing. Therefore, we expect that developers who contribute to more number of projects are more likely to become inactive.

Finally, projects employ developers performing different activities ranging from clerical to highly intellectual (von Krogh et al. 2003). Since enjoyment in programming is an important motivator for OSS developers to contribute (Shah 2006), the role of the developers is likely to influence their decisions to remain active in the project. Roles requiring intellectually challenging tasks may be more motivating for developers to contribute and to remain active in a project than roles with more mundane responsibilities.

2.1.2 Project Level Factors

We focus on two project level factors that may affect developers' perception of the vitality of the project – project age and project size. Our argument relies on the notions of the liability of newness and the liability of smallness of OSS projects (Chengalur-Smith et al. 2010). The liability of newness suggests that a newer project will be perceived as less legitimate because it has had less time to establish clear governance procedures such as recruitment strategies, rules for peer review process and conflict resolution. In addition such a project has had less time to establish its credibility among the developers and the supporting social network structure, such as a helpful user community, that may submit bug reports and feature requests (Chengalur-Smith et al. 2010). Such projects may find it difficult to retain developers. On the other hand, the projects that have had more time to mature may be in more advanced stages of development and more active (Crowston and Scozzi 2002). Such mature projects may appeal to the developers given their greater activity and vitality (Choi et al. 2010). Therefore, we expect older projects to have lower mean turnover rates.

OSS projects are also likely to be susceptible to the liability of smallness which suggests that smaller projects may be perceived as having less pragmatic legitimacy resulting in a difficulty to attract and retain developers (Chengalur-Smith et al. 2010). Size of the developer base is also important in attracting additional collaborators as it may be considered a sign of vitality of the project (Choi et al. 2010). Larger projects with more developers may provide more opportunities for learning and building reputation for developers along with giving them a sense of importance of the project (Chengalur-Smith et al. 2010). Therefore we expect larger projects to have lower mean turnover rates.

2.2 METHODOLOGY

To explore and explain the nature and impact of a developer and project variables on turnover, we used archival data. The sample of projects and participants was drawn from SourceForge (www.SourceForge.net). SourceForge provides open source developers with a centralized place to manage their development and includes communication tools, version control processes, and repositories for managing source code. It is one of the largest open source repositories, estimated to host over 168,000 projects (Madey and Christley 2008).

The sample contained data for 40 currently active projects on SourceForge and 201 developers. The projects were chosen on the basis of their overall activity levels (SourceForge provides a list of projects that have been most active during their development). All the selected projects had to be currently active (i.e. at least one developer currently working on them) and have CVS/SVN repositories on SourceForge that could be mined for developer activity levels. In many instances projects had separate websites and hosted their CVS/SVN repositories outside SourceForge. In other cases projects did not provide all the data that we needed. Such projects were dropped from our data set. SourceForge also provides a webpage for each developer which contains information about developers' joining date, current activity, projects that they contribute to and their roles in the projects.

2.2.1 Developer Level Variables

The following five developer level (level 1) variables, including the outcome variable, turnover, were collected –

- Turnover – We scanned for developer activity on CVS/SVN code repositories of the projects. Each project maintains such a code repository that contains the complete history of its development. The code repository can be scanned for individual developer activity history. Turnover was operationalized as a binary outcome variable. A developer was deemed active, coded as 0, if at least one CVS/SVN commit was made by him/her in a 2 month period (March 25th 2010 to May 25th 2010). On the other hand, s/he was deemed inactive, coded as 1, if s/he had made no commits to the code repository in that period. Since it is difficult to predict when turnover has happened, such an approach is reasonable. Joyce and Kraut (2006) also followed a similar approach in their study of turnover from online newsgroups, however they chose an observation period of six months to determine turnover. Our choice of a two month period seemed reasonable for an exploratory study in an open source context since such projects require more active developer attention than online newsgroups.
- Role of the Developer – We collected the data for the role of each individual developer from the project webpages. A project may employ developers for various roles that range in the level and kind of expertise required[6]. Since we did not find enough developers for each of the possible roles, and on noting that the two most frequently employed roles

[6] Some examples of roles developers may perform in the project are as administrators, developers, document writers, project managers, packagers, web designers, translators, support technician, cross platform porter, all hands person etc.

were those of Developer and Administrator, we created two dummy variables "Developer" and "Admin" with the base group "Other" (which included all other roles)[7].

- Number of Projects – We collected the data for the number of OSS projects undergoing active development that the developer was involved in by scanning individual developer webpages.
- Tenure – We approximate the tenure of a developer in months by using the date of joining SourceForge.net. The longer the developer has been on SourceForge, the greater is his/her chances of having gained enough technical and programming knowledge to overcome the contribution barrier of a project they may wish to contribute to[8].

2.2.2 Project Level Variables

The following project level variables (level 2) were collected –

Project Age – The date the project was registered is available on SourceForge. We calculate the age in number of months since its registration.

Size of Project – The number of developers listed on the project's webpage.

[7] As evident from Table 1, roughly 25% roles belonged to the *Other* category. Since *Developer* and *Admin* dummies are correlated we also analyzed the data by merging *Other* and *Admin* categories to create a single *Developer* dummy variable. In doing so we found that the HLM results did not change appreciably.

[8] Since it is difficult to judge when a developer actually starts working on a project, we used their tenure on SourceForge.net as a proxy. For future work, we are considering operationalizing tenure as the period elapsed since their first contribution to the focal project, which may be a more robust way of approximating tenure.

2.3 STATISTICAL MODELS AND RESULTS

Modeling the effects of developer level and project level variables together presents considerable conceptual and methodological challenges. The Hierarchical Linear Modeling (HLM) technique allows researchers to model developer level outcomes within projects and model any between project differences that arise. Newer versions of HLM also allow appropriate estimating techniques for dichotomous dependent variables, such as turnover. HLM also allows multivariate tests based on comparing model deviances (-2 Log Likelihood at convergence) using the Laplace Estimation method (Raudenbush and Bryk 2002). Deviance (-2LL) is a measure of model fit; the smaller the deviance is the better is the model fit.

The study was carried out in two parts and follows the approach recommended by Rumberger (1995). In the first part a developer model of turnover was developed and tested with logistic regression using only developer level variables. This allows an analysis focused only on developer level variables. However, this not only ignores project level variables but also assumes that the effects of developer level variables on turnover do not vary from project to project. This assumption was tested in the second part of the study.

In the second part of the study logistic HLM analysis was performed. The developer level model used in this part of the study was based on the results of the first part. It allowed us to focus the analysis on explaining between project differences in the predicted mean turnover rates (turnover characteristics adjusted for differences in developer characteristics between projects) and between project differences in the effects of developer level variables on turnover rates. Table 1 presents the descriptive statistics and bivariate correlations of variables at individual and project levels.

Table 1. Descriptive Statistics and Correlations

Developer level (n=201)	Mean	s.d.	1	2	4	5
1. Tenure	83.01	36.891	-			
2. Number of Projects	2.92	2.763	.325**	-		
3. Turnover (DV)	.62	.486	.136	-.026	-	
4. Developer	.42	.495	.038	-.081	.024	-
5. Admin	.33	.473	-.194**	.063	-.189**	-.605**
Project level (n=40)	Mean	s.d.	1	2		
1. Age	86.25	43.665	-			
2. Size	14.70	23.750	.323*	-		

*p < 0.05, **p < .001, two-tailed tests

2.3.1 Logistic Models

A series of linear logistic models were developed and tested to measure the effect of developer level variables on turnover behavior. Turnover is a binary dependent variable that can be expressed as a probability p_i, which takes on the value of unity if the developer *i* becomes inactive in the project, zero otherwise. The probability *p* is transformed into log of odds (or logit) which is expressed as:

$$Log\ [p_i / (1-p_i)] = \beta_0 + \beta_1\ Tenure + \beta_2\ Developer + \beta_3\ Admin + \beta_4\ Number\_of\_Projects$$

Logistic Regression was carried out in two steps. In the first step univariate estimates were computed for all the level 1 independent variables. In the next step we estimated the full multivariate model using the forced entry method. Table 2 presents the exponentiated logistic coefficients, which represent the ratio of predicted odds of turnover with a one unit increase in the independent variable to the predicted odds without one unit increase. Thus, a value of one signifies no change in the odds of turnover. A value greater than (less than) one indicates that the odds of turnover increase (decrease) due to a unit change in independent variable.

Table 2. Predicted odds of turnover

Variable	Univariate estimates	Multivariate estimates
Admin	.443*	.320*
Developer	1.104	.524
Tenure	1.008	1.007
Number_of_Projects	.981	.955
-2LL (initial = 266.583)		254.268
Cox and Snell R^2		.059
Nagelkerke R^2 $\Delta\chi 2$ = 12.311* ($p < .050$)		.081

*p < 0.05, **p < .001

The univariate and multivariate estimates of "Admin" are both significant and suggest that administrators have 44.3% lower odds of turnover than the "Other" category. This means that administrators are more than twice as likely to remain active than developers with "Other" roles. Since "Admin" was significant in the univariate and multivariate estimates it was retained for further HLM analysis.

2.3.2 HLM Models

HLM analysis requires two types of models: a level 1 model to estimate the effects of developer level variables on turnover and a level 2 model to estimate the effect of project level variables on the coefficients of the level 1 analysis. HLM analysis allows for the estimated coefficients from the level 1 model to vary across projects and any such difference can be modeled with project level variables. We begin the analysis by modeling the unconditional model (base model) with no predictors at either level.

2.3.3 Unconditional Model

$$Log\ [p_{ij} / (1\text{-}p_{ij})] = \beta_{0j}$$

$$\beta_{0j} = \gamma_{00} + u_{0j}$$

This model allows us to ascertain the variability in the outcome variable at each of the two levels i.e. within project and between project variability. The results are shown in Table 3.

Table 3. Unconditional Model

Fixed effect		Coefficient	se	p value
Average project mean γ00		.484	.183	0.012
Random effect	Variance component	df	χ2	p value
Project mean, uoj	.314	39	57.48	.028
Deviance (-2LL) Estimated parameters	631.288 2			

The Null hypothesis H_0: $\tau_{00} = 0$ is rejected (p = .028). This suggests that significant variation exists among projects in their turnover rates. The intraclass correlation coefficient (ICC) measures the proportion of total variance in the outcome that is between projects, i.e. level-2 (Snijders and Bosker, 2012). Resulting ICC values for our analysis suggest that 8.71% variation in turnover that can be explained by level-2 predictors resides *between* projects[9]. Further, for a project with a typical turnover rate (with $u_{0j} = 0$), the expected log odds of turnover is .484. This corresponds to a probability of $1/ (1 + e^{(.484)}) = .38$. This means that for a typical developer in a typical project there is a 38% chance of turnover in a 2 month period.

2.3.4 Conditional Model

This model allows part of the variation in the intercept β_0 (mean turnover rates) to be explained by project level variables (project age and size),

$$Log\ [p_{ij} / (1\text{-}p_{ij})] = \beta_{0j} + \beta_{1j}\ Admin$$

$$\beta_{0j} = \gamma_{00} + \gamma_{01}\ Proj\_Age + \gamma_{02}\ Proj\_Size + u_{0j}$$

$$\beta_{1j} = \gamma_{10}$$

All the variables were grand mean centered. This centering approach facilitates the interpretation of the results and also lessens multicollinearity in group level estimation by reducing the correlation between the group level intercept and slope estimates (Raudenbush 1989). Table 4 presents the results of the conditional model.

[9] Level-1 variance for logistic HLM models is given by $\pi^2/3$ while level-2 variance (τ_{00}) is the variance component of u_{00} (Snijders and Bosker, 2012).

Table 4. Conditional Model

Fixed effect	Coefficient	se		p value
Average project mean γ_{00}	.760**	.210		0.001
Proj_Age Slope γ_{01}	.010*	.004		0.035
Proj_Size Slope γ_{02}	-0.015*	0.006		0.028
Admin Slope γ_{10}	-1.013**	0.360		0.006
Random effect	Variance component	df	$\chi 2$	p value
Project mean, u_{oj}	0.102	37	41.355	.286
Deviance (-2LL) Estimated parameters	617.781 5			

The Null hypothesis H_0: $\tau_{00} = 0$ fails to be rejected (p = .286). This means that after controlling for project size and age no significant variation remains to be explained. The proportion of reduction in variance or variance explained at level 2 is .6751, implying that project size and age account for 67.51% of the explained variance at level 2. The Deviance (-2 Log Likelihood) is also significantly improved from the base model (ΔD = 13.50, χ^2_{df} = 3, p = .004), suggesting a good model fit and a fully identified model[10].

We find that project administrators are 1/ (1 + $e^{(1.013)}$) = 26.63% less likely to turnover in the 2 month period than the "other" group, after controlling for other effects in the model. Additionally, project age and size have small but significant positive and negative effects on

[10] A conditional model that included all developer level variables did not further improve deviance and was rejected in favor of the more parsimonious model presented here.

developer turnover respectively. A unit increase in project age increases the log-odds of turnover by 0.010, while a unit increase in project size reduces it by .015, all other things being equal.

2.4 LIMITATIONS AND FUTURE DIRECTIONS

Like all empirical work this study is limited in many ways. First, the sample is biased toward more active projects. Such projects may have well developed infrastructures allowing retention of active members and/or a constant inflow of newer active members. Including less active projects in the future should allow for more robust and generalizable results. Second, the use of binary variable for turnover leads to loss of variance information. Some developers are more active than others and are more likely to stay active and our analysis should take this into account. To address this issue in future models we will use each developer's participation history to calculate a probability of turnover. Similar techniques have been used in marketing research to calculate the probability that a customer with a given pattern of transactions is still alive (Schmittlein et al., 1987). Finally we will seek a *conceptual* integration of developer and project level factors in modeling turnover rather than just an *empirical* integration. While the initial empirical integration lays the groundwork for simultaneous study of individual and project level factors, additional conceptual integration is needed for a complete understanding of turnover in open source projects. We consider this study as an important first step toward this end.

2.5 CONCLUSION

In this preliminary study, we modeled turnover behavior of core open source contributors using logistic hierarchical linear modeling approach. We argue that taking both the developer and the project level factors into account will lead to a richer understanding of the issue of turnover in open source projects. Our analysis suggests that developer role, project size and project age are important predictors of turnover. We find that there exists a significant variation in mean turnover rates among projects on SourceForge and that project age and project size account for a sizable proportion of this variation. An understanding of factors that affect turnover in OSS projects may prove valuable for project managers and allow them to make more informed decisions in managing the volunteer developer work force. The IS research community will also benefit from the improved understanding of the interplay of factors across levels in managing volunteer contributions in virtual teams.

3.0 STUDY 2: THE ROLE OF OWNERSHIP AND SOCIAL IDENTITY IN PREDICTING DEVELOPER TURNOVER IN OPEN SOURCE SOFTWARE PROJECTS

3.1 INTRODUCTION

As mentioned earlier, in this study we propose a *conceptual* integration of developer and project level variables to analyze turnover. Previous OSS research has neglected the role of the organization structure (ownership) of OSS communities on member retention. Further, this research has neglected the inherently nested nature of the developer and project variables. In order to address these limitations and expand the existing research, this study develops a Hierarchical Linear Model (HLM) of turnover behavior in OSS. The analysis focuses on two levels: the developer level, which examines factors that may affect developers' decisions to become inactive, and the project level, which examines the factors that may influence the rates of turnover among projects. This study also compares various measures of developer turnover in the OSS context. In what follows, we present a brief background of the Ownership and Social Identity Theory literatures. This is followed by the hypotheses and the methodology sections where we outline the testable hypotheses and the empirical methods used in the study. Finally we conclude by noting the limitations of this work and suggesting the steps we intend to take in the future to further improve this study.

3.2 LITERATURE REVIEW

In this section, we outline the current state of research in the Employee Ownership literature and the Social Identity literature.

3.2.1 Employee Ownership

In Sociological and Economic literature on organizations, a firm is often seen as composed of four classes of actors: shareholders, board of directors, executives, managers and owners (Kang and Sorensen, 1999). Shareholders are often thought of as owners of the firm and hence this literature has often used “equity stake” or “share/stock value” as a proxy to operationalize their level of ownership. However, there are other possible forms of organization; for example, owners may be workers (employee owned companies). In entrepreneurial firms, one person may perform all the tasks (Kang and Sorensen, 1999). A common motivation to introduce employee ownership in traditional firms has been to increase worker and firm productivity (Pierce and Rodgers, 2004). In the OSS context, projects have tended to use ownership of modules for effectively managing the code production, avoiding chaotic code changes, and as a way to recognize the expertise of an owner-developer (Mockus et al., 2002).

Ownership literature has conceptualized ownership as a dual conceptualization; firstly as an objective and real state (i.e. formal ownership) and secondly as a perceived state (i.e. psychological). While formal ownership is viewed as an objective reality and is often seen is a “bundle of rights”, psychological ownership is viewed as a reflective and experienced state achieved due to the reality of possession (Pierce and Rodgers, 2004). This psychological state is more likely to be created when the formal ownership allows control over the owned object, knowledge of the object, and an investment in an object (Pierce et al. 2001). In the OSS context, formal ownership is accorded only to developers who have shown expertise with the project code and who have invested their time in developing it. Therefore, it is very likely that the formal ownership arrangement in OSS leads to a feeling of psychological ownership among developers.

Pierce et. al. (2001) argued that ownership can be seen as fulfilling three basic human desires:

1) Efficacy and Effectance: Ownership and the rights that come with it, allow individuals to feel in control and give them the ability to alter their environment, thus satisfying their innate need to be efficacious (Beggan, 1991).

2) Self-Identity: Possessions serve as a symbolic expression of self and they are closely connected to the perception of self-identity and people use ownership for the purpose of defining themselves and expressing their identity to others (Pierce et al. 2001). Possessions play a dominant role in forming an owner's identity and become a part of their extended selves (Belk, 1988). In fact, ownership and identity may be so strongly related that people may engage in territorial behaviors to identify and defend their possessions (Brown et. al. 2005). To quote Sartre (1969),

"I am what I have; What is mine is myself"

3) Having a Place (Belongingness): Possessions also allow people to fulfill their need to "dwell" and they may devote significant resources to targets that may potentially become their homes (Pierce et al., 2001). When people feel like owners in an organization, their need for belongingness is met by "having a place" in terms of their needs being met (Avey et. al. 2009).

Therefore, the positive effects of ownership are generally hypothesized because of the sense of fulfillment, self-esteem and the enhancement to ego it provides. When employees feel ownership in an organization, they feel a sense of responsibility toward it and tend to engage in positive behaviors (Avey et. al. 2009). Additionally, Beggan (1992) confirmed the existence of “mere ownership effect”, the hypothesis that owners rate a target object much more favorably than non-owners. Because people perceive their possessions as an extension of their self-identities, they maintain enhanced perceptions of what they own, in order to maintain their positive identities. The mere ownership effect serves as a basis of motivation to maintain a positive self-image (Beggan, 1992). Furthermore, he found this effect despite the time people had to ruminate about the object, thereby suggesting that mere knowledge of ownership is enough for this effect to take hold, irrespective of the time the object was owned.

3.2.2 Social Identity Theory

The argument that people seek motivation for self-enhancement through their possessions is analogous to the idea proposed by Tajfel and Turner (1985) in Social Identity Theory (SIT) that people make biased judgments in favor of their groups to enhance their individual self-esteem. SIT posits that people tend to classify themselves and others into various social categories, such as groups, teams or affiliation (Tajfel and Turner, 1985). A major reason for maintaining this classification is to find a sense of “belongingness” and maintain positive self-image. The desire to maintain a positive self-image is so strong that people tend to associate themselves with successful others even when the connection may be incidental or meaningless. The desire to maintain an association with another successful entity means that negative feedback on the group's functioning may cause people to end their association, the so called *cutting off the reflected failure* syndrome (Snyder et. al. 1986). On the other hand, De Cremer et. al. (2002) showed that under circumstances of negative feedback, some people may actually increase their association and responsibility to the group. They argued that the kind of reaction people show to group failure may well depend on how strongly people feel associated to the group.

Social Identity Theory (Tajfel and Turner, 1985) suggests that people's social identities may be positively or negatively affected according to the evaluations of those groups to which they belong. If the group's status (or success) is lower in comparison to other relevant groups, people's social identities may be threatened. In such cases SIT posits that people may respond in one of the following manner:

1) Individual Mobility: Individuals may leave or dissociate themselves from their erstwhile group and seek to join a better group. However, this option may not always be readily available to people, as joining other groups may not always be feasible (Ellemers et. al., 1988).

2) Social Competition: Individuals may respond by working harder to make the group successful, so that its status relative to other groups may be enhanced.

3) Social Creativity: Individuals may respond by changing the comparison criteria to a new dimension or changing the values assigned to the attributes so that previously negative comparisons are now perceived positive (Tajfel and Turner, 1985).

In the next section, we will develop our hypotheses regarding developer turnover in OSS context using the Ownership literature and Social Identity Theory as our bases.

3.3 HYPOTHESES DEVELOPMENT

OSS participants differ from each other, most conspicuously in the rights and duties and the accompanying ownership stakes they have in the project. In Sociology and Organizational Behavior literature, an employee's ownership stake in a firm is often seen as a "bundle" of rights, powers and privileges (Kang and Sorensen, 1999). Formal employee ownership in a firm is frequently associated with rights to (1) possess some share of the owned object, (2) exercise influence (control) over the owned object, and (3) acquire the information about the status of that which is owned (Pierce et al., 1991). This literature has also studied how *varying levels of employee ownership* affect employees' job attitudes such as turnover and work effectiveness. Workers-owners of the firm (shareholders) were found to have significantly higher levels of organizational commitment and lesser levels of turnover than non-shareholders (Long, 1980; Pierce et. al., 1991). Hammer et al. (1981), while studying the link between ownership and absenteeism concluded that ownership creates a mechanism through which employees perceive that their voices may be heard in bringing about organizational change. Thus formal ownership may create a perception of holding a position of influence and involvement in decision making, thereby generating a sense of responsibility toward the firm and higher levels of interest and commitment to it.

In the OSS context, participants may also have varying ownership levels and possess the corresponding levels of rights (for example, Google Code participants can be sole owners, part owners, committers or contributors)[11]. Owners (and part owners) possess the highest levels of

[11] http://code.google.com/p/support/wiki/Permissions

rights which include the ability to make changes to the project structure, code and workforce (i.e. they can move people in or out of the project). Committers possess the ability to make changes to the code but not the larger project structure or the workforce. Contributors, on the other hand, possess much restricted rights and can only post comments and point out issues in the code; however their rights may be upgraded to include code commits. This phenomenon is consistent with Russell's (1985) view that ownership systems can be differentiated by the extent to which they permit rights to be exercised. Prior literature in OSS has studied how participants are accorded progressively higher rights, duties and the accompanying ownership of the project code (von Krogh et al. 2003). To be accorded these rights, participants need to demonstrate their interest, skill and understanding of the code structure. Mockus et al. (2002) studied the process through which developers are *selected* and granted ownership of project modules based on their skills and expertise[12]. Since participants with higher levels of ownership are more likely to be committed to the development, we propose the following main effect of ownership (level 1 variable) on turnover:

Hypothesis 1: *An OSS participant's level of ownership in the project will be negatively associated with turnover. Specifically, we hypothesize that sole owners will be least likely to turnover from a project followed by part owners, committers and contributors respectively.*

[12] Different projects may follow different strategies to accord ownership rights. Apache project for example, follows an emergent ownership structure where some participants emerged as "de facto" owners of modules through their continued code development and commitment to a module. Mozilla on the other hand, follows an "enforced" ownership strategy where every change to the code is reviewed by the module owner (Mockus et al. 2002).

As mentioned earlier, OSS developers may work simultaneously on a number of projects with possibly different ownership levels in each. A given developer may be the sole owner in one project, co-owner in second, and committer in third. Yet another developer may work on a single project as an owner. Role theory argues such multiple relationships with multiple roles may be the cause of psychological stress. For example, Goode (1960) argued that "role strain" is a natural consequence of performing multiple roles. Role strain comprises of two overlapping problems: role overload which refers to constraints imposed by time, and role conflict which refers to discrepant expectations in performing roles irrespective of time (Sieber, 1974). Based on this we may assume that multiplication of roles causes burden on people and inhibits their ability to contribute to projects. However, the Theory of Role Accumulation suggests the following positive outcomes of multiple roles: (1) Overall status security and buffers against failure, (2) role privileges, (3) resources for status enhancement and role performance, and (4) enrichment of the personality and ego gratification (Sieber, 1974). Thus, role accumulation provides a psychological mechanism through which developers may be able to raise their performance levels. In addition, developers working on multiple projects may be able to apply skills learned from one context to the other with greater efficiency than others. However, Sieber (1974) does not deny the inevitability of role overload. Therefore we propose a curvilinear main effect for the number of projects a developer is involved in (level 2 variable) on turnover:

Hypothesis 2: *Overall, the number of projects a developer is associated with has a U-shaped relationship with turnover in a given project. Specifically, the probability of developer turnover will decrease up to a point as the number of projects the developer is associated with on increases, after which it will increase.*

However, as the number of projects a developer works on increase, their perception of level of “belongingness” may decrease and chances of individual mobility may increase. Such developers may not feel too attached to a project and may also have other projects they may seamlessly move into. Because “sole owners” have the greatest burden of responsibilities, in terms of managing and overseeing the project, they may find it increasingly difficult to devote time to a given project when there are other projects which require their attention at the same time. Since their sense of “belongingness” may be diminished, sole owners may be more likely to ignore certain projects in order to manage their workload better. Part owners, on the other hand, may rely on the shared ownership structure to leverage help from others in sharing their burden. Committers and Contributors may also be able to retain activity by sharing the workload with others. However, since the continuum of ownership levels from sole owners to contributors represents a decreasing burden of rights and duties, it is likely that lower levels of ownership may allow developers retain their associations with multiple projects.

Hypothesis 3: *The number of currently active projects a developer is associated with, will moderate the relationship between the level of ownership of the developer and turnover in that project. Specifically, as the number of associations increase, developers with higher level of ownership in the focal project will be more likely to leave than developers with lower levels of ownership.*

As proposed by SIT, people strive to maintain positive identities through their social associations. If their identities are threatened due to low group status or failure they may disassociate from the group. Therefore we propose the following main effect of project success (level 2 variable) on turnover,

Hypothesis 4: *Overall, the success of the OSS project is negatively related to turnover in that project. Specifically, as the project success increases the chances of a developer working in it leaving are reduced.*

However, SIT also proposes that people may also respond to group failure by committing more to the group thereby facilitating its success (Social Competition). Here we argue that the decisions developers take, i.e. whether to leave or stay in a failing project, may well depend on their level of ownership in the project. De Cremer et. al. (2002) found that when group members received negative feedback, their decision to dissociate from the group often depended on their level of identification with the group. Developer-Owners of an OSS project achieve their status either by starting a new project or by demonstrating their expertise through contribution and commitment to the project. Such developers are also likely to strive to maintain their position and justify their past association with the project. Indeed, project escalation literature has similarly argued that managers may maintain their commitment levels to sinking projects in order to justify the sunk costs (Keil et. al. 1995). In OSS context, the sunk costs are likely to be the psychological association with the project and the effort spent on it. Therefore we propose the following interaction effect of ownership (level-1 variable) and project success (level-2 variable) on turnover,

Hypothesis 5: *The success of the OSS project will moderate the relationship between the level of ownership of developer and turnover in that project. Specifically, as the project success decreases, developers with higher level of ownership will be less likely to leave than developers with lower levels of ownership.*

3.4 DATA AND MEASURES

Data for this research were collected from Google Code. Google code provides open source participants with a centralized place to manage their development and includes communication tools, version control processes, and repositories for source code. It is one of the largest open source repositories and is estimated to host over 237,802 projects (as of February 2012) out of which over 71,091 projects were active[13]. We designed a Web Crawler for the purpose of collecting the developer level and project level data using the R programming environment (R Development Core Team, 2011). Web Crawlers have been used in previous research to collect data on OSS projects (e.g., Setia et al. 2010). Additional developer activity data was collected by downloading the project repositories and parsing them using the CVSAnalY software[14]. This software allowed us to parse project repositories and store developer contribution data in a MySQL database.

In the initial stage, data were collected for 5,500 top listed projects on Google code in August, 2012[15]. These included project level data and a list of 13,989 unique developers who contributed to these projects. This set of 5,500 projects and 13,989 developers form our initial sample. In the next stage, additional project level data were collected for 22,948 other projects that the core set of developers additionally work on, but that do not fall in our core sample of 5,500 projects. Thus, all 13,989 developers fall under the 28,448 projects.

---

[13] We got these numbers by querying the February, 2012 Google code database dump, hosted by Flossmole. http://code.google.com/p/flossmole/

[14] http://git.libresoft.es/cvsanaly/

[15] http://code.google.com/hosting/search?q=&btn=Search+Projects

3.4.1 Sample Inclusion Criteria

Once we had captured the initial dataset, we applied various sample inclusion criteria on project characteristics in order to select the appropriate sample of projects and developers that reflect the population that we sought to generalize our results to.

We represent the sample inclusion criteria as a set of filters that were applied to the initial sample of 5500 projects (Figure 1 represents the process):

- *Filter 1:* We excluded projects of size 1, i.e. projects with only one developer, since involvement in such a project may not be meaningful as a group process such as social identity. Stewart et. al. (2006) also used a similar exclusion criterion.
- *Filter 2:* Out of a total of 2111 projects that were left post filter 1, about 82% used SVN based version control system, while 11% used Mercurial and 7% used GIT systems. Due to the differences in the way workflow is organized in these source code management systems, Rodriguez-Bustos and Aponte (2012) found differences among developer contributions when projects migrated from one system to the other. Therefore, in order to control for any differences among projects that use SVN versus GIT or Mercurial, we excluded projects that used Mercurial or GIT repositories.
- *Filter 3:* In the next stage we excluded either projects whose source code could not be read and parsed, or did not exist anymore, or were moved to different project hosting sites such as GitHub or SourceForge. In order to flag projects that had possibly migrated, we performed automated text analysis of project descriptions using keywords such as "migrate", "move", "GitHub", and "SourceForge". The project descriptions of projects

were then manually checked in order to make sure to exclude projects that had actually migrated to other hosting sites.

- *Filter 4:* In the next step, we excluded projects that either were course or class related and hence potentially temporary (e.g. https://code.google.com/p/swe574-group2-spring2013/), hosted only documents and not code, were test projects created by developers to test the Google code environment (e.g., https://code.google.com/p/211227358-testproject/), or were sponsored by a company such as Google. In order to flag projects that were possibly course related, we performed automated text analysis of project descriptions using keywords such as "course", "assignment", "student", "university", "school", "college". The project descriptions of projects were then manually checked in order to make sure to exclude projects that were actually course or assignment related and hence potentially temporary. This allowed us to retain projects that were the product of individual developers working under self-volition, and voluntary basis.
- *Filter 5:* In the final step, we excluded projects that showed no activity in the 30 day period prior to data collection. This was done to ensure that only currently active projects were included in the sample. Stewart et. al. (2006) also used a similar exclusion criterion.

The final sample consisted of 446 projects and 2949 unique developers. The project repositories of these 446 projects were parsed and stored in a MySQL database using the CVSAnalY software in order to analyze individual developer activities. The 2949 developers in this sample also worked on other 4445 "secondary" projects that did not overlap with the "focal" sample of 446 projects. Thus, all the 2949 unique developers work under 4891 projects.

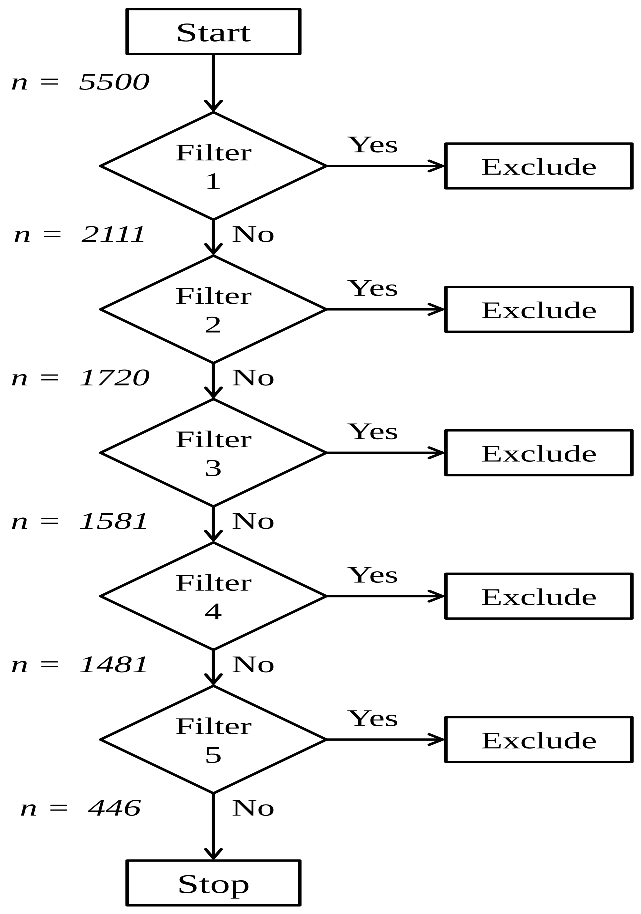

Figure 1. Sample Inclusion Process

3.4.2 Developer Level Variables

The following developer level variables, including the outcome variable, turnover, were collected –

a) Dependent Variable (Turnover): Measuring turnover in a voluntary non-contractual setting (such as OSS) is a non-trivial problem since the developers may go inactive for an elongated period of time, and then may return. The current research approach to measure turnover in such settings is to observe inactivity over an elongated, often arbitrary, period of time (e.g. Joyce and Kraut, 2006). However, this approach results in right-censored measurement of turnover, since a developer may return after the observation period has ended. Since, to the best of author's knowledge, there are no established methods to measure turnover in the OSS context, we use Schmittlein et. al's (1987) Pareto/NBD approach to operationalize turnover. Originally proposed in the Marketing literature to track a customer's lifetime value in a non-contractual setting, this approach assumes that customers buy at a steady (albeit stochastic) rate for a period of time and then become inactive. The time to dropout is modeled using the Pareto (exponential-gamma mixture) timing model, and the repeat buying behavior is modeled using the NBD (Poisson-Gamma mixture) counting model. This approach has been shown to work quite well in modeling actual customer behavior based on their past buying behavior (Fader et. al. 2005). We argue that a similar approach can be used to track developer contributions and model their lifetime and dropout behaviors.

In this approach a developer's activity on the code (i.e. code commit) may be considered an "event" (analogous to a "purchase" event). A developer who is active is considered "Alive", while a developer that is inactive for any reason is considered "Dead". A developer who is active at time $t = 0$ is observed up to a suitable time "T". During this observation period, the developer makes "X" number of code commits with the last commit coming at time "TX", $0 < TX \leq T$. The value "X" is considered the "Frequency" of commit activity while "TX" is considered "Recency". The Pareto/NBD approach then allows us to calculate the conditional probability that the developer is still alive after the observation period, $P(Alive|(X, TX, T)$, subject to the following assumptions outlined by Schmittlein et al. (1987):

- While alive, a developer makes commits to the code according to a Poisson process with a rate λ. The Poisson process implies exponentially distributed interpurchasing times with the resulting "lack of memory" property (Schmittlein et. al. 1987). This is a reasonable assumption in the OSS context since the development of code is not bound by any regular schedule, and a developer may have a constant probability to contribute to the code.

- Each developer remains alive for a lifetime which has an exponentially distributed duration with death rate μ. The events that could trigger "death" (such as a new career, lifestyle change, etc.) may arrive in a Poisson manner. The arrival of all possible death events is the superposition of the individual events, and is best approximated as a Poisson process (Karlin and Taylor, 1975). Thus, this assumption is also reasonable in the OSS context.

- The activity rate λ and the death rate μ for different developers are distributed according to different gamma distributions in the population. Further, λ and μ are distributed independently of each other. It is reasonable to assume that different developers have different activity and death rates. Some developers are more active than others and other developers may become disenchanted with the project sooner than others. The gamma is a flexible distribution that can capture most reasonable distributions. Further, a frequent developer may become disenchanted with the project sooner and die. On the other hand, another developer may become more strongly attached to the project. Thus, there is no a-priori reason to favor a positive correlation between λ and μ over negative (Schmittlein et. al. 1987).

Thus, given a developer's Frequency and Recency profile we can construct a developer's conditional probability of turnover, *P(DEATH)* as *1 – P(ALIVE|(X, TX, T))*. Each OSS project maintains a code repository that contains the complete history of its development including each individual developer's activity history. We downloaded the project SVN repositories and parsed them into a MySQL database using the CVSAnalY tool. To generate the conditional probability we chose to observe the developers for 3 different observation periods of 90, 180, and 270 days to capture their activities over a short, medium and long term respectively. Joyce and Kraut

(2006) also used a similar observation period of 180 days. We queried the MySQL database to generate each developer's Frequency (*X*) and Recency (*TX*) profile over the 90, 180 and 270 day observation periods.

We then calculated *PDEATH90*, *PDEATH180* and *PDEATH270* as three separate outcome variables for each developer in our sample using the "Buy-Till-You-Die"(BTYD) package in R (Dziurzynski, Wadsworth, Fader et. al. 2012)[16]. As we argue later in the paper, further analysis allowed us to select one of the most appropriate among the three outcome variables, which was then used for modeling the turnover behavior using developer and project level predictor variables.

There are several advantages to using the Pareto/NBD approach over heuristic approaches that have been used in the past. One heuristic approach that has been applied is to deem a developer inactive if he/she has been inactive for at least a given period of time (Joyce and Kraut, 2006; Sharma et. al. 2012). Such a heuristic ignores a developer's historic pattern of activity and does not allow us to generate a probability measure (Schmittlein et. al. 1987). It also leads to a loss in information by neglecting to differentiate between more frequently active developers from less active ones. Furthermore, once the developer has been deemed inactive it ignores their possibility of a comeback. Unlike the heuristic approaches, the Pareto/NBD approach not only takes into account "how many distinct days of activity" were shown by a developer but also the "recent most activity". An added advantage of the Pareto/NBD approach

[16] Since the outcome variables are proportions, it is recommended that the arcsine transformations (i.e. $sin^{-1}(\sqrt{PDEAD})$) be performed in order to normalize the distribution (Hogg and Craig, 1995). Our results remained unchanged when using the transformed version of the DVs. Hence we report our results using the untransformed versions for ease of interpretation.

is the ability to predict future activity levels of developers (expected number of commits in the future). This can allow researchers and OSS managers to not only predict the available active workforce in the future and manage the project accordingly, but also to calculate the expected number of commits for the project in the future by summing over the expectations of the current developer base(Schmittlein et. al. 1987).

b) Ownership level of developer: As mentioned earlier, Google Code lists the ownership status of a developer as an owner, co-owner, committer or contributor. We used 3 dummy variables, "*Part Owner Dummy*", "*Committer Dummy*" and "*Contributor Dummy*", to capture the different levels of ownership, with "*Sole Owner*" as the reference group.

c) Number of Projects a developer is associated with: We collected the number of projects a developer is associated with, "*Number of Projects*". This information is available on each developer's profile page on Google code along with the corresponding ownership level. Further, to explore any possible curvilinear effects (i.e. H2), we also calculated (*Number of Projects*)^2.

3.4.3 Project Level Variables

The following project level variables were collected –

Project Success: Project success in the OSS context can be measured using a variety of indicators, since it is a multidimensional construct (Stewart et. al. 2006). A simple and popular measure of a project's "popularity" or its extent of "use" in OSS literature has been the number of times a project's code has been downloaded (Crowston et. al. 2006). However, since older projects have had more time to accumulate the number of downloads, we divided this variable by project age and log-transformed it to correct the skew, i.e. "*Log (Downloads By Age)*".

In addition, successful project development also involves interacting with the user and developer communities and fulfilling their bugs and feature requests. Google code allows users and developers to report "issues" with the project. The proportion of issues resolved by the developer team can be a helpful measure of the strength of a project's processes and ultimately its success (Crowston et. al. 2006). Therefore, we calculated the ratio of issues closed by the developer team (i.e. resolved) to the number of issues reported, "*Proportion of Issues Resolved*", as a second measure of project success.

3.4.4 Project Level Control variables:

The following variables were used as controls:

a) Project Age: We controlled for "*Project Age*", i.e. the time in years from the first code commit made on the project. Chengalur-Smith et al (2010) argued that a new project may suffer from the *liability of newness* which suggests that such a project might be perceived as less legitimate because it has had less time to establish clear governance procedures such as recruitment strategies, rules for peer review process and conflict resolution. In addition such a project has had less time to establish its credibility among the developers and the supporting social network structure, such as a helpful user community, that may submit bug reports and feature requests. Such projects may find it difficult to retain developers.

b) Project size: Butler (2001) has shown that a community's size may impact the level of turnover in it. We controlled for the number of developers working on the project, "*Number of Developers*". Each project lists the number of developers associated with it.

c) Project License: License restrictiveness has been to shown to impact the level of developer interest in the project (Stewart et. al. 2006). The use of restrictive licenses, i.e. licenses with a *viral clause* (such as the GNU GPL), may affect developers' and users' perceptions of cost and benefits of developing and using the software by restricting compatibility of code with other software products and its commercialization potential. In accordance with Stewart et. al.'s (2006) work, we distinguish between projects with and without a license carrying a viral clause. In addition, there were a few projects (n=16) which did not clarify the license being used. We created 2 dummy variables "*Permissive License*" and "*Other License*" with restrictive licenses as the reference group.

We present the descriptive statistics of level-1 and level-2 variables in Tables 5 and 6 respectively. The correlations among developer Recency (X), Frequency (TX), PALIVE and the outcome variable PDEAD are presented in Table 7 for each of the three observation periods. We note that all the correlations in Table 7 were significant, and that the correlations are highest among the adjacent periods, as was to be expected. The pattern of correlations among X, TX and PALIVE/ PDEAD for any given period showed that both Frequency and Recency are correlated highly with PALIVE/PDEAD, however that Recency is more strongly correlated than Frequency. This suggests that a more recently active developer has more chances of being alive than a frequent developer whose last activity was further in the past. This leads to, at first glance somewhat counterintuitive, but reasonable suggestion that a developer who might have contributed heavily in past has lower probability of being alive than a less frequent developer who contributed very recently. We present some typical examples of developer Recency, Frequency and PALIVE values in Table 8. Comparing the developers 5 and 6 we find that even though developer 6 did have one activity in the 90 day period, he has lower chances of being alive than developer 5 who made no commits during this period. Another interesting observation can be made when comparing developers 4 and 5. Even though developer 4 has been more and recently active than developer 5, his chances of being alive are slightly lower than developer 5. This is because the Pareto/NBD model assumes different activity and death rates among developers. A developer who has a history of frequent contributions will have to not only maintain the contribution level but will have to be more recently active as well in order to get higher PALIVE values. On the other hand, a developer who has been a historically lethargic contributor might still have higher chances of being alive because the model assumes a slower activity rate for him. This is where the Pareto/NBD crucially differs from the heuristic approaches that have been used.

Table 5. Level-1 Descriptive Statistics (Developer Level). N=2949

Variable	Mean	S.D	Minimum	Maximum
Frequency of Activity during 90 day period (X90)	1.72	6.42	0	62
Recency of Activity during 90 day period (TX90)	10.40	25.56	0	90
P(Alive) at the end of 90 day period (PALIVE90)	0.22	0.24	0	1
P(Dead) at the end of 90 day period (PDEAD90)	0.78	0.24	0	1
Frequency of Activity during 180 day period (X180)	3.12	11.86	0	124
Recency of Activity during 180 day period (TX180)	24.11	54.87	0	180
P(Alive) at the end of 180 day period (PALIVE180)	0.17	0.27	0	1
P(Dead) at the end of 180 day period (PDEAD180)	0.83	0.27	0	1
Frequency of Activity during 270 day period (X270)	4.35	16.85	0	199
Recency of Activity during 270 day period (TX270)	36.45	80.85	0	270
P(Alive) at the end of 270 day period (PALIVE90)	0.13	0.25	0	1
P(Dead) at the end of 270 day period (PDEAD270)	0.87	0.25	0	1
Part Owner Dummy	0.32	0.46	0	1
Committer Dummy	0.55	0.50	0	1
Contributor Dummy	0.07	0.26	0	1
Number of Projects	3.26	5.09	1	144
(Number of Projects)^2	36.47	434.04	1	20736
Part Owner Dummy × Number of Projects	1.37	4.39	0	144
Committer Dummy × Number of Projects	1.51	3.21	0	99
Contributor Dummy × Number of Projects	0.14	0.68	0	12

Table 6. Level-2 Descriptive Statistics (Project Level). N=446

Variable	Mean	S.D	Minimum	Maximum
Project Age (in years)	2.97	1.98	0.03	15.51
Number of Developers	6.79	7.39	2	51
Permissive License Dummy	0.51	0.50	0	1
Other License Dummy	0.04	0.19	0	1
Proportion of Issues Resolved	0.53	0.34	0	1
Log (Downloads By Age)	2.92	1.39	-0.66	6.15

Table 7. Correlations Among Developer Frequency, Recency, P(Alive) and P(Dead)

	X90	TX90	PAlive90	PDead90	X180	TX180	PAlive180	PDead180	X270	TX270	PAlive270	PDead270
X90	1	.99	.55	-.55	.92	.89	.34	-.34	.89	.85	.22	-.22
TX90	.99	1	.60	-.60	.91	.89	.38	-.38	.88	.85	.25	-.25
PAlive90	.55	.60	1	-1	.51	.53	.56	-.56	.50	.51	.44	-.44
PDead90	-.55	-.60	-1	1	-.51	-.53	-.56	.56	-.50	-.51	-.44	.44
X180	.92	.91	.51	-.51	1	.99	.49	-.49	.96	.94	.34	-.34
TX180	.89	.89	.53	-.53	.99	1	.55	-.55	.96	.94	.40	-.40
PAlive180	.34	.38	.56	-.56	.49	.55	1	-1	.48	.51	.67	-.67
PDead180	-.34	-.38	-.56	.56	-.49	-.55	-1	1	-.48	-.51	-.67	.67
X270	.89	.88	.50	-.50	.96	.96	.48	-.48	1	.99	.41	-.41
TX270	.85	.85	.51	-.51	.94	.94	.51	-.51	.99	1	.49	-.49
PAlive270	.22	.25	.44	-.44	.34	.40	.67	-.67	.41	.49	1	-1
PDead270	-.22	-.25	-.44	.44	-.34	-.40	-.67	.67	-.41	-.49	-1	1

Note: All correlations are significant at .01 level (2-tailed).

Table 8. Typical examples of developer Recency, Frequency and PALIVE

Developer	X90	TX90	PALIVE90	X180	TX180	PALIVE180	X270	TX270	PALIVE270
1	18	90	1	26	171	.94	41	270	1
2	24	90	1	60	180	1	92	270	1
3	1	87	.98	1	87	.52	1	87	.28
4	0	0	.14	8	166	.93	9	242	.89
5	0	0	.14	0	0	.06	1	237	.91
6	1	1	.03	1	1	.00	1	1	.00
7	1	1	.01	8	158	.87	23	267	.99

3.5 METHODOLOGY

Modeling the effects of developer level and project level variables turnover presents considerable conceptual and methodological challenges. Most studies in OSS literature focus on activity levels in project by considering either the developer level variables (Hertel et. al, 2003; Hars et al., 2002), or project level factors such as license choice and organizational sponsorship (Stewart, et al 2006). To the best of the authors' knowledge, none of the studies have looked at turnover in open source projects by considering both levels of analysis (see Setia et al. 2010 for an exception). Including project level variables in a developer level model is likely to create aggregation bias, which can underestimate the effects of variables that are estimated at the inappropriate level. While including aggregated values of developer level variables in a project level model may fail to fully capture the effects of certain variables (Rumberger, 1995).

The Hierarchical Linear Modeling (HLM) technique allows researchers to model developer level outcomes within projects and model any between project differences that arise as a part of the nested design[17] (Raudenbush and Bryk, 2002). Although we had hypothesized that a developer's characteristics (ownership level, number of project associations) would impact turnover from project, there are reasons to believe that such effects may vary across projects. Thus, the effect of developer characteristics is nested within the effect of project level characteristics. A nested structure of the data also means that the independence of observations can no longer be assumed – developers working in one project may be more similar to each other than developers working on another project. HLM relaxes the independence assumption and allows information to be analyzed across multiple levels and hence is more robust for multilevel data than OLS (Luke, 2004). Finally, our theoretical framework proposes hypotheses that are composed of constructs operating and interacting at two levels, thereby suggesting a need for a multilevel model.

There is no single best way to build multilevel models and individual steps that a researcher should take in building the models depends on the research questions (Luke, 2004). A typical approach requires first testing an unconditional model and incrementally building conditional models (Setia et. al. 2010). An unconditional or a "Null" model is a simple one-way ANOVA model with random effects. It involves no predictors at any level and is useful to judge

[17] A nested design assumes that a developer works on only one among the 446 "focal" projects. If this assumption is not valid then a cross-classified should be considered (Raudenbush and Bryk, 2002). However, in our data set there were only 80 developers, i.e. 2.71% of total 2949, who worked on multiple focal projects. Since there is practically no variation for a cross-classified design, we instead chose to randomly delete "extra" developer associations within the focal projects. This way each developer worked on only one focal project in our sample as required by the nested design.

the proportion of variability in the DV (turnover) across the developer and project level predictors (i.e. within and between projects). The Null model can be useful tool to judge if a multilevel model is warranted in the first place. Once the need for multilevel modeling has been established, the researcher then proceeds to build a series of conditional models (i.e. models with predictors) in order to explain the variability in the DV. Typically, the models are built bottoms-up by constructing the level-1 model first to explain within-group (i.e. within-project) variability and then explaining between-group variability using level-2 predictors (Snijders and Bosker, 2012; Raudenbush and Bryk, 2002).

While entering the variables in the model, it is recommended that they should be centered. We also centered all our variables, except the dummies, before the analysis. Centering the variable allows for ease of interpretation, reduces multicollinearity concerns and enhances the quality of the results (Setia et. al. 2010). All the developer level variables were group-mean centered while the project level variables were grand-mean centered (Raudenbush and Bryk, 2002). Next, in order to test for any multicollinearity among predictor variables, we checked for correlations among variables and the corresponding VIF values. As can be seen in Tables 9 and 10, none of the correlation values were abnormally high. In addition, none of the VIF values were greater than 5. This suggested that multicollinearity was unlikely to be an issue. In the next section, we present the HLM models that were run and the accompanying results. We utilized full-maximum likelihood procedure to test our models in the HLM7.0 software (Bryk et. al. 1996).

Table 9. Correlations among Level-1 Predictors and VIF values

	Part Owner Dummy	Committer Dummy	Contributor Dummy	Number of Projects	VIF
Part Owner Dummy	1	-.75	-.19	.16	4.34
Committer Dummy	-.75	1	-.31	-.13	4.65
Contributor Dummy	-.19	-.31	1	-.10	2.09
Number of Projects	.16	-.13	-.10	1	1.02

Note: All correlations are significant at .01 level (2-tailed). None of the VIF values are above 10.

Table 10. Correlations among Level-2 Predictors and VIF values

	Project Age	Number of Developers	Permissive License Dummy	Other License Dummy	Proportion of Issues Resolved	Log (Downloads By Age)	VIF
Project Age	1	.21**	.03	-.13**	.21**	.19**	1.14
Number of Developers	.21**	1	.02	.08	.19**	-.06	1.10
Permissive License Dummy	.03	.02	1	-.19**	.04	.11*	1.06
Other License Dummy	-.13**	.08	-.19**	1	-.03	-.11*	1.08
Proportion of Issues Resolved	.21**	.19**	.04	-.03	1	.14**	1.09
Log (Downloads By Age)	.19**	-.06	-.11*	-.11*	.14**	1	1.09

Note: All correlations are significant at .01 level (2-tailed). None of the VIF values are above 10.

3.6 MODELS AND RESULTS

3.6.1 Unconditional (Null) Models

As mentioned earlier, an unconditional model does not allow any predictor variables and is only used to estimate the level of variance that resides within and between projects. We ran three different Null models for the three DVs, namely *PDEAD90*, *PDEAD180* and *PDEAD270*. This allowed us to select the best DV for further analysis with conditional models. Figure 2 shows the 3 unconditional models.

Dependent Variable	PDEAD90	PDEAD180	PDEAD270
Null Model	$PDead90_{ij} = \beta_{0j} + r_{ij}$ $\beta_{0j} = \gamma_{00} + u_{0j}$	$PDead180_{ij} = \beta_{0j} + r_{ij}$ $\beta_{0j} = \gamma_{00} + u_{0j}$	$PDead270_{ij} = \beta_{0j} + r_{ij}$ $\beta_{0j} = \gamma_{00} + u_{0j}$

Figure 2. Unconditional (Null) Models

Where, $PDead90_{ij}$ represents the outcome probability for i^{th} developer in the j^{th} project, β_{0j} is the mean outcome for the j^{th} project, r_{ij} is the unique effect of the i^{th} developer (error) with a variance σ^2, γ_{00} is the grand mean of turnover in the population and u_{0j} is the random effect of j^{th} project with variance $\tau_{00.}$ Table 11 presents the results of the 3 null models.

Table 11. Unconditional (Null) Models with DV observed over 90, 180 and 270 day periods.

Fixed Effects	PDEAD90	PDEAD180	PDEAD270
Intercept, γ_{00}	.774***	.822**	.857***
Random Effects			
Level-2 variance, $\tau_{00} = \text{var}(u_{oj})$ Level-1 variance, $\sigma^2 = \text{var}(r_{ij})$	.00422*** .05163	.00860*** .06614	.00727*** .05776

As can be seen in Table 11, the population mean of probability that a given developer is dead (PDEAD) increases from 77.4% to 85.7% with an increase in the observation length. Thus, projects tend to lose developers with time, a rather expected result. Note that our study was limited to tracking developers that were already associated with the projects and ignores new developers that may have joined the projects during the observation period. Thus it seems essential for projects to keep on attracting new members in order to survive. The level-2 variances, τ_{00}, in all 3 models are highly significant suggesting the need for further exploration with conditional models. In order to select the best observation period (best outcome variable) we noted the following in table 11:

- The total variance in the outcomes available for explanation ($\tau_{00} + \sigma^2$) increases from .05585 for the 90 day period to .07474 for the 180 day period, an increase of 33.82%. Surprisingly, the total variance for the 270 day period (.06503) is less than the 180 day period, a decrease of 12.99%.

- The Intraclass Correlation Coefficient (ICC)[18] values for the 3 models suggest that the proportion of explainable level-2 (i.e. between project) variance increases from 7.55% (90 day) to 11.50% (180 days) but drops slightly to 11.18% (270 day period).

These observations suggest that the 180 day period might be more suitable than 90 and 270 day periods on the basis of amount of variation available for explanation. In addition, we found that there were 87 *additional* developers that contributed in the 180 day period that did not contribute at all in the 90 day period. These 87 developers represent a sizable addition to the 517 developers who had made at least one commit during the 90 day period. The similar number for the 270 day period was 41, i.e. less than half. The 180 day observation period has also been used in previous research, although using the heuristic measures of turnover (Joyce and Kraut, 2006). Finally, the 180 day period, being half-way between the short-term and long-term periods, offers a reasonable compromise and is also practically feasible for researchers wishing to study turnover in non-contractual settings. The above arguments suggested to us that measuring the outcome over the 180 day period (i.e. *PDEAD180*) was preferable over others, given our data. Thus, we retained *PDEAD180* as the outcome variable to be further analyzed using the conditional models.

3.6.2 Conditional Models

The conditional model allows predictors at both levels so that parts of sources of variability in the outcome may be accounted by the measured variables. However, the conditional models for

[18] The ICC is the proportion of level-2 variance to the total variance (Snijders and Bosker, 2012).

HLM allow for an extraordinarily rich class of modeling possibilities (Raudenbush and Bryk, 2002). As mentioned earlier, a typical approach is to model within-project variability first using level-1 predictors, and then modeling the between-project variability using level-2 predictors. In order to test for the main effects of level-1 predictors (i.e. H1 and H2) we tested a main effect only one-way ANCOVA with random effects model (Model 2). Figure 3 presents Model 2.

Level-1 Model:

$$PDEAD180_{ij} = \beta_{0j} + \beta_{1j}\,(\mathit{Part\ Owner\ Dummy}) + \beta_{2j}\,(\mathit{Committer\ Dummy}) + \beta_{3j}\,(\mathit{Contributor\ Dummy}) + \beta_{4j}\,(\mathit{Number\ of\ Projects}) + \beta_{5j}\,(\mathit{Number\ of\ Projects})^{2} + r_{ij}$$

Level-2 Model:

$$\beta_{0j} = \gamma_{00} + u_{0j}$$

$$\beta_{1j} = \gamma_{10}$$

$$\beta_{2j} = \gamma_{20}$$

$$\beta_{3j} = \gamma_{30}$$

$$\beta_{4j} = \gamma_{40}$$

$$\beta_{5j} = \gamma_{50}$$

Figure 3. MODEL 2: One way ANCOVA with Random Effects (main effects)

Where, γ_{00} is the mean probability of turnover for a typical sole owner (reference group) associated with 3.26 projects across all projects[19]; γ_{10}, γ_{20} and γ_{30} represent the differences in probabilities of turnover among a typical part owner, committer and contributor as compared to a typical sole owner (reference group) respectively. Finally, γ_{50} and γ_{60} represent the main effects of

[19] Note that the variable *Number of Projects* was centered with a mean of 3.26.

Number of Projects and (*Number of Projects*)^{2}, i.e. linear and curvilinear effects, on probability of turnover. The third column of Table 12 presents the results of model 2[20].

[20] For ease of reference, Table 12 also presents the results of the null model (Model 1).

Table 12. Results of HLM estimation

Fixed Effects	Model 1	Model 2	Model 3	Model 4	Model 5
Intercept, γ_{00}	.774***	.651***	.628***	.616**	.620***
For H1: Ownership (reference group: "Sole Owner"),					
Part Owner, γ_{10}		.114***	.144***	.142***	.137***
Committer, γ_{20}		.231***	.252***	.261***	.253***
Contributor, γ_{30}		.275***	.299***	.312***	.297***
For H2: Developer's Project Associations,					
Number of Projects, γ_{40}		-.002	.004	.005	.004
(Number of Projects)^2, γ_{50}		.00002*	.00004**	.00004**	.00003*
For H3: Ownership × Number of Projects,					
Part Owner × Number of Projects, γ_{60}			-.010***	-.010**	-.009*
Committer × Number of Projects, γ_{70}			-.006*	-.007*	-.006
Contributor × Number of Projects, γ_{80}			-.008	-.011	-.008
For Project Level Control Variables,					
Project Age, γ_{01}				.003	.003
Permissive License, γ_{02}				.023	.023
Other License, γ_{03}				-.007	-.008
Number of Developers, γ_{04}				-.001	-.001
For H4: Project Success Variables,					
Proportion of Issues Resolved, γ_{05}				-.061***	-.040
Log (Downloads By Age), γ_{06}				-.011**	-.032**

For H5: Ownership × Project Success, Part Owner × Proportion of Issues Resolved, γ_{11} Part Owner × Log (Downloads By Age), γ_{12} Committer × Proportion of Issues Resolved, γ_{21} Committer × Log (Downloads By Age), γ_{22} Contributor × Proportion of Issues Resolved, γ_{31} Contributor × Log (Downloads By Age), γ_{32}					-.066 .024 .022 .018 .050 .035*
Random Effects					
Level-2 variance, $\tau_0^2 = \text{var}(u_{oj})$ Level-1 variance, $\sigma^2 = \text{var}(r_{ij})$	.00860*** .06614	.00998*** .06047	.00978*** .06041	.00884*** .06040	.00919*** .06002
Model Fit					
Deviance (-2 log likelihood) Deviation Difference ($\Delta\chi^2$) Estimated Parameters Level-1 R^2	599.467 3 	379.029 220.438*** 8 5.73%	372.246 6.783* 11 6.08%	352.835 19.411*** 17 7.35%	342.342 10.493 23 7.39%

Notes: *p<.10, **p<.05, ***p<.01. Deviation differences are calculated as the absolute differences in deviance values between the current and the previous model, e.g., ΔD3=|D3-D2| etc. Significance of this difference is tested after accounting for the estimated parameters in the models.

In HLM, the Deviance difference test allows comparison of two nested models on same data set. The difference in deviances of two models can be used as a chi-squared distributed test statistic with a degree of freedom that equals the difference in number of estimated parameters (Snijders and Bosker, 2012). The deviance test comparison between Model 1 and 2 suggested that model 2 was a significantly better fit to the data than model 1, $\Delta D = 220.438$, $\chi^2_{df} = 5$, $p < .01$.

The coefficients γ_{10}, γ_{20} and γ_{30} were all significant at the .01 level. The value $\gamma_{10} = .114$ suggests that a typical part owner was 11.4% more likely to turnover than a typical sole owner. In addition, a typical committer and contributor were 23.1% and 27.5% more likely to turnover than sole owner in the 180 day period. These findings support H1, which suggested that higher levels of ownership in the project are associated with lower levels of turnover.

The linear effect of *Number of Projects*, γ_{40}, was not significant. However, the curvilinear effect, γ_{50}, was significant. The positive value of γ_{50} suggested a convex or U-shaped effect supporting H2. Next, in order to test the interaction hypothesis between ownership and number of projects a developer is associated with (H3), we analyzed model 3 as an extension of model 2 but with the interaction terms. Figure 4 presents model 3.

Level-1 Model:

$$PDEAD180_{ij} = \beta_{0j} + \beta_{1j}\,(Part\ Owner\ Dummy) + \beta_{2j}\,(Committer\ Dummy) + \beta_{3j}\,(Contributor\ Dummy) + \beta_{4j}\,(Number\ of\ Projects) + \beta_{5j}\,(Number\ of\ Projects)^{2} + \beta_{6j}\,(Part\ Owner\ Dummy \times Number\ of\ Projects) + \beta_{7j}\,(Committer\ Dummy \times Number\ of\ Projects) + \beta_{8j}\,(Contributor \times Number\ of\ Projects) + r_{ij}$$

Level-2 Model:

$$\begin{aligned}
\beta_{0j} &= \gamma_{00} + u_{0j}\\
\beta_{1j} &= \gamma_{10}\\
\beta_{2j} &= \gamma_{20}\\
\beta_{3j} &= \gamma_{30}\\
\beta_{4j} &= \gamma_{40}\\
\beta_{5j} &= \gamma_{50}\\
\beta_{6j} &= \gamma_{60}\\
\beta_{7j} &= \gamma_{70}\\
\beta_{8j} &= \gamma_{80}
\end{aligned}$$

Figure 4. MODEL 3: One way ANCOVA with Random Effects (interaction effects)

Where, γ_{60}, γ_{70} *and* γ_{80} represent differences in slopes of part owners, committers and contributors with the slope for sole owners (γ_{40}) for the relationship between turnover and *Number of Projects,* respectively.

After including the 3 interaction terms (model 3) the model fit was significantly improved in comparison to model 2 as suggested by the deviance difference test[21], $\Delta D = 6.783$, $\chi^{2}_{df} = 3$, $p < .10$. The coefficients γ_{60} and γ_{70} were significant suggesting that the differences in

[21] Adding interaction terms of ownership with (*Number of Projects*)^{2} did not lead to an improved model and thus were not included.

slopes among part owners and committers with sole owners were significant. However, this difference was not significant for contributors. That is, there were significant differences in the prediction of turnover probability by *Number of Projects* between part owners and sole owners, and between committers and sole owners. The negative sign on part owner slope ($\gamma_{60} + \gamma_{40}$ = - 0.60) and committer slope ($\gamma_{60} + \gamma_{40}$ = - 0.20) suggested that part owners and committers were less likely to turnover than sole owners with an increase in *Number of Projects*, thereby supporting H3. The random effects for model 3 showed that, even though reduced in comparison to model 2, there was still significant residual variance remaining at level-2 (τ_0 = 00978, $p < .01$). This suggested modeling this variance with level-2 predictors.

In order to test for the effect of project success on turnover probability (H4) we included the variables *Proportion of Issues Resolved* and *Log (Downloads By Age)* as level-2 predictors, while controlling for developer characteristics. The project level variables that were controlled for were *Project Age*, *Permissive License*, *Other License* and *Number of Developers*. This *intercepts-as-outcome* (model 4) is presented in Figure 5, while the results are presented in the 5th column of Table 12.

Level-1 Model:

$$PDEAD180_{ij} = \beta_{0j} + \beta_{1j}\,(Part\ Owner\ Dummy) + \beta_{2j}\,(Committer\ Dummy) + \beta_{3j}\,(Contributor\ Dummy) + \beta_{4j}\,(Number\ of\ Projects) + \beta_{5j}\,(Number\ of\ Projects)^2 + \beta_{6j}\,(Part\ Owner\ Dummy \times Number\ of\ Projects) + \beta_{7j}\,(Committer\ Dummy \times Number\ of\ Projects) + \beta_{8j}\,(Contributor \times Number\ of\ Projects) + r_{ij}$$

Level-2 Model:

$$\beta_{0j} = \gamma_{00} + \gamma_{01}\,(Project\ Age) + \gamma_{02}\,(Permissive\ License\ Dummy) + \gamma_{03}\,(Other\ License\ Dummy) + \gamma_{04}\,(Number\ of\ Developers) + \gamma_{05}\,(Proportion\ of\ Issues\ Resolved) + \gamma_{06}\,(Log\ (Downloads\ by\ Age)) + u_{0j}$$

$$\beta_{1j} = \gamma_{10}$$
$$\beta_{2j} = \gamma_{20}$$
$$\beta_{3j} = \gamma_{30}$$
$$\beta_{4j} = \gamma_{40}$$
$$\beta_{5j} = \gamma_{50}$$
$$\beta_{6j} = \gamma_{60}$$
$$\beta_{7j} = \gamma_{70}$$
$$\beta_{8j} = \gamma_{80}$$

Figure 5. MODEL 4: Intercepts-as-outcome

Where, γ_{05} and γ_{06} represent the effects of *Proportion of Issues Resolved* and *Log (Downloads By Age)* on the mean probability of turnover, i.e. the main effect of project success; and γ_{01}, γ_{02}, γ_{03} and γ_{04} represent the effect of control variables.

The deviance difference test in Table 12 showed that model 4 was a significant improvement over model 3, ΔD = 19.411, $\chi^2_{df} = 6$, $p < .01$. We also observed significant negative effects of *Proportion of Issues Resolved* ($\gamma_{05} = -.061$, $p < .01$) and *Log (Downloads By Age)* ($\gamma_{05} =$

-.011, $p < .01$) on probability of turnover thereby supporting H4. Interestingly, none of the control variables had a significant effect on the outcome. The variance component of level-2 random effect was reduced compared to model 3, however it was still significant ($\tau_0 = 00884$, $p < .01$).

Next, in order to test if there were cross-level interactions (i.e. interactions among level-1 and level-2 predictors) as hypothesized in H5, we tested a slopes-as-outcome (model 5) as presented in Figure 5. The results are presented in the 6th column of Table 12.

The deviance difference test showed that model 5 was not better than model 4, $\mathit{\Delta D} = 10.493$, $\chi^2_{df} = 6$, $p > .10$, and hence was rejected in favor of the more parsimonious and better fitting model 4 (Raudenbush and Bryk, 2002). Thus, we did not find any support for H5.

Level-1 Model:

$$PDEAD180_{ij} = \beta_{0j} + \beta_{1j}\,(Part\ Owner\ Dummy) + \beta_{2j}\,(Committer\ Dummy) + \beta_{3j}\,(Contributor\ Dummy) + \beta_{4j}\,(Number\ of\ Projects) + \beta_{5j}\,(Number\ of\ Projects)^{2} + \beta_{6j}\,(Part\ Owner\ Dummy \times Number\ of\ Projects) + \beta_{7j}\,(Committer\ Dummy \times Number\ of\ Projects) + \beta_{8j}\,(Contributor \times Number\ of\ Projects) + r_{ij}$$

Level-2 Model:

$$\beta_{0j} = \gamma_{00} + \gamma_{01}\,(Project\ Age) + \gamma_{02}\,(Permissive\ License\ Dummy) + \gamma_{03}\,(Other\ License\ Dummy) + \gamma_{04}\,(Number\ of\ Developers) + \gamma_{05}\,(Proportion\ of\ Issues\ Resolved) + \gamma_{06}\,(Log\ (Downloads\ by\ Age)) + u_{0j}$$

$$\beta_{1j} = \gamma_{10} + \gamma_{11}\,(Proportion\ of\ Issues\ Resolved) + \gamma_{12}\,(Log\ (Downloads\ by\ Age))$$

$$\beta_{2j} = \gamma_{20} + \gamma_{21}\,(Proportion\ of\ Issues\ Resolved) + \gamma_{22}\,(Log\ (Downloads\ by\ Age))$$

$$\beta_{3j} = \gamma_{30} + \gamma_{31}\,(Proportion\ of\ Issues\ Resolved) + \gamma_{32}\,(Log\ (Downloads\ by\ Age))$$

$$\beta_{4j} = \gamma_{40}$$

$$\beta_{5j} = \gamma_{50}$$

$$\beta_{6j} = \gamma_{60}$$

$$\beta_{7j} = \gamma_{70}$$

$$\beta_{8j} = \gamma_{80}$$

Figure 6. MODEL 5: Intercepts-as-outcome

3.7 LIMITATIONS AND FUTURE WORK

This study is limited in several ways. First, we only chose to observe developer contributions as code commits. However, past research has shown the importance of other kinds of contributions, often by peripheral developers, such as code documentation, managing websites, wikis and mailing lists, popularizing projects through positive word-of-mouth etc. (Setia et. al 2010). However, since code commits are a necessary (but not sufficient) condition for the success of an open source project, understanding factors that may impact the actual code development are necessary.

Second, we only chose to observe projects that used SVN as their code management system and neglected projects that used other increasingly popular systems such as GIT and Mercurial. While it is our conjecture that our results should be generalizable to projects using other SCM systems, it remains a work for the future.

Third, our choice of project success measures is in no manner exhaustive. Past OSS literature has differentiated between the market success, technical success and team effectiveness of the project and why such a distinction is warranted (e.g. Grewal et. al. 2006, Stewart and Gosain 2006, Crowston et. al. 2004, Sagers 2004, Crowston and Scozzi 2002). It would be interesting to further explore what *kind* of project success matters more for which relationship. In addition, Shah (2006) argued that OSS developers are motivated for a variety of reasons and explored the differences between core and peripheral developers. Therefore, it would be interesting to see what kinds of project success or failure impacts the retention rates of which group.

. Fourth, we had excluded projects that did not show any activity in the 30 day period before data collection, an arbitrary cut-off. However, on revisiting some projects that were excluded it was found that the projects were still in-development. In fact there were some projects that were “resurrected” after long periods of inactivity. This has the potential to bias our results in favor of recently active projects.

Fifth, we assumed that any developer whose name was listed on the project was still associated with it. While this is true in a formal sense, some developers may have “left” the project even though their names still appeared on the project list. There were many developers in our data set that never made any code commits in any observation period. In such a case, their Pareto/NBD *PDEAD* values kept decreasing with increasing observation periods. On the other hand, there were some developers that “resurrected” themselves and became more active as observation period increased. This suggests that future researchers might be able to uncover interesting findings using longitudinal or time-series analyses.

Finally, we only analyzed projects that were not sponsored by an external agency. Increasingly, many for-profit companies are sponsoring open source projects and paying developers to work on them (West and O’Mahony, 2008). Our results cannot be generalized to such settings and more work is required to distinguish any differences in the phenomena analyzed in this study among sponsored projects.

4.0 OVERALL CONTRIBUTIONS

There are likely to be several contributions of this work. Table 13 presents a summary of the hypotheses and results of the empirical analyses. First, we merged the Ownership and Social Identity literatures to argue that the effect of ownership on member retention is moderated by the threat to the identity, i.e. project success. We showed that managing ownership structures are critical for projects to maintain an active developer base. Overall, sole owners were least likely to turnover followed by part owners, committers and contributors. However, as the number of project associations increased, lower ownership levels were associated with greater chances of retention. This presents a trade-off for project managers in terms of managing the ownership structures in the project. The implications are likely to be generalizable to other online community forms of production such as Wikis.

Second, we argued that the correct way to model OSS developer participation level is to acknowledge the inherently nested nature of the data. Analyzing effects at multiple (i.e. at individual and group) levels and how they may interact allows for a deeper understanding of how developers are likely to continue working on projects. For example, is there a difference in turnover rates among sole-owners in a small project (with few other developers) and sole-owners in a large project? In the former case, the sole owner may feel lower levels of accountability while in the latter case he/she may feel more accountable and responsible for the well-being of the project. This opens up interesting avenues as to how the size and the presence and number of other owners affect the performance of a developer. Since OSS developers work under multiple contexts, it is very likely that the extra-project context also matters. This will allow OSS researchers to begin exploring how developers regulate their commitment levels across multiple projects.

Table 13. **Summary for Hypothesized Relationships**

Hypothesis 1: An OSS participant's level of ownership in the project will be negatively associated with turnover. Specifically, we hypothesized that sole owners will be least likely to turnover from a project followed by part owners, committers and contributors respectively.	Supported
Hypothesis 2: Overall, the number of projects a developer is associated with has a U-shaped relationship with turnover in a given project. Specifically, the probability of developer turnover will decrease up to a point as the number of projects the developer is associated with on increases, after which it will increase.	Supported
Hypothesis 3: The number of currently active projects a developer is associated with, will moderate the relationship between the level of ownership of the developer and turnover in that project. Specifically, as the number of associations increase, developers with higher levels of ownership in the focal project will be more likely to leave than developers with lower levels of ownership.	Supported
Hypothesis 4: Overall, the success of the OSS project is negatively related to turnover in that project. Specifically, as the project success increases the chances of a developer working in it leaving are reduced.	Supported
Hypothesis 5: The success of the OSS project will moderate the relationship between the level of ownership of developer and turnover in that project. Specifically, as the project success decreases, developers with higher level of ownership will be less likely to leave than developers with lower levels of ownership.	Not Supported

Third, we introduced a practical way of operationalizing turnover and are the first to show the application of the Pareto-NBD model in an online community context. We argued that this approach is preferable over the heuristic measures that have been used in the past, and can become an important tool for future researchers. In order to assess the adequate length of observation period, we compared 3 potential outcomes spread over short, medium and long term, and concluded that a medium term observation period appears suitable in the OSS context. In fact, the Pareto-NBD model may be strained if very long histories of developer activities are taken into account (Schmittlein et al. 1987). If the complete past of a developer that has worked for years is taken into account, the death rate μ will be close to zero, thus allowing the developer to stay alive for a very long time in the future. In the Marketing context, Schmittlein et al. (1987) recommended an observation period not greater than 2 years even in presence of more data. Since other online voluntary contexts such as Wikis may inherently different rates of development and life cycles, it opens the door for researchers to assess appropriate periods in these fresh contexts.

BIBLIOGRAPHY

Abdel-Hamid, T., “Investigating the Impacts of Managerial Turnover/Succession on Software Project Performance”, Journal of Management Information Systems, 9(2), 1992.

Avey, J. Avolio, B. Crossley, C. and Luthans, F. “Psychological Ownership: Theoretical Extensions, Measurement, and Relation to Work Outcomes,” *Journal of Organizational Behavior*, (30) 2009, pp 173-191.

Beggan, J. K. “On the social nature of nonsocial perception: The mere ownership effect,” Journal of Personality and Social Psychology (62) 1992, pp 229–237.

Beggan, J. K. “Using what you own to get what you need: The role of possessions in satisfying control motivation,” *Journal of Social Behavior and Personality* (6:6) 1991, pp 129-146.

Belk, R. W. "Possessions and the Extended Self," *Journal of Consumer Research* (15) 1988, pp 139-168.

Brown G., Lawrence, T. B. and Robinson, S. L. “Territoriality in Organizations,” *Academy of Management Review* (30) 2005, 577-594.

Bryk, A., Raudenbush, S. and Congdon, R. “*HLM. Hierarchical Linear and Non Linear Modeling with the HLM/2L and HLM/3L Programs*,” Chicago: Scientific Software International, 1996.

Butler, B. "Membership Size, Communication Activity, and Sustainability: A Resource-Based Model of Online Social Structures," *Information Systems Research* (12:4) 2001, pp 346-362.

Chengalur-Smith, I. N., Sidorova A. and Daniel S. (2010) “Sustainability of Open-Source Projects: A Longitudinal Study” *Journal of the Association for Information Systems* Vol. 11: Iss. 11, Article 5.

Conte, M. A. and Svejnar, J. “The Performance Effects of Employee Ownership Plans,” in: Paying for Productivity: A look at the Evidence, A.S. Blinder (ed.), Brookings Institution, Washington DC, 1990.

Crowston, K. and Scozzi, B. “Open Source Software Projects as Virtual Organizations: Competency Rallying for Software Development,” IEE Proceedings-Software, 149(1), 2002.

Crowston, K., Annabi, H., Howison, J., and Masango, C. "Effective Work Practices for Software Engineering: Free/Libre Open Source Software Development," *Proceedings of the 2004*

ACM Workshop on Interdisciplinary Software Engineering Research, ACM New York, Newport Beach, CA, 2004.

Crowston, K., Wei, K., Li, Q., Eseryel, U. and Howison, J. “Coordination of Free/Libre Open Source Software Development,” *Proceedings of the International Conference on Information Systems,* Las Vegas, NV, 2005.

Daniel, S. L., Agarwal, R., and Stewart, K.J. "The Diverse Effects of Diversity in Global, Distributed Collectives: A Study of User Participation in Open Source Projects," forthcoming at *Information Systems Research,* 2012.

Ellemers, N., Knippenberg, V.A., De Vries, N. and Wilke, H. “Social identification and permeability of group boundaries,” *European Journal of Social Psychology* (18) 1988, pp 497-513.

Fader, P., Hardie, B. and Lee, K. “Counting Your Customers the Easy Way: An Alternative to the Pareto/NBD Model,” Marketing Science, vol. 24 (2), pp.275-284, 2005.

Feller, J., Fitzgerald, B., Hissam, S. and Lakhani, K. “Perspectives on Free and Open Source Software,” Cambridge, MA: MIT Press.

Goode, W. “A Theory of Role Strain,” *American Sociological Review* (25) 1960, pp 483-496.

Grewal R., Lillien, G., and Mallapragada G., “Location, Location, Location: How Network Embeddedness Affects Project Success in Open Source Systems”, *Management Science,* 52 (7), 2006.

Hall, T., Beecham, S., Verner, J. and Wilson, D. “The impact of staff turnover on software projects: the importance of understanding what makes software practitioners tick”, Proceedings of the 2008 ACM SIGMIS CPR conference on Computer personnel, 2008

Hars, A. and Ou, S. “Working for free? Motivations of participating in open source projects,” (Proceedings of the 34th Hawaii International Conference on System Sciences)” *International Journal of Electronic Commerce* (6:3) 2002, pp 25–39.

Hertel, G., Niedner, S., and Herrmann, S. "Motivation of Software Developers in Open Source Projects: an Internet-Based Survey of Contributors to the Linux Kernel," *Research Policy* (32) 2003, pp 1159-1177.

Hogg, R., McKean, J. and Craig, A. “Introduction to Mathematical Statistics,” 5th ed., Prentice Hall, 2005.

Joyce, E., and Kraut, R.E. 2006. "Predicting Continued Participation in Newsgroups," Journal of Computer Mediated Communication (11), pp. 723-747.

Kang, D.L. and Sorenson, A.B. “Ownership organization and firm performance,” *Annu. Rev. Sociol* (25) 1999, pp 121-144.

Karlin, S. and Taylor, H. “A First Course in Stochastic Processes,” Academic Press, New York, 1975.

Keil, M., Truex III, D. and Mixon R. “The Effects of Sunk Cost and Project Completion on Information Technology Project Escalation*,” IEEE Transactions on Engineering Management,* 42(4), 1995.

Lakhani, K. and Wolf, R., “Why Hackers Do What They Do: Understanding Motivation Efforts in Free/F/OSS Projects,” Working Paper, MIT Sloan School of Management, 2003.

Long, R. “The Effects of Employee Ownership on Organizational Identification, Job Attitudes, and Organizational Performance: A Tentative Framework and Empirical Findings,” *Human Relations*, 31, 1978.

Long, R. “Job Attitudes and Organizational Performance under Employee Ownership,” *The Academy of Management Journal* (23:4) 1980, pp. 726-737.

Luke, D. “Multilevel Modeling,” Quantitative Applications in the Social Sciences, Sage, 2004.

March, J. G., and Simon, H. A. *Organizations*. New York: Wiley, 1958.

Mockus, A., Fielding, R. T., and Herbsleb, J. “D. Two case studies of open source software development: Apache and Mozilla,” *ACM Transactions on Software Engineering and Methodology*, (11) 2002, pp 309–346.

O’Mahony, S. and Ferraro, F. “The Emergence of Governance in an Open Source Community,” *Academy of Management Journal* (50:5) 2007, pp 1079–1106.

Pierce, J. and Rodgers, L. “The Psychology of Ownership and Worker-Owner Productivity,” *Group & Organization Management* (29:5) 2004, pp 588-613.

Pierce, J. L., Kostova, T. and Dirks, K. “Toward a Theory of Psychological Ownership in Organizations,” *Academy of Management Review* (26) 2001, pp 298-310.

R Development Core Team. 2011. “R: A Language and Environment for Statistical Computing,” the R Foundation for Statistical Computing, Vienna, Austria

Raudenbush, S.W. and Bryk, A.S. *Hierarchical Linear Models* (Second Edition), Thousand Oaks, Sage Publications, 2002.

Robles, G., and Gonzalez-Barahona, J.M. "Contributor Turnover in Libre Software Projects," in: *IFIP International Federation for Information Processing: Open Source Systems*, Springer, Boston, 2006.

Rodriguez-Bustos, C. and Aponte, J. “How Distributed Version Control Systems Impact Open Source Software Projects,” in the 9th IEEE Conference on Mining Software Repositories, pp. 36-39, 2012.

Rothschild-Whitt, J. “The collectivist organization: An alternative to rational-bureaucratic models,” *American Sociological Review* (44) 1979, pp 509–527.

Rothschild, J. and Russell, R. “Alternatives to bureaucracy: Democratic participation in the economy”, in: *Annual review of sociology*, Short, J. F. (ed.), Annual Reviews, Palo Alto, CA, 1986.

Sagers, G., "The Influence of Network Governance Factors on Success in Open Source Software Development Projects" (2004). *ICIS 2004 Proceedings.*

Sartre, J.P. Being and Nothingness: A Phenomenological Essay on Ontology, Philosophical Library: New York, 1969 (First Published in 1943).

Setia, P., Rajagopalan, B., Sambamurthy, V. and Calantone R. "A Theory and Empirical Test of Peripheral Developer Contributions to the Open Source Development Model," *Information Systems Research* (Articles in Advance) 2010, pp 1-23.

Schmittlein, D.C., Morrison, D.G., and Colombo, R. 1987. "Counting Your Customers: Who Are They and What Will They Do Next?," Management Science (33:1).

Sharma, P.N., Hulland, J. and Daniel, S. "Examining Turnover in Open Source Software Projects Using Logistic Hierarchical Linear Modeling Approach," in I. Hammouda et. al. (Eds): Open Source Systems: Long Term Sustainability, IFIP Advances in Information and Communication Technology (OSS 2012), v. 378, pp. 331-337, Springer Berlin Heidelberg, 2012.

Shaw, J. "Turnover Rates and Organizational Performance: A Review, Critique and Research Agenda," *Organizational Psychology Review*, 1(3), 2011.

Sieber, S. "Toward a Theory of Role Accumulation," *American Sociological Review* (39) 1974, pp 567-578.

Singh, P. "The Small World Effect: The Influence of Macro Level Properties of Developer Collaboration Networks on Open Source Project Success." *ACM Transactions of Software Engineering and Methodology*, 20(2), 2010.

Singh, P., Tan, Y. and Mookerjee, V. "Network Effects: The Influence of Structural Social Capital on Open Source Project Success", *Management Information Systems Quarterly*, 35(4), 2011.

Snijders, T. and Bosker, R. "Multilevel Analysis: An Introduction to Basic and Advanced Multilevel Modeling," Second Edition, Sage, 2012.

Stewart, K. and Gosain, S. "The impact of ideology on effectiveness in open source software development teams," *MIS Quarterly* (30:2) 2006, pp 291-314.

Stewart, K.J., Ammeter, T.A. and Maruping, L. 2006. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," Information Systems Research (17:2), pp. 126-144.

Tajfel, H. and John T. "The social identity theory of intergroup behavior," in: Worcehl, S. and Austin, W.G. (eds.), *Psychology of Intergroup Relations,* 2nd edition, Nelson-Hall, Chicago, 1985.

Ton, Z. and Huckman, R. "Managing the Impact of Employee Turnover on Performance: The Role of Process Conformance," *Organization Science*, 19(1), 2008.

von Hippel, E. and von Krogh, G. "Open source software and the "private-collective" innovation model: Issues for organization science," *Organization Science* (14) 2003, pp 209–223.

von Krogh, G., Spaeth, S., and Lakhani, K. R. "Community, joining, and specialization in open source software innovation: A case study," *Research Policy* (32) 2003, pp 1217–1241.

Wagner, I. and Rosen C. "Employee Ownership-Its Effects on Corporate Performance," *Employment Relations Today*, 12, 1985.

West, J and O'Mahony, S. "The Role of Participation Architecture in Growing Sponsored Open Source Communities," *Industry & Innovation*, 15:2, pp.145-168, 2008.