July 2009

UNDERSTANDING PARTICIPATION BEHAVIOR AND STATUS ATTAINMENT OF OPEN SOURCE SOFTWARE DEVELOPERS – A LATENT CLASS GROWTH MODELING APPROACH

Israr Qureshi
Hong Kong Polytechnic University, msisrar@inet.polyu.edu.hk

Yulin Fang
City University Hong Kong, ylfang@cityu.edu.hk

UNDERSTANDING PARTICIPATION BEHAVIOR AND STATUS ATTAINMENT OF OPEN SOURCE SOFTWARE DEVELOPERS – A LATENT CLASS GROWTH MODELING APPROACH

Abstract

The success of open source software (OSS) projects heavily depends on voluntary participation by a large number of developers. Developers new to an OSS community must participate by engaging in community interactions before they are qualified by the community as core developers. This exploratory study examines new peripheral developers' temporal participation behavior and its impacts on the time taken to attain core developer status. Using the novel latent class growth modeling approach on 133 peripheral developers across 40 OSS projects, we found that these peripheral developers differed in the initial levels and growth trajectories of participation, and distinct classes of participation behavior were identified. We also found that different classes of developers differ in their time taken to attain core developer status. Implications to research and practice are discussed.

INTRODUCTION

The open source software development (OSS) model originated in the 1970s, partially as a defensive reaction to the move by some private software companies to appropriate publicly-available software into their proprietary applications (Stallman and Lessig 2002). Over the last decade this intriguing software development model has emerged as a viable alternative to the commercial software projects (Fitzgerald 2006), and has attracted increasing academic and corporate attention (Sen 2007; Stewart, Ammeter and Maruping 2006). Some OSS projects have achieved remarkable adoption success. Among the best known OSS projects are the Linux operating system, as well as the Apache web server, which answers 70% of all the Web pages requests through the Internet (Netcraft 2004). For the commercial market, Gartner group estimates that the market for open source software services could reach US $4.3 billion by 2010. Other report that 60 per cent of the largest companies in North America had planned to implement OSS applications, half of these for mission-critical functions (Schadler 2004).

The notable success of open source software projects would not have been accomplished without individual developers' voluntary participation (Roberts, Hann and Slaughter 2006). Indeed, research has identified that failure in OSS development is frequently due to shortage of volunteer participation, whereas successful OSS projects often feature a large number of active participants (Crowston, Annabi and Howison 2003; Krishnamurthy 2002; Markus, Manville and Agres 2000). For this reason, considerable research has been motivated towards developing a deep understanding of OSS developer participation (von Krogh and von Hippel 2006).

A two-tier structure of OSS developers with differential roles of participation have been identified in the literature: the tier of *core* developers with authorization to submit code changes and that of *peripheral* developers with the permission to participate in the mailing list only (Lee and Cole 2003). Both types of developers are important for the success of OSS projects: while core developers have direct administrative responsibilities in maintaining the software codebase, peripheral developers make indirect contributions through their participation in the mailing list (Lee and Cole 2003). These tiers of developers are ecologically dependent on each other. On the one hand, core developers draw intellectual input from peripheral developers by relying on the periphery to generate patches of computer codes and bug reports. On the other hand, candidates of core developers must be drawn from the large pool of peripheral developers, per evaluation and nomination by the existing core developers (Fang and Neufeld 2009).

OSS research to date has tended to exclusively focus on core developer participation by understanding motivations to participate (Franke and von Hippel 2003; Hertel, Niedner and Herrmann 2003; Shah 2006; Von Hippel 2001; Von Krogh, Spaeth and Lakhani 2003) and performance impacts of participating (Fielding 1999; Roberts et al. 2006). However, our understanding of peripheral developers' participation behavior and performance outcomes have been limited despite their aforementioned importance, with

very few exceptions (Fang and Neufeld 2009; Von Krogh et al. 2003). It is important to understand peripheral developers' participation behavior as it is through continued participation that peripheral developers make themselves likely to be nominated as future core developers, which in turn sustains the coding activities in the OSS project (Lee and Cole 2003). Yet, the few studies with a partial focus on peripheral developers either established the association between levels of participation and individual performance, or identified types of participation required for a peripheral developer to become a core developer, without providing a clear picture about how their participation behaviors are changed over time and how such temporal changes affect their prospect as core developers (Von Krogh et al. 2003). It is likely that peripheral developers with different "career prospects" might exhibit differential temporal participation patterns; developing such a temporal perspective can enrich our understanding of peripheral developers' participation dynamics. The objective of the present exploratory study is thus to contribute to the OSS literature by *identifying* peripheral developers' temporal participation patterns and *exploring* their relationships with individual performance in terms of attainment of core developer status.

To address this research question, we base our conceptual argument on the existing OSS literature and empirically model the longitudinal participation trajectories of 133 peripheral developers in 40 OSS projects with differing progresses regarding attainment of core developer status, using the latent class growth analysis (LCGA) technique (Muthén and Muthén 2000; Nagin 1999). The nature of our research question entails an analytical technique that supports both the person-centered approach, which model individual trajectories based on intra-individual changes over time and classify individuals into distinct categories based on inter-individual differences in behavioral patterns (i.e., different classes of participation trajectories), as well as the variable-centered approach, which describes the relationships among variables (i.e., participation trajectories and developer status attainment). However, the existing methods in the information systems field has either only focused on the variable-center approach, such as regression and structural equation modeling (Gefen, Straub and Boudreau 2000), or on the person-centered approach, such as cluster analysis (Jain, Ramamurthy, Ryu and Yasai-Ardekani 1998; Malhotra, Gosain and El Sawy 2005) that account for inter-individual differences only and does not explain intra-individual changes. LCGA is an analytical technique that summarizes longitudinal data by modeling intra-individual and inter-individual variability in developmental trajectories through a small number of classes defined by unique sizes (initial values) and shapes (growth) (Nagin 1999), thus effectively consolidating the two approaches. This technique has found widespread usage in research on psychology and sociology (Kreuter and Muthén 2008; Pinquart and Schindler 2007; Wang and Bodner 2007; Wu and Witkiewitz 2008.). However, it has not yet been used in the information systems field. The second objective of this study is thus to introduce LCGA by demonstrating its usability within the context of OSS developer participation.

The remainder of the paper is organized as follows. In the next section, we review the OSS literature and identify four classes of peripheral developers with differing initial levels of participation and differing temporal patterns of participation. Hypotheses are developed to empirically explore the existence of these four classes of developers. We then introduce the latent class growth analysis technique and conduct an empirical investigation by drawing on a longitudinal dataset of 133 peripheral developers from 40 projects. Finally, empirical results are discussed in conjunction with the existing OSS literature.

CONCEPTUAL BACKGROUND AND HYPOTHESES

An OSS project involves a decentralized community of volunteer developers who collaborate to produce a software product using Internet-based tools such as e-mail, mailing lists, Web-based concurrent versioning systems (CVS), and bug reporting software. To date, OSS researchers and practitioners have been primarily interested in three sub-areas of research: (1) individual developer participation; (2) competitive dynamics; and (3) innovation processes, governance and organization [see (von Krogh and von Hippel 2006) for a summary of these areas]. While much additional research is required within each area, the focus of the present study is on developer participation and more precisely on temporal participation trajectories of peripheral developers prior to their attainment of the core developer status.

Peripheral developers differ from core developers in that they have limited access to codebase. Research has found that, while access to certain areas (e.g., CVS systems) was restricted to core developers who took on key technical activities and demonstrated advanced technical knowledge, access to the OSS community was free and open to everyone, resulting in a large pool of peripheral developers (Von Krogh et al. 2003). Participation by peripheral developers in an OSS community usually take the form of interacting with core developers and other peripheral developers on a variety of technical issues, such as reporting bugs, offering bug fixes, discussing on feature development (Von Krogh et al. 2003). However, only a small number of peripheral developers whose performance is successfully recognized and valued by the existing core developers can eventually be granted core developer status (Fang and Neufeld 2009), implying that joining the core developer community is not effortless. Software development is a knowledge-intensive activity that often requires high levels of domain knowledge, experience, and intensive learning by those contributing to it (Fichman and Kemerer 1997; Pliskin, Balaila and Kenigshtein 1991), and integration of newcomers into the process is arduous. Only those who continuously involved in the development process over a period of time could contribute in a meaningful way, and many might find it too effort-taking to join the core developer team of the project (Kohanski 1998). Using qualitative investigation, limited prior research has indicated that peripheral developers are more likely to be granted core status if they participate according to a "joining script" in terms of intensity and type of participation they engaged in (Von Krogh et al. 2003) and those who participate continuously in an active fashion are more likely to be granted core status (Fang and Neufeld 2009).

While the prior research has been successful in establishing the association between peripheral developers' level of participation and their likelihood of attaining a core status, they do not offer a longitudinal perspective as to whether all the peripheral developers who eventually obtained core developer status follow similar participation processes. Yet, it is noteworthy that different types of peripheral developers might follow different participation trajectories, which in turn have implications on the developer's success with core status attainment. It is important to uncover how project members join a particular project, and "the nature and emergence of social categories in such projects" (Von Hippel and Von Krogh 2003) [p.218].

Summarizing the existing literature, we identify four classes of peripheral developers who may exhibit differential participation trajectories, and term them as *domain experts, quick learners, goal drifters,* and *community lurkers*. Domain experts come into a project with sufficient relevant knowledge and skills that they can apply to generate immediate results. For instance, Fang and Neufeld (2009) identified that some developers had possessed deep software development expertise prior to joining a community. Shah (2006) reported similar findings. Thanks to the existing expertise, domain experts can effectively engage themselves into the community discussion as soon as they started participating, and are likely to keep on participating actively as their earlier contributions to the community is recognized (Fang and Neufeld 2009). Thus, domain experts may start with a high initial level of participation in the community, and remain steadily active over time.

Quick learners, on the other hand, join a community with a specific, clear objective in mind, e.g., fixing a bug or adding a feature to the existing code so that their own needs for an enhanced version would be satisfied. While they do come with general competence and skills, quick learners lack project-specific knowledge that would otherwise enable them to perform as soon as they joined the community. Thus, they stay relatively silent on the developer mailing list, at least for a short initial period, in order to familiarize themselves with the specific project context and gain project-specific knowledge (Shah 2006). Having learned about the technical details of the project, they would contribute more actively, than other contributors, to an ongoing technical discussion as a way of increasing their recognition (Von Krogh et al. 2003). Indeed, numerous research has identified that one of the major benefits from participating in OSS communities is learning and competence development (Hertel et al. 2003; Lakhani and Wolf 2005; Shah 2006). Thus, quick learners may start with a relative medium level of participation in the community, but become increasingly active over time.

Goal drifters join a community in a similar capacity as quick learners, i.e., they do not come with the ability to get started contributing immediately. They differ from quick learners in that they are interested

in participating, yet without a predefined focus. Thus, it may take them longer to familiarize with what they thought they would need to know, and participate not as intense as quick learners over time. For instance, research has found that a number of new participants solicit assignments upon their joining a community, implying that they are interested yet do not know what they should do (Von Krogh et al. 2003).

The fourth type of developers is termed as community lurkers. Lurkers never actively participate in the community discussion. They choose to remain peripheral as their goal is not for contributing to the community, but for benefiting from the intellectual discussion in the community. For instance, a group of developers have been identified as observing and learning from certain OSS projects not for improving the focal software per se, but for understanding its underlying mechanisms so as to apply the knowledge somewhere else (Ye and Kishida 2003). These peripheral developers began with very low levels of participation and remained at that level.

Given the discussion on the four types of peripheral developers, we develop hypotheses that distinguish peripheral developers in terms of their initial levels of participation as well as their growth patterns. We expect that domain experts' initial participation should be more frequent than the other classes of developers because by definition they join a community with hands-on skills that enable them to participate immediately. In contrast, we expect that community lurkers' initial participation should be the least frequent because they choose to remain peripheral for reasons not related to contributing to the community. Similarly, we expect to see differences between different classes of developers in terms of their participation growth patterns. Despite quick learners' relatively low frequency of participation at the initial stage, they are likely to increase their participation over time as they keep learning project particulars and become competent in contributing to the community. In contrast, goal drifters, who began with a similar modest level of initial participation, may increase their participation at a slower rate, because they are not specialized enough to keep involved in a specialized area. Community lurkers, on the other hand, may never increase their participation over time, but remain peripheral throughout. Given that peripheral developers' participation in the community is exclusively manifested as interactions via mailing list with other developers, including both core developers and other peripheral developers, and the impact of interactions with core developer and that with other peripheral developer is likely be different in terms of status attainment by the prospective peripheral developer. We separate hypotheses between peripheral developers and core developers for the purpose of exploration.
Thus, we hypothesize:

H1: There are significant differences in peripheral developers' initial level of interactions with core developers

H2: There are significant differences in peripheral developers' growth of interactions with core developers over time

H3: There are significant differences in peripheral developers' initial level of interactions with other peripheral developers

H4: There are significant differences in peripheral developers' growth of interactions with other peripheral developers over time

Furthermore, to the extent that these four classes of peripheral developers do exist and their initial levels and temporal growth patterns may differ from one another, we hypothesize:

H5: Distinct classes of peripheral developers can be identified based on their initial level of interaction and growth over time with the core developers

H6: Distinct classes of peripheral developers can be identified based on their initial level of interaction and growth over time with the other peripheral developers

The prior research has suggested that more active participation leads to higher possibility of advancing an OSS developer's performance rank (Roberts et al. 2006). Similarly, it stands to reason that peripheral developers who participate in mailing list more actively should attain core developer status sooner than others because their performance is generally better recognized. Since different classes of peripheral developers engage in different levels of initial participation and have different growth trajectories of ongoing participation, we expect that the time it takes for peripheral developers to attain core developer status will differ across different classes of developers, such that those with high initial interaction and high levels of ongoing interaction will gain the core developer status sooner. Thus, we hypothesize that:

H7: Peripheral developers that have high initial interactions with core developers and continue to interact at higher levels with core developers will be promoted sooner than others

H8: Peripheral developers that have high initial interactions with other developers and continue to interact at higher levels with other peripheral developers will be promoted sooner than others.

RESEARCH METHODOLOGY

To test these hypotheses, we need an empirical method that can (1) *estimate* initial levels of participation and participation trajectories for each individual developer, (2) *identify* classes of peripheral developers based on trajectories (growth patterns) of their interactions with core developers and other peripheral developers, and (3) *examine* relationships of identified classes with time taken for status attainment.

The objective of such an analysis is to capture information about inter-individual differences in intra-individual cumulative pattern of interactions (Muthén and Muthén 2000; Nesselroade 1991). Such a technique is useful when observed differences in the patterns are a result of unobserved heterogeneity in the population (Muthén and Muthén 2000; Nagin 1999). Thus, heterogeneity in the observed interaction patterns may emerge from the unobserved difference amongst the developers towards, for example, utility, convenience, ease of use and other aspects of email as a medium of interactions.

Population heterogeneity, such as gender, race, and organizational designation, is either observable or available from company records, and thus can be explicitly represented by variables in a model. When population heterogeneity is unobservable, it cannot be accounted for in the model using simple regression or structural equation modeling techniques; however, latent class analysis framework can take it into account by using latent classes in the model (Muthén and Muthén 2000; Nagin 1999). This is achieved through a use of categorical latent variable that represent the latent classes (i.e. unobserved heterogeneity). The basic idea involve in latent class analysis is that that each latent class corresponds to an unobservable subpopulation that has its own growth trajectory that is define by a set of parameter values.

Latent Class Analysis

At the simplest level latent class analysis (LCA) describes how the probabilities of a set of observed variables or latent variables measured by observed indicators vary across groups of individuals where group membership is not observed (Muthén and Muthén 2000). For example, the observed variables may be various measure of community participation: frequency of participation and duration of participation. These participation variables may define latent classes that differ on relative participation, i.e. members of a group may participate in an OSS community more frequently yet their interaction may last only for a shorter period each time, whereas members of another group may participate less frequently but their interactions may last longer each time. LCA refers to these unobserved groups of individuals as latent classes. The objective of LCA is to find the optimal number of latent classes that can describe the associations among a set of observed variables. In the analysis, classes are added (or reduced) stepwise until the model fits the data well, and probabilities of being in each class are provided. This basic LCA may be extended to include outcomes of class membership, such as status attainment. Clogg (1995) provides an excellent overview of this method.

Latent Class Growth Analysis

Latent class growth analysis (LCGA) combines techniques of LCA and latent curve modeling (LCM) (Bollen and Curran 2006). In this technique a single outcome variable or a latent variable is measured at multiple time points to define different growth trajectories for each individual. LCM helps the researcher identify the pattern of changes over time by utilizing the set of repeated observed measures to estimate "an unobserved trajectory that gave rise to the repeated measures" (Bollen and Curran 2006, p. 34). The primary interest is not in the repeated measures themselves but rather in the unobserved path of change, which is referred to as latent trajectory (MacCallum, Kim, Malarkey and Kiecolt-Glaser 1997). To that extent LCM resembles the traditional latent variable SEM approach where indicators of a latent construct are used to understand the unobserved construct. LCM models estimate random intercepts and random slopes (linear or higher order) for each individual (i.e., subject) in the sample so that trajectories over time for each individual can be constructed.

LCGA builds on LCM and LCA. LGCA represents a latent class analysis in which the latent classes correspond to differences in growth trajectories for the outcome variable. For example, in a two-class model, one class may have high intercept and moderate linear growth of the outcome variable while the other may have low intercept but quadratic growth. The object of the analysis is to estimate the different growth curve patterns, and based on these patterns, the class membership of each individual (Nagin 1999). This model is flexible and more than one growth process can be used to identify classes. In addition, the model may include antecedents and outcome variables that might be related to the class membership (Muthén and Muthén 2000).

Sampling

We sampled developers from OSS projects hosted in Source Forge (SF) (http://sourceforge.net), the largest Web-based hosting service for OSS projects, and a major data source for empirical OSS studies (Colazo and Fang 2009; Koch and Schneider 2002; Mockus, Fielding and Herbsleb 2002; Newby, Greenberg and Jones 2002). Due to the longitudinal nature of our study that focused on temporal interaction among developers, we needed to sample developers from healthy, collaborative OSS projects that have tractable activity data in both CVS repository and mailing list. To accomplish that, we followed the approach introduced by Colazo and Fang by focusing on projects hosted in SF that met three criteria: they must be collaboratively developed, ported, and had activity data publicly available in CVS and mailing list (2009). This effort results in 62 OSS projects comprised of 870 core developers. As the focus of our study was participation trajectories of peripheral developers who were eventually promoted to core developer, these 870 core developers became our sample frame, of which 206 developers were successfully identified in both the developer mailing list and CVS. The time taken for the 206 developers to achieve the core developer status (hereafter termed as promotion time) ranged from 1 week to 207 weeks.

We captured mailing-list interactions as and when they actually happened but for modeling purpose we used weekly interval in order to avoid idiosyncrasies associated with specific day of a week. For example, developers who have a full time job may interact intensely over a week-end compare to other week-days. Similarly in order to be consistent with "development trajectory" philosophy of latent growth models, we used cumulative email interactions instead of actual week to week interaction. In order to identify growth patterns in mailing-list interaction a sufficiently long period of observation should be identified. We decided to use a cut-off of seven weeks, i.e. we used only those peripheral developers whose promotion time was more than seven weeks. This condensed the final sample to 133 peripheral developers spanning over 40 projects[1].

Analysis Technique

We used Mplus (5.2 version) software for our analysis because it uses generalized structural equation modeling framework and implementation is flexible to incorporate continuous, categorical, and count

[1] We repeated our analysis for peripheral developer with promotion time greater six weeks, the results were comparable but parameters were less stable. For promotion time less than six weeks, we could only fit simple patent growth model, model with quadratic, cubic and higher order suffered from identification problem.

variables (Muthén and Muthén 2007). To test our hypotheses, we constructed the LGCA model visualized in figure 1, and performed following stepwise analysis.

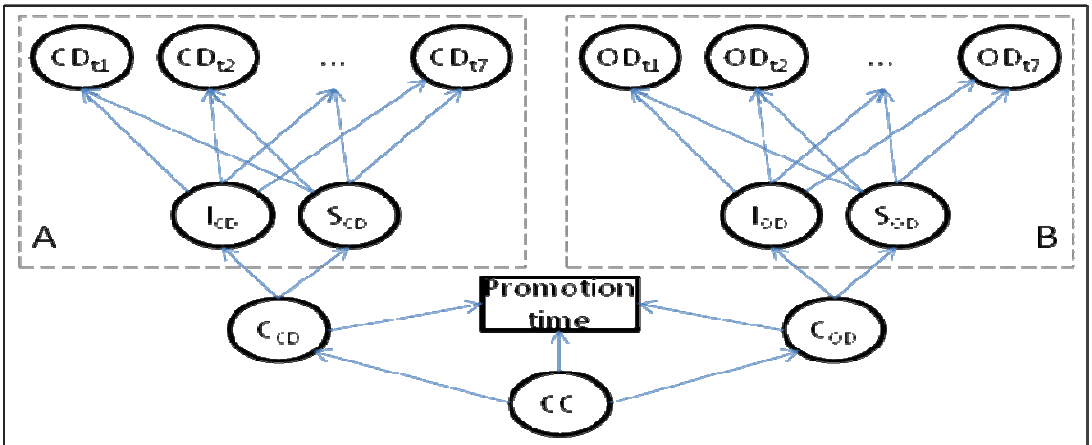

Figure 1: Research Model

Note: Block 'A' represents growth model for peripheral developers interactions with core developers, Block 'B' represents growth model for interactions with other peripheral developers. CD_{t1} ... CD_{t7} are cumulative number of emails sent to core developers at time t1... t7. CO_{t1} ... CO_{t7} are cumulative number of emails sent to other peripheral developers at t1...t7. I_{CD} and S_{CD} are respectively intercept and slope latent variables for interaction–with-core-developer process, I_{OD} and S_{OD} are respectively intercept and slope latent variables for interaction–with-other-peripheral-developer process. C_{CD} and C_{OD} are latent class variables estimated using LCGA. CC is a combination class variable calculated by combining C_{CD} and C_{OD} classes. Arrows from various latent class variables to promotion indicate promotion times would be different based on class membership

Step 1- Estimation of latent curve models (LCM): As a first step two separate models shown in block 'A' and 'B' in the Figure 1 were estimated. We tried to fit linear (LCM1), quadratic (LCM2) and cubic (LCM3) models. Table 1 provide model fit indices for these models. Based on model fit indices it can be concluded that LCM3 is best fit model for both the processes. Table 1 also provide information about variances in intercepts and slopes. There is significant variance in intercepts (i.e. initial level of interactions) for the process of interactions with core developer ($var(I_{CD})$ =21.92, $p<0.001$). Thus, hypothesis H1, which stated that there are significant differences in peripheral developers' initial level of interactions with core developers, was supported. We also found support for hypothesis H2, as all the three components of slope (i.e. linear [$var(L_{CD})$ =21.82, $p <0.001$], quadratic [$var(Q_{CD})$ =.06, $p <0.01$], and cubic [$var(C_{CD})$ =.012, $p <0.001$]) for peripheral developers' interactions with core developer have significant variations. Similarly, Table 1 indicates that there are significant variations in intercept, and linear, quadratic and cubic components of slope for peripheral developers' interactions with other peripheral developer. Thus, hypothesis H3 and H4 were supported.

Table 1: LCM Models

Models	CFI	TLI	RMSEA	Variance		
				Intercept	Lin	Quad/Cubic
Interactions with core developer						
LCM1	.562	.600	.842	26.82***	36.58**	---
LCM2	.804	.784	.619	21.57***	.30.28***	.65***/---
LCM3	**.992**	**.987**	**.12**	**21.92*** **	**21.82*** **	**.06**/ .012*** **
Interactions with other peripheral developer						
LCM1	.537	.577	.935	209.69***	46.95***	---/---
LCM2	.738	.710	.774	78.935***	121.13***	1.73***/---
LCM3	**.985**	**.977**	**.15**	**80.19*** **	**91.51*** **	**.21***/.041*** **

LCM1, LCM2, LCM3 represent linear, quadratic, and cubic LCM models. For variance: *** p<.001, ** p<.01. Row with bold numbers indicates the best fit model for that process. For example, LCM3 is the best fit model for both processes modelled.

Step 2- Estimation of classes for each growth process: Once latent curve model is estimated, the next step is to identify number of classes for each growth process. The latent class growth analysis for interactions with core developer yielded four classes (we used various information criteria provided by Mplus to decide best solution for number of classes, details of analysis is not included because of space constraint and could be obtained from the author team). These four classes are shown in Figure 2. As there are four clearly identifiable classes based on intercept and slope of growth trajectories, hypothesis H5 was supported.

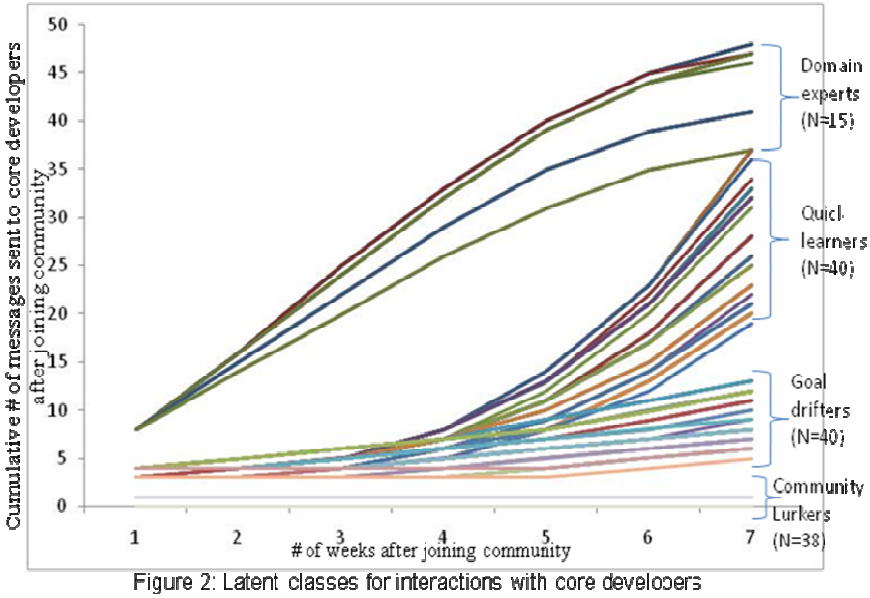

Figure 2: Latent classes for interactions with core developers

Similarly, LCGA for interactions with other peripheral developers also resulted in four classes as shown in the Figure 3[2]. Thus, hypothesis H6 was supported.

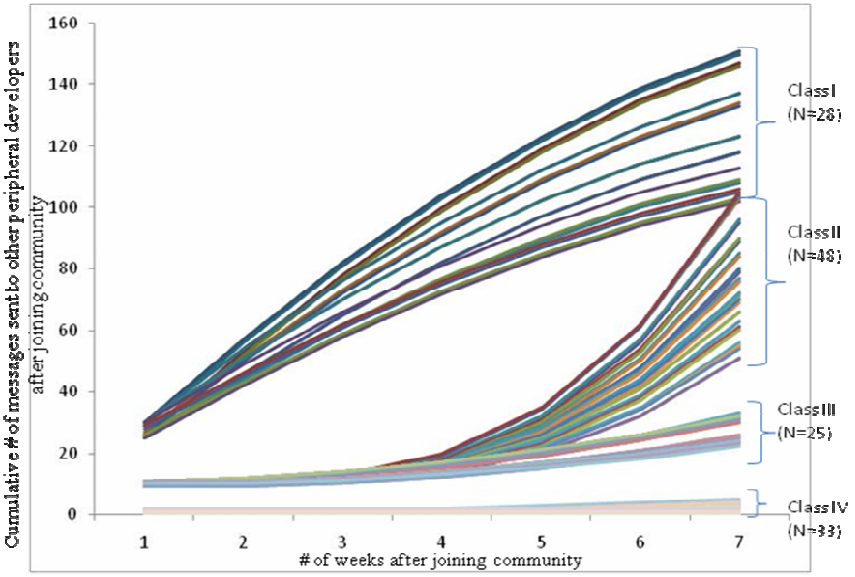

Figure 3: Latent classes for interactions with other peripheral developers

[2] These classes are based on the community building activity. However, due to space constrain, we do not discuss the mechanism behind formation of these four classes. We intend to include detail explanation of significance of these four classes in later expanded version of this paper.

Step 3-Relationship of classes with promotion time: The next step is to establish whether each of the identified classes differ on some observed outcome variable. Table 2 presents mean promotion time for each of these classes. For interaction with core developer, mean promotion time for class 'experts' is lowest (7.5 weeks) and that for lurkers is highest (83.2). Using t-test, we found that mean promotion time for each class is significantly different from all the other three classes. Thus, hypothesis H7 was supported. Similarly, Table 2 also shows that there are significant differences in mean promotion time for classes based on interactions with other peripheral developers. Thus, hypothesis H8 was supported.

Table 2: Mean promotion time (PT) in weeks for various classes

Interactions with core Developer		Interactions with other developers	
Class	Mean PT	Class	Mean PT
Domain experts	7.5***	*I*	8.7***
Quick learners	13.4***	*II*	17.9***
Goal drifters	28.7***	*III*	35.5***
Community lurkers	83.2***	*IV*	86.4***

*** indicate 'mean PT' for this class is significantly ($p<.001$) different from mean PT of all other three classes.

Discussion and Conclusion

By using latent class growth modeling approach, the present study explores the participation behaviour of peripheral OSS developers before they attain core developer status. Our empirical results add to the existing OSS literature by revealing several important findings. First, the results support our hypotheses that there are significant differences in the peripheral developers' initial levels of interactions, as well as subsequent changes in interactions, with core developers and other peripheral developers respectively. Second, our results empirically identify that there are four distinct classes of peripheral developers characterized with differential initial levels of participation and temporal participation behavior, in terms of their interactions with core developers and other peripheral developers, respectively. Specifically, the participation trajectories of the observed classes are highly consistent with the patterns of domain experts, quick learners, goal drifters, and community lurkers, as were summarized from the literature. Finally, our findings reveal that the different classes of peripheral developers are associated with different time taken for attainment of core developer status. Domain experts, featured with high initial levels of participation and stable growth of participation, attain core developer status most quickly. By contrast, community lurkers, featured with low initial levels and subsequent stagnating growth of participation, take the longest time to advance their status. The two classes of developers, featured with modest levels of initial participation and delayed temporal growth of participation, rank between domain experts and lurkers.

In addition, it is noteworthy that since same individuals were classified into four classes based on interactions with core developer and separately into four classes based on interactions with other peripheral developers, there should be a degree of overlap in these two schemes of classification. Depending on the degree of overlap, the number of combined classes could range between four and sixteen (four by four). We found in our post-hoc analysis that there were six combination classes identified in our sample (Table 3), representing a good degree of overlaps between two classification schemes, except for two additional classes (classes 2 and 4 in table 3). In class 2, developers classified as quick learner by their interaction with core developers were classified as domain expert by their interaction with other peripheral developers. In class 4, developers classified as goal drifter by their interaction with core developers were classified as quick learner by their interaction with other peripheral developers. The emergence of these two additional classes may imply that interaction with core developers might be more demanding than interaction with other peripheral developers, such that those who are qualified as domain experts for their interaction with peripheral developers may not be qualified in the interaction with core developers. Similarly, those perceived as quick learner in the peripheral developer group might not be so in their interaction with core developers.

Table 3: Combination classes sorted by mean promotion time (PT)				
Class	*Interaction with core developer*	*Interaction with other peripheral developer*	*Class size*	*Mean PT*
#1	Domain expert	Domain expert	15	7.53
#2	Quick learner	Domain expert	13	10.08
#3	Quick learner	Quick learner	27	14.93
#4	Goal drifter	Quick learner	21	21.71
#5	Goal drifter	Goal drifter	22	34.14
#6	Lurker	Lurker	35	86.4

Our study makes several important contributions to the OSS literature. First, although peripheral developers play an important role in OSS communities (Lee and Cole 2003; Von Hippel and Von Krogh 2003), little research has explicitly focused on understanding peripheral developers' participation behavior, with only very few exceptions (Fang and Neufeld 2009; Von Krogh et al. 2003). Even the few exceptions, which qualitatively identified key participation activities associated with peripheral developers' promotion, fail to offer a longitudinal view of these developers' participation patterns, leaving the dynamics of developer participation unexamined. Our study is among the first to provide a nuanced understanding that distinguishes peripheral developers' temporal participation behavior. On the one hand, by synthesizing the existing OSS literature, we theoretically identify the four classes of OSS developers with distinct temporal patterns of participation (domain experts, quick learners, goal drifters, and community lurkers), thus offering a conceptual contribution to the existing OSS literature. On the other hand, we empirically identify several classes of peripheral developers who follow different participation trajectories that are generally consistent with what were expected, thus adding empirical support to our reasoning. Although it is premature to establish that the four empirically identified classes of developers capture the four conceptual categories identified in the literature, our empirical results do evidence that all peripheral developers do not participate in a similar pattern. Instead, they differ significantly. Future research can conduct in-depth qualitative analysis of the mailing-list interactions to identify the latent variables that maximize intra-class similarity and inter-class differences.

Second, our study provides a more nuanced understanding of the relationship between developer participation and status attainment. Prior OSS research generally established that developer participation is positively associated with their status advancement (Fang and Neufeld 2009; Roberts et al. 2006), without taking a step further by understanding differences that underlie developers' participation growth patterns and the subsequent implications to the speed of status advancement. Our study provides strong evidence that peripheral developers' status attainment depends not only on how much he/she participates as studied in the prior literature, but also on the growth trajectories of their participation. Such differential performance impacts of the distinct classes of peripheral developers provide further justifications of the theoretical and empirical importance of understanding the latent classes of peripheral developers.

Third, our study offers a novel methodological contribution by introducing the latent class growth modeling approach. The information systems field has been populated with a variety of analysis techniques. They are either variance-based approaches or person-centered approaches; no existing methods combine the two approaches by simultaneously estimating temporal growth curves, clustering classes based on growth patterns, and examining the relationships of clustered classes with outcomes. Latent class growth modeling is such an analytical approach that enables us to distinguish classes based on growth trajectories. Our study demonstrates the usage of this approach by applying it to the OSS context.

There are several opportunities for future research. First, as noted earlier, although it is evidenced that several latent classes of peripheral developers exhibit conceptually expected behavioral patterns, further research is needed to verify the underlying factors clustering these classes are consistent with our theorizing. Second, although we conceptualized four latent classes of peripheral developers, empirically

six classes were identified by combining interactions with the core and the peripheral developers. Further study is required to have a deeper understanding of the two additional classes. Qualitative analysis on the interactions of these peripheral developers could add more insights.

In conclusion, the present study explores the relationships between temporal participation behavior and status attainment of OSS peripheral developers by applying the latent class growth modeling approach. We identify distinct classes of peripheral developers with differential participation behavior at the beginning and over time, and find significant differences between classes of developers in terms of the time required for attaining the core developer status. The results advance our theoretical understanding of OSS peripheral developers' participation behavior from a longitudinal perspective, as well as contribute the novel latent class growth modeling approach to the information systems field in general and the OSS research in particular.

References

Bollen, K.A., and Curran, P.J. "Latent curve models: A structural equation perspective," John Wiley and Sons, Hoboken, NJ, 2006.

Colazo, J., and Fang, Y. "The Impact of License Choice on Open Source Software Development Activities," *Journal of the American Society for Information Science and Technology* (forthcoming) 2009.

Crowston, K., Annabi, H., and Howison, J. "Defining Open Source Software Project Success.," International Conference on Information Systems, Seattle,, 2003, pp. 327-340.

Fang, Y., and Neufeld, D. "Understanding Sustained Participation in Open Source Software Projects," *Journal of Management Information Systems* (forthcoming) 2009.

Fichman, R.G., and Kemerer, C.F. "The Assimilation of Software Process Innovations: An Organizational Learning Perspective," *Management Science* (43:10) 1997, pp 1345-1363.

Fielding, R.T. "Shared Leadership in the Apache Project," *Communications of the ACM* (42:4) 1999, pp 42-43.

Fitzgerald, B. "The Transformation of Open Source Software," *MIS Quarterly* (30:3) 2006, pp 587-598.

Franke, N., and von Hippel, E. "Satisfying heterogeneous user needs via innovation toolkits: The case of Apache security software," *Research Policy* (32:7) 2003, pp 1199-1215.

Gefen, D., Straub, D., and Boudreau, M.-C. "Structural equation modeling and regression: Guidelines for research practice," *Communications of the Association for Information Systems* (4:7) 2000, pp 1-78.

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.

Jain, H., Ramamurthy, K., Ryu, H.S., and Yasai-Ardekani, M. "Success of data resource management in distributed environments: An empirical investigation," *MIS Quarterly* (22:1) 1998, pp 1-29.

Koch, S., and Schneider, G. "Effort, Cooperation and Coordination in an Open Source Software Project: GNOME," *Information Systems Journal* (12:1) 2002, pp 27-42.

Kohanski, D. *Moths in the Machine* St. Martin's Press, New York, NY., 1998.

Kreuter, F., and Muthén, B. "Analyzing Criminal Trajectory Profiles: Bridging Multilevel and Group-based Approaches Using Growth Mixture Modeling.," *Journal of Quantitative Criminology* (24:1) 2008, p 1.

Krishnamurthy, S. "Cave or Community? An empirical examination of 100 mature open source projects," University of Washington, Bothell.

Lakhani, K., and Wolf, B. "Why hackers do what they do: Understanding motivation and effort in free/open source software projects.," in: *Perspectives on Free and Open Source Software,* J. Feller, B. Fitzgerald, S. Hissam and K. Lakhani (eds.), MIT Press, Boston, MA, 2005.

Lee, G.K., and Cole, R.E. "From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development," *Organization Science* (14:6) 2003, pp 633-649.

MacCallum, R., Kim, C., Malarkey, W., and Kiecolt-Glaser, J. "Studying Multivariate Change Using Multilevel Models and Latent Curve Models," in: *Multivariate Behavioral Research*, Lawrence Earlbaum, 1997, pp. 215-253.

Malhotra, A., Gosain, S., and El Sawy, O.A. "Absorptive capacity configurations in supply chains: Gearing for partner-enabled market knowledge creation," *MIS Quarterly* (29:1) 2005, pp 145-187.
Markus, M.L., Manville, B., and Agres, C.E. "What makes a virtual organization work," *California Management Review* (Fall) 2000, p 13.
Mockus, A., Fielding, R.T., and Herbsleb, J. "Two case studies of Open Source Software development: Apache and Mozilla," *ACM Transactions on Software Engineering and Methodology* (11:3) 2002, pp 309-346.
Muthén, B., and Muthén, L. " Integrating Person-Centered and Variable-Centered Analyses: Growth Mixture Modeling With Latent Trajectory Classes," *Alcoholism: Clinical and Experimental Research* (24:6) 2000, pp 882-891.
Muthén, L.K., and Muthén, B.O. "Mplus User's Guide," Muthén & Muthén, Los Angeles, CA, 2007.
Nagin, D.S. "Analzying developmental trajectories: A semi-parametric group-based approach," *Psychological Methods* (4) 1999, pp 139-157.
Nesselroade, J.R. " Interindividual differences in intraindividual change," in: *Best methods for the analysis of change: Recent advances, unanswered questions, future directions,* J.L. Horn (ed.), American Psychological Association, Washington, D.C., 1991, pp. 92-105.
Netcraft "February 2004 Web Server Survey," Netcraft, 2004.
Newby, G.B., Greenberg, J., and Jones, P. "Open source software development and Lotka's Law: Bibliometric patterns in programming," *Journal of the American Society for Information Science and Technology* (54:2) 2002, pp 169-178.
Pinquart, M., and Schindler, I. "Changes of Life Satisfaction in the Transition to Retirement: A Latent-Class Approach," *Psychology and Aging* (22:3) 2007, p 442.
Pliskin, N., Balaila, I., and Kenigshtein, I. "The knowledge contribution of engineers to software development: a case study.," *IEEE Transactions on Engineering Management* (38:4) 1991, p 344–348.
Roberts, J.A., Hann, I.-H., and Slaughter, S.A. "Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects," *Management Science* (52:7) 2006, pp 984-999.
Schadler, T. "Open Source moves into the mainstream," Forrester Research, Inc., Cambridge, MA, pp. 1-5.
Sen, R. "A Strategic Analysis of Competition Between Open Source and Proprietary Software," *Journal of Management Information Systems* (24:1) 2007, pp 233-257.
Shah, S.K. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development," *Management Science* (52:7) 2006, pp 1000-1014.
Stallman, R.M., and Lessig, L. *Free software, free society: Selected essays of Richard M. Stallman* GNU Press, Boston, Massachusets, 2002.
Stewart, K.J., Ammeter, A.P., and Maruping, L.M. "Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activity in Open Source Software Projects," *Information Systems Research* (17:2) 2006, pp 126-144.
Von Hippel, E. "Innovation by User Communities: Learning from Open Source Software," *MIT Sloan Management Review*:Summer) 2001, pp 82-86.
Von Hippel, E., and Von Krogh, G. "Open source software and the "private-collective" innovation model: issues for organization science," *Organization Science* (14:2) 2003, pp 209-223.
Von Krogh, G., Spaeth, S., and Lakhani, K. "Community, joining and specialization in open source software innovation," *Research Policy* (32) 2003, pp 1217-1241.
von Krogh, G., and von Hippel, E. "The Promise of Research on Open Source Software," *Management Science* (52:7) 2006, pp 975-983.
Wang, M., and Bodner, T.E. "Growth Mixture Modeling: Identifying and Predicting Unobserved Subpopulations With Longitudinal Data," *Organizational Research Methods* (10:4) 2007, p 635.
Wu, J., and Witkiewitz, K. "Network Support for Drinking: An Application of Multiple Groups Growth Mixture Modeling to Examine Client-Treatment Matching," *Journal of Studies on Alcohol and Drugs* (69:1) 2008., p 5.
Ye, Y., and Kishida, K. "Toward an understanding of the motivation of Open Source Software Developers," *IEE Proceedings - Software*) 2003, pp 419-429.