8-2013

Open source software development process model – A grounded theory approach

Keng SIAU
Singapore Management University, klsiau@smu.edu.sg

Y. TIAN

Open Source Software Development Process Model -- A Grounded Theory Approach

Completed Research Paper

Keng Siau
Missouri University of Science and Technology
siauk@mst.edu

Yuhong Tian
Express Scripts Holding Company
yhtian@yahoo.com

ABSTRACT

The open source movement has provided software users with more choices, lower software acquisition cost, more flexible software customization, and possibly higher quality software product. Although the development of open source software is dynamic and it encourages innovations, the process can be chaotic. An Open Source Software Development (OSSD) process model to enhance the survivability of OSSD projects is needed. This research uses the grounded theory approach to derive a Phase-Role-Skill-Responsibility (PRSR) OSSD process model. The three OSSD process phases -- Launch Stage, Before the First Release, and Between Releases -- address the characteristics of the OSSD process as well as factors that influence the OSSD process. In the PRSR model, different roles/actors are required to have different skills and responsibilities corresponding to each of the three OSSD process phases. This qualitative research contributes to the software development literature as well as open source practice.

Keywords

Open Source Software Development (OSSD), Software Development Process Model, Grounded Theory

INTRODUCTION

With the increasing popularity of open source software such as Mozilla Firefox, Linux, Apache, and mySQL in the business world, open source software is attracting more and more attentions in the academic community as well (Aksulu and Wade, 2010). Advocates of open source software development argued that it could improve software product quality, encourage software evolution and innovation, and enhance business values (Mehra and Mookerjee, 2012; Allen, 2012; Shaikh and Cornford, 2012; Chengalur-Smith, Nevo and Demertzoglou, 2010; Amrit, 2009; Raymond, 2001; O'Reilly, 1999; Long and Siau 2005, 2006, 2007, 2008; Long, Siau, and Howell, 2007). Among others, one key advantage of open source software is its assumed high quality due partly to the intensive peer review or "many eyeballs". Members of the open source communities give direct, specific and immediate feedback to the software code written by others. All members have access to the source code and can submit code patches. This software development model is called "Bazaar" model by Raymond (2001). Traditional cathedral models (Raymond, 2001) of information system development are in short of second party review. OSSD, or parallel development through Internet (Krogh, 2003), has practically been proven to be a very successful counterstrategy to this problem.

Although this "Bazaar" model of OSSD (Open Source Software Development) naturally encourages innovation and software evolution, it also causes some problems. Lacking of conformity and unstructured project management tends to decrease the survivability of the OSSD project. In OSSD, there are no formally documented norms or customs except for the implied customs and taboos learned through experience, not to mention any comprehensive development process guidelines. Cusumano (2004) believed that many OSSD projects are "semi-organized chaos". The state of affairs of OSSD urgently calls for an increased understanding of the unique OSSD process and the development of OSSD process models to help guide OSSD project management.

The nature of OSSD projects is dynamic. Collaborations in OSSD are extraordinary, which makes a systematic software development process very imperative. The unique features of OSSD imply that many existing standardized or quasi-standardized software development processes such as CMM (Capability Maturity Model) and UP (Unified Process), cannot be readily applied to OSSD.

Although managing an OSSD project has some similar characteristics as that of traditional software development. The differences are very apparent. OSSD has its own unique characteristics. To name just a few, first, the virtual development team of an open source project can become very large and change frequently, which require appropriately breaking up the project into distinct components and documenting the specifications clearly so that programming teams can work on them without much communication. Second, the voluntary, peer-to-peer decentralized virtual team (Kwok and Gao, 2004) demands different project leadership (Pauleen, 2003) than that of the usual software development team. Third, the lifecycle of OSS is supposed to be longer because the publicly available source codes are usually continuously being updated (Lerner and Tirole, 2002). As Berglund and Priestley (2001, p134) pointed out: "The nature of open-source development still remains somewhat uncharted territory… but is typically (among other characteristics) robust, public, just-in-time, user-driven, global, community oriented, critical-mass dependent, non-directional in its growth, developed from the bottom up, and change-prone." These situations call for the study of OSSD process and the development of OSSD process model. This research aims to identify key factors and variables that affect the OSSD process, and to develop a process model to guide OSSD practice.

LITERATURE REVIEW

A software process refers to the activities performed during the development of a software product or system (Curtis et al., 1992). Two of the well-known software processes are CMM and UP. Both CMM and UP focus on the developmental details. They are complex, sophisticated, and detail oriented, which limit the possibility of them being adopted in practice (Fuggeta, 2000). Many researchers call for people and context oriented research in software process research to deal with the dynamic, uncertain nature of software development practice. There are some narrowly focused software processes but they are formulated for some specific scenarios of traditional cathedral model of software development. For example, Agile Programming deals with changing requirements, while eXtreme Programming (XP) emphasizes on quickly creating prototype that has limited function and then building on it. OSSD is different when compared to traditional "cathedral" model of software development. The previous software process propositions were not designed with OSSD in mind and may not be readily applied to OSSD.

Many of the prior research works on open source focused on the motivations of programmers as well as the organizations that adopted OSS (open source software) (Lerner and Tirole, 2002; Kogut and Metiu, 2001; Dedrick and West, 2003). There are some research works that focus on OSSD issues. Factors that were found to be related to OSSD include documentations (Berglund and Priestley, 2001), size of the community, work distribution among the community, problem reporting, code ownership, defect density, and time to resolve problem (Mockus and Herbsleb, 2002). Other researchers studied the key success factors of OSSD. Trust and ability of individuals to coordinate and safeguard interactions (Stewart and Gosain, 2004) were found to be keys to OSSD success. Crowston *et al*. (2003) showed that user, product, process, developers, use, and recognition are identified and shown to be the success factors of OSSD project. They also point out that OSSD process is not a "one-off event" as traditional software development usually is, but rather an on-going activity that involves continuously updating and releasing of new versions of the software. OSSD project features "band-wagon" effect (Madey *et al.*, 2002), i.e., "rich get richer". Good developers prefer to join a successful project. "Herding" effect of OSSD (Oh and Jeon, 2004) is used to describe the herding exit of OSSD project by its team members when the project is not promising. This is the leading susceptibility and vulnerability of an OSSD project (Oh and Jeon, 2004). Therefore, it is crucial for an OSSD project to achieve critical mass shortly after it is launched.

A definitive feature of OSSD is its virtual developmental community. Admittedly traditional software development projects can also be conducted under virtual development environment. The difference is that OSSD is solely accomplished under virtual environment and totally subjected to its norm.

The effect of virtual environment on OSSD is mixed. "Band-wagon" effect and "Herding" effect are in fact due to the virtual, opened membership communities. The communication media in the virtual environment is less effective than in the physical environment (Olson and Olson, 2000). In addition, knowledge sharing is supposed to have lower quality since members are normally anonymous (Wasko and Faraj, 2005). The voluntary team members and virtual environment are challenges for the project manager and require different leadership. Research also showed some positive effects of virtual environment. For example, the e-media equalizes the participants and favor innovative ideas, task-orientation, and rational decision-making (Yamauchi *et al.*, 2000; Sproull and Kiesler, 1986). The asynchronous communication allows more time for reflection before sending out a message, which facilitates quality knowledge sharing (Wasko and Faraj, 2005).

The previous research on OSSD can be categorized into: OSSD project success factors, virtual communities, OS membership, OS leadership, and OS product (Table 1).

Category	Research	Focus	Identified Relevant Factors/ Results
Project success factors	Berglund and Priestley, 2001	OS documentation process	User control/driven, social structure of the community, live, and face-to-face communication forum
	Mockus and Herbsleb, 2002	OS development	Size of the community; work distribution among the community; problem reporting; code ownership; defect density; time to resolve problem reported
	Crowston *et al*., 2003	OS success factors	user, product, process, developer, use, and recognition
	Madey *et al.*, 2002	OS success factor	Band-wagon" effect: Initial success of an OSSD project induces more success down the road because good developers prefer to join a successful project
	Sagers *et al.*, 2004	OS network governance	Ability of individuals to coordinate and safeguard interactions
	Jensen and Scacchi, 2004	OSSD	Issues of OSSD: collaborations, leadership and control, and conflict resolutions of intra-community and inter-community in OSSD
	Amrit, 2009	OS success factor	Studied whether coordination mechanisms in commercial software development are applicable to OOSD projects
	Chengalur-Smith, Nevo, and Demertzoglou, 2010	OS software	Examined the antecedents of business value of OS infrastructure technologies
	Mehra and Mookerjee, 2012	OS optimization	An analytical model based on optimal control theory to characterize the employment contract that features the best mix of open source participation and wage payments
Virtual Community	Stewart and Gosain, 2004	Virtual community of OS	Ideology, Trust between members
	Wasko and Faraj, 2005	Knowledge sharing on virtual community	Quality knowledge sharing in virtual community is difficult to maintain; Gaining professional reputation is the leading factor why an individual virtual community participant contributes knowledge
	Yamauchi *et al.*, 2000	Collaboration in OS community	Reduction of social context cues by electronic means equalized participants and facilitates the proposition of innovative idea. As a result, OS Virtual community tends to have rational decision-making
	Olson and Olson, 2000	Computer mediated communication	Face to face meeting is generally deemed as richer communications than computer mediated communication
OS membership	Oh and Jeon, 2004	OS membership dynamic	Herded exits by its participants may be the leading susceptibility and vulnerability of an OSSD project
	Chin and Cooke, 2004	Job satisfactory of OS virtual community members	Intrinsic and extrinsic motivational factors, project coordination, project characteristics and group trust of members in the OS community have effects on member's job satisfaction, and ultimately the OS project success
Leadership	Pauleen, 2003	OS leadership	Virtual leaders have to build some level of personal relationships with the virtual team members before initiating working relationships with them. Theoretical steps involved in building relationships with virtual team members

OSS Product	Lerner and Tirole, 2002	OS product /OS economics	Lifecycle of open source software is supposed to be longer because publicly available source codes are usually continuously being updated
	Wu and Lin, 2001	OS product and OS process framework	Information about OS product, a relatively complete framework of OSSD process framework starting with the launcher's "personal itch" and ending in the official version release
	Allen, 2012	OS product	New networks of commercial and semi-commercial players create viable business ecosystems around successful OS business products

Table 1. Previous research on OSSD

Even though previous research provided some interesting results and conceptual understanding, these studies usually addressed only a fraction of the issues or factors related to the OSSD process. There are few comprehensive research works on the factors affecting OSSD process, and there is no comprehensive OSSD model that has been developed. Among the previous research in OSSD, Wu and Lin (2001, p34) is possibly the only one that presented a relatively complete framework of OSSD process starting with the launcher's "personal itch" and ending in the official version release. However, the framework is very generic and does not address any of the specific procedural guidelines of the process especially during the very important project initiation stage. This research, using the grounded theory approach, fills this literature gap and develops a comprehensive OSSD process model.

RESEARCH METHODOLOGY AND DATA

Grounded theory is a qualitative research method to generate or discover a theory based on systematic analysis of the data. It intends to bring rigor to qualitative research yet to retain its unique characteristic of rich insights. For a research in its theory building stage, qualitative method such as grounded theory is very appropriate. The essential procedure of grounded theory is coding, which includes open coding (examining and categorizing the data), axial coding (identifying relationship between the categories), and selective coding (focused analysis of core category).

Data Source and Site Selection

The basic communication tool of OSSD project is Internet mailing list. The archives of the communications endow us with substantial data for a qualitative grounded theory research. For this research, we need a project that 1) has substantial data (archives) that cover most, if not all, of the project life cycle; 2) is not closely related to a profit-seeking company (to be representative of a typical OSSD project; and 3) is not a project that lasts longer than ten years, so that the data is manageable for this research. Based on the above criteria, we identified QuantLib as the best candidate. This project is a free/open-source library for quantitative finance. We used five years of messages in the project's development archive – a total of 820 messages.

Using a variety of data sources in grounded theory research is suggested, i.e., data should be collected from multiple sites and by different methods such as interview, observation and documentations (Dey, 1999). Therefore, in addition to QuantLib project, we collected data from interview archives of OSSD practitioners from different OSSD projects on OSSD community site, Slashdot.org. 29 interview archives were selected based on theoretical relevance and purpose of this research (Orlikowski, 1993). Each interview is about 1-2 pages. All the interviewees in the selected interviews were OSS project leaders or developers. Several email interviews with the QuantLib team were conducted by one of the researchers.

In addition to cost effectiveness and convenience, there are a number of advantages of using secondary/documentary-based data (Szabo and Strang, 1997). It avoids researcher's bias stems from the process of data collection. More efforts can be placed on the process of analysis and interpretation of findings. Successful examples of secondary/documentary-based grounded theory analysis include Lock (1997) and Turner (1983). For a research using second-hand data, the original data set needs to be large enough to allow for the process of constant comparisons and theoretical sampling (Goulding, 2002). Our research meets this requirement because we had sufficient data available.

Data Coding

The open coding step involves carefully inspecting the text of the data files to formulate categories. Normally, conceptual categories are created by grouping several categories. Constant comparisons are used each time a category is identified. The newly identified category is compared to previously created categories and conceptual categories to see if it can be merged with one of them or renamed. When theoretical saturation point is reached (i.e., no new categories emerge from the data), the data analysis moves on to the second step -- axial coding. Axial coding actually is inseparable from open coding in that the recording of theoretical memoranda in open coding stage is in fact part of the axial coding. In axial coding, the researcher further inspects the data in terms of the conditions, context, action/interaction strategies, and consequence of the strategies for each category (Strauss and Corbin, 1990) and comes up with ideas to further refining and re-organizing of the existing categories in regard to their relationships. Finally, during selective coding stage, the core category is identified and analyzed, and a theoretical model of OSSD process is developed.

RESULTS AND DISCUSSIONS

open coding – conceptual categories

Initially, the open coding step results in nine high level conceptual categories and 60 properties. Further investigation into the data source of these categories ended up with five conceptual categories.

Virtual Environment

A number of incidents in the data can be attributed to the virtual environment of OSSD. Virtual communication may sometime involve typos, delayed responses, redundant posts, or simply forget to attach files. On the other hand, it allows formal and comprehensive summary of suggestions and bugs that can be readily delivered to every team members and users. In addition, knowledge sharing mechanisms such as hyperlink can be used to share information to facilitate their work.

In the virtual environment, people do not have the same opportunity to develop bond as they do in physical setting. Therefore, team members have to put effort in building rapport with each other especially in the project initiation stage. OSSD depends almost entirely on its website to disseminate relevant information. For the same reason, OSSD requires more codified process published on its website. Frequent announcements of project progress and releases not only can prevent job duplication but also can motivate members and boost morale.

Finally, the importance of leadership in the OSSD virtual environment can never be over emphasized. Leaders need to build rapport with existing members and potential new members, and to show their passion and commitment especially at the beginning of a project when few people are involved. Different from most of the physical software development project leaders, the OSSD leader usually is also the project initiator, main designer, and main developer. Therefore he or she has to do more of the technical job in addition to administration. Another requirement for the project leader is that s/he must accept the democratic decision making process, which is the norm in virtual communities.

Voluntary Members

Unlike the paid employee in commercial software project, OSSD members are mostly volunteers. Therefore, OSSD leader usually is unable to recruit members competitively as he or she wishes. The only way an OSSD project can attract competent developers is to build up its critical mass and become visible and successful (Madey *et al*., 2002). For the same reason, job usually is assigned by invitation rather than by mandatory order. Pushing for job completion is usually indirect, implied, and with no or flexible deadlines set by the leader. Therefore, the leader can rarely be assured of on-time completion of jobs unless s/he is doing them by himself/herself. Consequently, both the product release date and the release version are usually very flexible. The relationship between members and the project is pretty loose, which leads to relatively high turnover rate. It is not unusual to see 50% or even higher turnover rate in an OSSD project. As a result, documentation and consistent coding style are stressed emphatically. Since most, if not all, of the members have a regular paid job, issues such as working time and software ownership may arise. In addition, the voluntary members' or users' inputs are usually sought by the project leader in the democratic OSSD decision-making process.

Product

The OSSD process must consider the features of its ultimate product. Since the source codes of the product are open, peer-to-peer review of the program is made possible. Summary of bugs or suggestions for function extensions posted by either the developers or users help improve the quality of the OSSD product. The prevailing norm of "release early, release more" in OSSD certainly exerts an impact on its process. OSSD product features continuous updates and releases, and good version

control can never be ignored even after the first version is released. Other issues such as OS software license and software ownership must also be taken care of at various OSSD process stages.

Users

OSSD project seldom has a physical user body as in traditional software project, especially during the early stage of the project. As Wu and Lin (2001) pointed out, OSSD project usually starts from the initiator's "personal itches". It is usually not until the project goes through its critical-mass building stage and starts to attract people to its mailing list discussion that a real sense of user group comes into being. Therefore, soliciting user requirements for OSSD is very different from traditional software development in that it is conducted more or less on an on-going basis rather than by aggregating in the project initiate stage.

OSSD users, empowered by the Internet resources, are generally more knowledgeable and most are capable of coding and making extensions to the OSSD products. In OSSD project, users usually involve in the software development process more deeply than they would do in traditional software development process.

OSSD Process

The above four categories – virtual environment, voluntary members, product, and users – are factors that affect the OSSD process. These factors differentiate OSSD from traditional software development. In conclusion, the challenge of managing a software project in a pure virtual environment and under the control of its democratic norm, the voluntary, open-membership team members, the knowledgeable, deeply involved user and unique user-developer interactions, and the open-source, continuous released product consist of the characteristics of OSSD. Hence, the OSSD process should account for all of these identified categories.

Axial Coding -- Relationships between Categories

Axial coding identified the relationships among the five categories shown in figure 1. As discussed earlier, Virtual Environment, Voluntary Members, Product and Users all affect the OSSD process. The Virtual Environment and Voluntary Members have both direct impacts and indirect impacts that are mediated through products and users as shown in figure 1.

Selective Coding - Core Categories and a PRSR OSSD Process Model

Selective coding further refined the "core" category (Glaser and Strauss, 1967), namely the OSSD Process as shown in Table 2. Further refinery on this core category is needed in grounded theory's selective coding stage (Glaser and Strauss, 1967). All the other four non-core categories have to be integrated around this core category (Figure 2).

Considering the four non-core categories – virtual community, voluntary members, product, and users created from grounded theory open coding and axial coding analysis, we developed a three stage OSSD process model, i.e., PRSR (Phase, Role, Skills, and Responsibilities) model during selective coding as shown in Table 2. Based on the in-depth grounded theory analysis above, we believe it is more appropriate to break OSSD process into three phases: Launch Stage, Before the First Release, and Between Releases. This is drastically different from traditional software lifecycle, but it appropriately addresses the four identified non-core categories from the grounded theory research and data. Due to the virtual environment of OSSD, there is a distinct OSSD project launch phase when the virtual community comes into being and members start interacting, and the project starts to accumulate members. After the virtual community is formed and stabilized, the next goal is to deliver the first release as soon as possible. This phase, which we called "Before the First Release" is the stage for the structuration of the virtual community. During this phase, members start to be involved in OSSD activities. Then the project enters the third phase, which is generally the longest, almost endless phase in some projects. This phase is relatively more stable than the previous two. The objective is to maintain the previous releases and to add extensions to the product, i.e., seeking for more and frequent releases. As most OSSD projects undergo a very long period of constant releases, we use "Between Releases" to name this phase. The key activities of each phase are listed in Table 2.

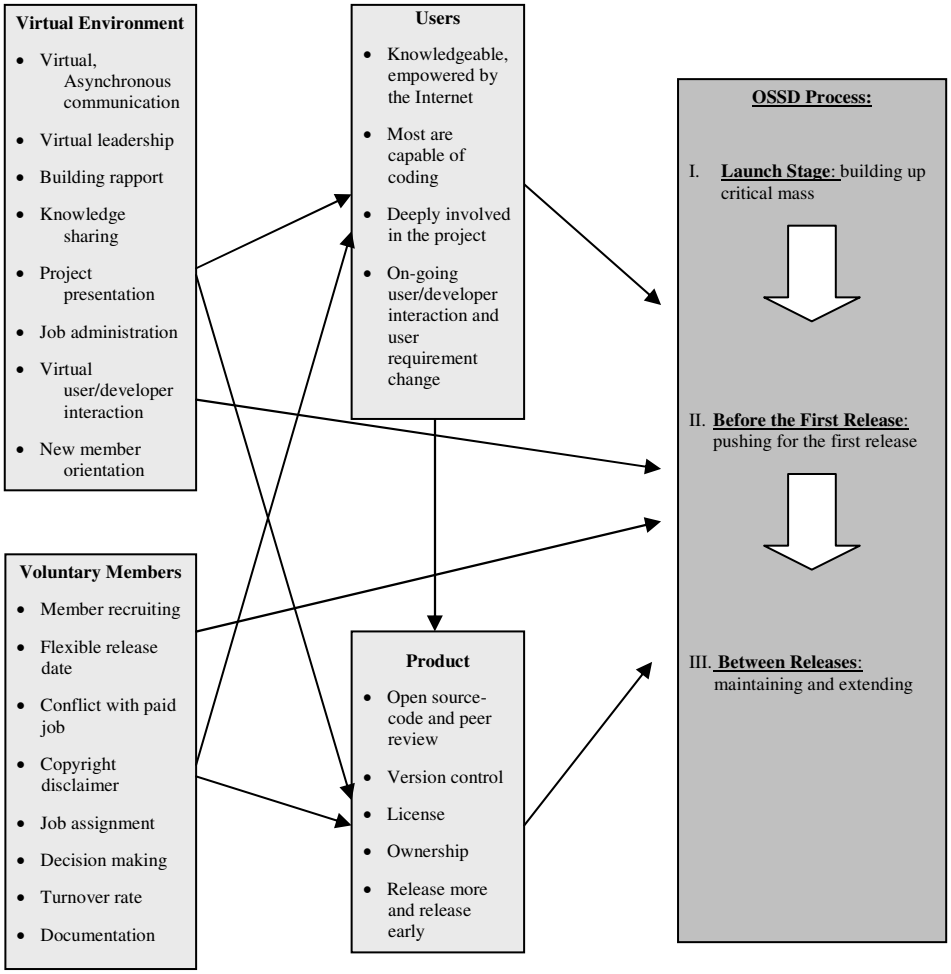

Figure 1. Relationships between categories

Phase	Roles	Skills	Responsibilities
Phase I: Launch Stage	**Leader**	• Virtual leadership/social skill • Broad knowledge of related/similar products • Software design/development skill • License/Legal skill and knowledge	• Project website/mailing list creation • Prepare/post project description, platforms • License issue • Encourage/invite discussion, taking care of virtual communication issues • Invite volunteers • Welcome new members • Build rapport • Project promotion .
	Team member	• Virtual social skill • Software design/development skill	• Get software ownership disclaimer • Subscribe and contribute to developer and user mailing lists discussions • Design/code • Documentation • Building rapport with other members

	User	• Basic knowledge of future product	• Subscribe and contribute to user mailing list discussion • Provide user requirements • Prepare bug report
Phase II: Before the First Release	**Leader**	• Virtual Leadership/social skill • Software development skill • Job administration skill	• Roughly set a date for release • Push for job done • Direct mailing list discussions • Make decisions when there is no consensus • Job administration-avoid conflict and redundant job • Ensure software quality and good documentation • Team administration –new member "orientation" • Get to know members and develop friendship with them
	Team member	• Virtual social skill • Software development skill	• Code • Documentation • Peer review/bug report • Knowledge sharing • Cooperative, get job done on time • Develop virtual friendship with other members
	User	• Software testing skill	• Test the product before release • Prepare bug report • Continuously provide user requirements
Phase III: Between Releases	**Leader**	• Software design/development skill • Software version control skill • Job administration skill • Broad knowledge of related/similar products	• Look for ways to extend project • Make the decision when there is no consensus • Maintain/document previous product versions
	Team member	• Software design/development skill • Software version control skill • Broad knowledge of related/similar products	• Code • Documentation • Peer review/bug report • Knowledge sharing • Cooperative, get job done on time • Look for and discuss ways of enhancing the software • Maintain/document previous product versions
	User	• Software testing skill • Broad knowledge of related/similar products	• Test product releases • Bug report • Propose new user requirement • Provide suggestions for new extensions

Table 2. A proposed OSSD process model -- PRSR model

The PRSR model we proposed first divides the OSSD process into three phases. Then we recognized that there are three actors in OSSD project: leader, team members, and users. The roles of the actors can become overlapped, i.e., a leader can be a team member and user of the software product. During the three OSSD phases, each role is required to have different skills and different responsibilities.

The PRSR OSSD process model differs from traditional software development process in that it is tailored for OSSD. The proposed three OSSD process phases reflect the typical stages of an open source project. The roles of software leader, developer, and user are usually overlapped in many OSSD projects. Therefore, differentiation of these roles is important. The

skills and responsibilities of each role are created by integrating the four non-core categories, i.e., virtual environment, voluntary members, product, and users (see Table 2). The resulting OSSD process model is representative and illustrative with the definition of the roles and skills, and the responsibilities of each role during the three OSSD phases.

CONTRIBUTIONS, LIMITATIONS, AND FUTURE RESEARCH

Through an in-depth grounded theory analysis of an OSSD project mailing list archives and some OSSD practitioners' online interviews, this research identified the factors affecting the OSSD process and proposed a Phase-Role-Skill-Responsibility (PRSR) OSSD process model.

Previous research have pointed out that OSSD projects are organized loosely (Cusumano, 2004), subjected to constant change (Berglund and Priestley, 2001), and on-going (Crowston *et al*., 2003). Our grounded theory analysis shows that virtual community, voluntary members, and unique features of product and users of OSSD all have an effect on OSSD. A process model for OSSD must consider these dynamic and "bazaar" features of OSSD. The PRSR model, rather than providing detailed, rigid software development steps, focuses on defining the roles, skills, and responsibilities along the three OSSD phases. We believe that if everybody involved in the project knows what his/her role is, what skills he/she needs to possess, and what responsibilities he/she needs to fulfill over the three distinct project phases, then there will be less "semi-organized chaos" (Cusumano, 2004). The PRSR OOSD process model helps to mitigate and alleviate the project chaos and at the same time supports the dynamic nature of OSSD. The PRSR OSSD process model provides a useful guideline for OSSD project management practice and helps to increase the sustainability of open source projects. This research also contributes insights and knowledge that can be used for future theory development on software development process – particularly OSSD.

Grounded theory approach is an endeavor to make a qualitative research rigor and yet retaining the feature of enriched insights. As a theory building approach, grounded theory analysis normally ends up with preliminary model. Also, the process of grounded theory analysis is still subjected to the limitation of researcher's personal judgment. Therefore, in the next stage of the research, in addition to incorporating more OSSD projects to our data sets, we plan to use Delphi method to solicit OSSD practitioners' opinions on the PRSR model. The results will be useful in validating and refining the model.

REFERENCES

1. Aksulu, A. and Wade, M.R. (2010) A Comprehensive Review and Synthesis of Open Source Research, *Journal of the Association for Information Systems*, 11:11, Article 6.
2. Allen, J.P. (2012) Democratizing Business Software: Small Business Ecosystems for Open Source Applications. *Communications of the Association for Information Systems*, Vol. 30, Article 28.
3. Amrit, C. (2009) Coordination in Open Source versus Commercial Software Development, *AMCIS 2009 Proceedings,* Paper 222.
4. Berglund E. and Priestley M. (2001) Open-Source Documentation: in search of user-driven, just-in-time writing, In *proceedings of SIGDOC 2001*, October 21-24, in Santa Fe, NM
5. Boudreau, MC. (2002). Using grounded theory in IS research, Paper presented at the AoM/LAoM Annual Conference, July.
6. Chengalur-Smith, Indushobha; Nevo, Saggi; and Demertzoglou, Pindaro (2010) An Empirical Analysis of the Business Value of Open Source Infrastructure Technologies, *Journal of the Association for Information Systems,*11:11, Article 3.
7. Chin, P. and Cooke, D. (2004) Satisfaction and coordination in virtual communities, in *Proceedings of the Tenth Americas Conference on Information Systems*, 2699-2708
8. Crowston, K., Annabi, H., Howison, J. (2003) Defining open source software project success. *Proceedings of the Twenty-fourth International Conference on Information Systems*, 327-340
9. Curtis, B; Kellner, M; and Over, J. (1992) Process modeling, *Communications of the ACM*, 35, 9, 75-90
10. Cusumano, M. (2004) Reflections on Free and Open Software, *Communications of the ACM*, 47, 10, 25-28
11. Dey, Ian (1999) *grounding grounded theory-guidelines for qualitative inquiry*, London, Academic press
12. Fuggetta, Alfonso (2000) Software process: a roadmap, *Proceedings of the Conference on The Future of Software Engineering*, Limerick, Ireland, 25 - 34

13. Glaser, B.G. and Strauss, A.L. (1967) *The Discovery of Grounded Theory: Strategies for Qualitative Research*, Aldine Publishing Company, New York, NY.
14. Goulding, C. (2002) *Gounded theory: A practical Guide for management, Business and Market Researchers*, London: SAGE publications.
15. Jensen, C. and Scacchi, W. (2004) Collaboration, leadership, control, and conflict negotiation in the netbeans.org community. In *Proceedings of the 4th Workshop on Open Source Software Engineerin*g, 48-52
16. Krogh, G. (2003) Open-source software development. *MIT Sloan Management Review* Spring,14-18
17. Kogut, B. and Metiu, A., (2001) Open-source software development and distributed innovation, *Oxford Review of Economic Policy,*17, 2, summer, 248-264.
18. Kwok, J.S. and Gao, S. (2004) Knowledge sharing community in P2P network: a study of motivational perspective, *Journal of Knowledge Management*, 8, 1, 94-102
19. Lerner, J. and Tirole, J. (2002) Some Simple Economics of Open Source, *Journal of Industrial Economics,* 52, 2, June, 197-234.
20. Lock, K. (1997). Constructing opportunities for contribution: structuring intertextual coherence and "problematizing" in organizational studies, *Academy of Management Journal*, 40, 5, 1023-1062.
21. Long, Y., Siau, K. (2005) Impact on Open Source Software Performance: A View from Social Structure, *Proceedings of Americas Conference on Information Systems (AMCIS 2005)*, Omaha, NE, USA, August 11-14, , 3049-3053.
22. Long, Y., Siau, K., (2006) Social Network Dynamics for Open Source Software Projects, *Proceedings of Americas Conference on Information Systems (AMCIS 2006)*, Acapulco, Mexico, August 4-6, 820-830.
23. Long, Y., Siau, K., Howell, K. (2007), Measuring Knowledge Sharing in Open Source Software Development Teams, *Proceedings of Americas Conference on Information Systems (AMCIS 2007)*, Colorado, USA, August 9-12, 1-12.
24. Long, Y. and Siau, K. (2007) Social Network Structures in Open Source Software Development Teams, *Journal of Database Management*, 18, 2, 25-40.
25. Long, Y., Siau, K. (2008) Impacts of Social Network Structure on Knowledge Sharing in Open Source Software Development Teams, *Proceedings of Americas Conference on Information Systems (AMCIS 2008)*, Toronto, Canada, USA, August 14-17.
26. Madey, G., Freeh, V., Tynan, R. (2002) The open source software development phenomenon: an analysis based on social network theory. *Proceedings of Eighth Americas Conference in Information System*, 1806-1813.
27. Mehra, A. and Mookerjee, V.M. (2012) Human Capital Development for Programmers Using Open Source Software, *MIS Quarterly*, 36, 1, 107-122.
28. Mockus, A. and Herbsleb, J. (2002) Two Case Studies of Open Source Software Development: Apache and Mozilla, *ACM Transactions on Software Engineering and Methodology*, 11, 3, 309-346
29. Oh, W. and Jeon, S. (2004) Membership dynamics and network stability in the open source community: the Ising perspective. The P*roceedings of 25th International Conference on Information Systems,* 413-425
30. Olson, G. and Olson, J. (2000) Distance matters, *Human-Computer Interaction*, 15, 139-178
31. O'Reilly, T. (1999) Lessons from open source software development, *Communications of the ACM* 42,4, 33-37
32. Orlikowski, W.J. (1993) Case tools as organizational change: Investigating incremental and radical changes in system development, *MIS Quarterly*, 17, 3, 309-341.
33. Pauleen, D.J. (2003). An inductively derived model of leader-initiated relationship building with virtual team members, *Journal of Management Information Systems*, 20, 3, 227-256.
34. Raymond, E. (2001) *The cathedral and the bazaar*, O' Reilly, Sebastopol, CA
35. Sagers, G.W.; Wasko, M.M., and Dickey, M. H. (2004) Coordination efforts in Virtual Communities: Examining network governance in open source, *Proceedings of the Tenth Americas Conference on Information System*s, New York, 2695-2698.
36. Scott, J.E. (2000) Facilitating interorganizational learning with information technology, *Journal of Management Information Systems*, 17, 2, 81-113.
37. Shaikh, M. and Cornford, T. (2012) Strategic drivers of open source software adoption in the public sector: Challenges and Opportunities, *ECIS 2012 Proceedings,* Paper 237.
38. Sproull, L.S. and Kiesler, S. (1986) Reducing social context cues: electronic mail in organizational communication, *Management Science* 32, 11, 1492-1512

39. Stewart, K and Gosain, S. (2004) An exploratory study of ideology and trust in open source development groups, Twenty Second International Conference on Information Systems, 507-512.
40. Szabo, V. and Strang V.R. (1997) Secondary analysis of qualitative data, *Advances in Nursing Science*, 20, 2, 66-74
41. Turner, B.A. (1983) The use of grounded theory for the qualitative analysis of organizational behavior, *Journal of Management Studies*, 20, 3, 333-348.
42. Vreede, G.D., Jones, N., and Mgaya R.J. (1999) Exploring the Application and Acceptance of Group Support Systems In Africa. *Journal of Management Information Systems*. 15, 3, 197-234.
43. Wasko, M.M and Faraj, S. (2005) Why should I share: Examine social capital and knowledge contribution in electronic networks of practice. *MIS Quarterly*, 29, 1, 35-58.
44. Wu, M. and Lin Y. (2001) Open source software development: an overview, IEEE Computing Practices, June
45. Yamauchi, Y., Yokozawa, M., Shinohara, T. and Ishida, Toru. (2000) Collaboration with lean media: how open-source software succeeds, *Proceedings of the 2000 ACM conference on computer supported cooperative work,* 329-338