Open Source Software Development: The New Training Ground?

Dr. Hala Annabi -University of Washington, -hpannabi@u.washington.edu
Dr. Sean T. McGann, Ohio University, mcgann@ohio.edu

ABSTRACT

Individuals interested in software development are joining Open Source Software (OSS) projects to learn how to develop software. OSS projects provide an experiential learning opportunity as learning occurs through the actual building of code. Also, OSS project are often globally distributed including members from a variety of countries. This makes OSS an especially suitable setting to prepare students for participating in global software development projects. This paper reports on findings from a study analyzing learning processes in OSS project and how these processes may serve as a training ground for Information Systems students.

INTRODUCTION

OSS is a broad term used to describe software that is developed and released under a form of "open source" license. There are many licenses with a range of different features, all of which allow inspection of the software's source code. There are thousands of OSS projects that span a range of applications; the Linux operating system and the Apache Web Server are probably the most well-known. Many OSS groups have been highly successful in meeting the challenges of developing large and complex software system.

What is perhaps most interesting about OSS for IS educators is that OSS projects are recognized as a medium through which individuals and groups may "acquire and develop new skills and experiences" in a global team [1]. In the current digital environment and due to the open nature of OSS projects, individuals of various types and levels of skills and interests may collaborate to produce software. This collaboration provides an informal learning ground where intentional and unintentional learning occurs. Members of the OSS community, particularly non-experts observe and interact with experts and other non-experts engaged in software development projects. As well, individuals are exposed to developers from various countries and geographic regions. These interactions assist individuals in further developing their skills and understanding of software development while working in a global context. Specifically, OSS projects have been helpful to teenager, young college students and professionals. Tuomi [2] noted that "[OSS] projects have also created large pools of highly skilled software professionals, often through a somewhat miraculous alchemy that has transformed teenagers into ***globally leading system architects***, sometimes with little supporting formal computer education".

Reports suggest that a significant percentage of OSS participants are under the age of 22 in support of Tuomi's observations. A survey conducted by University of Maastricht (International Institute of Infonomics) and Berlecon Research GmbH in 2002 (included 2784 responses from individuals engaged in OSS) indicated that respondents *started* working with open source at the age of 22.9 (median 22.00): "7% started below an age of 16 years, one third was between 16 and 20 years old, another third between 21 and 25, and a quarter was more than 26 years old when starting OS/FS development" [3]. The report also indicated that 83% of all respondents were IT professionals, and students represented the second largest group at 16%. In a different survey, Robles et al [4] report among their sample: 37.2% of respondents are less than 'college graduate', and 54.6% are less than 'university graduate'. Thus, it is not surprising the University of Maastricht's report indicated that the initial motivation for participating in OSS projects stems from a desire to learn. The report "…found an initial motivation for participation in the OS/FS community that rather aims at individual skills and the exchange of information and knowledge with other developers, but over time a maturing of the whole community with regard to both, commercial (material) and political aspects. To learn and to share knowledge have also been the most important issues of OS/FS developers' expectations from other developers."

These findings raise the question of whether OSS projects could become the new training/learning ground for software development's young professionals and students. These ideas have significant implications to IS education. Before reaching any conclusions regarding OSS as a learning environment we must first understand the nature of learning in this environment and how we can best utilize it in our formal educational setting. Two specific questions to start off with are:

1. What is the nature of learning process in OSS?

2. What are the benefits and limitation of the learning process from an educational perspective?

This paper explores the first question by drawing on a study of the learning process in OSS [5]. Findings from the first question will guide our discussion of the 2nd and 3rth questions.

STUDY BACKGROUND

The study is an in-depth investigation of the Apache Web Server, a successful OSS project during the first year of development. The study was guided by an interdisciplinary theoretical framework to explain the process through which groups and individuals of OSS projects learn. Drawing on multiple areas of study, the framework uses an input-process-output structure that integrates four research strands: organizational learning, shared mental models, group research, and asynchronous learning networks. The framework includes group structure, organizational level, and group design inputs (see figure 1). These inputs affect the nature of learning opportunity episodes (LOE) (triggers, process and outcomes) in the group which include the group learning process. The learning process results in group and individual learning. The framework indicates that outcomes of learning recursively affect group structure inputs. Details of the framework and study are beyond the scope of this paper due to space limitations. Please refer to Annabi [5, 6] for more on the study.

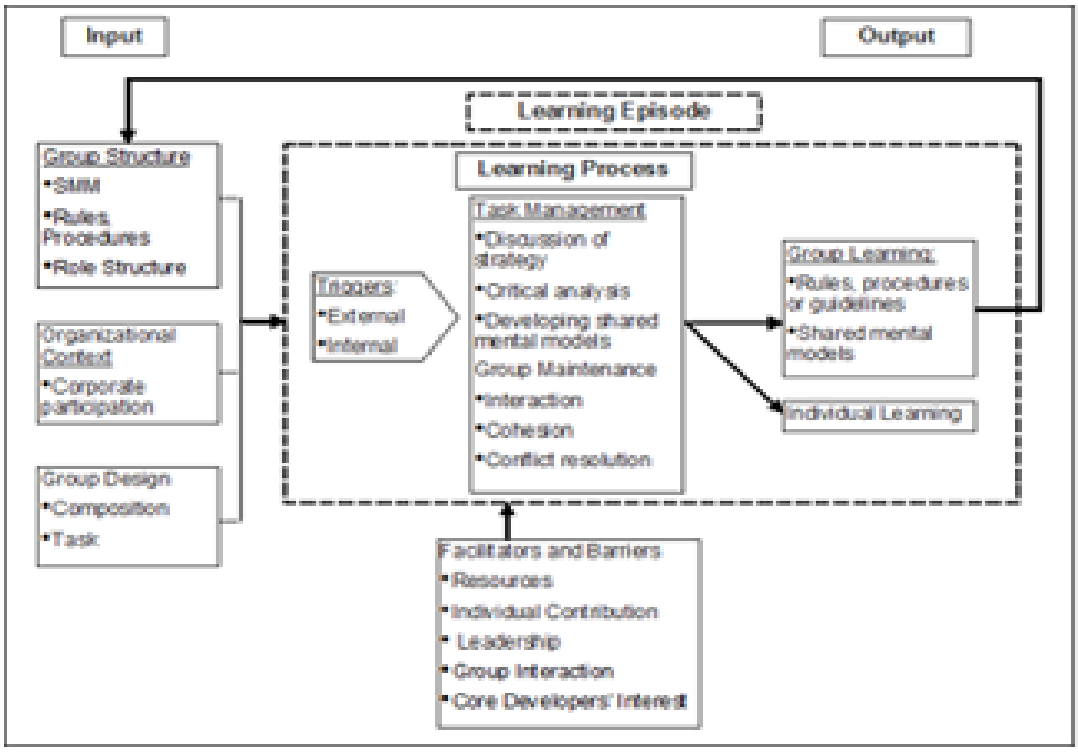

Figure 1 The Learning Process in Open Source Software Groups [5]

METHODS

This study employed a qualitative case study design to better understand the phenomenon of learning. More specifically, we employed a single embedded case study design, based on a theoretical sample strategy for case selection [7]. For details of the method and rational to use this design please refer to Annabi [5, 6]. The embedded unit of analysis is the learning opportunity episode (LOE) which is a group event that occurs over time as a result of a learning trigger [5, 6]. The study developed and used three content analytic frameworks to analyze OSS group interaction from the group's mailing list to explore the learning process in these groups. The content analytic frameworks revealed learning behaviors and practices and factors that impede or enhance learning in OSS groups.

RESULTS AND DISCUSSION

The Apache group had no formal role structure, procedures, or guidelines to guide group

membership, rules for task management, coding style and structure, system requirements or work plans at the start of the project. Individuals interested in the project joined a mailing list (new-httpd@hyperreal.com) where members contributed ideas, code, bug report and bug fixes based on needs and interests. During the period of observation, 6,649 messages were posted to the mailing list, and the group produced 38 versions of Apache as a result of 236 of patches, bug fixes, bug reports, and documentation. Messages posted to the mailing list and code submissions came from 8 core developers (Apache experts) and 46 active (co-developers) and occasional (active users) contributors (included both expert software developers and novices). The study identified 178 LOE. In the following section we will report on the characteristics of group learning process observed in the 178 LOEs analyzed in the study.

Q1 - What is the nature of the learning process in OSS groups?

Learning in the Apache Web Server project was a complex and latent phenomenon. Learning occurred within a social process focused on either developing the product (e.g. writing code and documentation) or developing processes by which to develop the code (e.g. coordinating efforts). Of the 178 episodes collected, 28% focused exclusively on process, 44% focused exclusively on product, and 28% focused on both process and product. Eighty-six percent of learning group activities involved developing shared mental models of the code, resulting in various code releases. This is where most learning occurred as experts explained the structure of the code and the various modules to each other and novices joining the group. Experts often explained the rational for the structure of the code and various coding practices they adopt. As well, experts and novices often assisted each other to further develop their ideas and address problems by pooling their ideas and skills. They sometimes did so by presenting lessons learned from other projects and contexts. In addition to developing the code, the group developed some rules and guidelines for coordinating individual efforts of developers (e.g. voting procedures, numbering scheme) to produce quality product. Lastly, different individuals exercised their abilities to manage various stakeholders in the project. Specific characteristics of interest to IS educators are elaborated on in the next two sections.

Experiential learning

Learning activities were embedded in getting the job done. There were no formal learning activities (e.g classes or seminars). As suggested earlier, our analysis discovered that learning opportunities had a focus on either developing the group ***product*** (e.g. writing code and documentation), developing ***processes*** for producing the product (e.g. contribution guidelines, voting procedures), or developing ***both*** product and process. Lessons learned regarding the product focused on coding style, the way function and modules should work and operate, assessing quality, and overall systems design. Knowledge was shared through writing and examining code. As developers write the code they learn individually. Other developers also learn by examining the code written by other individuals. Discussions about the code provided a reflective space for both the writer and the examiners of the code. This does not necessarily occur for every patch. These discussions tended to happen for complex and critical modules the most. All involved and interested in developing the code had the opportunity to question errors or gaps in their understanding; they also had a chance to share their knowledge and understanding.

In addition to learning about writing the code, individuals learned about the process through which code was developed. Apache project members, at various stages of development, examined the processes by which they coordinated their efforts, planned for release and marketing of releases, invited members to participate, and maintained a productive social environment. Things that often came into play were issues of varying cultural context (e.g. terminology and metaphors used, social norms, work norms), varying individual expertise and interests, managing time zones, and differing holidays. At times, members had to manage conflict as well.

Group interaction is necessary for learning

Figure 2 suggests that the distribution of LOE is correlated with the distribution of level of interaction over time. This further suggests that for learning to occur in the group, the group has to interact. Periods marked by limited group interaction (operationalized by number of messages) are also associated with periods of fewer learning opportunities. In analyzing the content of group interaction we found that there are a number of behaviors necessary to facilitate learning in the group. In addition to writing code and documentation, individuals learned and "taught". Learning and teaching behaviors fell into three categories; critical analysis, discussion of strategy, and developing shared mental models (for a detailed list of these behaviors please refer to Annabi [5]). These behaviors focused on sharing the knowledge individuals hold with the group as they relate to the code. As well, these behaviors focused on explaining the concepts to others in the context of developing

the code and coordinating efforts. Experts in particular modeled how to appropriately explain the code structure and programming practices using behaviors that education literature highlights as effective teaching and knowledge sharing behaviors. More specifically, experts confirmed understanding by asking and answering questions, sharing war stories, presenting content from external knowledge sources, and identifying and clarifying misconceptions. As well, experts models leadership behavior by attending to social needs of a global group membership by addressing conflict, injecting affective behaviors to group interactions, and appreciating and supporting team members contributions.

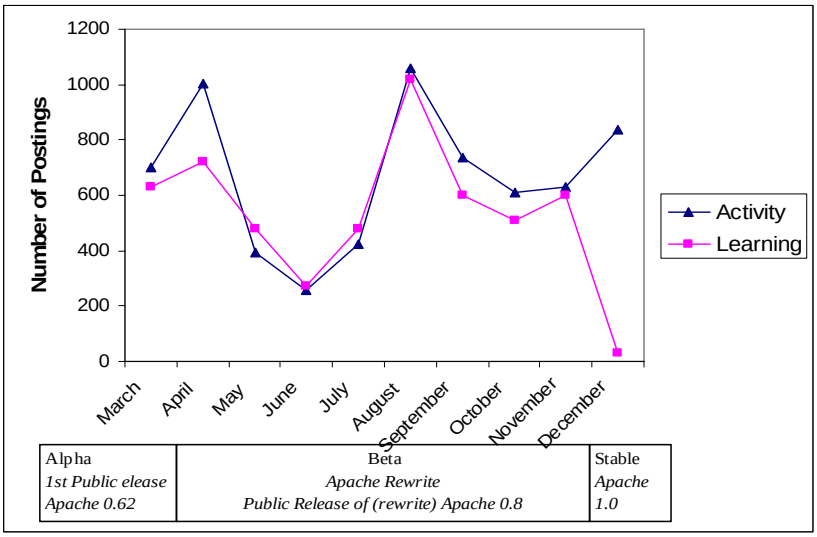

Figure 2 Distribution of Learning Opportunity Episodes vs. Level of Interaction Over Time [5]

Analysis of group interactions also revealed the importance of having the right mix of expertise. The relevant expertise is important in two ways, to identify when errors or misunderstanding occur, or to identify when a task could not be done for lack of expertise. There were instances in the project where misinformation was shared where experts played a significant role in correcting. If the right expertise was not present, errors would have persisted and group frustrations would have elevated. Another instance, experts recognized the lack of expertise. This is especially important from an educational and learning perspective so as "ineffective" practices do not persist.

Facilitators of the learning process

Most contributions to learning opportunity episodes and code development in our study came from core-developers. Seventy-six percent of mailing-list postings (containing the learning behaviors discussed in the previous section) and 72% of code submissions were made by core-developers (the experts in the project). Events that triggered learning in Apache were generated from core-developers, as 75% of triggers were internal learning triggers. We also observed the movement of co-developers to being part of the core development group. Similarly, we observed active users becoming more active and becoming co-developers. This movement was indicated by an increase in these individuals contribution to the code and their contribution to the learning process by increasing their sharing knowledge, confirming understanding, and conflict resolution behaviors. This confirms Lave and Wenger's [8] Legitimate Peripheral Participation model of learning. Novices observe experts until they are capable of moving closer to the center of practice and making significant contributions.

Q2 - What are the benefits and limitation of the learning process from an educational perspective?

The findings of the study suggest that OSS provides an interesting setting for a real world hands- on learning experience. Learners are exposed to both the technical and social aspects of developing software. They learn how to write code, understand the design of the software, while learning how to participate or manage a global software development group. The findings suggest that individuals can learn valuable technical and social skills as well as learning

behaviors through legitimate peripheral participation. Learners have access to experts developing code through their observation of and interaction with the development team and process. Learners learn specific technical knowledge about writing code and documentation as we well as behavior to share such knowledge and analyze code. Additionally, they may observe social skills necessary while participating in such software projects.

Further, OSS provides lower barriers to entering a project compared to projects in formal organizations as any individual can participate as an active user if they wish. Since most of the projects are conducted in an independent and distributed environment, individuals having access to information technologies do not have to relocate or change their work or school schedules.

The challenges OSS provides to an effective learning environment are significant. Although the capital costs may be low, there is a significant learning curve associated with these projects as there is often little documentation on the overall system design or documentation.. Necessary for overcoming this obstacle is the support of the core developers (the experts). The core developers have a significant role to play in explaining the code and the appropriate practices. In Apache, the core group of developers where both knowledgeable of the code, and had the skills to explain it to new comers and to each other. This is not necessarily the case in most OSS projects. There is a need to provide the support for learners in these instances. If the appropriate support in explaining the conceptual foundation to the technical process in which they engage, their learning experience becomes limited.

Also noteworthy is that fact that some social practices shared are sometimes ineffective or inappropriate and become a liability for the new learners. Particularly, the informal nature of the environment most often does not emulate a formal system development business environment in that there is often no explicit system design, no deadlines, informal project management (individuals do the tasks that interest them).

CONCLUSION

There is no doubt that OSS development has the potential for providing an effective learning experience for IS students interested or engaged in software development. Participating in OSS projects provide an alternative or supplemental low cost real-world experience. Students are exposed to both the technical and social aspects of developing software in a global team. The challenges highlighted above when the appropriate expertise is not available however, do pose a challenge. Our students and professionals are participating in these experiences regardless of these challenges and are sometimes affected by these limitations.

To capitalize on the benefits and minimize the limitations IS educators can potentially fill a role. There are two aspects that we must consider. First, since many of our students, current young professionals, and future professionals engage in these groups, we must define our role in educating our students and future professionals in the OSS environment. Are there particular skills we can provide to make OSS participant better aware of the nature of learning in these environments? Can we utilize the learning behaviors identified from our study and similar studies and teach them to our students and professionals?

Secondly, there is potential for IS programs to integrate the experience of participating in OSS projects into the curriculum. This could be done as an experiential piece of the curriculum to augment or substitute for internships. If this option is pursued, it is pivotal that the learning experience be structured in a way that attention is given to both the technical and social aspects of this experience. Conceptual content for both topics should be provided as part of the formal learning experience. Also, the formal learning experience provided should contain a reflection cycle where instructors and other students reflect together on the nature of activities and learning taking place in these projects.

In conclusion, OSS is an interesting and important phenomenon. The nature of learning in OSS projects provides potential benefits for IS education. These potential benefits however come with some limitations. IS educators must further understand how learning is taking place in this environment. As well, it is important that we be creative in imagining the possibilities for capturing the benefits of the learning opportunity OSS provides. Consequently, further investigation and experimentation with these learning environments can provide exciting learning opportunities for IS education.

REFERENCES

1. Arent, J. and Nørbjerg, J. (2000) Software Process Improvement as Organizational

Knowledge Creation: A Multiple Case Analysis. *Proceedings of the 33rd Hawaii International Conference on System Sciences*. http://doi.ieeecomputersociety.org/10.1109/HICSS.2005.57e

2. Tuomi, I. (2005) What did we learn from open source? *First Monday*. 10(10). http://www.firstmonday.org/issues/special10_10/tuomi/index.html
3. University of Maastricht (International Institute of Infonomics) and Berlecon Research GmbH. (July 2002). FLOSS Final Report. http://www.infonomics.nl/FLOSS/report/
4. Robles, G., Scheider, H., Tretkowski, I. and Weber, N. (2001). Who Is Doing It?: A Research on Libre Software Developers. http://widi.berlios.de/paper/study.html
5. Annabi, H. (2005). Moving from Individual Contribution to Group Learning: The Early Years of the Apache Web Server. Unpublished Ph.D. Dissertation, Syracuse University, Syracuse, NY.
6. Annabi, H., Crowston, K. and Heckman, R (2006) From Individual Contribution to Group Learning: The early Years of Apache Web Server, The Second International Conference on OSS, Lake Como Italy, June.
7. Yin, R. 1984. Case Study Research. Sage Publications, Beverly Hills, CA.
8. Lave, J., & Wenger, E. (1991). *Situated learning: Legitimate peripheral participation*. Cambridge: Cambridge University Press.