Healthy Community and Healthy Commons: 'Opensourcing' as a Sustainable Model of Software Production

Damrongsak Naparat, Michael Cahalane, Patrick Finnegan
UNSW Australia Business School
Sydney, Australia
E-mail: {d.naparat, m.cahalane, p.finnegan}@unsw.edu.au

Abstract

Many commercial software firms rely on opensourcing as a viable model of software production. Opensourcing is a specific form of interaction between firms and open source software (OSS) communities for collaboratively producing software. The existing literature has identified opensourcing as a viable form of software production, which could be a substitute for "in-house" or "outsourced" software development. However, little is known about how opensourcing works or is sustained in the long term. The objective of this research is to explain the factors affecting the sustainability of opensourcing as a model of software production. The study employs a single case study of hospital software in Thailand to understand how firms and the communities can live symbiotically and sustain their collaboration to peer-produce vertical domain software. The analysis reveals six mechanisms (positive experience, trust in the leadership of the project leader, the demonstration of reciprocity, marketing the community, enriching knowledge, and face-to-face meetings) and demonstrates how they operate in conjunction with each other to sustain opensourcing.

Keywords

Opensourcing, sustainable, mechanism-based theorising, software production, vertical domain software

INTRODUCTION

Opensourcing is a particular form of crowdsourcing. Crowdsourcing involves "a company or institution taking a function once performed by employees and outsourcing it to an undefined (and generally large) network of people in the form of an open call" (Howe 2006). Opensourcing is a crowdsourcing method for software production where commercial firms and OSS communities collaboratively develop software of interest to firms (Ågerfalk and Fitzgerald 2008). Opensourcing has implications for both profit and not-for-profit firms since it allows them to leverage the intelligence and innovation of 'the crowd' by incorporating such innovation into their software products. Large firms that have practiced opensourcing include Apple and IBM (West 2003).

Ågerfalk and Fitzgerald (2008) have identified opensourcing as an interesting phenomenon. This is because opensourcing could be a viable alternative form of software production, especially for vertical domain software. However, the major challenge of opensourcing is its sustainability as it relies on the open source software (OSS) development model. Specifically, when software is peer-produced (e.g., OSS development), it is subject to several threats, such as under provisioning of contributions, unilateral appropriation of software, and misapplication of software, that could prevent its continuance (Benker 2002; O'Mahony 2003; Shah 2006).

A definition for "sustainable opensourcing" is lacking; hence we define it as the ability of an opensourcing project to (1) prevent the community from being damaged so that it exhibits the continuity of its software development and maintain software of the firms' interest (healthy community), and (2) nurture and improve the information commons of the project over a predetermined period of time (healthy commons). The first dimension of sustainable opensourcing concerns a healthy community, meaning an open source project has a significant number of participants (critical mass) to perform software development and maintenance activities. The healthy community is related to (i) how the community attracts participants and (ii) how the community sustains participation. The second dimension, a healthy commons, is concerned with the quality of contributions and the quality of software produced by the community. Information goods such as software are not subject to the overgrazing problem; however, the lack of improvement or nurturing of the commons can result in the tragedy of the commons in the context of software production (cf. Andreev et al. 2010). We argue that high-quality contributions are the crucial ingredient to produce high-quality software. Therefore, for opensourcing to be sustainable, its community and its commons must be healthy.

The objective of this paper is to explain the factors affecting the sustainability of opensourcing as a model for software production, especially for the development of vertical domain software. The paper begins by outlining the challenges that could prevent opensourcing from being sustainable, with the perspective that those challenges could severely affect either the health of the community or the health of the commons. We argue that sustainable opensourcing can be better understood through process theory by revealing the mechanisms that sustain opensourcing. This is followed by a description of the case study and research methodology. We use a single case study of the development of a hospital information system (HIS) in Thailand to discover mechanisms that help sustain opensourcing. This is because HIS development is a good demonstration of vertical domain software development, where business processes and user requirements are complex and are not known amongst developers. Hence there was a need for the software company to collaboratively produce software with a community of practitioners from the health care industry. The findings are then presented in the form of six theoretical propositions that, collectively, explain how each mechanism works in conjunction with others to sustain opensourcing. Finally, we provide recommendations for practitioners interested in pursuing opensourcing as a model for software production.

CHALLENGES TO SUSTAINABLE OPENSOURCING

Software production models such as “in-house” development or outsourcing rely on formal governance mechanisms to enable and sustain the process of software production. When software is produced within a firm, “hierarchical” governance mechanisms (e.g., management controls and incentive structures) (cf. Williamson 1996) ensure that the software development process can deliver a desirable outcome. In outsourcing, software is sourced through markets where the price mechanism (cf. Williamson 1996) plays a role in choosing the supplier. When the supplier is identified, it is the outsourcing contract that governs the production of software (Bush et al. 2008). Software outsourcing can be repeated as many times as necessary through market mechanisms, thus, while outsourcing providers may change, outsourcing is considered a sustainable form of software production. In contrast, opensourcing relies on a hybrid form of governance, consisting of hierarchical and community-based social mechanisms to produce software (Naparat and Finnegan 2013). We argue that the core problem with this hybrid governance is that while hierarchy seems to work perfectly for controlling and coordinating paid developers, who are sponsored or employed, firms do not have formal control over the voluntary participants in opensourcing. Lacking complete control over the opensourcing community presents several challenges related to the sustainability of opensourcing as a model for software production. Those challenges are (i) under provisioning (low level of contribution), (ii) unilateral appropriation of software, and (iii) misappropriation of software (Benker 2002; O’Mahony 2003; Shah 2006).

First, the under provisioning challenge is related to, in Benkler’s (2002) terms, the motivation problem. The OSS literature reports different types of motives that drive participants to participate in OSS development. Broadly, these motives could be categorised into intrinsic (e.g., learning, fun) and extrinsic rewards (e.g., money, reputation, use of software) (Von Krogh et al. 2012). Some of these motives last longer than the others, thus sustaining participants’ contributions over a longer period (cf. Fang and Neufeld 2009). Moreover, participants’ motivation can be undermined, and when their motivation is low and continuing to participate does not seem to be beneficial, participants leave the community (Benkler 2002, Shah 2006). This is important because the loss of participants adversely affects the health of the community.

Second, unilateral appropriation is evident when a particular individual attempts to direct the software development to benefit themself in a way that prevents other participants from benefiting (Benkler 2002). Dahlander and Wallin (2006) suggest that for firms to push their agenda, they need to assign their employees to work with OSS communities. These employees need to gain legitimacy and take important positions in those projects in order to influence the work of the community. However, focusing too much on a firm’s agenda could lead to unilateral appropriation, which in turn may introduce conflicts between firm-based developers and voluntary developers. Shaikh and Cornford (2010) found that such conflict arises when firm-based (paid) developers attempt to coerce voluntary participants to meet a firm’s requirement. Ågerfalk and Fitzgerald (2008) suggest that it is important that firms, before pushing their own agenda, require consent from others in the community. Unilateral appropriation could lead to a greater degree of monitoring and scrutinising of code, and stronger peer review, resulting in the community incurring the cost for monitoring (e.g., more time for peer review) (Shaikh and Cornford 2010). In sum, the lack of trust and higher cost of software production can lead to the departure of participants (Adler 2001; Benkler 2002), thus damaging the health of the community.

Finally, the misappropriation of software is another challenge that could prevent opensourcing from being sustainable. According to Shah (2006), while firms can establish software development projects and allow outsiders to contribute to the projects, the use of non-open source licences makes participants feel reluctant to participate because they think that the license would make it difficult for them to get the permission to use the software. Even in several exemplary open source projects, such as the Linux kernel or Debian Linux, the communities have to establish foundations and employ an open source licence as a legal means of protecting their projects from being misappropriated by individuals (O'Mahony 2003). We argue that misappropriation of software leads to the departure of participants because it prevents other participants from benefiting from the software produced. Again, this dramatically damages the health of the community.

RESEARCH DESIGN

The research objective is to explain the factors affecting the sustainability of opensourcing as a model of software production. We use process theory as it allows us to unearth mechanisms and deep structures that allow us to achieve the explanation necessary to meet our research objective. The identification of mechanisms is necessary because mechanisms are "the nuts and bolts processes by which cause and affect relationships in the social world come about" (Gross 2009, p. 386). Therefore, we employ mechanism-based theorising to explore the "nuts" and "bolts" that sustain opensourcing.

Mechanism-based theorising seeks to explain how and why a process (P) with certain inputs (I) can produce outcomes (O) (Gross 2009; Hedström and Sweberg 1998; Hedström and Ylikoski 2010). In the context of the use of mechanism-based theorising in a social situation, Gross (2009) explains that;

> [a] social mechanism is a more or less general sequence or set of social events or processes analysed at a lower order of complexity or aggregation by which—in certain circumstances—some cause X tends to bring about some effect Y in the realm of human social relations. This sequence or set may or may be analytically reducible to the actions of individuals who enact it, may underwrite formal or substantive causal processes, and may be observed, unobserved, or in principle unobservable. (p. 364)

Gross (2009) argues that social mechanisms are composed of a chain of four elements: actors, problem situations, habitual responses, and resources—or the A-P-H-R chain. The identification of the A-P-H-R and their relationships allow a mechanism to be revealed.

The search for mechanisms can be carried out at the level of an individual. Therefore, we adopt DBO theory to explain an individual's behaviour. DBO theory is composed of three important theoretical terms: desire (D), belief (B), and opportunity (O), which are required to analyse actions and interactions of individuals. DBO theory places emphasis on individuals' actions and argues that human actions are carried out through a combination of desires, beliefs, and opportunities (Hedström 2005).

In opensourcing, participants encounter a series of problems related to software production. Participants employ their resources (e.g., skills, knowledge, time) to respond to these problems by performing individual and collective actions to reach solutions.

Data Collection

We followed an in-depth single case study for this research. A single case study is suitable for explanatory research because it allows the researcher to trace the case closely over time (Yin 2003). A single case study was conducted with the Hospital Operating System (HOS) community (details in the next section). The HOS community was chosen because it produced software in a continuing manner for more than 10 years, which we considered to be long enough to demonstrate sustainability.

The data collection for the study was conducted over the course of 18 months, from July 2011 to December 2012. Multiple data collection techniques were used to collect data: (i) semi-structured interviews, (ii) document analysis, and (iii) online/offline field observations, as they allow for data triangulation, to enhance validity and reliability of the mechanisms underpinning sustainable opensourcing. We conducted 33 semi-structured interviews with four groups of participants. These included (i) the project founder and the company's CEO, who represented the sponsoring firm; (ii) eight Open Source Technology Ltd (OST) employees who participated in the project; (iii) 20 practitioners from rural hospitals who were voluntary participants; (iv) former participants; and (v) officers from central public health care authorities (see Table 1). The interviews with different types of participants allowed us to minimise the "elite bias" problem (cf. Miles and Huberman 1994). The interviews were aided by the use of interview guides. Each interview lasted 60–90 minutes. Documents related to the HOS project were gathered through the Internet and via personal contacts. Documents included news from online newspapers, user requirement logs, bug fix logs, and charts and diagrams depicting business processes, as well as the project and the sponsoring firm's websites. The online field observation was conducted through the community's online web forum and the community's Facebook group. Moreover, we conducted offline field observations by participating at several informal meetings amongst the community, formal HOS user trainings, and leisure activities. The field observations allowed the researcher to gain the trust of informants, which resulted in informants being more willing to reveal their opinions (Jorgensen 1989).

Transcribed interviews, gathered documents, and field notes yielded more than 500 pages of transcripts. Data analysis was conducted using open/axial coding and selective coding techniques (cf. Charmaz 2008; Corbin and Strauss 2008). Concepts and their relationships were identified via an open/axial coding process, thereby revealing construct categories and subcategories. Next, mechanisms were identified from the constructs through the use of the A-P-H-R chain and the DBO theory (cf. Gross 2009; Hedström 2005). The researcher followed the approach of Corbin and Strauss (2008) and identified relationships between concepts, contexts, conditions, actions, interactions, and actors who perform the actions. The identification of relationships amongst these elements allowed the researcher to discover mechanisms, how they operate, what the outcomes of the mechanisms were, and how the mechanisms were linked or how they operated in conjunction with one another. Finally, "core" mechanisms that sustain opensourcing were identified through selective coding.

Table 1: Interviews conducted in the study

Type of Participant	No. of Interviews	Interview Duration (Hours : Minutes)	Description
Project founder	2	1:40	Rationale for opensourcing
CEO	1	1:32	Roles of firm
Paid developers	11	11:37	Roles, short-/long-term motivations
Voluntary participants	14	Approx. 20 hours	Roles, short-/long-term motivations
Former participants	3	2:32	Reasons for leaving/rejoining the community
Provincial public health officer	1	1:20	Perspective on OSS medical software, provincial health policies
National health security officer	1	0:30	Perspective on OSS medical software, national health policies
Total	33	Approx. 40 hours	

THE CASE STUDY AND ANALYSIS

The Case Study Background

The Hospital Operating System (HOS) project is an opensourcing project that employs a community-driven model for Information and Communications Technology (ICT) at rural hospitals. The primary goal of the project is to develop hospital information systems software for small rural hospitals. The HOS project began in 1999 as a 17-month, government-funded research project, with its initial plan to implement the system at 10 hospitals. HOS was implemented at 17 hospitals nationwide when the research projected was completed. In 2000, the project founder, Kongkiat Kespetchara, turned the project into a commercial venture and founded Open Source Technology Co. Ltd. (OST). The project continued to use the community-driven model to crowdsource production. Participants in the HOS community included practitioners from rural hospitals (e.g., doctors, nurses, pharmacists) and paid software engineers (OST's employees). HOS proved itself as an exemplary sustainable opensourcing project because its collaborative software development model has worked for more than 10 years and is still in operation. Rural hospitals using HOS gained the benefit of process improvement by reducing patient risk and lost data, as well as increasing data accuracy and financial returns. HOS was released under a General Public License (GPL), but OST sells services around HOS, including software installation and configuration, custom module development, and providing related computer and networking hardware.

Analysis

The analysis revealed a chain of six mechanisms working in conjunction with each other to sustain opensourcing (see Table 2). Following the mechanism-based theorising approach, the presentation of each mechanism includes the description of a situational problem, the definition of the mechanism, a discussion of the operation of the mechanism to solve a situational problem, and the presentation of the supporting evidence from the case study.

Table 2: Mechanisms that sustain opensourcing

Mechanism	Description	Sustainability Dimension
Positive Experience	Positive knowledge, perception, and attitude that participants have accumulated from their previous participation	Healthy Community
Trust in the Leadership of the Project Leader	The participants' firm belief that the project leader is visionary, knowledgeable, and has enough resources and control to steer the opensourcing project in a way that benefits the community at large	Healthy Community Healthy Commons
Demonstration of Reciprocity	Examples of reciprocal acts performed by participants	Healthy Community
Marketing the Community	How the opensourcing community employs methods to create awareness and earn a good reputation for the project.	Healthy Community
Enriching Knowledge	How the opensourcing community improves the domain-specific and technical knowledge of its participants	Healthy Commons
Face-to-Face Meetings	Regular/ongoing offline face-to-face meetings organised by the opensourcing community to allow participants to meet each other in person	Healthy Community

The Positive-Experience Mechanism

It is found that the erosion of participants' motivation is a challenging issue that the opensourcing community needs to overcome. It was unavoidable that some participants left the community during the last 10 years. However, the community managed to retain the majority of participants, especially active participants. In addition, the community has attracted new participants. The analysis revealed that the positive experience of participants plays a critical role in maintaining participants' motivation.

We argue that a positive experience comes about through positive knowledge, perceptions, and attitudes that participants have from their previous participation in the project. Positive experience creates the desire for participants to continue to participate. Participants continue their participation with the expectation that they will receive the same or better experience from their future participation. There are two sources that bring a positive experience to participants: (i) the timely delivery of software solutions to participants, and (ii) good community culture.

First, the HOS community consistently delivered software and software updates to satisfy its participants and end users. Since participants were primarily driven by use of software, to be able to obtain a high-quality software solution satisfied their needs. For instance, one participant mentioned "although HOS is a free piece of software, its quality is as high as (if not better than) expensive commercial packages." Moreover, the community responded to bugs and error reports by providing bug fixes and solutions to errors in a timely manner. As for software maintenance, the community provided prompt and accurate solutions for any technical problems raised by participants. This led to a good impression for participants—the feeling that they were part of the community, that their contributions were valued, and that their problems were not ignored. Thus, they wanted to continue to participate. For instance, in discussing his positive experience, one long-term participant mentioned, "I remember having some great difficulties setting up HOS. There was one participant who was willing to help me fix the problem, which took quite a long time to finish and that was during the weekend. I appreciated that greatly."

Next, it was part of the community culture that the community valued participants' contributions, especially their requirements and domain-specific knowledge. Participants had their say on which user requirements they wanted the community to implement and how they should do it. All essential business processes and user requirements specified by participants were implemented. This impressed the participants, and they realised that their contributions and efforts were not wasted. Participants were engaged, and they felt that they were part of the community. This led to a positive experience and instilled in the participants the desire to continue their work. Another positive aspect of the community culture was that the HOS participants regarded each other as friends. The presence of friendship created a pleasurable community climate. Most of the participants mentioned this; for instance, one participant noted, "The HOS community culture is very good. We live as friends or kin. I wonder if I could experience the same thing in other communities."

Accumulated positive experience creates positive emotional feelings for participants, such as the feeling of passion in working with the community or loyalty to the community. Such emotions play a crucial role in sustaining participation and encouraging participants to spread positive images about the software and the community to the public, allowing the community to attract new participants**.** One long-term participant mentioned, "I love the HOS community . . . I will never leave the community or switch to other software." He even had his signature attached to every post on the web forum, stating, "I will love HOS until the day I die." Another participant, who was very passionate about using and contributing to the community, said, "I want every rural hospital to use HOS. Whenever I meet staff from other hospitals, I always persuade them to use HOS and join the HOS community."

Positive experience keeps the community healthy by maintaining current participants and attracting new participants. Therefore, our first proposition is presented as follows:

Proposition 1: Positive experience sustains opensourcing by instilling in the participants a desire to (i) continue to participate, and (ii) spread a positive image of the community, thus attracting new participants.

Trust in the Project Leader

We found that the ability of the community to eliminate and solve conflicts amongst participants creates a positive experience for them. Disagreements amongst participants normally led to unsatisfactory emotions towards each other, thus undermining their motivation to continue participating in the project.

Our analysis revealed that trust in the leadership ability of the project leader allowed the project leader to take a role in resolving conflicts very efficiently and effectively. The ability of the project leader to resolve conflicts quickly and satisfy participants influenced the participants' positive experience and their desire to continue participating. The fact that the project leader was the same person as the company owner meant that the leader had hierarchical power to mobilise the company's resources (e.g., paid developers, computer infrastructures) to serve the community. The project leader also gained trust from participants by being a visionary, knowledgeable, and a trustworthy person. As a trustworthy and respectful leader, voluntary participants were inclined to follow his decisions, and this minimised conflicts and resistance that could hamper the software development process.

The following evidence from our case study demonstrates how trust in the leadership allows the project leader to resolve conflicts and maintain a positive experience for the participants.

During the informal conversations amongst long-term volunteer participants, one participant mentioned, "Trust in the community was shaken because HOS development seemed to stall without any reason." Another participant added, "There were no fixes or updates released. It was miserable when this happened." Later, voluntary participants realised that OST put all of their efforts to produce another commercial HOS software and parked the HOS project. The response to this was "[w]e [volunteer participants] got together and wrote e-mails to the CEO and the founder, stating that OST should pay attention to the community's needs and continue the HOS development. We understand that OST has limited human-resource capacity and has to survive financially, but OST cannot totally ignore the HOS community. Otherwise, we have to switch to other software and discontinue our participation." Later the project leader called for a face-to-face meeting between the company and key long-term participants. At the meeting, the project leader assured the community that the HOS development would continue. After the meeting, the project leader mobilised the company's resources to get the project active once again. Moreover, the project leader made another promise to the community that "OST will continue to support the HOS project even if there was only one hospital using it, and it will always be open source" This statement dismissed doubts and questions that voluntary participants had in mind, such as "How long will OST keep supporting us?" and "Should I stay in the community?"

While conflicts undermine participants' motivation and deflect participants from software development tasks, trust in the leadership allows conflicts to be resolved quickly. We argue that the sustenance of participants' motivation promotes a healthy community. Moreover, because conflicts are quickly resolved, participants can focus on software development, thereby improving the health of the commons. Therefore, our second proposition is presented as follows:

Proposition 2: Trust in the leadership of the project leader sustains opensourcing by allowing the project leader to quickly solve conflicts, thereby (i) allowing participants to focus on software development and improving software quality and (ii) creating a positive experience for the participants.

The Demonstration of Reciprocity

We argue that the crucial problem of opensourcing is to sustain participation. As opensourcing lacks a managerial hierarchy, and an authoritative chain of command, and financial reward is only effective for paid developers, the ongoing participation of voluntary participants becomes a challenge. Our analysis revealed that it is the trust in reciprocity that triggers the exchange of resources to produce software at the beginning. However, it is the demonstration of reciprocity that sustains the ongoing participation.

To clarify, when participants see the demonstration of reciprocity, it affirms their trust in other participants. In addition, it increases their trust in the act of reciprocity. The demonstration of reciprocity thus instils in the participants a desire to continue to contribute because they believe that others will reciprocate.

The company and the community have demonstrated that they have lived up to each other's expectations and have not failed to reciprocate. Voluntary participants demonstrated that they regularly performed their tasks, including specifying user requirements, reporting bugs, conducting software testing, and providing user-to-user help. Similarly, the company demonstrated that it could transform the business processes and user requirements into a software solution. Bugs and errors were fixed through numerous software updates. From 2000 to 2012, the community released three major HOS versions, with numerous bug fixes and minor releases in between. Prompt responses and reciprocal acts from other participants (in the context of user-to-user help) also increased the participants' trust in the principle of reciprocity, which in turn instilled in them a desire to continue to participate.

Therefore, we present our third proposition as follows:

Proposition 3: The demonstration of reciprocity sustains opensourcing by increasing trust in other participants, thereby increasing their expectation of reciprocity.

Marketing the Community

Our analysis shows that the ability of the community to attract new participants and retain current participants is critical for the community to be healthy and viable. A proactive marketing of the community mechanism was employed to create public awareness and attract new participants. At the community building stage (i.e., early 2000), the company took an active role in recruiting practitioners to build up the community. Direct mails to rural hospitals and attending conferences allowed the company to receive attention from a significant number of practitioners, which was enough to create a critical mass of voluntary participants. Moreover, the company attended several medical conferences to publicise the HOS project in order to attract more participants. The company also attempted to internationalise the project and attract more international participants by producing an English version of HOS. In addition, HOS participants used 'word of mouth' to spread their positive experience about participating in the community, as well as the high quality of the software. Word of mouth was a very effective viral marketing mechanism for the HOS community as it helped attract new participants and expand their network of users.

The company received several prestigious national awards for the development of the HOS. These awards illustrated that community-based software development was efficient and effective and the community was capable of producing high-quality software. Good reputation of the project nourished participants' motivation and ensured that they did not waste their efforts.

It can be seen that the marketing the community mechanism allows the community to maintain current participants and attract new participants, thus improving the health of the community. Therefore, we present our fourth proposition as follows:

Proposition 4: The "marketing the community" mechanism sustains opensourcing by (i) building public awareness about the high-quality software product and the production process, thus (ii) attracting new participants and (iii) helping retain current participants through increasing reputation reward.

Enriching Knowledge

The analysis revealed that the ongoing knowledge transfer between the company and the community is critical as it allows participants to improve the quality of their contributions, as well as to perform more difficult tasks, which leads to a better pool of software (healthy commons).

There were two types of knowledge that were transferred amongst participants: domain-specific knowledge and technical knowledge. Domain-specific knowledge included information, know-how, business processes, and insight into the public healthcare domain. This type of knowledge was embedded locally in practitioners (i.e., voluntary participants). Domain-specific knowledge changed and evolved constantly. Therefore, participants needed to constantly transfer domain-specific knowledge to the community and to the company.

In addition, it was revealed that the company organised various training sessions with varying degrees of difficulty, ranging from fundamental to very advanced, to transfer technical knowledge to HOS users and voluntary participants. For example, the basic training included using HOS functions and basic HOS system administration and maintenance. More intermediate training sessions were infrastructure management (operating system installation and configuration and database management), SQL programming, using data-query tools, and using advanced report-building tools. The advanced training sessions included Java programming and HOS module programming. Moreover, we found that community-led mutual learning amongst peers allowed participants to gain domain-specific knowledge and technical knowledge. This learning was achieved through online and offline discussions on various topics. Through these discussions, knowledge was transferred and circulated amongst participants. The HOS community also had a peer-mentoring programme—meaning more experienced and knowledgeable participants assumed a mentoring role to help less-experienced participants (typically those who were new to HOS). We found that the effective way to do peer mentoring was by demonstrating how to perform tasks. The demonstrations were conducted both online and offline. The use of real-time interactive technologies (e.g., video conferencing, real-time messaging, and remote login) facilitated online mentoring. Offline mentoring was conducted through face-to-face meetings. However, offline mentoring normally occurred between participants who lived in nearby geographic locations. Otherwise, mentoring took place at special events, such as formal training sessions or the HOS annual conference.

We found that technology transfer allowed participants to perform more tasks. Many community members began participating by providing user requirements. However, after several formal SQL programming training sessions, they also began to contribute SQL code to the community, which significantly improved the data-reporting functionality of the HOS. Moreover, formal Java programming training sessions transferred sufficient programming skills to participants for them to be able to contribute the HOS core code. This demonstrated that voluntary participants could carry out more difficult tasks if their skills and knowledge were enriched. Many participants gained HOS system administration and maintenance skills from peer mentoring and formal user training, so now they provide user-to-user help to novice participants. This emphasises the importance of "enriching knowledge" in improving participants' knowledge and skills.

By improving participants' knowledge and skills, participants can do more tasks and improve the quality of their contribution, thus promoting a healthy commons. Therefore, we present our fifth proposition as follows:

Proposition 5: The enriching knowledge mechanism sustains opensourcing by (i) transferring and improving participants' domain-specific and technical knowledge, thus (ii) allowing them to improve their contributions to the development of software.

Face-to-Face Meetings

An important finding was that face-to-face meetings are crucial for sustaining opensourcing as they allow participants to effectively communicate with each other, as well as develop and strengthen their casual relationships (healthy community). The community organised face-to-face meetings on a regular basis, including an annual meeting, training sessions, leisure activities, and site visits to hospitals. Although participants established good relationships through online interactions, offline interactions significantly enhanced their relationships. Participants wanted to see each other in person if there was an opportunity. Spending time together to perform leisure activities introduced another aspect of being a community participant, socialising.

The HOS community regularly organised leisure activities after formal meetings and training sessions. Activities such as playing soccer encouraged participants to mix and to work as a team while also having fun. Staying at the same hotel allowed participants to have opportunities to have informal conversations and to get to know each other better. Informal dinners, drinks, or excursions to various tourist attractions were useful as icebreakers. Participants, once they were more acquainted with each other, were more open to establishing close relationships. As evidenced from the field observations, several participants became good friends and were more eager to help each other.

In summary, personal relationships go hand in hand with work relationships. It creates a good community culture where participants actually work with their friends, rather than just colleagues. The duality of work and fun, or work relationship and casual relationship, leads to a positive experience for the participants. Personal relationships are key to enhancing the community culture and to sustaining participants' motivation, thus enhancing the health of the community. Therefore, we present our sixth proposition as follows:

Proposition 6: The face-to-face meeting mechanism sustains opensourcing by (i) building casual relationships in addition to work relationships, thereby (ii) enhancing the community culture and participant experience and (iii) reinforcing participants' desires to continue participating.

CONCLUSION

This study aimed to explain the factors affecting the sustainability of opensourcing as a model of software production. To achieve this, we employed mechanism-based theorising and a case study of the production of hospital software to reveal mechanisms underpinning sustainable opensourcing. We have contributed to the opensourcing literature and practice in several ways. First, acknowledging the lack of a definition of "sustainable opensourcing," we have defined the term and argued that sustainable opensourcing comprises two dimensions: "the healthy community" and "the healthy commons." Second, we have presented a chain of six mechanisms underpinning sustainable opensourcing. We have revealed that positive experience, the demonstration of reciprocity, marketing the community, and face-to-face meetings significantly enhance the health of the community as they retain current participants and attract new participants. Enriching participants' knowledge plays a significant role in sustaining opensourcing by allowing the community to improve the quality of contributions, thus promoting a healthy commons. Furthermore, "trust in the leadership" facilitates a healthy community and a healthy commons. Third, findings from the study emphasise the role of face-to-face meetings and casual relationships in sustaining opensourcing. Although the community uses a number of online collaboration and communication platforms to communicate effectively, face-to-face meetings are still necessary to allow participants to strengthen their relationships and develop trust. Although participants are motivated by utilitarian rewards, such as money and software use, personal relationships are also crucial as they enhance the community culture and hold the community together over the long term.

This study is limited by the use of a single case study. Moreover, Thai culture and the domain of business (medical) are expected to influence the result of the study. We then call for more case studies conducted in different settings. Future qualitative studies should be conducted in other vertical domains and/or different cultural settings. Findings from future studies will allow the theory proposed in this study to be refined and improved. Having presented theoretical propositions, this study provides a good foundation for future quantitative research. Future quantitative research can delineate the hypotheses from the propositions to perform theory testing.

REFERENCES

Ågerfalk, P. J., and Fitzgerald, B. 2008. "Outsourcing to an Unknown Workforce: Exploring Opensourcing as a Global Sourcing Strategy," *MIS Quarterly* (32:2), June, pp. 385–409.

Adler, P. S. 2001. "Market, Hierarchy, and Trust: The Knowledge Economy and the Future of Capitalism," *Organization Science* (12:2), April, pp. 215–234.

Andreev, P., Feller, J., Finnegan, P., and Moretz, J. 2010. "Conceptualizing the Commons-based Peer Production of Software: An Activity Theoretic Analysis," *ICIS 2010 Proceedings*.

Benkler, Y. 2002. "Coase's Penguin, or, Linux and 'The Nature of the Firm,'" *The Yale Law Journal* (112:3), December, pp. 369–446.

Bush, A. A., Tiwana, A., and Tsuji, H. 2008. "An Empirical Investigation of the Drivers of Software Outsourcing Decisions in Japanese Organizations," *Information and Software Technology* (50:6), May, pp. 499–510.

Charmaz, K. 2008. "Constructionism and Grounded Theory," in *Handbook of Constructionist Research*. New York, NY: The Guilford Press.

Corbin, J., and Strauss, A. 2008. *Basics of Qualitative Research 3e*. Thousand Oaks, CA: SAGE Publications.

Dahlander, L., and Wallin, M. W. 2006. "A Man on The Inside: Unlocking Communities as Complementary Assets," *Research Policy* (35:8), October, pp. 1243–1259.

Fang, Y., and Neufeld, D. 2009. "Understanding Sustained Participation in Open Source Software Projects," *Journal of Management Information Systems* (25:4), Spring, pp. 9–50.

Gross, N. 2009. "A Pragmatist Theory of Social Mechanisms," *American Sociological Review* (74:3), June, pp. 358–379.

Hedström, P. 2005. *Dissecting the Social: On the Principles of Analytical Sociology*. New York, NY: Cambridge University Press.

Hedström, P., and Swedberg, R. 1998. *Social Mechanisms. An Analytical Approach to Social Theory*. Cambridge, UK: Cambridge University Press.

Hedström, P., and Ylikoski, P. 2010. "Causal Mechanisms in the Social Sciences," *Annual Review of Sociology* (36:1), April, pp. 49–67.

Howe, J. 2006. "Crowdsourcing: A Definition," Retrieved 8 March, 2011 from http://crowdsourcing.typepad.com/cs/2006/06/crowdsourcing_a.html

Jorgensen, D. L. 1989. *Participant Observation : A Methodology for Human Studies*. Thousand Oaks, CA: SAGE Publications.

Miles, M. B., and Huberman, A. M. 1994. *Qualitative data analysis: An expanded sourcebook 2e*. Thousand Oaks, CA: SAGE Publications.

Naparat, D., and Finnegan, P. 2013. "Crowdsourcing Software Requirements and Development: A Mechanism-based Exploration of 'Opensourcing,'" *AMCIS 2013 Proceedings.*

O'Mahony, S. 2003. "Guarding the Commons: How Community Managed Software Projects Protect Their Work," *Research Policy* (32:7), July, pp. 1179–1198.

Shah, S. K. 2006. "Motivation, Governance, and the Viability of Hybrid Forms in Open Source Software Development," *Management Science* (52:7), July, pp. 1000–1014.

Shaikh, M., and Cornford, T. 2010. "'Letting Go of Control' to Embrace Open Source: Implications for Company and Community," *HICSS 2010 Proceedings*.

Von Krogh, G., Haefliger, S., Spaeth, S., and Wallin, M. W. 2012. "Carrots and Rainbows: Motivation and Social Practice in Open Source Software Development," *MIS Quarterly* (36:2), June, pp. 649–676.

West, J. 2003. "How Open Is Open Enough?: Melding Proprietary and Open Source Platform Strategies," *Research Policy* (32:7), July, pp. 1259–1285.

Williamson, O. E. 1996. *The Mechanisms of Governance*. New York, NY: Oxford University Press.

Yin, R. 2003. *Case Study Research: Design and Methods 3e (Applied Social Research Methods, Vol. 5)*. Thousand Oaks, CA: SAGE Publications.

COPYRIGHT