Journal of Leadership and Management®
An International Journal of Strategic, Behavioral and Public Aspects of Leadership in Management
Cabell's Listed
A journal of the Institute of Leadership and Management Association, PO Box 564, Douglassville PA 19518. 484-332-3331. https://www.jleadershipmanagement.org

IT-Enabled Open Innovation Types & Behaviors

U. Yeliz Eseryel, College of Business/East Carolina University, eseryelu17@ecu.edu

Online published: July 2024 Print published: July 2024 Editors: Adam Szpaderski & CJ Rhoads	Authorship Roles and Conflict of Interest Statement is on file at the Journal of Leadership and Management offices, available on request. Contact editors@jleadershipmanagement.org

ABSTRACT

Companies increasingly move towards open innovation strategies. Many organizations adopt disruptive business models by emulating Open source software (OSS) development processes or by adopting OSS products. Disruptive business models and management innovation are increasingly built around information technologies (IT). Yet, extant organizational innovation literature has several limitations that make it challenging for organizations to learn from open innovation communities: (1) It ignores the role of IT in innovation process, except for the use of ideation tools. (2) It does not measure process-level elements that can be emulated by others, such as innovation behaviors. Rather, it mostly measures the outcomes in the form of micro- or macro-level technology and marketing measures. (3) It does not provide tools for researchers or organizations to observe and measure open innovation activities. In this study, we conduct a comparative case study using cross-sectional data. This study addresses these three gaps in the extant research by: (1) Identifying the IT used for open innovation, and how IT is used, (2) determining individuals' open innovation behaviors, which can be emulated by others, and (3) providing a content analysis schema that can be used by researchers and practitioners alike to determine the innovativeness of an open innovation community and identify who the key innovators are. We answer the following two research questions: How do successful open innovation communities innovate through information technologies? We identify three types of open innovation behaviors: action-based, synergistic, and peripheral action-based. We provide an open innovation content analysis schema that can be used to evaluate the innovativeness of individuals and open innovation communities.

KEYWORDS

Open Innovation, innovation behaviors, Open Source Software Development Communities, knowledge management, IT-enabled innovation, comparative case study

Acknowledgement

The author would like to thank the second coder for their help with content analysis schema reliability establishment. The following sources of funding provided partial support for data collection and data analysis: NSF IIS 05-27457 (Investigating the Dynamics of Free/ Libre Open Source Software Development Teams) and IIS 04-14468 (Effective Work Practices for Open Source Software Development Teams).

Introduction

Constant evolution of technology and rapid globalization has changed the nature of competition in industries. These innovations have allowed companies to begin organizing their research and development processes more efficiently. (Trott & Hartmann, 2009). For example, there is a move toward open innovation strategies to shorten innovation cycles (Gassmann & Enkel, 2001). Increased openness enables organizations to become more innovative. They add knowledge from external sources, including their stakeholders, such as the users of their products. Open innovation is the process of systematically encouraging and exploiting a wide range of internal and external knowledge sources for accelerating innovation (Chesbrough, 2003; Conboy & Morgan, 2011).

Information technology (IT) is a critical facilitator of open innovation (Chesbrough, 2003; Nambisan & Sawhney, 2007). Openness, by definition, requires companies to increase the number of external knowledge sources into their innovation processes (Conboy & Morgan, 2011). Thus, information technology (IT) is a Critical facilitator and natural enabler of open innovation (Chesbrough, 2003; Nambisan & Sawhney, 2007). For example, information technologies can make a company's innovation process easy and cheaply transparent. Furthermore, it allows outsiders to easily observe and take part in these processes.

Organizational innovation research stream largely ignores the role of IT in innovation, except for idea generation tools (Crossan & Apaydin, 2010). The information systems (IS) field recognizes how and when IT contributes to innovation success (Banker, 2007; Durmusoglu & Barczak, 2011; Kleis, Chwelos, Ramirez, & Cockburn, 2012; Pavlou & Sawy, 2006). Yet, little is known about the role of IT in expediting innovation in contexts such as software development (Conboy & Morgan, 2011).

In the software development context, open source software development (OSS) communities are prime examples of open innovation. OSS development communities provide a competitive advantage to the organizations utilizing them (Hedgebeth, 2007; Lundell, Lings, & Syberfeldt, 2011). OSS combines private investment and collective action model of innovation (von Hippel & von Krogh, 2003). OSS development communities collaborate globally via IT to develop software code that is visible and editable. OSS communities typically have a core developer team, that contributes a lot, along with many peripheral members. In OSS communities, the core team systematically draws knowledge users, and peripheral developers.

The benefits of the open innovation provided by the OSS communities are beyond the company level. OSS altered global competition in the computer software and hardware industries and even in adjacent industries (von Krogh & Spaeth, 2007). One example is consumer goods, where embedded OSS is becoming increasingly common. OSS growth influenced the society and economy (von Krogh & Spaeth, 2007) and transformed the software industry (Dinkelacker, Garg, Miller, & Nelson, 2002; Wasserman, 2009). It brought open innovation practices into software development (von Hippel & von Krogh, 2003). All these changes motivate both companies and governments to emulate the open innovation model presented by OSS development teams (Chesbrough, 2003; Davenport, 1997; Goldman & Gabriel, 2005). Large corporations and software companies incorporate OSS solutions into their product portfolio (von Krogh & Spaeth, 2007), or adopt new software development approaches utilizing OSS approaches and technology (Dinkelacker et al., 2002). Governments adopt policies to increase OSS development and use (Comino & Manenti, 2005; Cook & Horobin, 2006). Yet, to successfully emulate the open innovation practices of OSS communities, companies and governments need to (1) understand how these communities innovate, and (2) how they use IT for open innovation.

To answer these two questions, Eseryel (2014) adapted the innovation framework of Nonaka and Takeuchi (1995) for the open innovation setting. She developed a content analysis schema to analyze open innovation practices. Yet, her study is limited in its generalizability because it is a single case study (Eseryel, 2014). It's not clear whether the findings can be generalized to teams with differences in their communities, in terms of size and dynamics. Yet, both membership and participation may change over time, which may change the open innovation patterns (Wei, et al., 2021).

Thus, the goal of this article is two-fold: (1) Develop an open innovation framework building on the work of Eseryel (2014), (2) Choosing two open innovation projects that differ in their community leadership and membership to observe whether the open innovation types, behaviors, and dynamics hold in these opposite cases. Our research question is: *How do successful open innovation communities innovate through information technologies?*

LITERATURE

The Challenges of the Extant Innovation Literature for Investigating Open Innovation

The way innovation is often described and operationalized creates confusion on what it is and how it happens. The confusion about the nature of innovation is evident in the profusion of definitions and operationalizations (Garcia & Calantone, 2002). Garcia and Calantone (2002) identified 15 constructs and at least 51 distinct scale items in only 21 empirical studies on innovativeness.

Innovation is typically operationalized by measuring the nature of the technology or the market positioning of the product. Innovation literature operationalized its core construct with micro and macro level marketing and technology measures, disregarding the innovation process. The four types of innovation measures found in the literature are summarized next. *Micro-level marketing measures* used for empirical analysis of innovation include the newness of the customers, market approach or competitors from the perspective of the firm (Cooper, 1979), the newness of the product from the firm's perspective (Cooper & de Brentani, 1991), firm's experience of selling the product (Green, et al.,1995), and the newness of the product technology to the customer (Ali, et al., 1995). *Micro-level technology measures u*sed for empirical innovation analysis include the newness of the technology (Goldenberg et al., 1999), novelty of the software code (von Hippel & von Krogh, 2003), the technology knowledge base for the firm (Green et al., 1995), the newness of the production process for the firm (Cooper, 1979), modification of the technology currently in use at the firm (Colarelli O'Connor, 1998), level of technical difference from a firm's other products (M. Lee & Na, 1994), and the complexity of manufacturing technology (More, 1982). *Macro-level marketing measures* used to operationalize innovation include its newness to the world (Atuahene-Gima, 1995), newness to the competitive environment (Cooper & de Brentani, 1991), the consistence of the innovation with existing customer values (Souder & Song, 1997), the lack of actual demand together with the existence of potential demand (Cooper, 1979) and newness to the market (Cooper, 1979). *Macro-level technological measures* used for operationalizing innovation include the level of science and technology knowledge base within the general scientific community (Green et al., 1995), the level of modification of technology used in other industries (Colarelli O'Connor, 1998) and the extent to which innovation incorporates a substantially different core technology relative to the previous product generation (Chandy & Tellis, 2000). Kogabayev and Maziliauskas' (2017) overview of innovation theories presents two other categorizations of innovation, that are worth mentioning: Mensch categorize innovations into (1) basic innovations, (2) improving innovation, and (3) fake innovations. Basic innovations are the major innovations that shake the market by bringing in previously unknown processes and products. Improving innovation refers to small but important product, process, or service improvements. Fake innovations externally modify products or processes without changing their consumer characteristics (p.68). A multidimensional innovation model categorizes the concept across three dimensions; (1) product versus process, (2) administrative versus technological, and incremental versus radical (Cooper, 1998).

To summarize, innovativeness helps companies succeed in the market (Goldenberg et al., 1999). Some companies and governments want to learn how to innovate, but they do not know which behaviors to encourage, or how to train their employees. Yet, the measures used in empirical studies of innovation black box the innovation process. Understanding open innovation requires opening the black box. We need to understand what innovating is at its smallest visible component: a single innovation behavior of a person.

Conceptualizing Innovation

To conceptualize innovation, one needs to find an answer to two questions (1) What does 'to innovate' mean? and (2) What is the nature of the innovation process?

Earlier, we discussed innovation categorizations. These categorizations emerge from studies where researchers investigated innovations after the fact and categorize them. In our study, we aim at capturing the act of innovating, while it is happening. Therefore, we went back to some of the earlier literature that captures what happens at the core of innovating.

There is no generally accepted definition of innovation. According to the American Oxford Dictionary, the verb 'to innovate' means to 'make changes in something established'. The change, when considered within software development context, could be the introduction of *new ideas*, *features*, or *methods*. Afuah (2003) referred to innovating as incorporating new knowledge into products, processes, and services. Similarly, Twiss and Goodridge (1989) described innovating as achieving novelty and extending from *the emergence of an idea* to its commercialization. Urabe (1988) suggested that 'innovation consisted of the *generation of a new idea* and its implementation into a new product, process or service'. To summarize, to innovate refers to developing new ideas and then making changes in products, processes, and services by incorporating these new ideas. Therefore, the innovation process should start from (1) idea generation, which may then go through the processes of idea development, evaluation, and idea selection, and (2) incorporating that idea into an implemented product, process, or service.

Urabe (1988) describes the nature of innovation as 'never a one-time phenomenon, but a long and cumulative process of a great number of decision-making processes, ranging from the phase of generation of a new idea to its implementation phase'. This suggests that should not wait to identify one very impactful activity and study it to understand the innovation process. Rather, we need to capture all incremental idea generation, development, evaluation, selection, and implementation activities to capture innovation.

Operationalizing Open Innovation Behaviors

Open innovation is described as the process of systematically encouraging and exploiting a wide range of internal and external knowledge sources for accelerating innovation (Chesbrough, 2003; Conboy & Morgan, 2011; Eseryel, 2014 p.806). Therefore, combining the openness aspect of Chesbrough's definition with our innovation definition, we can develop a more thorough definition of open innovation: ***Open innovation is the process of systematically exploiting internal and external knowledge sources for accelerating the generation, development, evaluation, and selection of ideas, which are then implemented to develop or improve products, processes, or services.***

Our definition, therefore, includes the following components: (1) exploiting internal/internal knowledge sources for (2) idea generation, development, evaluation, & selection, and (3) incorporating that idea into an implemented product, process, or service.

OSS Open Innovation Communities

This section first introduces the general structure of OSS communities (although no two communities are exactly the same). We then describe how these communities innovate according to the extant research. OSS communities have a core-team of developers, and many individuals, who may become users, developers, and emerge as leaders. In their periphery, OSS communities have active and passive users (Crowston & Howison, 2006). Many communities do not have a hierarchical structure. Yet, members in the core versus periphery may have different behaviors even if they share some common behaviors. For example, Wei, et al. (2017) analyzed their communication behavior. They found that core and peripheral members may be similar in that they use more positive politeness strategies than negative ones. But they differ in the strategies they use to protect their positive face in positive politeness (Wei, et al, 2017).

While they are sometimes referred to as 'teams', they differ highly from organizational teams: For example, their most often used task-coordination method is to assign tasks to themselves (Crowston, Li, et al., 2007). The members care so much about keeping all community members in the loop that even when a group of community members physically come together, they sit around a table and communicate and work using the mailing lists, and other information technologies (Crowston, et al., 2005; Crowston, Howison, et al., 2007). OSS community decision-making processes are also uniquely different: Eseryel et al. (2020) identified five unique decision-making processes used by these communities: shortcut, implicit development, implicit evaluation, complete, and abandoned. Finally, participation in communication and decision-making by members differed according to task types and relevant triggers that start the communication process (Wei, et al., 2021).

The roles of OSS community members and leaders may change over time. For example, users may become developers, and then emerge as leaders based on their action-based transformational leadership (Eseryel & Eseryel, 2013): OSS leaders do not outline a vision and coordinate others to do the work. They continuously work on their vision by developing software code (action-based). Other developers and users 'follow' their leadership by modeling them and contributing to their work as well. Research showed that action-based leadership transforms both others' perception of the leader, and it strategically influences system development effectiveness and IT vision (Eseryel & Eseryel, 2013). In their theory of functional and visionary leadership for self-managing virtual teams, Eseryel et al. (2020) identified functional leaders who lead with substantive contributions within the existing organizational structures (e.g., shared mental models and norms). To change these structures, any action-based transformational leader (i.e., visionary leader) should first emerge as functional leader to get community followership. Functional and Visionary Leadership theory explains the OSS community leadership very well.

OSS Communities and Open Innovation

Knowledge creation and sharing is at the heart of open innovation. OSS community members use all four modes of creating and sharing knowledge: socialization, externalization, combination, and internalization

(Eseryel, 2014). Yet, Eseryel (2014) finds that for these four modes, community members use behaviors that are much more different than those observed in organizations (see for example, Nonaka and Takeuchi, 1995).

While individuals closer to the core may likely contribute more to innovation, the research strongly emphasizes the importance of the peripheral members. For example, Eseryel (2014) found that users may create nearly as much new knowledge as the core developers do (p.824). The OSS core developers design and develop the software. The users typically contribute by submitting patches, and reporting bugs (Lee & Cole, 2003), sharing their knowledge (AlMarzouq et al.,1998; Crowston & Howison, 2005; Gacek, et al., 2001; Mockus, et al., 2002). Users are peripheral members who increase software popularity by trial and error, and simply by using the product (Setia et al., 2012). They contribute to open innovation with fresh ideas (Chesbrough, 2003; Setia et al., 2012).

The community members exchange much information online, and they develop shared models on the software, and its development (Scozzi, et al., 2008). The knowledge creation within the OSS teams manifests through a plethora of learning opportunities (Hars & Ou, 2001; Hermann, et al., 2000; Himanen, et al., 2001; Kohanski, 1998), and socialization practices (Crowston & Annabi, 2005; Weber, 2004). Learning is the first step of knowledge creation. Learning begins at OSS communities through feedback and error correction (Kogut, 2000; Lee & Cole, 2003). Feedback and error correction, as well as critical evaluation of knowledge are important to innovation communities (Lee & Cole, 2003). Developers review the source code submitted by others and give feedback or fix errors as needed (Lee & Cole, 2003). This is basically a peer-review process. Developers depend solely on IT for communication, coordination, and work. All of these are archived online and publicly available online. Therefore, developers can exchange information, compare facts, search, and discuss any change (Lee & Cole, 2003). Archived knowledge extends the transparencies beyond synchronous interactions. The stakeholders can go back in time and experience the same knowledge transfer, as if the knowledge creation is happening while they are reading the mailing lists. This results in a continuous cycle of knowledge growth, which is key for (open) innovation.

A Framework to Codify OSS Open-Innovation

In the IS literature, innovation and knowledge management are viewed as two separate literature streams. At the core of (open) innovation lies knowledge creation, transfer, and management (Eseryel, 2014). Thus, we adapted Eseryel's (2014) framework for knowledge-creation modes and behaviors in OSS.

Knowledge-creation mode	Knowledge-creation behavior
Socialization (tacit to tacit)	Report bugs
	Submit patch
	Commit patches
Externalization (tacit to explicit)	Contribute to problem resolution by providing information
	Make suggestions and troubleshoot for the developers and users
	Explain logic behind suggestions
	Ask questions related to someone's suggestion
	Mentor and guide others
Combination (explicit to explicit)	Communicate issue resolution
	Communicate patch commit
	Refer the users to other knowledge sources
Internalization (explicit to tacit)	Document changes
	Write tests
	Develop webpages
	Contribute to wiki

Table 1. Knowledge-creation modes and behaviors in OSS Open Innovation Communities (Adopted from Eseryel, 2014)

According to this framework, there are four knowledge-creation modes (adopted from Nonaka and Takeuchi, 1995), and matching OSS knowledge creation behaviors for each mode. An extensive discussion of how OSS knowledge creation can be coded, and the coding schema can be found in Eseryel (2014, p.821). Below, we discuss only those OSS knowledge-creation behaviors that are relevant to open innovation. To secure this connection, we utilize the definition of open innovation that we developed in the previous section: Process of systematically exploiting internal and external knowledge sources for accelerating the generation,

development, evaluation, and selection of ideas, which are then implemented to develop or improve products, processes, or services. There are two types of knowledge that can be exploited: tacit knowledge refers to people's knowledge that they may find difficult to explain. Explicit knowledge refers to knowledge codified and explained, such as documentation. Using the categorization of four knowledge creation modes; socialization, externalization, combination, and internalization (Nonaka and Takeuchi, 1995) Eseryel identified 15 knowledge creation behaviors in OSS open innovation communities.

Knowledge Creation Behaviors in OSS Open Innovation Communities

Socialization

Socialization can be observed as tacit knowledge transfer through working together, such as in an internship (Nonaka and Takeuchi, 1995). Tacit technical skills can be transferred through observation, imitation, and practice, without needing to be made verbally explicit. OSS communities have their own version of an apprenticeship model. An informal OSS apprenticeship begins with testing and reporting bugs in the software, followed by submitting patches to fix the identified bugs (Ducheneaut, 2005; von Krogh et al., 2003). An OSS apprentice who follows the tacit rules and stays active over a sufficiently long enough time gains trust of the OSS members and may receive privileges given to members. While the software code is explicit knowledge (Haefliger, von Krogh, & Spaeth, 2008; Morner & von Krogh, 2009), observing how it is structured and built conveys tacit knowledge (Morner & von Krogh, 2009, p. 443). Explicit and tacit knowledge are entities on the same continuum rather than being distinct opposites (Jha, 2002; Nonaka & von Krogh, 2009).

In OSS communities, observing community-based decision process is another way technical and procedural tacit knowledge is transferred. Even the way decisions are made is very different than decision-making process theories of organizational theory (Eseryel, et al., 2020).

Studying the source-code artifact and participating in decision-making about the software code requires an intensive intellectual engagement with the source-code. (Eseryel, 2014). The source-code artifact carries within tacit knowledge, which may act as a coordination mechanism between those who interact with the source-code (Bolici, et al., 2016). Studying the artifact may show one the reasoning and choices used in its development, thereby transferring tacit knowledge. For example, one can examine which algorithms others use and how they structure the source code (Morner & von Krogh, 2009). This intellectual examination causes the software code, which is packed with tacit knowledge of its developers gets transferred to the person who engaged intellectually with the software to decipher this tacit knowledge.

Eseryel (2014) identified three distinct socialization behaviors for the IT-enabled context of open-source software development communities. These include (1) reporting bugs or improvement needs, (2) submitting patches, and (3) committing one's own patches Eseryel (2014, p.816).

Externalization

Externalization is a process of converting tacit knowledge into explicit concepts through the act of writing, dialoging, and collective reflection (Nonaka & Takeuchi, 1995). In the brick-and-mortar setting, externalization happened often using metaphors and analogies, which created shared understandings. Eseryel (2014) did not find evidence for the use of metaphors or analogies in the OSS interactions. Instead, externalization of tacit knowledge is found in problem conceptualization (& Reinhardt, 2006), new idea creation (Hemetsberger & Reinhardt, 2006) and problem resolution (Eseryel, 2014). The knowledge externalization was observed often in mailing lists, as individuals explained their ideas, evaluated, rejected, corrected, or defended certain ideas (Hemetsberger & Reinhardt, 2006). The externalization differed quite a bit in OSS innovation setting, from the company settings shared by Nonaka and Takeuchi (1995). In fact, the externalization in open innovation takes the form of detailed and clearly explained knowledge. Eseryel (2014, p.816) identified five ways in which software developers externalize their tacit knowledge as part of the problem solution process: (1) contribution to problem resolution by providing information, (2) making suggestions and troubleshooting for others, (3) explaining logic behind one's suggestions, (4) asking questions related to someone's suggestion, and (5) mentoring and guiding others.

Combination

Combination refers to creation of new knowledge by reconfiguring explicit information through sorting, adding, combining, and categorizing it (Nonaka and Takeuchi, 1995). In organizations, combination may

include summary documents, reports, or meetings. The combination activities create systemic knowledge in central repositories to enable new knowledge creation. For example, data mining and business intelligence techniques support further decision making. Three types of knowledge combination were identified in OSS development communities' archival data (Eseryel, 2014, p.816); (1) communicating issue resolution, (2) communicating patch commits, and (3) referring the users to other knowledge sources.

Internalization

Internalization refers to converting explicit knowledge into tacit knowledge (Nonaka & Takeuchi, 1995). Internalization happens by learning by doing. In software engineering, typically junior developers or interns may be tasked with testing and documenting an existing system (Kautz & Thaysen, 2001). These activities are excellent ways of internalizing an unfamiliar software environment, and thus recommended for software engineering education (Jansen & Saiedian, 2006). Documenting an unfamiliar software requires intellectual engagement by experimentation with the system to learn more about it, before a person can codify the information explicitly (Eseryel, 2014). Therefore, the act of documenting results in an increase of tacit knowledge in the mind of the documenter, who had to understand the system by studying it. The output of this process is an explicit artifact, i.e., document(s). Eseryel (2014, p.816) identified four types of internalization were identified by prior literature that applies to the OSS development communities: (1) documenting changes, (2) writing tests, (3) developing webpages, and (4) contributing to the community wiki.

RESEARCH METHOD

This section introduces the study context, provides an overview of the study, and describes the case selection, archival data collection, data reduction, and data analysis.

Study Context

For this study, two cases that have clear differences that allow for theoretical replication, but that are similar to each in terms of general structure: Two cases were OSS development communities within the Apache Software Foundation. Apache constitutes a revelatory example of open innovation (Thomas & Hunt, 2004). The Apache Software Foundation (ASF), a non-profit organization, was formed to transfer best practices of the successful Apache web server team (Fielding, 1999) to more than 300 open source communities under the ASF umbrella.

Case Selection

A preliminary study was conducted via interviews with long-term Apache Software Foundation members (1) to inform the researcher about the context, (2) to improve the researcher sensitivity, i.e., “ability of the researcher to pick up subtle nuances in the data that infer to a meaning” (Corbin & Strauss, 2008, p. 19), and (3) to select the appropriate cases for this longitudinal study (Pettigrew, 1990). This study helped identify the two very successful examples of open innovation to be investigated.

Two cases were identified to allow for theoretical replication (Guba & Lincoln, 1994). In theoretical replication, two cases are expected to show different results for predictable reasons. To protect the identity of the participants, the cases are given a pseudo name of Delta, and Stable. Stable project had stable leadership and community membership over the period of this study. The project had two key leaders, one of whom was a founder, and the other a long term leader. These perceived leaders were actively contributing to the software development and to the community. Delta project differed in its leadership. People who were in the same two roles (the founder and the other long-term leader) were slowing their participation and one new leader had emerged at the time of the study. Delta project also differed in its periphery: While Stable project had a decent sized periphery with 44 active users, Delta had 129 active users in its periphery.

To account for other factors that may potentially influence open innovation, two projects were selected to show similarities. Both Delta and Stable developed modular software, had a stable software release, their communities were vibrant, and at the development stage (Helfat & Peteraf, 2003). Finally, their membership satisfied the minimum size requirement to allow for interaction (Hare, 1976). In the first period, Delta had 11 core team members and Stable had 7. Delta had 129 users and Stable had 44 users who contributed to the project. Software development projects at the development stage presents an environment with (1) clear norms and roles, (2) active & dynamic software development environment, (3) new software development

activities, and (4) abundant communication. Thus, the selected cases allowed for open innovation as defined by von Hippel and von Krogh (2003).

Research Process

This study presents a cross-sectional comparative case-study of two OSS communities. Figure 10 presents the process steps and the goals & the outcomes of each of the 5 steps: (1) Preliminary study was conducted to for context familiarity, and case selection (Pettigrew, 1990). (2) interviews with key informants were conducted for each project. This step helped identify the leadership and the innovativeness of the project (3) Based on members' input, two types of threads were collected that preceded the interview: Critical incident threads, and regular activity threads. (4) Interviews were followed with content analysis of the archival data preceding the interviews. The content analysis allowed the observation of the community's open innovation practices that coincide whereas interviews allowed to capture member perceptions. (6) Lastly, conclusions were drawn from the two case studies after within- and cross-case analyses. The findings were synthesized and compared with the extant literature to systematically discuss the findings.

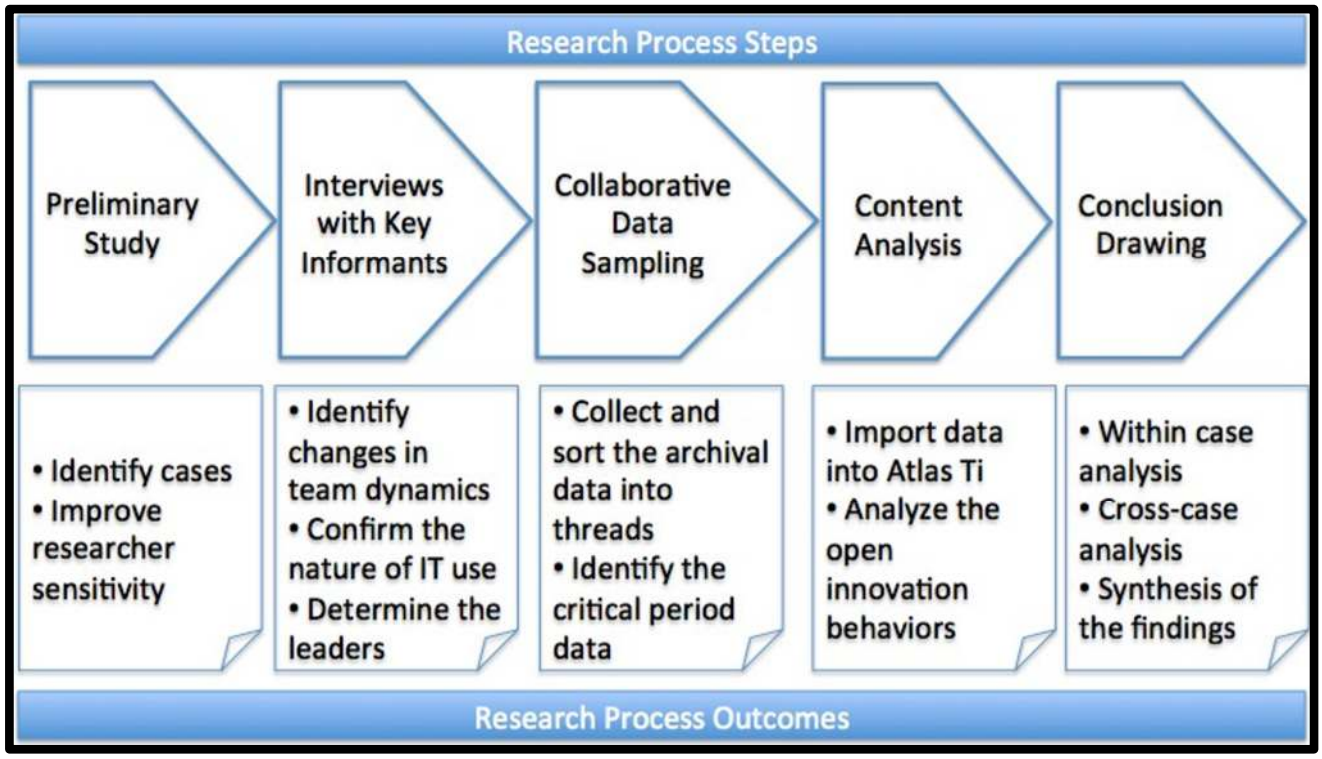


Figure 10: The Research Process Steps & Outcomes

Data

In this study, four types of information technology artifacts were analyzed for evidence of open innovation: Communication media: (1) developers' mailing list, (2) users' mailing list, (3) JIRA issue tracker, and (4) SVN (software version control). Developers' mailing list captured core-team based knowledge creation. Users' mailing list locates the inflow and outflow of knowledge between the core team and the extended community. These media, and the responses to them were organized by thread per the subject. The JIRA (issue tracker) helped the community coordinate efforts and ideation, which resides at the core of open innovation (Setia et al., 2012). SVN, the software versioning system that enables the developers to simultaneously work on and submit changes to the software. SVN allows identification of members' software code contribution behaviors, a form of knowledge contribution (Morner & von Krogh, 2009). Both SVN and JIRA provided systematic information on various aspects of the open innovation process, such as ideation (Setia et al., 2012), quality improvement (Setia et al., 2012) and action-based innovation (Eseryel & Eseryel, 2013).

Archival Data Sampling

Tables 1 and 2 present the data sampled from the Stable and Delta projects respectively. All data were organized in the form of threads based on their subject line. Single-email threads were eliminated, as they did not present opportunities for community based open innovation. The Stable project continued its leadership and membership in a similar way for a long time. At the Delta project, a major change occurred in the form of an inflow of a number of new and highly active members, and simultaneously the project founder reduced their activity and leadership. We expect the identified open innovation types and the open innovation behaviors to be about the same in both projects. Yet, we expect the projects to differ in how leaders, developers, and peripheral users contribute to the open innovation across two cases.

THE STABLE PROJECT (485 Messages Analyzed)			
	Regular Events	**Critical Events**	**Total Events**
SVN (Work)	20	43	63
JIRA	20	16	36
Developers' M. List	20	5	25
Users' M. List	20	1	21
TOTAL	**80**	**65**	**145 Events (485 Messages)**

Table 2: Sampling of the Archival Data Analyzed for the Stable Project

Eseryel (2014) identified two types of contributions as relevant for open innovation: ongoing regular contributions and critical contributions. Individuals' ongoing regular contributions contributed to incremental innovation. People's contributions to the critical events were also key to the success of the open innovation project. An example may be fixing a major bug that blocks the release of the software. Developers versus users' open innovation patterns between regular versus critical events differ (Eseryel, 2014).

Sampling 20 threads or fewer per archival data type allowed theoretical saturation (Eseryel, 2014), where adding new threads did not contribute further to the theory (Morse, 2004). We sampled160 regular-event threads from both projects and matching critical event threads in the same time period (104 threads). This sampling strategy resulted in the analysis of a total of 246 event threads (1066 messages); 485 messages from the Stable project

Table 2) and 581 messages from the Delta project (Table 3).

THE DELTA PROJECT (581 Messages Analyzed)			
	Regular Events	**Critical Events**	**Total Events**
SVN (Work)	20	17	37
JIRA	20	12	32
Developers' M. List	20	3	23
Users' M. List	20	7	27
TOTAL	**80**	**39**	**119 Events (581 Messages)**

Table 3: Sampling of the Archival Data Analyzed for The Delta Project

Critical events were identified first by filtering the (JIRA) issue trackers for 'major' issues (an important problem, which still allowed the users to use the software), 'critical' issues (an important issue which allowed a minor release, but not a major one), or 'blocker' issues (an issue that blocked all releases). Critical events

were identified within the six-months preceding the interview. The events were reviewed by a key informant of each community, who selected the most critical ones. All relevant developer and user discussions relevant to the critical events were included in their analyses.

Data Analysis

Interview Analysis

The interviews were used to identify the community leaders and to better understand the nature of the open innovation communities. The interviews were conducted with key informants of the team who were present at Apache conferences. All interviews were transcribed and then analyzed deductively by two independent content analyzers using Atlas-Ti content analysis software. Perceived leaders were identified by calculating perceived leadership indices (PLI) for each member at each interview period (Sarker et al., 2002). PLI was calculated for each participant by dividing the number of times they were identified as a leader by interviewees, divided by the total number of interviewees (Sarker et al., 2002). All members' perceived leadership index scores ranged between zero (non-leader) and one (perceived as a leader by all interviewees). Following Heckman and Misiolek (2005), members with higher than or equal to 0.5 PLI were identified as leaders, because 50% or more of the group members perceived them as leaders.

Content Analysis Schema Development to Analyze Archival Data

Our first task was to develop an open innovation content analysis schema. We adapted the knowledge creation for Open Innovation content analysis schema (see Table 1), developed by Eseryel (2014).

The data were organized in threads and content analysis was done using the Atlas-Ti software at a thematic level. Thematic level coding allows the selection of a whole message or any sub part as the unit of coding as long as the identified unit captures a meaning. Deductive coding was conducted using the IT-enabled knowledge creation for open innovation framework (Eseryel, 2014). The content analysis schema was reliable at 87% (Eseryel, 2014), which is considered high level of reliability (Neuendorf, 2002). Then, inductive coding was conducted using the open innovation definition developed earlier to allow for emergent codes.

The researcher and another independent coder established coding reliability for this study using a test sample of 20 episodes, which is more than the 10% of data (Potter & Levine-Donnerstein, 1999). In the second sample coding, the reliability of coding surpassed the original rate of 87%.

All open innovation behaviors were then re-grouped and categorized based on the open innovation definition provided earlier. Based on this effort, we ended up removing one externalization behavior, and all three combination behaviors from the content analysis schema. The rest of the codes were regrouped based on similarities in the type of open innovation. This re-grouping helped categorized 12 open innovation behaviors into three types.

Open Innovation Type	Description	Open Innovation Behavior
Action-Based Open Innovation	Innovating by contributing directly to the software development work (Eseryel & Eseryel, 2013).	Software development (Patch Submission in JIRA or Listserv)
		Commit own patch (SVN)
Synergistic Open Innovation	Developing innovative ideas and solutions through intense interaction with others.	Report a bug or recommend a feature (JIRA or Listserv)
		Ask questions related to another person's suggestion
		Contribute to decision-making by providing information
		Explain the logic behind one's own suggestions
		Make suggestions & troubleshoot
		Documentation (Test Development) (SVN)
		Review and commit user's patch (SVN)
Peripheral Action-Based Open Innovation	Action-based behaviors that do not involve changing the code, but that improve the overall product quality.	Documentation: Change log (SVN)
		Documentation: Webpage (SVN)
		Documentation: Wiki (SVN)

Table 4: Content Analysis Schema for Open Innovation Coding

After the reliability of the coding was established and the content analysis schema was adapted to cover only open innovation behaviors (Table 4), complete coding was done by the researcher using the reliable content analysis schema, as is commonly accepted (e.g., Lapointe & Rivard, 2005; Levina & Vaast, 2005; Pawlowski & Robey, 2004). Since data analysis is at the heart of building theory from case studies (Eisenhardt, 1989), it is best done by the researcher (Eseryel, 2014).

Table 4 presents the content analysis (coding) schema used for this study. The schema includes open innovation types, their descriptions, and the open innovation behaviors that fit each type. Among the fifteen types of open innovation behaviors, three behaviors were coded for action-based open innovation, five behaviors made up the synergistic open innovation, and seven different behaviors were coded for the peripheral open innovation. The coded data from Atlas-Ti were then transferred to Microsoft Excel and tabulated.

The findings were synthesized first by conducting within case studies and then cross-case comparisons (Eisenhardt, 1989). The analysis process used adopted data analysis steps recommended by Miles and Huberman (1994).

FINDINGS

This section presents the answers to the two research questions. Our research question was "*How do successful open innovation communities innovate through information technologies?*" Our first answer to the 'how' question is 'with three types of open innovation behaviors.' The identification of these behaviors, and their definitions are explained below.

Three Types of Open Innovation Behaviors

Table 4 names and describes the three types of open innovation and lists relevant open innovation behaviors. These were developed by following the process outlined in the 'Content Analysis Schema Development to Analyze Archival Data' sub-section of the research methods section. After finalizing the open innovation coding schema, we analyzed both communities using the reliable schema.

Table 5 presents the three open innovation types and the frequency of the behaviors constituting them based on archival data analysis of both projects when both projects' communities were relatively stable. All three innovation types would fall into "Improving innovation" among Mensch's three types of innovation (from Kogabayev & Maziliauskas, 2017). The innovation processes can be described as incremental innovation, according to Cooper's (1998) categorization (from Kogabayev & Maziliauskas, 2017) Below, we describe each type in detail.

Open Innovation Process	OI Behaviors in STABLE	Frequency	OI Behaviors in DELTA	Frequency
Action-Based	54	22%	60	22%
Synergistic	189	78%	191	69%
Peripheral Action-Based	1	0%	27	10%
Total	**243**	**100%**	**278**	**100%**

Table 5. The Frequency of Open Innovation Behaviors in Both Teams Preceding the Interview

Type-I: Action-based Open Innovation

The first type of open innovation is ***action-based open innovation***, where individuals innovated by contributing to the work of software development. The related behaviors are provided in Table 6. Action-based open innovation type was the second most frequently used open innovation type following the Synergistic innovation that will be described next.

Open Innovation Type	Description	Open Innovation Behavior
Action-Based Open Innovation	Innovating by contributing directly to the software development work (Eseryel & Eseryel, 2013).	Software development (Patch Submission in JIRA or Listserv)
		Commit one's own patch (SVN)

Table 6. The Action-based Open Innovation Behaviors

22% of innovation behaviors constitute action-based innovation in both projects (Table 5). Action-based open innovation refers to incremental innovation contributed by individuals through their work on software development. The term 'action-based' emerged from the work of Eseryel (2013), who described transformational leaders, who emerged as leaders, i.e., started to be perceived by the community as leaders, despite lacking a formal management or leadership role, due to their action-embedded contributions to the software development. These individuals accomplished their vision for the software purely by working towards the development, rather than by communicating a grand-vision and coordinating others to execute this vision. This is a key distinguisher between open innovation and closed innovation that is seen in hierarchical, organizational settings. The outcome of action-based open innovation served two purposes: (1) The produced software code directly contributed to the development of an incremental product innovation, and (2) The software code became a boundary object for community members to study and transfer their tacit knowledge. Thus, action-based open innovation created an opportunity for other members to transfer the innovator's tacit knowledge. The member who studied the new code may have learned about the thought processes and principles the innovator used to write the code. Thereby the transferred knowledge became this member's own (i.e., that of the member who studied the new code) tacit knowledge. One interviewee mentioned how they learned from a key developer by inspecting her work.

> *"You look at the code she writes, and you really admire what she does. I would like to be like her. So, I try to replicate her thought process sometimes."*

Another interviewee said:

> *"This is what you should do: start by testing, (and) reporting bugs. And then submit small patches, download and analyze others' work and lurk into what is going on the (mailing) list. This is how you are expected to learn about the project. Nobody will explain you all they know, or give you tasks to do."*

The quote above provides a representative example of how individuals transferred tacit knowledge from others in the community through intellectual engagement with the software artifact that the community members create. Yet, this knowledge transfer did not happen through an apprentice working in the physical location and observing and replicating a mentor's work, as it would be expected from a brick-and-mortar apprenticeship model. Rather, the transfer of tacit knowledge happened primarily through inspecting the final output of the software development act, namely the software artifact.

Information Technologies enable this kind of innovation because of the affordances it provides developers: The SVN system allows the developers to view the software code. It also allows multiple people to work on the code at the same time without a need for a coordinator. When multiple people make changes on the same functionality at the same time, it allows them to consolidate the changes. Therefore, members do not need to sit next to a developer writing code to learn from them. IT turns code into a boundary object between members. Studying the boundary object enables members the affordance of tacit-knowledge transfer that was in the past only possible by being in the same location.

Type-II: Synergistic Open Innovation

The second type of open innovation was ***Synergistic open innovation***. The innovation behaviors that fell into this category (see Table 7) included individuals' behaviors while innovating together through intense interaction and knowledge exchange with internal and external community members. Synergistic open innovation was the type of innovation that could precede the action-based open innovation. Yet, in open innovation communities, action-based innovation without preceding Synergistic open innovation is very common (Howison & Crowston, 2014).

In was the most frequently used open innovation process observed in the data: Among the open innovation behaviors, 78% (189 instances) of the Stable Project's and 69% (191 instances) of the Delta Projects were of Synergistic type (Table 5). This showed that the most frequently used open innovation behavior type involved individuals making their tacit knowledge very explicit to share with other, and then they built on each other's explicated knowledge.

Open Innovation Type	Description	Open Innovation Behavior
Synergistic Open Innovation	Developing innovative ideas and solutions through intense interaction with others.	Report a bug or recommend a feature (JIRA or Listserv)
		Ask questions related to another person's suggestion
		Contribute to decision-making by providing information
		Explain the logic behind one's own suggestions
		Make suggestions & troubleshoot
		Documentation (Test Development) (SVN)
		Review and commit user's patch (SVN)

Table 7. Synergistic Open Innovation Behaviors

For example, in the quote provided below, 'community member A' wrote about a problem she identified in the code and provided a solution. By doing that, member A made her tacit knowledge explicit. When another member, for example member B, read these ideas, he learned about the correct implementation of the spans interface, which means that there was transfer between the two members of general software development

knowledge. Member A also discussed how an improper implementation of the spans class may influence various features of the software being developed. Thus, there was a transfer of specific knowledge related to this software both members were working on.

> *"In TermSpans (or the anonymous Spans class returned by SpansTermQuery, depending on the version), the skipTo() method is improperly implemented if the target doc is less than or equal to the current doc: public boolean skipTo(int target) throws IOException {*
>
> *//are we already at the correct position? if (doc >= target) {*
>
> *return true;*
>
> *}*
>
> *...*
>
> *This violates the correct behavior (as described in the Spans interface documentation), that skipTo() should always move forwards, in other words the correct implementation would be:*
>
> *if (doc >= target) { return next();*
>
> *}*
>
> *This bug causes particular problems if one wants to use the payloads feature - this is because if one loads a payload, then performs a skipTo() to the same document, then tries to load the "next" payload, the spans hasn't changed position, and it attempts to load the same payload again (which is an error)."*

If member B only read member A's ideas, then the following knowledge transfer would have happened: By explaining her ideas, member A made her tacit knowledge explicit. Then by reading A's ideas, member B transferred member A's explicit knowledge and they became his tacit knowledge. Member B could have then joined the discussion and provided additional information that may change the way issue should be solved. This would have been an example of member B explicating his tacit knowledge. Further, this would have constituted an example of B's Synergistic innovation behavior since member B was building on member A's observations.

In Synergistic open innovation, the role of information technologies was to create interaction among people, which then resulted in a community-based innovation process. While the conversion of tacit knowledge to explicit knowledge (i.e., externalization) is known to be, by definition, very challenging, if not impossible, the archival data analysis indicated that the developers often explicated their ideas clearly, by using a combination of language and code samples. In fact, the interviews indicated that the developers were conscious about how they externalized their knowledge to get the community buy in. As one interviewee put it:

> *"When you put forth an idea that you would like to get implemented, you need to communicate it clearly so that everyone knows where you're coming from. That's why I try to be very clear about my communication on the [mailing] lists. I always explain what the alternatives are, why my solution is the best one, and how I will go about it and why. Doing so, you make sure that everyone gets behind the idea faster, so that you can go forward with it."*

Conscious efforts to externalize one's knowledge to get acceptance from others were often visible in the mailing list. For example, after giving an extensive explanation and evaluation for his/her recommendation, one developer asks:

> *"What do you think? We can do this change in [version] 3.0 so that we don't have to take care of backward compatibility issues, that is of course if everybody agrees to make the change."*

Externalization manifested mostly as part of the problem identification, definition, and resolution, which are Synergistic steps of software development. The example below presents an instance, where a developer

presented why the "company rule" did not work correctly, why this was a problem and identified two potential solutions. [We inserted words in all capitals between square brackets to mark the ending of different parts of the text we took from the archival data]:

> *" The COMPANY rule in StandardTokenizer is defined like this:*
>
> *// Company names like AT&T and Excite@Home. COMPANY = (ALPHA) t ("&"|"@") (ALPHA)*
>
> *While this works perfect for AT&T and Excite@Home, it doesn't work well for strings like widget&javascript&html.* ***[PROBLEM DEFINITION]*** *Now, the latter is obviously wrongly typed, and should have been separated by spaces, but that's what a user typed in a document, and now we need to treat it right....*
>
> *That got me thinking on whether this rule is properly defined, and what's the purpose of it. Obviously, it's an attempt to not break legal company names on "&" and "@", but I'm not sure it covers all company name formats. For example, AT&T can be written as "AT & T" (with spaces) and I've also seen cases where it's written as ATT* ***[PROBLEM DEFINITION]***
>
> *This rule slows StandardTokenizer's tokenization time, and eventually does not produce consistent results.* ***[IDENTIFICATION OF THE NEGATIVE OUTCOME OF THE IDENTIFIED PROBLEM]*** *If we think it's important to detect these tokens, then let's at least make it consistent by either*
>
> *-changing the rule to (ALPHA)(("&"|"@") (ALPHA))+, thereby recognizing "AT&T", and "widget&javascript&html" as COMPANY.* ***. [POTENTIAL SOLUTION-1 WAS PRESENTED]***
>
> *That at least will allow developers to put a CompanyTokenFilter (for example) after the tokenizer to break on "&" and "@" whenever there are more than 2 parts. We could also modify StandardFilter (which already handles ACRONYM) to handle COMPANY that way.* ***[KNOWLEDGE DISSEMINATION ABOUT THE RECOMMENDED SOLUTION WAS PRESENTED]***
>
> *-changing the rule to (ALPHA)("&"|"@") (ALPHA)(*P* | "!" | "?") so that we recognize company names only if the pattern is followed by a space, dot, dash, underscore, exclamation mark or question mark.).* ***[POTENTIAL SOLUTION-2 WAS PRESENTED ABOVE]***
>
> *That'll still recognize AT&T, but won't recognize widget&javascript&html as COMPANY (which is good* ***[KNOWLEDGE DISSEMINATION ABOUT THE RECOMMENDED SOLUTION]***
>
> *What do you think?* ***[EFFORT TO BRAINSTORM POTENTIAL SOLUTIONS]"***

Often the problem resolution seemed to be complex with many dependencies to consider. Similar to the example above, often several solution alternatives were shared, and their pros and cons were discussed to find the most effective and efficient way of resolving the problem. Others contributed to the problem resolution by providing information, such as other aspects to consider, similar to the contribution by another developer here:

> *"COMPANY identifies AT&T, Excite@Home but it also identifies R&D, AD&D, Q&A all are not really COMPANY. So, there's a semantic error in the name of the rule (I know we shouldn't refer to the names too strictly, but still).* ***[KNOWLEDGE CONTRIBUTION FOR PROBLEM RESOLUTION]"***

In this specific instance, the additional input provided by another user completely changed the recommended solution to the removal of the original rule. This shows how Synergistic innovation behaviors helped create better solutions for software problems.

Synergistic open innovation was used to find the most effective and innovative solutions for software development building on knowledge from internal and external community members. Synergistic open

innovation was also used for externalizing one's process-related tacit knowledge; about how OSS development worked in general, and how the given OSS development community operated, more specifically. The archival data provided ample evidence for how developers mentored others on both the software-based knowledge and community processes. An experienced developer mentioned during an interview:

> *"Now I spend most of my time mentoring others. Teaching them about how we do things around here, how to develop better software, etc."*

Information technologies allowed the affordance of instant sharing of knowledge across different time-zones and geographies and building on each other's comments using the threaded discussions on the mailing lists and on the JIRA issue trackers.

Type-III: Peripheral Action-Based Open Innovation

The last type of open innovation is ***peripheral action-based open innovation***. The related behaviors are provided in Table 8. These behaviors do not constitute developing code. They are documentation development activities, which originally seem secondary, and are typically done by peripheral members to learn about the project. Therefore, their name starts with peripheral. They refer to the development of software-related components that contribute to the quality of the software, such as documentation, project Website, and Wiki. While these behaviors may seem like less important efforts, developers considered these as part of action-based leadership (Eseryel, 2014). Therefore, we refer to them as peripheral action-based open innovation.

While the Stable project had almost non-existent peripheral action-based innovation (one instance), 10% (27 instances) of the Delta project constituted peripheral action-based innovation (Table 5).

Writing documentation, such as documenting changes, writing tests, developing web page and wikis were the ways in which the community members internalized the knowledge of others. When read by others, these documentations enabled third parties to internalize knowledge created by community members. Internalization happened through contributions to websites and wikis, which were all submitted through SVN.

Open Innovation Type	Description	Open Innovation Behavior
Peripheral Action-Based Innovation	Action-based behaviors that do not change the code, but that improve the overall product quality.	Documentation: Change log (SVN)
		Documentation: Webpage (SVN)
		Documentation: Wiki (SVN)

Table 8. The Peripheral Action-Based Innovation Behaviors

Peripheral action-based innovation provided newcomers an opportunity to learn by doing. Newcomers learned about the inner workings of the software and the general process by which the community members collaborated when they contributed to documentation, testing, contributing to wiki and Website. They have to learn by doing, which gives them an opportunity to study others' work and thereby transfer their tacit knowledge to build their own tacit knowledge.

Leaders', Developers', and Users' Contributions to the Open Innovation Process

In this section, we describe how community members contributed to open innovation process. We had described having selected two cases that differ in a given way for theoretical replication (Guba & Lincoln, 1994). In theoretical replication, two cases are expected to show different results for predictable reasons. As we expected, we observed the same types and behaviors of open innovation across both cases, which were described in the previous section.

For the contributions of leaders, developers and users to OI, we expect to see differences between two cases for the following reasons: Stable project had two key leaders, one of whom was a founder, and the other a

long term leader. These perceived leaders were actively contributing to the software development and to the community before the interview. Delta project differed in its leadership. People who were in the same two roles (the founder and the other long-term leader) were slowing their participation and one new leader had emerged at the time of the study. Delta project also differed in its periphery, because of a recent influx of new users: While Stable project had a decent sized periphery with 44 active users, Delta had 129 active users in its periphery. For that reason, we expect these two project communities to differ in terms of who participates in what kind of open innovation.

Because Eseryel (2014) found that the contribution patterns of the community members changed between regular events and critical events, we will report the patterns in two different types of events separately.

How Did the STABLE Community Members Contribute to Three Open Innovation Types During the Regular & Critical Events?

Stable Community OI Behaviors During REGULAR Events

Error! Reference source not found.-A and Table 9-A show the number of open innovation behaviors exhibited by leaders, developers, and users (i.e., peripheral members) in the Stable community during the **regular events.** During the regular events, 18% of all community open innovation behaviors were ***action-based*** (21 instances) and 82% were ***synergistic*** (98 instances). Therefore, most of the community OI behaviors were synergistic. The biggest contributors to synergistic OI were the 11 developers (50%).

(A)STABLE COMMUNITY OI BEHAVIORS DURING **REGULAR** EVENTS	LEADERS (2)	Average Leader Contribution	DEVELOPERS (11)	*Average Developer Contribution*	USERS (44)	*Average User Contribution*	Everyone (57)	*Average Contribution By any Participant*
Action-Based	0	0	20	*2*	1	*0*	21	*0*
Synergistic	26	13	49	*4*	23	*1*	98	*2*
Peripheral Action-Based	0	0	0	*0*	0	*0*	0	*0*
TOTAL	**26**	**13**	**69**	***6***	**24**	***1***	**119**	***2***
(B) STABLE COMMUNITY OI BEHAVIORS DURING **CRITICAL** EVENTS	LEADERS (2)	Average Leader Contribution	DEVELOPERS (11)	*Average Developer Contribution*	USERS (44)	*Average User Contribution*	Everyone (57)	*Average Contribution By*
Action-Based	15	8	17	*2*	1	*0*	33	*1*
Synergistic	44	22	41	*4*	6	*0*	91	*2*
Peripheral Action-Based	0	0	1	*0*	0	*0*	1	*0*
TOTAL	**59**	**30**	**59**	***5***	**7**	***0***	**125**	***2***

Table 9. Stable Community OI Behaviors During (A) Regular Events, and (B) Critical Events

Two leaders did not contribute to ***action-based open innovation***, 11 core developers contributed a combined 95% (20 instances), and all 44 users contributed 5% (one instance). Two leaders contributed 27% of all ***synergistic open innovation*** (26 instances), 11 core developers contributed a combined 50% (49 instances), and all 44 users contributed 23% (23 instances). An average leader innovated synergistically 13 times, an average developer 4 times, and an average user only once. Thus, while all developers' combined synergistic OI behaviors were 89% more than that of the leaders (Figure 11-A), on the average, a project leader innovated synergistically about three times as much as an average developer, and 13 times as much as an average user

(Table 9-A). None of the participants contributed to ***peripheral action-based open innovation*** (**Error! Reference source not found.**-A).

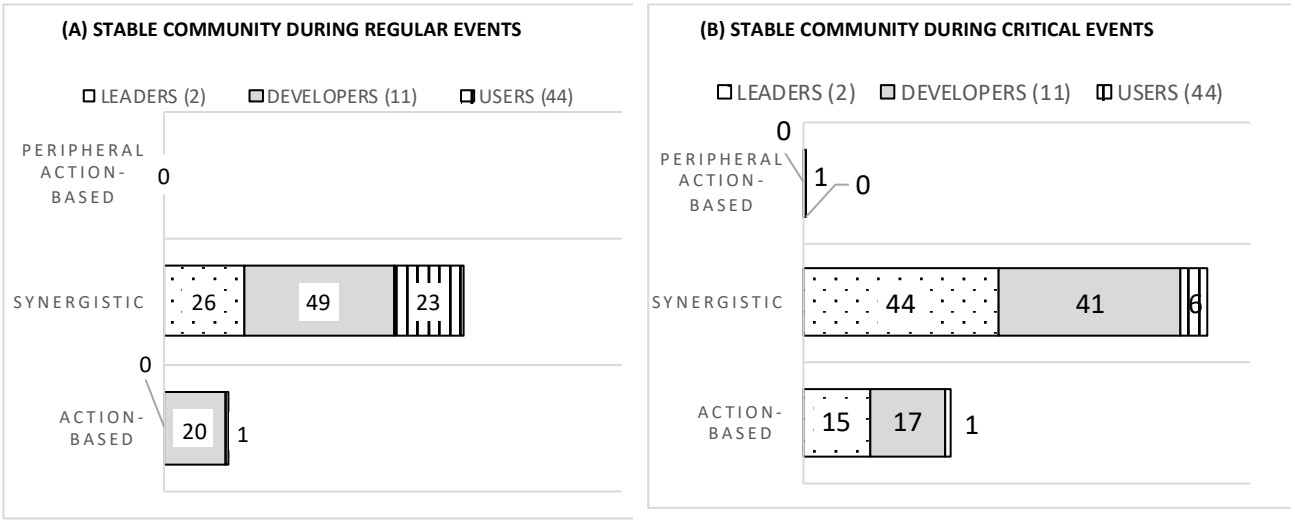


Figure 11. The Distribution of Leadership Behavior Types Among the Stable Community During (A) Regular, and (B) Critical Events

Stable Community OI Behaviors During CRITICAL Events

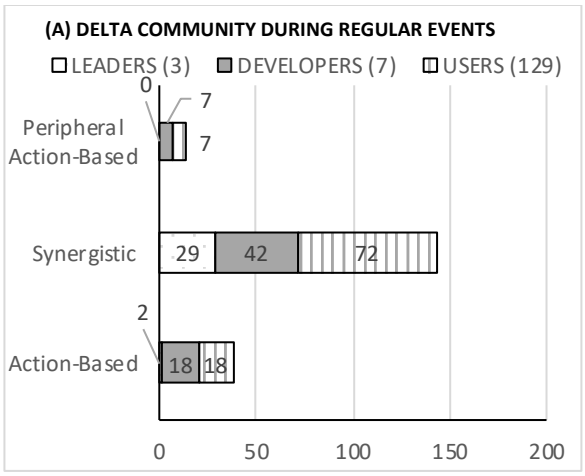
Error! Reference source not found.-B and Table 9-B present the number of open innovation behaviors exhibited by leaders, developers, and users (i.e., peripheral members) in the Stable community during the **critical events.** During the critical events, 26% of all community open innovation behaviors were ***action-based*** (33 instances) and 73% were ***synergistic*** (91 instances), and 1% were ***peripheral action-based open innovation*** (one instance). Most of the OI behaviors were ***synergistic***, similar to that observed during regular events.

Action-based open innovation behaviors were visibly higher (26% versus 18% of all OI) during critical events, compared to the regular events. Differently from the regular periods, two leaders contributed visibly (45%) to ***action-based open innovation*** (15 instances), 11 core developers contributed a combined 52% (one instance), and all 44 users contributed 3% of the action-based innovation (1 instance). Two leaders contributed as much as the 11 core developers (48%, and 45% respectively) to the ***synergistic OI***. 44 users contributed 7% of all synergistic open innovation (6 instances). **Error! Reference source not found.**-B shows that an average leader innovated synergistically 22 times, an average developer 4 times, and an average user negligibly (0.1 times). Thus, while all developers' combined synergistic OI behaviors were only 7% less than that of the leaders (Figure 11-A), on the average, a project leader innovated synergistically about six times as much as an average developer, and 220 times as much as an average user. Only one instance of ***peripheral action-based open innovation*** was contributed, which was by a developer (**Error! Reference source not found.**-B).

How Did the DELTA Community Members Contribute to Three Open Innovation Types During the Regular & Critical Events?

This section discusses the behaviors during regular and critical events.

DELTA Community OI Behaviors During REGULAR Events



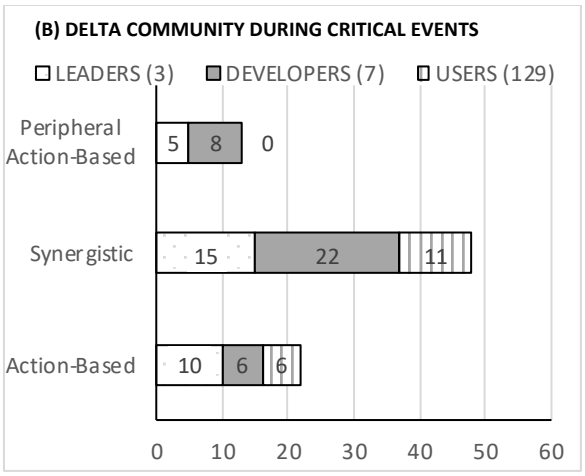


Figure 12-A and Table 10-A show the number of open innovation behaviors exhibited by leaders, developers, and users (i.e., peripheral members) in the Delta community during the **regular events**. During the regular events, 19% of all community open innovation behaviors were ***action-based*** (38 instances) and 73% were ***synergistic*** (143 instances), and 7% were ***peripheral action-based*** (14 instances). Therefore, most of the community OI behaviors were synergistic. The biggest contributors to synergistic OI were interestingly the 129 users (50%).

(A) DELTA COMMUNITY OI BEHAVIORS DURING **REGULAR** EVENTS	LEADERS (3)	Average Leader Contribution	DEVELOPERS (7)	*Average Developer Contribution*	USERS (129)	*Average User Contribution*	Everyone (139)	*Average Contribution by any Participant*
Action-Based	2	1	18	*2*	18	*0.4*	38	*1*
Synergistic	29	15	42	*4*	72	*2*	143	*3*
Peripheral Action-Based	0.0	0.0	7	*1*	7	*0.2*	14	*0.2*
TOTAL	**31**	**10**	**67**	***10***	**97**	***1***	**195**	***1***
(B) DELTA COMMUNITY OI BEHAVIORS DURING **CRITICAL** EVENTS	LEADERS (3)	Average Leader Contribution	DEVELOPERS (7)	*Average Developer Contribution*	USERS (129)	*Average User Contribution*	Everyone (139)	*Average Contribution by any Participant*
Action-Based	10	5	6	*1*	6	*0.1*	22	*0.4*
Synergistic	15	8	22	*2*	11	*0.3*	48	*1*
Peripheral Action-Based	5	3	8	*1*	0	*0.0*	13	*0.2*

TOTAL	**30**	**10**	**36**	***5***	**17**	***0.1***	**83**	***1***

Table 10. Delta Community OI Behaviors During (A) Regular Events, and (B) Critical Events

Three leaders contributed 5% of the ***action-based open innovation*** (18 instances), 7 developers contributed a combined 47% (18 instances), and 129 users contributed 47% (18 instances). Three leaders contributed 20% of all ***synergistic open innovation*** (29 instances), 7 developers contributed a combined 29% (42 instances), and all 129 users contributed 50% (72 instances). Developers and users shared the ***peripheral action-based open innovation*** equally at 50% (7 instances each). During regular events an average leader innovated action-based once, an average developer twice, and an average user negligibly (o.4 times). Thus, while all developers' combined action-based OI behaviors were nine times those of the leaders, an average developer innovated action-based about only twice as much as an average leader. Similarly, an average leader innovated action-based about 2.5 times as much as an average user (Table 10-A).

During regular events an average leader innovated synergistically 15 times, an average developer four times, and an average user twice. Thus, while all developers' combined synergistic OI behaviors were 45% more than that of the leaders, on the average a project leader innovated synergistically about four times as much as an average developer, and about eight times as much as an average user (Table 10-A).

During regular events an average leader did not contribute to peripheral action-based OI. An average developer contributed to peripheral action-based OI once and an average user negligibly (0.2 times). Thus, while all developers' combined peripheral action-based OI behaviors were the same as that of the users, on the average a developer innovated peripherally about five times as much as an average user (Table 10-A).

In total, during regular events across all community members, we observed visibly higher synergistic OI behaviors (143 instances) than action-based (38). Action based OI behaviors were followed by peripheral action-based OI behaviors (14 instances).

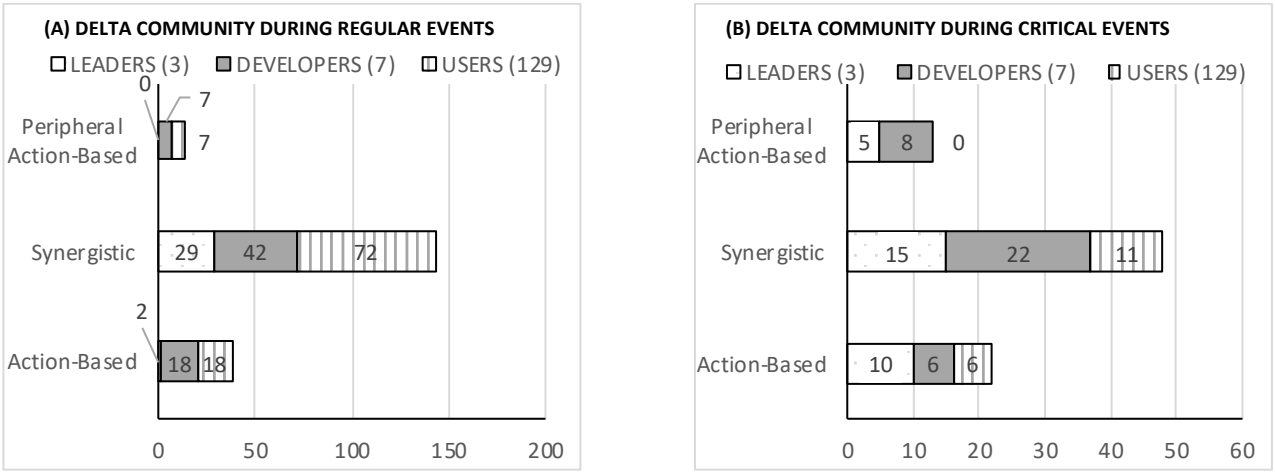
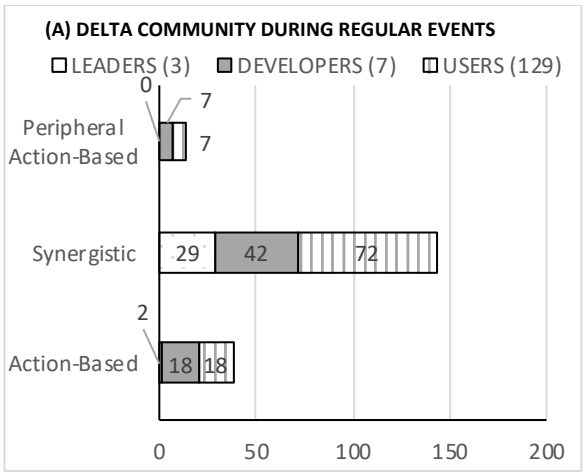


*Figure 12. The Distribution of Leadership Behavior Types Among the **Delta** Community Leaders, Developers, and Users During (A) Regular Events, and (B) Critical Events*

DELTA Community OI Behaviors During CRITICAL Events



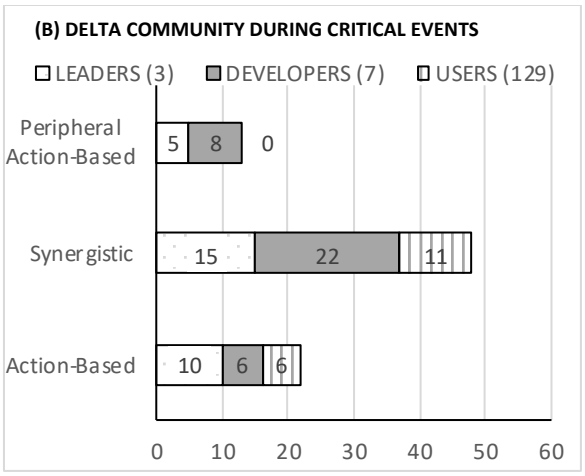


Figure 12-B and Table 10-B present the number of open innovation behaviors exhibited by leaders, developers, and users (i.e., peripheral members) in the Delta community during the **critical events.** During the critical events, 27% of all community open innovation behaviors were ***action-based*** (22 instances) and 58% were ***synergistic*** (48 instances), and 16% were ***peripheral action-based open innovation*** (13 instances). Most of the OI behaviors were ***synergistic***, similar to that observed during regular events.

Action-based open innovation behaviors were visibly higher (27% versus 19% of all OI) during critical events, compared to the regular events. Three leaders contributed visibly more (45% of all action-based OI) to ***action-based open innovation*** (10 instances) in the critical events compared to the regular events (5% of all action-based OI, 2 instances). Seven developers and 129 users each contributed about 27% each (6 instances each) to action-based OI during critical events.

Differently from the regular events, both the leaders and the developers increased their contribution to the ***synergistic OI.*** Three leaders contributed 31% (15 instances), and 7 developers contributed 46% (22 instances) to the synergistic OI. 129 users contributed 23% of all synergistic open innovation (11 instances). Leaders contributed 38% (5 instances) of the ***peripheral action-based open innovation***, and the developers contributed the remaining 62% (22 instances).

During critical events an average Delta leader innovated action-based five times, an average developer once, and an average user negligibly (o.1 times). Thus, while all leaders' combined action-based OI behaviors were 1.6 times those of the developers, and 1.6 timers those of the users, an average leader innovated action-based five times that of an average developer and 50 times that of an average user (Table 10-B).

During critical events an average leader innovated synergistically eight times, an average developer twice and an average user negligibly (0.3 times). Thus, while all developers' combined synergistic OI behaviors were 45% more than that of the leaders, on the average a project leader innovated synergistically about four times as much as an average developer, and about 26 times as much as an average user (Table 10-B).

During critical events an average leader innovated peripheral action-based three times, an average developer once, and an average user did not contribute to peripheral OI. (Table 10-A).

In total, the most observed OI behavior during critical events was synergistic (48 instances), followed by action-based (22 instances). The least observed OI behavior was peripheral action-based (13 instances).

How do the Two Community Members Compare to Each Other Across All Event Types?

Error! Reference source not found. and Table 10 present the OI behaviors of respectively the Stable project's and Delta project's community members across both regular (in part A of both tables) and critical events (in part B of both tables). Stable members innovated mostly synergistically across both event types.

Community Similarities in Action-Based OI Behaviors

Table 11 presents action-based OI behaviors in both communities. In both communities and in both event types, action-based OI behaviors come second to synergistic OI behaviors.

Across both projects, leaders contributed minimally to action-based OI during regular events (0 for Stable, and 5% for Delta). Yet, their contribution to action-based OI during critical events was substantive (46% versus 45% of all community's action-based OI during critical period was contributed by the Stable and Delta leaders respectively). Both projects' developers contributed substantively to action-based OI during regular events, yet the developers' total contribution to action-based OI during the critical events were almost halved (45% reduction in Stable, and 43% reduction in Delta). Both teams' user contribution to action-based OI dropped by about the same percentage. (40% in Stable, and 42% in Delta)

OI Type	Community	Event Type	Leader Contributions (# of Events)	% of All Team's Contribution	Developer Contribution (# of Events)	% of All Team's Contribution	User Contribution (#of events)	% of All Team's Contribution	*Total Contribution*
Action-Based OI	**STABLE**	Regular	0.0	0	20.0	95%	1.0	5%	*100%*
		Critical	15.0	46%	17.0	52%	1.0	3%	*100%*
	DELTA	Regular	2.0	5%	18.0	47%	18.0	47%	*100%*
		Critical	10.0	45%	6.0	27%	6.0	27%	*100%*

Table 11. Action-Based OI Behaviors in Both Communities

Community Differences in Action-Based OI Behaviors

Delta community seems to have a stronger contribution from users. Delta has 2.9 times the user size of Stable (129 users versus 44 users). Further, the Delta users contributed substantively to action-based OI compared to Stable users: During regular events, Stable users contributed to 1 event (5% of all members' contributions). Delta users contributed to 18 regular events (47% of all members' contributions). Both community users' contribution dropped by about the same percentage, as mentioned in the previous section. Yet, Delta user contribution to critical events still constituted 27% of all members. This is significant because the Delta developers also contributed 27%. As a result, while both Stable and Delta leaders contributed very similarly to regular and critical event action-based OI, in stable, the remainder is distributed differently in both communities: In Stable, remainder of the action-based OI was covered by the 44 developers (95% in regular, and 52% in critical events). Yet in Delta, the action-based OI was split evenly between developers and users. During the regular events, both Delta developers and users contributed 47% each, and during critical events, both Delta developers and users contributed 27% each.

OI Type	Community	Event Type	Leader Contributions (# of Events)	Contribution Per Leader	% of All Team's Contribution	% Contribution by Avg Leader
Synergistic OI	**STABLE**	Regular	26	13	27%	13.27%
		Critical	44	22	48%	24.18%
	DELTA	Regular	29	10	20%	6.76%
		Critical	15	5	31%	10.42%
OI Type	Community	Event Type	Developer Contribution (# of Events)	Contribution Per Developer	% of All Team's Contribution	% Contribution by Avg Developer
Synergistic OI	**STABLE**	Regular	49	4	50%	0%
		Critical	41	4	45%	0%
	DELTA	Regular	42	6	29%	0%
		Critical	22	3	46%	0%
OI Type	Community	Event Type	User Contribution (#of events)	Contribution Per User	% of All Team's Contribution	% Contribution by Avg User
Synergistic OI	**STABLE**	Regular	23	1	23%	1%
		Critical	6	0	7%	0%
	DELTA	Regular	72	1	50%	0%
		Critical	11	0	23%	0%

Table 12.Synergistic OI Behaviors in Both Communities

Community Similarities in Synergistic OI Behaviors

In both communities, synergistic OI behaviors were the most often observed OI behaviors in both type of events (Table 12). In both communities, and in both event types an average leader contributes more than an average developer, and an average developer contributes more than an average user to synergistic OI. In both communities, an average user contributes to one synergistic OI to a regular event, but negligibly to a critical event. In both communities, all the leaders combined provide more of the synergistic OI to the critical events than they do to the regular events. This is reversed for users: all community users combined provide more of the synergistic OI to the regular events than they do to the critical events.

The difference in the % contribution to synergistic OI between two communities' developers and users stem from the number of active developers and users. While in both communities, the average user contributed less than 0.5% of the synergistic OI behaviors, Delta users contribute 50% of all community synergistic OI, and Stable users contribute 23%. This is because Delta has 239 users and Stable has 44. We see the same pattern in developers' contribution to synergistic OI. An average developer provides synergistic OI to three to six events. Yet, Stable has 11 developers, who contribute 50% of the regular event synergistic OI. Since Delta has only 7 developers, they contribute 29% to the regular event synergistic OI.

Comparison of Peripheral Action-Based OI Behaviors

Table 13 shows a comparative view of peripheral action-based OI behaviors. In both communities and in both type of events, peripheral action-based OI behaviors are the least used among the three types of OI behaviors. Secondly, the group that contributed the most to peripheral OI behaviors is the developers.

OI Type	Community	Event Type	Leader Contributions (# of Events)	% of All Team's Contribution	Developer Contribution (# of Events)	% of All Team's Contribution	User Contribution (#of events)	% of All Team's Contribution	*Total Contribution*
Peripheral Action-Based	**STABLE**	Regular	0.0	0%	0.0	0%	0.0	0%	*0%*
		Critical	0.0	0%	1.0	100%	0.0	0%	*100%*
	DELTA	Regular	0.0	0%	7.0	50%	7.0	50%	*100%*
		Critical	5.0	38%	8.0	62%	0.0	0%	*100%*

Table 13. Peripheral Action-Based OI Behaviors in Both Communities

However, Delta community shows some similarity to the pattern that community members exhibited related to action-based OI: First, leaders increased their contribution between regular and critical events from zero to 38%. Second, the developers' and users' contribution to the regular events are the same.

DISCUSSION

This paper contributed to opening the black box of open innovation process. Specifically, this study addressed how open innovation community leaders, members and extended community contribute to open innovation through information technologies. We next present theoretical and practical contributions of our findings.

Contributions to Theory

This study helped us gain additional insights in the form of a fine-grained understanding of how community members at the core and the periphery of the OI communities innovated.

Our study confirmed the previous literature which suggested that perceived leaders contribute directly to software development in the form of work, and as a result they increased software development effectiveness and transformed technology vision (Eseryel & Eseryel, 2013). It extended these findings by suggesting that the leaders' contributions further increase the innovativeness of the project. Leaders contribute to OI both with their individual efforts, and by synergistically working with developers and users to innovate.

We confirmed that the users contribute specifically to improving software quality with patches, and bug reports (Setia, et al., 2012), which allows the users to socialize and learn the ropes (Eseryel, 2014). We extend these findings to suggest that users also contribute to open innovation. Moreover, our study provides a better understanding of how users innovate, and their overall contribution to the community regarding OI: First, users contribute mostly to the synergistic OI, meaning users innovate mostly in collaboration with others. It also means that users participate mostly in identifying problems and brainstorming solutions, by providing more information, or offering potential solutions. The second most frequently used OI type by users was action-based OI, which includes their direct contribution to the software code. Users contributed more to regular events than critical events across all OI types. This is probably due to users having limited knowledge and expertise to solve complex problems. Based on extant research, we had expected (1) users to exhibit peripheral OI behaviors the most, and (2) that most of the peripheral OI to be contributed by the users. Our findings were quite the opposite: Peripheral OI was the least frequently used OI behavior type by users. Moreover, most peripheral OI was contributed by developers.

Eseryel (2014) had suggested that even small numbers of innovative behavior contributed by each of the extended community members mattered. She observed that especially for the communities with large user bases, small contributions may add up to 50% the knowledge creation (Eseryel, in 2014). While the two communities in this study differed in their contributions to innovation, this study still supported the general observation: Keeping many users involved in the project makes a difference as we found in our

analysis of synergistic OI. Even though an average Delta user contributed less than 0.5% of synergistic OI, Delta users' combined contribution constituted half of all regular event synergistic OI. We also extend this observation about users to developers: Many qualified developers visibly increase open innovation.

In both communities, and across both types of events, the most frequently used OI type was synergistic, followed by action-based OI. The peripheral action-based OI was the least frequently used OI type. This brings a very interesting nuance to the literature because "overwhelming majority of work in Open Source [was] accomplished with only one developer working on any one task (Howison & Crowston, 2014, p.29). Individuals emerged as leaders based on their action-based work (Eseryel & Eseryel, 2013). Similarly, in analyzing OSS decision-making processes, Eseryel et al. (2020) found that 23% of all decisions followed a short-cut process, which meant that a person identified the problem and solution at once without involving other developers in the decision-making process. Yet, despite all these, most OSS open innovation takes place synergistically.

We found that generally leaders focused their innovation efforts on the critical events. This may likely be due to the leaders' high level of expertise, and their ability and interest in resolving these issues. The leaders showed a general trend of a visible decrease in innovation in general and action-based innovation during the regular events. Previous literature had created an expectation of high action-based innovation by leaders (Eseryel & Eseryel, 2013), not accounting for the differences in the types of software development activity (critical versus regular events).

The gap in innovation, caused by lessened activity of leaders during the regular events is often filled by the developers and extended community members. It may be that the leaders purposefully reduce their innovations in regular events to encourage the inputs of the rest of the community members. Alternatively, the high level of existing community innovation during the regular events may be enabling the leaders to focus their innovation efforts to more critical events.

Another contribution to theory is adaptation of Nonaka and Takeuchi's (1995) framework in open innovation settings. Despite the abundance of literature that built on Nonaka and Takeuchi's (1995) work, empirical operationalization of the knowledge creation modes was rather rare (e.g., Nonaka, Byosiere, Borucki, & Konno, 1994), with few exceptions such as Eseryel's (2014) study on IT-enabled knowledge creation. Our study builds and extends the work of Nonaka and Takeuchi (1996) and Eseryel (2014) to capture open innovation behaviors within IT-enabled open innovation communities for software development.

Contributions to Practice

This paper contributes to the practice by explicating how open innovation communities innovate through information and communication technologies. The implication of this study for information systems managers and practitioners, who incorporate OSS development teams into their organizational practice, is the need for: (1) supporting the existing users and encouraging them to stay involved, in an effort to keep the number of external knowledge contributors high; (2) keeping a large number of highly skilled developers who can deal with critical events.; and (3) getting the extended community members more active in the regular events, especially with regard to action-based innovation.

The types of open innovation identified here may inform which skills these organizations focus on during recruitment and training of team members (such as action focus, software development skills, strong documentation skills, and ability to communicate one's ideas clearly to other developers and users).

The managers/leaders should be aware that by investing in developers and users, it is possible to increase the group innovativeness. Leaders could also use OSS leaders' tactics that encourage community participation in OI and tackle critical issues: While we recommend that leaders be involved in all three types of OIs, they can focus their activities mostly on critical events, therefore helping solve issues where their expertise is needed. By limiting their involvement in regular events strategically, they can allow others to participate in OI, therefore increasing the innovation as a whole.

Lastly, this research showed that part of the innovation activities involve interaction with the IT artifact (JIRA, SVN, mailing lists) to produce and IT innovation outcome. Therefore, the development of strong IT skills is key to open innovation communities. Furthermore, having a variety of information technologies with features that enable all three types of innovation is the basis for a healthy open innovation community.

Limitations and Future Research

A major limitation of this study is that it analyzes data at a single point in time using one set of interviews, and archival data preceding that period. This limitation exists in many studies on open innovation communities. Yet, open innovation communities allow for major inflow and outflow of peripheral members. They further allow the peripheral members to become developers, or even leaders over time. Major changes in the community may cause the community to be much more innovative. Major changes may also cause the community to be much less innovative. Therefore, cross-sectional studies are needed to identify how community changes influence open innovative behaviors of users, developers, and leaders.

Secondly this study did not allow for a strong comparison across communities for peripheral action-based OI. This is because Stable community did not use peripheral OI during our study except for one event. This is interesting because interviews with Stable showed that they really appreciated peripheral action-based innovation. Investigating the Stable community during another time-period may allow for a better comparison, if that time period is one where peripheral OI is frequently used.

This paper presented findings from two comparative case studies. The findings of this study can thus be generalized to similar contexts, where the task is highly technical and knowledge-intensive, where the team is distributed globally and highly dependent on information technologies. The findings of this study can also be generalized to the companies that are using open source practices inside their organization.

References

Afuah, A. (2003). Innovation Management: Strategies, Implementation, and Profits (2nd ed.). Oxford University Press.

Ali, A., Krapfel, R., & LaBahn, D. (1995). Product innovativeness and entry strategy: Impact on cycle time and break-even time. *Journal of Product Innovation Management, 12*, 54-69.

AlMarzouq, M., Zheng, L., Rong, G., & Grover, V. (2005). Open source: Concepts, benefits, and challenges. *Communications of the Association for Information Systems, 16*(37), 756-784.

Atuahene-Gima, K. (1995). An exploratory analysis of the impact of market orientation on new product performance: A contingency approach. *Journal of Product Innovation Management,12*, 275-293.

Banker, R. D. (2007, April 09). [Email exchange with the author on research method used for the article "The Evolution of Research on Information Systems: A Fiftieth-Year Survey of the Literature in Management Science.]

Bolici, F., Howison, J., & Crowston, K. (2016). Stigmergic coordination in FLOSS development teams: Integrating explicit and implicit mechanisms. *Cognitive Systems Research*, *38*, 14-22. https://doi.org/10.1016/j.cogsys.2015.12.003.

Bonaccorsi, A., & Rossi, C. (2006). Comparing motivations of individual programmers and firms to take part in the open source movement: From community to business. *Knowledge, Technology & Policy, 18*(4), 40-64.

Carlsson, B. (2007). Innovation systems: a survey of the literature from a Schumpeterian perspective. In H. Hanusch & A. Pyk (Eds.), *Elgar companion to neo-Schumpeterian economics* (Vol. 857, pp. 857-871). Edward Elgar Publishing Limited.

Chandy, R. K., & Tellis, G. J. (2000). The incumbent's curse: incumbency, size, and radical product innovation. *Journal of Marketing, 64*, 1-17.

Chesbrough, H. M. (2003). *Open Innovation*. Boston, MA: Harvard Business School Press.

Colarelli O'Connor, G. (1998). Market learning and radical innovation: A cross case comparison of eight radical innovation projects. *Journal of Product Innovation Management, 15*(2), 151-166.

Comino, S., & Manenti, F. M. (2005). Government policies supporting open source software for the mass market. *Review of Industrial Organization, 26*(2), 217-240.

Conboy, K., & Morgan, L. (2011). Beyond the customer: Opening the agile systems development process. *Information and Software Technology, 53*(5), 535-542. http://dx.doi.org/10.1016/j.infsof.2010.10.007

Cook, I., & Horobin, G. (2006). Implementing eGovernment without promoting dependence: opensource software in developing countries in Southeast Asia. *Public Administration and Development, 26*(4), 279-289. https://doi.org/10.1002/pad.403

Cooper, R. G. (1979). The dimensions of industrial new product success and failure. *Journal of Marketing, 43*, 93-103.

Cooper, R. (1998). Benchmarking new product performance: Results of the best practices study. *European Management Journal*, *16*(1), 1-17. https://doi.org/10.1016/S0263-2373(97)00069-8

Cooper, R. G., & de Brentani, U. (1991). New industrial financial services: What distinguishes the winners. *Journal of Product Innovation Management, 8*, 75-90.

Corbin, J., & Strauss, A. (2008). *Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory* (3rd ed.). Thousand Oaks, CA: Sage.

Cox, A. (1998). Cathedrals, bazaars and the town council. Retrieved from http://slashdot.org/features/98/10/13/l423253.shtml

Crossan, M. M., & Apaydin, M. (2010). A multi-dimensional framework of organizational innovation: A systematic review of the literature. *Journal of Management Studies, 47*(6), 1154-1191.

Crowston, K., & Annabi, H. (2005). *Effective work practices for FLOSS development: A model and propositions.* Paper presented at the In Thirty-Eighth Hawaii International Conference on System Science (HICSS-38). Kona, HI.

Crowston, K., & Howison, J. (2005). The social structure of free and open source software development. *First Monday, 10*(2). http://firstmonday.org/htbin/cgiwrap/bin/ojs/index.php/fm/article/viewArticle/1207/1127

Crowston, K., & Howison, J. (2006). Hierarchy and centralization in free and open source software team communications. *Knowledge, Technology & Policy, 18*(4), 65-85.

Crowston, K., Howison, J., Masango, C., & Eseryel, U. Y. (2005, 7-10 August). *Face-to- face interactions in self-organizing distributed teams.* Paper presented at the OCIS Division, Academy of Management Conference, Honolulu, HI.

Crowston, K., Howison, J., Masango, C., & Eseryel, U. Y. (2007). The Balancing Act: The role of face-to-face meetings in technology-supported self-organizing distributed teams. *IEEE Transactions on Professional Communications*, *50*(3), 185-203. https://doi.org/10.1109/TPC.2007.902654

Crowston, K., Li, Q., Wei, K., Eseryel, U. Y., & Howison, J. (2007). Self-organization of teams in free/libre open source software development. *Information and Software Technology Journal, Special Issue on Qualitative Software Engineering Research*, *49*(6), 564-575. https://doi.org/10.1016/j.infsof.2007.02.004

Davenport, T. H. (1997). Knowledge management at Ernst & Young. *Knowledge Management Case Study.* Retrieved from http://www.bus.utexas.edu/kman/e_y.htm

Dinkelacker, J., Garg, P. K., Miller, R., &Nelson, D. (2002). *Progressive open source.* Paper presented at the Proceedings of the International Conference on Software Engineering (ICSE '02), Orlando.

Ducheneaut, N. (2005). Socialization in an open source software community: A socio-technical analysis. *Computer Supported Cooperative Work, 14*(4), 323-368.

Durmusoglu, S. S., & Barczak, G. (2011). The use of information technology tools in new product development phases: Analysis of effects on new product innovativeness, quality, and market performance. *Industrial Marketing Management, 40*(2), 321-330.

Eisenhardt, K. M. (1989). Building Theories from Case Study Research. *Academy of Management Review, 14*(4), 532-550.

Eseryel, U. Y. (2013). Information Technology Self-Leadership. Proceedings of the Nineteenth American Conference on Information Systems,

Eseryel, U. Y. (2014). IT-enabled knowledge creation for open innovation. *Journal of the Association for Information Systems*, *15*(11), 805-834. https://doi.org/10.17705/1jais.00378

Eseryel, U. Y., & Biernath, P. (2024). The influence of transformational IT leadership on the IT leadership of followers. *Journal of Leadership and Management*, *10*(1), 11-29.

Eseryel, U. Y., & Eseryel, D. (2013). Action-embedded transformational leadership in self-managing global information technology teams. *The Journal of Strategic Information Systems*, *22*(2), 103-120. https://doi.org/10.1016/j.jsis.2013.02.001

Eseryel, U. Y., Wei, K., & Crowston, K. (2020). Decision-making processes in community-based free/libre open source software development teams with internal governance: An extension to decision-making theory. *Communications of the AIS*, *46*, 484-510. https://doi.org/10.17705/1CAIS.04620

Fielding, R. T. (1999). Shared leadership in the Apache project. *Communications of the ACM, 42*(4), 42-43.

Gacek, C., Lawrie, T., & Arief, B. (2001). *The many meanings of open source* (1). Retrieved from http://www.dirc.org.uk/ publications/techreports/papers/1.pdf

Garcia, R., & Calantone, R. (2002). A critical look at technological innovation typology and innovativeness terminology: a literature review. *Journal of Product Innovation Management,19*(2), 110-132. https://doi.org/10.1111/1540-5885.1920110

Gassmann, O., & Enkel, E. (2001). Towards a theory of open innovation: three core process archetypes. Retrieved from http://www.alexandria.unisg.ch/Publikationen/274

Goldenberg, J., Lehmann, D. R., & Mazursky, D. (1999). *The primacy of idea itself as a predictor of new product success*. Marketing Science Institute Working Paper.

Goldman, R., & Gabriel, R. P. (2005). *Innovation Happens Elsewhere: Open Source as Business Strategy*. San Francisco: Morgan Kaufmann.

Green, S. G., Gavin, M. B., & Aiman-Smith, L. (1995). Assessing a multidimensional measure of radical technological innovation. *IEEE Transactions on Engineering Management, 42*(3), 204-213.

Guba, E. G., & Lincoln, Y. S. (1994). Competing paradigms in qualitative research. In N. K. Denzin & Y. S. Lincoln (Eds.), *Handbook of Qualitative Research* (pp. 105-117). Thousand Oaks: Sage.

Haefliger, S., von Krogh, G., & Spaeth, S. (2008). Code Reuse in Open Source Software. *Management Science, 54*(1), 180-193.
Hare, A. P. (1976). *Handbook of small group research* (2nd ed.). New York: Free Press. Hars, A., & Ou, S. (2001). *Working for Free? - Motivations of Participating in Open Source*
Projects. Paper presented at the Proceedings of the 34th Hawaii International Conference on System Sciences.
Heckman, R., & Misiolek, N. I. (2005). *Leaders and followers in student online project teams*. Paper presented at the 38th Hawaii International Conference on System Sciences. https://csdl2.computer.org/comp/proceedings/hicss/2005/2268/01/22680004c.pdf
Hedgebeth, D. (2007). Gaining competitive advantage in a knowledge-based economy through the utilization of open source software. *VINE, 37*(3), 284-294.
Helfat, C. E., & Peteraf, M. A. (2003). The dynamic resource-based view: Capability lifecycles. *Strategic Management Journal, 24*(10), 997-1010.
Hemetsberger, A., & Reinhardt, C. (2006). Learning and Knowledge-building in Open-source Communities: A Social-experiential Approach. *Management Learning, 37*(2), 187-214. https://doi.org/10.1177/1350507606063442
Hermann, S., Hertel, G., & Niedner, S. (2000). Linux Study Homepage. Retrieved from www.psychologie.uni-kiel.de/linux-study/writeup.html
Himanen, P., Torvalds, L., & Castells, M. (2001). The Hacker Ethic. New York: Random House.
Howison, J., & Crowston, K. (2014). Collaboration through open superposition: A theory of the open source way. *MIS Quarterly*, *38*(1), 29-50. https://www.jstor.org/stable/26554867
Jansen, D. S., & Saiedian, H. (2006). Test-driven learning: intrinsic integration of testing into the CS/SE curriculum. Paper presented at the Proceedings of the 37th SIGCSE technical symposium on Computer science education, Houston, Texas, USA.
Jha, S. R. (2002). Reconsidering Michael Polanyi's *Philosophy*. Pittsburgh: University of Pittsburgh Press.
Kautz, K., & Thaysen, K. (2001). Knowledge, learning and IT support in a small software company. *Journal of Knowledge Management, 5*(4), 349-357.
Kleis, L., Chwelos, P., Ramirez, R. V., & Cockburn, I. (2012). Information technology and intangible output: The impact of IT investment on innovation productivity. *Information Systems Research, 23*(1), 42-59.
Kogabayev, T., & Maziliauskas, A. (2017). The definition and classification of innovation. *HOLISTICA–Journal of Business and Public Administration*, *8*(1), 59-72. https://doi.org/10.1515/hjbpa-2017-0005
Kogut, B. (2000). The network as knowledge: Generative rules and the emergence of structure. *Strategic Management Journal, 21*(405-425).
Kohanski, D. (1998). *Moths in the Machine*. New York: St. Martin's Griffin.
Lapointe, L., & Rivard, S. (2005). A multilevel model of resistance to information technology implementation. *MIS Quarterly, 29*(3), 461-491.
Lee, G. K., & Cole, R. E. (2003). From a Firm-Based to a Community-Based Model of Knowledge Creation: The Case of the Linux Kernel Development. *Organization Science, 14*(6), 633-649.
Lee, M., & Na, D. (1994). Determinants of technical success in product development when innovative radicalness is considered. *Journal of Product Innovation Management, 11*, 62-68.
Ljungberg, J. (2000). Open source movements as a model for organizing. *European Journal of Information Systems, 9*(4), 208-216.
Lundell, B., Lings, B., & Syberfeldt, A. (2011). Practitioner perceptions of open source software in the embedded systems area. *Journal of Systems and Software, 84*(9), 1540-1549. https://doi.org/10.1016/j.jss.2011.03.020
Miles, M. B., & Huberman, A. M. (1994). *Qualitative Data Analysis* (2 ed.). Thousand Oaks, CA: Sage.
Mockus, A., Fielding, R. T., & Herbsleb, J. D. (2002). Two case studies of Open Source Software development: Apache and Mozilla. *ACM Transactions on Software Engineering and Methodology, 11*(309-346).
More, R. A. (1982). Risk factors in accepted and rejected new industrial products. *Industrial Marketing Management, 11*, 9-15.
Morner, M., & von Krogh, G. (2009). A Note on Knowledge Creation in Open-Source Software Projects: What Can We Learn from Luhmann's Theory of Social Systems? *Systemic Practice and Action Research, 22*(6), 431-443.
Morse, J. M. (2004). Theoretical saturation. In M. S. Lewis-Beck, A. Bryman, & T. F. Liao (Eds.), *Encyclopedia of Social Science Research Methods* (Vol. 1-3). Thousand Oaks, CA: Sage Publications.
Nambisan, S., & Sawhney, M. (2007). *The Global Brain: Your Roadmap for Innovating Faster and Smarter in a Networked World*. Philadelphia: Wharton School Publishing. Neuendorf, K. A. (2002). *The content analysis guidebook*. Thousand Oaks, CA: Sage Publications.
Nonaka, I., Byosiere, P., Borucki, C. C., & Konno, N. (1994). Organizational knowledge creation theory: A first comprehensive test. *International Business Review, 3*(4), 337- 351.
Nonaka, I., & Takeuchi, H. (1995). *The Knowledge-Creating Company: How Japanese Companies Create the Dynamics of Innovation*. New York: Oxford University Press, inc.

Nonaka, I., & von Krogh, G. (2009). Tacit Knowledge and Knowledge Conversion: Controversy and Advancement in Organizational Knowledge Creation Theory. *Organization Science, 20*(3), 635-652.

Pavlou, P., & Sawy, O. (2006). From IT leveraging competence to competitive advantage in turbulent environments: The case of new product development. *Information Systems Research, 17*(3), 198-220.

Pawlowski, S. D., & Robey, D. (2004). Bridging User Organizations: Knowledge Brokering and the Work of Information Technology Professionals. *MIS Quarterly, 28*(4), 645- 672.

Pettigrew, A. M. (1990). Longitudinal field research on change: theory and practice. *Organization Science, 1*(3), 267-292.

Potter, W. J., & Levine-Donnerstein, D. (1999). Rethinking validity and reliability in content analysis. *Journal of Applied Communication Research, 27*, 258-284.

Sarker, S., Grewal, R., & Sarker, S. (2002). *Emergence of leaders in virtual teams: what matters?* Paper presented at the Proceedings of the 35th Hawaii International Conference on System Sciences

Scozzi, B., Crowston, K., Eseryel, U. Y., & Li, Q. (2008, Jan 7-10). *Shared mental models among open source software developers.* Paper presented at the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008), Hawaii.

Setia, P., Rajogopalan, B., & Sambamurthy, V. (2012). How peripheral developers contribute to open-source software development. *Information Systems Research, 23*(1), 144- 163.

Souder, W. E., & Song, M. X. (1997). Contingent product design, and marketing strategies influencing new product success, and failure in US, and Japanese electronic firms. *Journal of Product Innovation Management, 14*, 21-34.

The Apache Software Foundation. (2012). The Apache Software Foundation Board of Directors. Retrieved from https://www.apache.org/foundation/board/

Thomas, D., & Hunt, A. (2004). Open source ecosystems. *IEEE Software, 32*(1), 89-91. Trott, P., & Hartmann, D. (2009). Why 'open innovation' is old wine in new bottles. *International Journal of Innovation Management, 13*(4), 715-736.

Twiss, B., & Goodridge, M. (1989). Managing Technology for Competitive Advantage. Trans-Atlantic Publications.

Urabe, K. (1988). Innovation and Management: International Comparison. Walter De Gruyter.

von Hippel, E., & von Krogh, G. (2003). Open source software and the 'private-collective' model: Issues for organization science. *Organization Science, 14*(2), 209-223.

von Krogh, G. (2002). The communal resource and information systems. *Journal of Strategic Information Systems, 11*(2), 85-107.

von Krogh, G., & Spaeth, S. (2007). The open source software phenomenon: characteristics that promote research. *Journal of Strategic Information Systems, 16*(3), 236-253. https://doi.org/10.1016/j.jsis.2007.06.001

von Krogh, G., Spaeth, S., & Lakhani, K. R. (2003). Community, joining, and specialization in opensource software innovation: a case study. *Research Policy, 32*(7), 1217-1241.

Wasserman, A. I. (2009). Building a business of open source software. In C. Petti (Ed.), *Cases in Technological Entrepreneurship* (pp.107-121). Edward Elgar Publishing.

Weber, S. (2004). *The Success of Open Source*: Harvard University Press.

Wei, K., Crowson, K., Eseryel, U. Y., & Heckman, R. (2017). Roles and politeness behavior in community-based free/libre open source software development. *Information & Management*, *54*(5), 573-582. https://doi.org/10.1016/j.im.2016.11.006

Wei, K., Crowston, K., & Eseryel, U. Y. (2021). Participation in community-based free/libre open source software development tasks: the impact of task characteristics. *Internet Research*, *31*(4), 1177-1202. https://doi.org/10.1108/INTR-03-2020-0112