Participation in Community-Based Free/Libre Open Source Software Development Tasks: The Impact of Task Characteristics

Kangning Wei, U. Yeliz Eseryel and Kevin Crowston
Shandong University, East Carolina University and Syracuse University

Abstract

Purpose – This paper explores how task characteristics in terms of trigger type and task topic influence individual participation in community-based Free/Libre Open Source Software (FLOSS) development by considering participation in individual tasks rather than entire projects.

Design/methodology/approach – A quantitative study was designed using choose tasks that were carried out via the email discourse on the developers' email fora in five FLOSS projects. Choice process episodes were selected as the unit of analysis and were coded for the task trigger and topic. The impact of these factors on participation (i.e., the numbers of participants and messages) was assessed by regression.

Findings – The results reveal differences in participation related to different task triggers and task topics. Further, the results suggest the mediating role of the number of participants in the relationships between task characteristics and the number of messages. We also speculate that project type serves as a boundary condition restricting the impacts of task characteristics on the number of participants and propose this relationship for future research.

Research limitations/implications – Empirical support was provided to the important effects of different task characteristics on individual participation behaviors in FLOSS development tasks.

Practical implications – The findings can help FLOSS participants understand participation patterns in different tasks and choose the types of tasks to attend to.

Originality/value – This research explores the impact of task characteristics on participation in FLOSS development at the task level, while prior research on participation in FLOSS development has focused mainly on factors at the individual and/or project levels.

Keywords: Free/Libre Open Source Software (FLOSS); task characteristics; participation

1. Introduction

Community-based Free/Libre Open Source Software development (referred to simply as FLOSS throughout this paper) has attracted great interest among researchers who seek to understand this novel model of openness, often with an interest in transferring the model to other settings (e.g., Economides and Katsamakas, 2006; Oh and Jeon, 2007). Developer and user voluntary participation and involvement is essential for the success of FLOSS development (Xu *et al.*, 2009; Ehls and Herstatt, 2015), and has been studied extensively in FLOSS research (e.g., Zhang *et al.*, 2013; Xu and Jones, 2010; Bagozzi and Dholakia, 2006; Barcellini *et al.*, 2014).

Extant research has focused on identifying individual or project-level factors influencing individuals' voluntary participation in FLOSS development projects as a whole. Factors examined include intrinsic and extrinsic motivations for getting involved (Roberts *et al.*, 2006; Hertel *et al.*, 2003; Lakhani and von Hippel, 2003), cognitive and affective trust (Xu and Jones, 2010), initial access level of developers (Fang and Neufeld, 2009), ideology (Bagozzi and Dholakia, 2006), software licensing (Stewart *et al.*, 2006; Santos *et al.*, 2013; Belenzon and Schankerman, 2015), and leadership effectiveness (Xu *et al.*, 2009). Beyond initial participation, a few studies have examined sustained participation and found that social interactions among members and the benefits obtained from social interactions are the main drivers of this kind of participation (Zhang *et al.*, 2013; Fang and Neufeld, 2009), and that both the core and the peripheral members of FLOSS teams use a variety of politeness strategies that create respect and intimacy to maintain their social interaction (Wei *et al.*, 2017). Taken together, this body of literature has contributed to the understanding of individual participation in FLOSS development by focusing on the important characteristics of participants and of projects.

However, given the volunteer nature of FLOSS projects, just deciding to participate or not in a project has little impact: what matters is what work or tasks volunteers choose to participate in and what they actually do for the project. FLOSS development is task-oriented and task characteristics are major configuration factors in FLOSS development (Howison and Crowston, 2014), suggesting the importance of adopting a task perspective for FLOSS research.

Task has long been a key consideration in group research more generally. Zigurs and Buckland (1998) defines a group task as "the behavior requirements for accomplishing stated goals, via some process, using given information" (p.316). This definition emphasizes the importance of task characteristics presented to the group, i.e., the specific attributes or dimensions that describe different tasks (Griffin *et al.*, 1981). Prior non-FLOSS organizational research has established that task characteristics such as task type, task complexity and urgency have great impacts on individual and group behaviors (Campbell, 1988; Luciano *et al.*, 2018).

Despite the practical and theoretical importance of tasks, the details of how developers choose tasks have received only limited attention in FLOSS research (Ehls and Herstatt, 2015). For example, in a longitudinal study of the Jazz project, Licorish and MacDonell (2017) found that software practitioners engaged most intensively (i.e., exchanged more messages) in enhancement tasks, followed by defect-fixing tasks and support tasks, which demonstrated that software practitioners' engagement depended on the nature of the work they were performing. Similarly, Howison and Crowston (2014) argued that decisions to participate or not in a particular FLOSS development task will be influenced by the specific characteristics of the tasks in addition to individual or project-level factors. Research has found that newcomers in FLOSS development are not confident about choosing their initial tasks because they do not have enough information about

the tasks (Steinmacher *et al.*, 2015). While indicative of the value of studying tasks, these studies are just a start.

The objective of this paper is to increase our understanding of participation in FLOSS development at the level of particular tasks. We specifically ask the following research question: *how do different task characteristics affect individuals' participation in completing FLOSS development tasks?* To answer this question, we focus on community-based FLOSS development and examine participation in tasks from a process perspective as explained in Section 2. Our results lead us to a re-evaluation of the functioning of FLOSS development teams, one that shed light on the nature of expertise in the projects that provides insights for future research on participation in FLOSS development.

2. Theoretical Background

We start by developing a framework for understanding tasks and their characteristics, drawing on research in small groups, deferring the development of specific hypotheses to the following section.

2.1 Task Characteristics and Participation Behavior

A range of possibly relevant task characteristics has been identified and substantial studies have demonstrated that individuals' behaviors vary according to these task characteristics (Venkatesh *et al.*, 2016). For example, Deng and Joshi (2016) found that in the context of crowdsourcing work environment, crowdsourcing task characteristics (e.g., job autonomy, task variety, task significance, etc.) shape individuals' continued participation. Speier et al. (2003) investigated the moderating role of task complexity on the relationships between interruptions and computer-supported task performance and found that interruptions facilitate performance on simpler tasks while inhibiting performance on more complex tasks.

Our research is an assessment of the relationships between selected task characteristics and individuals' participation behavior in FLOSS development tasks. We start with one task characteristic that has been studied extensively in group research, namely task type (e.g., Fang *et al.*, 2005-6; Licorish and MacDonell, 2017). Numerous classifications of task types have been proposed to describe differences in the tasks performed by teams (Stewart and Barrick, 2000). Most of the major classifications were developed between the 1950s and 1980s (Zigurs and Buckland (1998) offer a brief summary of these classifications).

2.2 McGrath (1984)'s Task Type Framework

Among classifications of task types, McGrath's (1984) task circumplex framework is one of the most cited. This framework argues that group tasks can be classified into four categories: *generate*, *choose*, *negotiate* and *execute*. These categories differ along two dimensions. The first dimension reflects the degree to which the tasks have cognitive versus behavioral performance requirements. For instance, *choose* tasks are cognitive since correct or preferred answers need to be selected (a cognitive task) and agreed upon, while *execute* tasks are behavioral since they need physical movement, coordination or dexterity (Straus, 1999). The second dimension indicates the degree to which the task is cooperative or conflictual. For instance, *negotiate* tasks are conflictual as they resolve differing interests or viewpoints while *generate* tasks can be cooperative (Straus, 1999).

As a well-established taxonomy of group tasks, McGrath's framework captures basic characteristics of task processes and has been widely used as a conceptual foundation to study different tasks types in both traditional organizational settings (e.g., Nouri *et al.*, 2013) and technology-mediated communication settings (e.g., Zigurs and Buckland, 1998; Barlow and Dennis, 2016). However, in the limited research that considers the different types of software development tasks when investigating individuals' or team behaviors, most of them classify task

types not on a theoretical framework. Rather, task types were identified based on what concrete work the developers or users do, such as bug-fixing tasks (Crowston, 2008), information-providing tasks (e.g., Lakhani and von Hippel, 2003), and defect, enhancement and support tasks (e.g., Licorish and MacDonell, 2017). To better understand participation behavior, researchers have called for in-depth examination of the nature and structure of software tasks by applying theoretical task frameworks in software development research (Licorish and MacDonell, 2017).

For this paper, we opted to study tasks from the "*choose*" quadrant of the McGrath's task circumplex (1984) as exemplars of the range of tasks involved in software development (i.e., we hold task type constant). Choose tasks require coordination effort from the team members in order to decide a correct answer (e.g., intellective tasks) or to seek consensus on a preferred answer (e.g., judgement tasks) (Straus, 1999). We selected choose tasks as the research focus for three reasons.

First, we argue that among the four task categories in McGrath (1984)'s task circumplex, choose tasks will be particularly common in FLOSS development. Software development tasks in general have been seen as conceptual in nature (Zmud, 1980; de Reuver *et al.*, 2018), i.e., on the cognitive side of McGrath's (1984) task circumplex framework.

Second, choose tasks are well suited for our study. It requires considerable participation and coordination in order to achieve agreement on a choice. Further, the tasks are not all the same. People may spend different amount of time along the cognitive-behavioral continuum for different tasks in a same category (Stewart and Barrick, 2000), which means that we should see variation in the effort devoted to finish the tasks.

Finally, from a practical perspective, choose tasks are easy to identify from a process view of tasks. Prior research in decision making has provided several ways to identify the steps in choice

processes (Poole and Roth, 1989; e.g., Mintzberg *et al.*, 1976), which enable us to identify these tasks in a consistent way.

2.3 Task Characteristics for FLOSS Choose Tasks

Dennis et al. (2008) argues that task "is best thought of in terms of the fundamental communication processes that must be performed" (p. 579). From this perspective, finishing a task requires participants not only to share information, but also to conduct a cognitive process collectively to assess and act on the information. Different kinds of tasks involve different cognitive requirements, which is consistent with McGrath's cognitive-behavioral dimension of the task circumplex. In this research, we build on this process view of tasks to analyze choose tasks in FLOSS development. We focus on the following two specific characteristics of tasks, trigger type and task topic, which each classify tasks into two categories.

2.3.1 Trigger type

In a voluntary and self-managing environment such as the FLOSS development, choose tasks are usually not prescribed. Instead, they emerge from the interaction among participants. That is, tasks start with some stimulus that evokes them (Mintzberg, 1973; Smart and Vertinsky, 1977), which we label as a *task trigger*. A trigger is an event that prompts activities to happen at a particular time (Dix *et al.*, 2004). Different triggers can be expected to provoke different task processes.

Triggers that evoke a choose task can be classified into three types along a continuum of the degree of pressure to make a choice: opportunity and crisis triggers form the two ends of the continuum, with problem triggers in between (Mintzberg *et al.*, 1976). Opportunity triggers are ideas that are considered on a voluntary basis to improve an already secure situation (Mintzberg *et al.*, 1976). At the other extreme, crisis triggers have high time pressure and resource demands that require immediate attention (Nutt, 1984; Mintzberg *et al.*, 1976). Problem-triggered tasks face

milder pressure than crises and can have multiple stimuli (Mintzberg *et al.*, 1976). Usually, problems require actions in a timelier manner than opportunities do.

Given the voluntary and distributed nature of community-based FLOSS development, human resources and time pressure are usually not constraints in FLOSS development (Colazo and Fang, 2010). Unlike commercial software development, community-based FLOSS projects usually do not set strict deadlines (Scacchi, 2002), instead having loosely defined timelines that are adjusted frequently (Michlmayr *et al.*, 2007). Therefore, crisis, which is characterized by high time pressure and resources, seems unlikely to play an important role in evoking tasks in FLOSS contexts. Our examination of FLOSS choose tasks confirmed this argument. Of the 300 tasks collected from 5 projects (described below in section 4.1), we found only one task that might be classified as crisis-triggered, which was about a lawsuit between Fire and AOL regarding the logo and trademark infringement. Therefore, acknowledging the possibility of having crisis-triggered tasks, such as security issues that might bring severe consequences (Shaikh and Vaast, 2016), we only focus on two types of triggers in this paper: *problem* and *opportunity*. The most common examples of *opportunities* in FLOSS development include suggesting new features to be included in the software. Identification of bugs, or individuals' emails asking for help in resolving a problem they ran into are common examples of *problems* that might trigger a task (Annabi *et al.*, 2008).

Applying McGrath's (1984) task circumplex framework, we argue that problem-triggered tasks contain more behavioral requirements while opportunity-triggered tasks require more cognitive or conceptual requirements. Problem-triggered tasks are action-oriented (Hackman, 1968) and usually seek correct answers, while opportunities are ambiguous in nature (Cohen *et al.*, 1972), seeking preferred alternatives rather than strictly correct answers. Therefore, the ends of the problem-triggered tasks are clear, and immediate actions/behaviors might be needed to solve

these problems. In contrast, people have less clarity about what actions are appropriate for opportunity-triggered tasks, and thus spend a large proportion of time on cognitive work such as discussing the pros and cons of the approach to solve this kind of tasks.

2.3.2 Task topic

After the need for working on a task (i.e., a trigger) is identified, a task-resolution process is initiated, and a set of actions and resources are deployed to finish the task. At this point, depending on the topic of the tasks, other members of the project team may decide to get involved in the task, and the task may require more or less discussion to work on.

To identify the characteristics that distinguish different tasks in FLOSS development, we apply Wood (1986)'s theoretical model of tasks. Wood (1986) argued that all tasks contain three essential components: *products* (entities produced by behaviors that can be observed independently of the behaviors that produce them), *required acts* (behavior(s) required to create the defined product), and *information cues* (facts that can be processed to make conscious judgments). The required acts and information cues are task inputs that set limits on knowledge, skills and resources that required for completing a task successfully (Wood, 1986). For example, in a bug-fixing task, the product is the piece of software code that fixes the bug. The required acts are the activities to develop such codes. The information cues are the information that can be used to make judgments during the performance of the bug-fixing task.

In this research, we apply Wood's framework to determine different task topics by looking for differences in the information cues, required acts and products of the tasks. Specifically, we consider two topics of choose tasks in software development activities: *tactical tasks,* the day-to-day programming activities that maintain efficient operations of developing and testing software functionality, and *strategic tasks*, the tasks concerned with the long-time health of a project (Drury

et al., 2012). Tactical tasks constitute the primary work of the team, that is, software development, e.g., bug fixes, additions of new features or product enhancements through a change in software. Strategic tasks are tasks about the strategic direction for the project, such as social, organizational and legal issues, or alliances and partnerships. An example of strategic tasks is to discuss and decide a release date for the developing software.

3. Research Model and Hypotheses Development

In this section, we apply the task framework developed above to develop specific hypotheses about the impact of task characteristics (i.e., trigger type and task topic) on participation in FLOSS development tasks. In this research, we consider participation as having two aspects: people who participate in the tasks and communication that they engage in in carrying out the tasks. We focus on these two aspects of participation for the following two reasons.

First, it is well established in FLOSS research that community-based FLOSS teams cannot survive without sufficient voluntary participation from individuals (Fang and Neufeld, 2009). FLOSS teams are highly dynamic, similar to some online communities (Faraj *et al.*, 2011). This dynamism may indicate a turnover of leaders and members, which generally affects community performance negatively (Ransbotham and Kane, 2011). Similarly, it is difficult to discuss and complete a task (e.g., solving a problem, or designing a solution for a bug) without a sufficient number of participants, especially given the voluntary nature of FLOSS participation.

Second, researchers have emphasized the importance of information-exchange process of conducting tasks (e.g., Dennis *et al.*, 2008; Clarke and O'Connor, 2012; Xu, 2016). How interaction happens depends on the community. In FLOSS development, members participate from around the world, meet face-to-face infrequently if at all, and thus interact primarily via text-based information technologies (Wayner, 2000). In this context, a task cannot be completed unless other

members engage in the task-related communication process by reading and responding to others' messages.

In general, these two aspects indicate the needs for knowledge diversity and idea/information generation, and more generally reflect members' participation in different task activities (Licorish and MacDonell, 2017). We next develop specific hypotheses regarding the impacts of trigger type and task topic on participation.

3.1 Trigger Type

As we argued earlier, problem-triggered tasks contain more behavioral requirements while opportunity-triggered ones require more cognitive or conceptual requirements. A specific example of opportunity-triggered tasks is a feature-enhancement task, which provides people an opportunity to discuss if a new feature is desired in a software program. Information such as users' desires for the new feature, the feature requirements and software design feasibility need to be gathered and shared. Licorish and MacDonell (2017) argued that tasks related to new features enhancement involve extensive intellectual and cognitive processes. Therefore, more people might be needed in opportunity-triggered tasks in order to provide the various information required. Further, compared with problem tasks, some parts of opportunity-triggered tasks do not need specific technical knowledge. For instance, if users want to just provide input on the desirability of a new feature, they do not need to understand programming since they are not required to write the code. Therefore, more people might be able to attend opportunity-triggered tasks due to the low requirements on technical knowledge.

Further, organizational decision-making researchers have argued that opportunities might indicate positive situations in which gains could be made, while problems indicate situations in which an expected loss might occur (Fredrickson, 1985; Chattopadhyay *et al.*, 2001). Dutton and

Jackson (1987) argued that problems are aversive stimuli from which people tend to withdraw, while "opportunities bestow status and prestige to those who deal with them" (p.82). These emotions might attract people to attend an opportunity-triggered situations and deter people from being involved with problem-triggered situations (Dutton and Jackson, 1987), since a major extrinsic motivation for joining FLOSS development is to gain reputation (Crowston *et al.*, 2012). Therefore, we argue that individuals will be more likely to be attracted to opportunity-triggered tasks than to problem-triggered tasks. Hence, we propose,

H1a: Problem-triggered choose tasks in community-based FLOSS development teams will involve fewer participants than opportunity-triggered choose tasks.

We next consider communication needed to complete a choose task. In software development context, problems such as defects or bugs identified in the software code might threaten the software functionality and impact user's acceptance of the software. Therefore, immediate actions for correct answers might be needed to solve these problems. Once the correct answer is found by one or more team members, there is usually no need to debate over the solution (Straus, 1999). Therefore, the need to coordinate participants' activities may be limited (Straus, 1999). However, for opportunity-triggered tasks, participants seek preferred alternatives rather than correct answers. The ends and means of this type of tasks are not clearly defined, which requires the team to spend a great amount of time in discussing and deciding the merits of each alternative (Stewart and Barrick, 2000), thus increasing communication.

Further, opportunity triggers are ambiguous. Processes dealing with such triggers thus seem likely to resemble the garbage can model of decision making (Nutt, 1984), which was proposed by Cohen and his colleagues in a study of organizational anarchies (Cohen *et al.*, 1972). This model proposes a decision process wherein task triggers, solutions, participants and choices dump

together in a relatively independent fashion, and a solution is made when elements from these four streams coincide under certain organizational structures. When discussing an opportunity-triggered task such as adding a new feature in FLOSS development, people may dump all their information and resources and discuss multiple issues in parallel regardless of their relevance. Therefore, we expect opportunity-triggered tasks will involve more communication than problem-triggered tasks. Hence,

H1b: Problem-triggered choose tasks in community-based FLOSS development teams will involve less communication than opportunity-triggered choose tasks.

3.2 Task Topic

After a trigger, a task-resolution process is initiated by deploying different resources and actions to finish the task. As noted above, the differences between tactical tasks and strategic tasks can be analyzed using Wood (1986)'s framework. Tactical tasks and strategic tasks differ in all three components in terms of products, required acts and information cues.

First, while tactical tasks result in a tangible technical product of software code, strategic tasks result in non-tangible non-technical product of consensus on an aspect of the team or its process. Considering information cues, we note that tactical tasks require technical cues, whereas strategic tasks require social or process-related cues. To process technical cues and to develop software require participants to possess domain-specific knowledge about not only the functionalities but also the inner workings of the software (Von Krogh *et al.*, 2003; Zhang *et al.*, 2013), which might create barriers for contribution to tactical tasks. On the other hand, strategic tasks face greater uncertainty and require participation and discussions from a broader team of participants (Moe *et al.*, 2012). Some FLOSS projects even have established a formal way to deal with these tasks so

that enough participants are involved. For example, GNOME project has committees and task forces composed of volunteers to complete important strategic tasks (German, 2003).

Finally, required acts for tactical tasks deal with actions related to adding new features, fixing bugs, updating documentation and so on (Howison and Crowston, 2014). In the FLOSS development context, Howison and Crowston (2014) observed that when a task such as fixing a bug or submitting a patch is clear to a developer, s/he prefers work alone on the task rather than working with others; if the task is too complex or difficult for one to finish, s/he prefers to defer rather than to collaborate. Medappa and Srivastava (2019) similarly argued that for this type of tasks, developers prefer to work in a sequential manner rather than engaging in co-work. Thus, the participants in a tactical task are limited, though others may have to be involved, e.g., to clarify the bug, for quality control or final acceptance of a patch. In contrast, for strategic tasks, required acts are more open ended and there is not an inherent preference for sole action. In summary, we hypothesize:

H2a: Tactical choose tasks in community-based FLOSS development teams will involve fewer participants than strategic choose tasks.

Considering communication exchanged in completing the tasks, we note the impact of all three task elements. The importance of cues and products has been discussed above. Required acts for tactical tasks include development-related acts such as identifying potential technical solutions, evaluating different solutions and selecting the best solution. These acts are relatively well structured as they are based on specific routines of software development procedures, such as design and testing. Further, while team members communicate and make choices through mailing lists or discussion fora about both tactical and strategic tasks, tactical tasks can take advantage of an additional communications channel, that is, the software code itself. Software code is an active

communication artifact in the sense that interacting with the software (e.g., by testing it) gives developers direct feedback and provides explication of knowledge and insights without direct discussion with other team members (Bolici *et al.*, 2016). In FLOSS development in particular, each developer can access the software code (i.e., the artifact of the work) at any time to inspect the changes made by the other developers (Bolici *et al.*, 2016). As a result, developers making changes do not have to explain in detail what they have done: if others are curious, they can examine the code themselves. Based on this argument, we suggest that tactical tasks should have less need for explicit communication. Similarly, prior research on FLOSS development has found that much FLOSS software development work does not require much explicit coordination (Krishnamurthy, 2002; Howison and Crowston, 2014).

In contrast, strategic tasks are more often related to the strategic direction of the project, which faces greater uncertainty, as the information required in such tasks is usually incomplete. Required acts for strategic tasks include defining the issue, identifying relevant information and trying to build consensus. Acts are less structured, meaning that the task process may extend over a considerable period of time and involve many back-and-forth among developers. Moreover, for strategic tasks, mailing lists or discussion fora might be the only channels through which team members share knowledge with each other as these tasks do not benefit from communication via the software code. Therefore, the participants need to go through the more complex process of explicating all their knowledge and the knowledge of other relevant parties, such as asking questions to make sure they understand each other's messages. As a result, we hypothesize:

H2b: Tactical choose tasks in community-based FLOSS development teams will involve less communication than strategic choose tasks.

4. Research Method

To test the hypotheses developed above, we designed a quantitative study using messages exchanged among developers and users in five community-based FLOSS projects.

4.1 Project Selection

We sought projects that would provide a meaningful basis for comparison across the two task characteristics. As previously noted, FLOSS business models are diverse. To control for unwanted systematic variance, we chose community-based projects (the focus of our study) that were roughly similar in age and were all at production/stable development stage. Projects at this stage have relatively well-developed membership and sufficient team interaction history to have established choice processes, yet the software code still has room for improvement, which enabled us to observe rich team-interaction processes. To control for the possibility that the development tools used might structure the choice process, we selected projects that were all hosted on SourceForge (http://www.sourceforge.net), a FLOSS development site popular at the time of data collection that provides a consistent ICT infrastructure to developers. Finally, acknowledging that the level of participation is heavily skewed across different projects (Crowston and Howison, 2005), we purposefully selected projects that develop different types of software. Specifically, we selected projects that developed Enterprise Resource Planning (ERP) systems and projects developing Instant Messenger (IM) clients.

Following the above criteria, we randomly selected 3 established projects from the IM category: Gaim (currently known as Pidgin), aMSN and Fire; and 2 from the ERP category[1]: WebERP and OFBiz (currently known as Apache OFBiz[2]). Table I provides a comparison of these projects.

Table I. Project Summary[a]

Project Name	**Gaim (Pidgin)**	**Fire**	**aMSN**	**WebERP**	**OFBiz**[b]
Type	IM Client	IM Client	IM Client	ERP	ERP
Lines of Code[c]	199,413	169,233	142,283	77,540	240,731
Primary programming language	C	C, C++, Objective C	Tcl/Tk	PHP	Java
Webpage	Pidgin.im	Fire. sourceforge.net	www. amsnproject.net	www. weberp.org	ofbiz. apache. org
Bytes of documentation included in distribution	None	166K	None	417K	1.1M
Type	Multi-Protocol	Multi-Protocol	Single-Protocol	N/A	N/A
Project License	GNU General Public License (GPL)	GPL	GPL v2	GPL	Apache v2
Developers	10	12	41[d]	27	35
Initial Release	Nov. 1998	Apr. 1999	May 2002	Jan. 2003	Nov. 2001

a. Except as noted, data on Gaim (Pidgin), OFBiz and WebERP were collected from Openhub.net using the compare projects function.

b. Source: https://www.openhub.net/p/Apache-OFBiz

c. Lines of code were determined using the cloc program applied to the source code download of the release closest to the date of data collection.

d. Source: http://www.amsn-project.net/current-developers.php

4.2 Data and Unit of Analysis

We studied choose tasks that were carried out via the email discourse on the developers' email fora. To support communication and coordination among voluntary participants, FLOSS development teams use a variety of electronic means, such as email lists, trackers and coding tools (Barcellini *et al.*, 2014; Storey *et al.*, 2016). Among these sources, archives of email lists or

[1] We had initially also selected the Compiere project for the ERP category. However, during data analysis we came to realize that Compiere was not a community-based project like the others, since it was started by a company and had both community and commercial aspects in its development. To avoid possible bias introduced by this project, we decided to remove it from our study, resulting in an unbalanced design with 3 IM and 2 ERP projects.

[2] At the time of the study, OFBiz was not under the Apache umbrella but was a community-based FLOSS project like the other selected projects.

discussion fora offer rich information for researchers to explore social aspects of community-based FLOSS development (Guzzi *et al.*, 2013). These sources have been used to study a variety of topics, such as decision processes (Eseryel *et al.*, 2020), group maintenance strategies used in online communication (Wei *et al.*, 2017), sustained participation (Fang and Neufeld, 2009), and strategic interaction in knowledge-sharing processes (Kuk, 2006), among many others.

Email data were obtained from the FLOSSmole website (http://flossmole.org/). Though we cannot completely rule out the possibility of off-list discussions occurring through other channels (e.g., IRC, IM, phone or face-to-face meetings), at the time of data collection, FLOSS developers on SourceForge used email as the main communication tool for collaborating and communicating among developers and users (Zhang *et al.*, 2013). This practice means that any discussions that took place outside of the email fora would be invisible not only to us as researchers, but also to numerous developers as well. Further, our analysis of the mailing list interactions did not reveal references to any off-line discussions, suggesting that the data source we used provided a complete view of the choice process, at least for the choices made there.

We selected the choice process episode as our primary unit of coding and analysis of choose tasks, defined as a sequence of messages that begins with a task trigger that presents an opportunity or a problem that needs to be worked on, includes the required acts of issue discussion and possibly ends with a choice announcement (Annabi *et al.*, 2008). To give an example, a trigger may be a feature request or a report of a software bug. A choice announcement may be either a statement of the intention to do something or a notice of an actual implementation of a fix. Note that some choice processes did not result in a choice that was announced to the community, while others had multiple announcements as the choice was revised. The messages in an episode capture the

interactions among team members that constitute the process of making that choice and finishing the task from start to finish.

Choice process episodes were identified from the continuous stream of available messages through an initial coding process done independently by two of the authors. We started the analysis by reading through the messages until we identified a message containing a trigger or an announcement. Once we found a trigger or an announcement, we identified the sequence of messages that embodied the team process for the choose task. We observed that teams generally organize discussions in email threads, occasionally initiating new threads with the same or similar subject line. Therefore, we developed a choice process episode by combining one or more threads that used the same or a similar subject line as the initial message and that discussed the same main issue. Our explorative evaluation of the threads showed that any such follow-ups were typically posted within the following month, but extreme cases could be as many as 3 months. We therefore searched for messages on the same or similar content up to three months after the posting date of the last message on a thread. Since we were analyzing the messages retrospectively, we could collect all messages related to the task over time.

The process of identifying messages to include in each episode proceeded iteratively. Two researchers collected messages, shared the process they used with the research team, and revised their processes based on feedback from the team. The pairwise inter-coder reliability among two independent coders using percent agreement for each variable (Neuendorf, 2002) reached 85% and 80% respectively on task triggers and choice announcements. All differences between the coders were reconciled through discussion to obtain the sample of episodes for analysis.

Sampling of choice process episodes was stratified by project time: we chose 20 episodes from the beginning, middle and end periods of each project[3] based on a concern that the choice process effort might be different at different stages of the software development (e.g., initial collaboration vs. a more established team). The sample size was chosen to balance analysis feasibility with sufficient power for comparisons. With 60 episodes per project, we have reasonable power for comparison across projects while keeping the coding effort feasible.

This initial coding process collected 300 choice process episodes, each a collection of messages with a trigger and a choice announcement if any. Since the subject of this research is the participation and amount of communication in a software development task, we only consider tasks that were completed. In our sample, all the tasks that did not make final choices (i.e., did not have choice announcements) were removed from further analysis. As a result, 31 choice process episodes were removed and 269 were kept (163 IM choice process episodes and 106 ERP ones). Table II describes the distribution of the episodes across the five projects.

Table II. Distribution and Descriptive Statistics of Choice Episodes for the Five Projects

Project Name	**Project Type**	**No. of Choice Episodes**	**Number of Participants**				**Amount of Communication (Number of Messages)**			
			Min	Max	Mean	Median	Min	Max	Mean	Median
aMSN	IM	57	1	14	4.02	3	2	49	8.54	4
Fire	IM	56	2	8	3.29	3	2	18	5.84	4
Gaim	IM	50	2	13	4.48	4	2	26	7.54	6
	IM in total	*163*	*1*	*14*	*3.91*	*3*	*2*	*49*	*7.31*	*5*
OFBiz	ERP	55	2	8	3.16	3	2	19	6.33	4
WebERP	ERP	51	2	8	3.35	3	2	21	6.33	4
	ERP in total	*106*	*2*	*8*	*3.25*	*3*	*2*	*21*	*6.33*	*4*
Total		269	1	14	3.65	3	2	49	6.92	4

[3] For each project, the beginning and the ending periods were the first and last 20 choice process episodes found as of the time of data collection (i.e., from the start of the project's on-line presence to the most recent period). The middle period for each project consisted of 20 episodes surrounding a major software release approximately halfway between the beginning and ending periods. We chose to sample around a release period because making a release is one of the key team choices for a FLOSS project.

4.3 Measurements

In this section, we describe the independent and dependent variables and how they were coded. Table III describes the variables.

Table III. Variable Description and Measures

Variable	Variable Description	Measures
Dependent variable		
Number of participants	Count variable: the number of unique participants involved in a task	The number of unique participants involved in a task
Amount of communication	Count variable: the number of messages that make up a choice episode, which starts with a trigger, regardless of who sending the message	The total number of messages posted in the choice episode
Independent variable		
Trigger	Binary variable: the task is triggered by problems or by opportunity	Problem (0); opportunity (1)
Task Topic	Binary variable: the task is tactical or strategic in nature	Tactical (0); Strategic (1)
Control variable		
Duration	Task completion time	The number of days the messages spanned in a choice episode
Period	Three different data collection periods based on software development cycle	Beginning (0); middle (1); end (2)
Project	A dummy variable for every project in our sample	WebERP (1); OFBiz (2); Gaim (3); Fire (4); aMSN (5)

Dependent variables. As we discussed above, we capture two aspects of participation in a choose task: people participating in the task and communication exchanged in carrying out the task. We examine the first aspect by calculating the *number of participants* attracted to a particular task. In our context, the task process was captured by choice process episodes. Therefore, the number of participants was measured by the number of unique participants who posted messages in the choice process episode. Communication in carrying out the task was measured by the volume of communication, i.e., the total number of messages exchanged in the choice process episode.

Independent variables. *Task trigger* is a binary variable capturing whether a task was triggered by a problem or an opportunity. For each choice process episode, the three authors coded the trigger. A trigger was identified as a problem (coded as 0) based on the following criteria: 1) when there are problems or questions to deal with (e.g., that the software code does not run correctly for

the developers or the users); 2) when software bugs were identified; or 3) when there were strategic issues that were challenges to deal with rather than opportunities (for example, when there seems to be a breach of licensing agreements). On the other hand, 1) a clear identification of a desired functionality or change in the code that provides new or changed functionality; and 2) strategic issues that talked about plans for or issues with the projects were identified as an opportunity (coded as 1). As a result, 163 episodes were coded as problem-triggered choose tasks and 106 were coded as opportunity-triggered ones.

Task topic is also a binary variable. For each episode, three researchers coded each episode as either a tactical (coded as 0) or strategic task (coded as 1) based on the topic discussed in the task. Tactical tasks were identified as the team discussing and making choices on one of the following questions, identified inductively from the analysis of messages in the choice process episode: 1) bug reports, 2) feature requests, 3) problem reports, 4) patch submissions, and 5) to-do lists. Choice announcements for tactical tasks reflected either acceptance/rejection of a need for software code modification or acceptance/rejection of a submitted code modification.

Strategic tasks were identified as discussing and making choices on one of the following questions: 1) system design, 2) infrastructure/process, 3) business function, 4) release management, and 5) other issues. Strategic choice announcements reflected acceptance/rejection of a long-term strategic proposal for system design, infrastructure change and process improvement or resource allocation including task assignment and time schedule. As a result, 207 episodes were coded as tactical tasks and 62 as strategic ones. During the coding process, any disagreements about coding were discussed among the researchers until they were addressed.

Control variables. We also included several control variables to account for the influences of time and project type. First, our sampling strategy involved collecting tasks from three different

periods of the projects: the beginning, the middle time around a major release, and the end. We expect that the different stages of software development might impact individuals' participation and their efforts. A three-category indicator variable, *period*, was used (0=beginning period, 1=middle, and 2=end) to control for potential time effects. Second, *duration*, which was also time-related, captured task completion time by measuring the number of days the messages spanned in a choice process episode. It controlled for the possibility that tasks that took a longer time to reach a conclusion would thus attract more participants and produce more messages. Lastly, we controlled the projects by introducing a five-category indicator variable, *project* (1=WebERP, 2=OFBiz, 3=Gaim, 4=Fire and 5=aMSN), to control for differences in the average participation across projects. Table IV lists the descriptive information of the number of participants, the amount of communication, and duration across projects, as well as the correlation among these variables.

Table IV. Descriptive Statistics and Spearman Correlations

	Mean	SD	Min.	Max.	1	2	3
1. Participants	3.65	2.15	1	14	1		
2. Messages	6.92	6.43	2	49	0.794**	1	
3. Duration	3.33	4.26	1	28	.333**	.417**	1

5. Results

5.1 Descriptive Statistics

Table V lists the descriptive statistics for the outcome variables for the different categories of triggers and task topics. Both outcome variables, the number of participants and the volume of communication, are count variables. Both are over-dispersed, i.e., their variances are bigger than their means, counter to the equal mean and variance expected for a Poisson variable. We used a Lagrange Multiplier test that fits a negative binomial model with ancillary parameter equal to zero (0) (Orme and Combs-Orme, 2009) to test the severity of over-dispersion. The results indicated that over-dispersion should not be a problem for participation counts ($p=0.915$ for ancillary parameter > 0). However, the results indicated a statistically-significant over-dispersion for the

volume of communication (p=0.002 for ancillary parameter > 0). Therefore, to test our hypotheses, we conducted Poisson regression on H1a and H2a regarding the number of participants, and negative binomial regression on H1b and H2b regarding the volume of communication, since negative binomial method is more suitable to over-dispersed count data (Stanko, 2016). We present each regression separately below[4].

Table V. Descriptive Statistics for Trigger Type and Task Topic

Project Type	Trigger type/Task Topic	No. of Episodes	No. of Participants		No. of Messages	
			Mean	Std	Mean	Std
IM	Problem	105	3.31	1.80	5.86	4.67
ERP	Problem	58	3.17	1.47	5.78	4.72
	Problem tasks in total	*163*	*3.26*	*1.68*	*5.83*	*4.67*
IM	Opportunity	58	4.98	3.04	9.93	10.01
ERP	Opportunity	48	3.35	1.56	7.00	4.84
	Opportunity tasks in total	*106*	*4.25*	*2.60*	*8.60*	*8.19*
IM	Tactical	131	3.39	1.67	5.78	3.99
ERP	Tactical	76	3.20	1.42	6.55	4.99
	Tactical tasks in total	*207*	*3.32*	*1.58*	*6.06*	*4.39*
IM	Strategic	32	6.03	3.71	13.56	12.66
ERP	Strategic	30	3.40	1.71	5.77	4.25
	Strategic tasks in total	*62*	*4.76*	*3.19*	*9.79*	*10.27*

5.2 Results of Hypothesis Testing

We used Poisson regression on two models to test hypotheses H1a and H2a regarding the impacts of task characteristics on the number of participants in choose tasks. Variables were added to the regression models in a stepwise way. Model 1(a) included only control variables, namely, duration, the period from which the episodes were taken (the end period was used as default), and the project (aMSN was used as the baseline), while model 2(a) represented a full test of the proposed factors predicting participation. The results are shown in Table VI as incidence rate ratios, i.e., a coefficient of 1 indicates no influence of the factor on the outcome; coefficients greater than 1 show a positive impact and coefficients less than 1 indicate a negative impact. The Nagelkerke pseudo-R^2 for the regression was 0.46, showing good predictive performance.

[4] For robustness check, we also analyzed the data using a structural equation model. The results were identical to the regression analyses. Therefore, for simplicity, we only presented the results from the regression analysis.

Table VI. Poisson Regression Model Results Using Number of Participants as Dependent Variable

	Model 1(a)	Model 2(a)	Model 1(b)	Model 2(b)
Constant	4.941*** (0.081)	7.537*** (0.100)	3.982***(0.071)	5.917*** (0.088)
Control Variables				
Duration	1.010 (0.007)	1.012 (0.007)	1.010 (0.007)	1.012 (0.007)
Period (end as the reference period)				
Beginning	0.647***(0.078)	0.643***(0.078)	0.654***(0.078)	0.645*** (0.078)
Middle	0.728*** (0.076)	0.789** (0.078)	0.731***(0.765)	0.792** (0.078)
Project (aMSN as the reference project)				
Fire	0.810* (0.099)	0.863 (0.100)		
Gaim	1.127 (0.094)	1.056 (0.096)		
OFBiz	0.787*(0.101)	0.743**(0.101)		
WebERP	0.833 (0.101)	0.798* (0.102)		
Project type - IM			1.201**(0.067)	1.270*** (0.067)
Direct Effects				
H1a: Trigger type-Problem		0.771*** (0.066)		0.760*** (0.065)
H2a: Task topic-Tactical		0.680*** (0.072)		0.663*** (0.071)
Log Likehood	-520.21	-499.55	-526.03	-501.87
LR Chi-square	55.22***	96.54***	43.57***	91.89***
N	269	269	269	269

Note: 1) Exponentiation of the coefficients are shown with standard errors in parentheses;
2) ***p<0.001, **p<0.01, *p<0.05

The findings revealed that, problem-triggered tasks involved significantly fewer participants than opportunity-triggered tasks (p=0.000), supporting H1a. Tactical tasks involved significantly fewer participants than strategic tasks (p=0.000), thus supporting H2a. Regarding the control variables, first, the duration of a choice process episode was found to have no impact on the number of participants. Second, the results showed that periods during which the choice process episodes were collected played a role in driving members' participation. More specifically, tasks from both beginning and middle periods showed the participation of significantly fewer people than the end period. This result is consistent with the growth in popularity and team size of the projects. Third, compared to aMSN project in the IM category, the projects in the ERP category (i.e., OFBiz and WebERP) involved significantly fewer participants, while the other two projects in the IM category (i.e., Fire and Gaim) showed no difference. The results indicate that there was a difference in participation between the two different types of projects we selected.

Given the previous finding, we restructured the control variable *project* into a two-category indicator variable (IM=0, and ERP=1) and reran the analysis. The results are shown in Table VI Model 1(b) and 2(b). Consistent with the results from Model 1(a) and 1(b), H1a and H2a were supported. Further, the result suggest that choose tasks in IM projects involved more participants than those in ERP projects. We will explore this issue in more depth in section 5.4.

We used negative binomial regressions to test H1b and H2b. It is reasonable to expect that more participants will lead to more communication. Therefore, we included the number of participants as a control variable in these tests. Variables were added to the models as for the previous test. The results are shown in Table VII. Model 1 included only duration, the number of participants, periods (the end period was used as baseline), and the project (aMSN was used as baseline) as control variables. In model 2, we added the direct impacts of trigger type and task topic to test H1b and H2b. The findings showed that controlling for the number of participants, the number of messages posted were not different between problem vs. opportunity triggered tasks and between tactical vs. strategic tasks. Therefore, neither of the hypotheses H1b and H2b was supported. Interestingly, in this regression there was a small positive coefficient for duration, which means the longer the time needed for completing a choose task, the more messages were generated. There was no significant difference across the five projects. The Nagelkerke pseudo-R^2 was 0.91, indicating that the number of messages was nearly perfectly predicted by the number of participants and duration.

Table VII. Negative Binomial Model Results Using Number of Messages as Dependent Variable

	Model 1	Model 2
Constant	2.257*** (0.109)	2.169*** (0.153)
Control Variables		
Duration	1.017* (0.007)	1.016* (0.007)
No. of participants	1.271***(0.013)	1.275***(0.015)
Period (end as the reference period)		
Beginning	0.990 (0.077)	0.993 (0.078)
Middle	1.046 (0.074)	1.050 (0.074)
Project (aMSN as the reference project)		
Fire	0.995 (0.091)	0.981 (0.092)
Gaim	0.854 (0.091)	0.844 (0.092)
OFBiz	1.150 (0.091)	1.140 (0.092)
WebERP	1.045 (0.092)	1.040 (0.093)
Direct Effects		
H1b: Trigger type-Problem		0.948 (0.063)
H2b: Task topic-Tactical		1.092 (0.074)
Log Likehood	-631.33	-630.07
LR Chi-square	290.83***	293.35***
N	269	269

Note: 1) Exponentiation of the coefficients are shown with standard errors in parentheses;
2) ***p<0.001, **p<0.01, *p<0.05

5.3 The Mediating Effect of the Number of Participants

Since we expected and found that more participants were associated with more communication, we examined the mediating role of the number of participants between task characteristics and the number of messages. First, we assessed a direct link from the number of participants to the number of messages. Then we applied the bootstrap mediation-test suggested by Hayes (2013). We examined the total and direct effects of each task characteristic on the number of messages and the indirect effects through the number of participants. This method allows for testing each independent variable (IV) in a separate model. In each model, we selected one factor from task characteristics as the main IV to be tested and treated the other (together with duration, period and project) as covariates to both the dependent variable and the mediator. Table VIII summarizes the mediation test results.

Table VIII. Mediating Effect of the Number of Participants

	Total Effects		**Direct Effects**		**Indirect Effects**		
Independent Variable	Coefficient	T-value	Coefficient	T-value	Point Estimate	Bias-corrected bootstrap 95% Confidence Intervals	
						Lower	Upper
Trigger Type-Opportunity	3.036	4.072***	0.424	0.972	2.611	1.157	4.203
Task Topic-Strategic	4.004	4.717***	-0.005	-0.010	4.009	2.048	6.202

Note: Duration, period and project were treated as dummies in the mediation analysis. *** p<0.001, **p<0.01, * p<0.05.

From the results we can see that trigger type and task topic both had significant total effects on the number of messages (p =0.000). However, when the number of participants was introduced to each model as a mediator, none had a significant direct impact on the number of messages. The indirect effects indicated the mediation effects through the number of participants. The results showed the indirect effects for both task characteristics were significant, with the point estimates of 2.611 (for trigger type) and 4.009 (for task topic) respectively, and 95 percent bias-correct bootstrap confidence intervals of 1.157 – 4.203 (for trigger type) and 2.048 – 6.202 (for task topic) respectively. Therefore, we concluded that the number of participants fully mediated the impacts of the two task characteristics on the amount of communication.

5.4 Post-hoc Examination of the Moderating Role of Project Type

Our results confirmed the hypotheses that problem-triggered tasks and tactical tasks both involved fewer participants than opportunity-triggered tasks and strategic tasks respectively. An interesting finding from the analysis was that it seemed the numbers of participants were significantly different across IM and ERP projects, which made us speculate that the effect of task characteristics on the number of participants in a choose task might depend on the project type. Applying RPOCESS developed by Hayes (2013), we conducted a post-hoc examination of the

moderating effect of the project type[5] in terms of IM vs. ERP. The purpose of this analysis was to draw further insights regarding the impact of task characteristics on the number of participants within project contexts, rather than making statistical inferences. The results summarized in Table IX confirmed our speculation.

We can see that project type moderated the relationship between trigger type and the number of participants ($\beta=-1.126$, $t=-2.258$, $p=0.025$), and it also moderated the relationship between task topic and the number of participants ($\beta=-2.522$, $t=-4.682$, $p=0.000$). To present the interaction effects more clearly, we plotted them in Figure 1.

Table IX. Moderating Effect of Project Type on the Relationship between Task Characteristics and the Number of Participants

	Coefficient	t	LLCI	ULCI
Task trigger → the number of participants				
Constant	4.151	14.252***	3.578	4.725
Period-Beginning	-1.531	-5.203***	-2.110	-0.951
Period-Middle	-1.034	-3.471***	-1.621	-.448
Duration	0.035	1.252	-0.020	0.090
Trigger type	1.479	4.595***	0.845	2.113
Project type	-0.290	-0.909	-0.919	0.339
Project type * trigger type	-1.126	-2.258*	-2.108	-0.144
Conditional effects of trigger type at values of the moderator				
IM	1.479	4.595***	0.845	2.113
ERP	0.353	0.925	-0.398	1.104
Task topic → the number of participants				
Constant	4.101	15.837***	3.591	4.611
Period-Beginning	-1.664	-6.040***	-2.207	-1.122
Period-Middle	-1.048	-3.757***	-1.597	-0.499
Duration	0.060	2.262*	0.008	0.112
Task topic	2.753	7.531***	2.033	3.473
Project type	-0.171	-0.643	-0.693	0.352
Project type * task topic	-2.522	-4.682***	-3.583	-1.461
Conditional effects of trigger type at values of the moderator				
IM	2.753	7.531***	2.033	3.473
ERP	0.231	0.585	-0.547	1.009

Note: 1) ***p<0.001, **p<0.01, *p<0.05.
2) LLCI: the lower limit confidence level; ULCI: the upper limit confidence level

[5] Since the mediation test showed that the number of participants fully mediated the relationship between the task characteristics and the number of messages, we only focused on the number of participants in the moderation analysis.

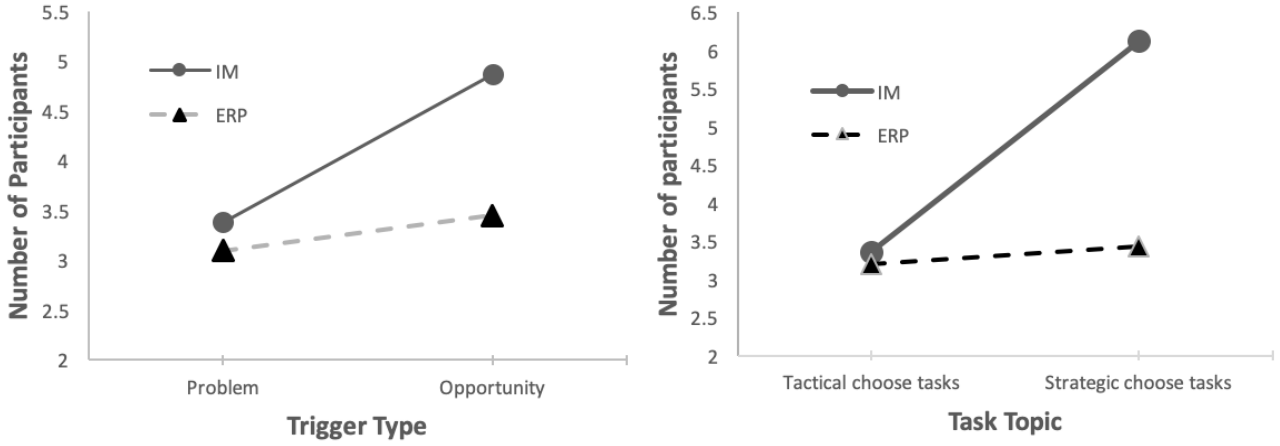

Figure 1. **Interaction Effects between Task Characteristics and Project Types**

The results show that opportunity-triggered tasks in the IM projects involved more participants than problem-triggered tasks ($\beta=1.479$, $t=4.595$, $p=0.000$) as expected, while there was no significant difference in the ERP projects ($\beta=0.353$, $t=0.925$, $p=0.356$). Similarly, strategic tasks in the IM projects involved more participants than tactical tasks ($\beta=2.753$, $t=7.531$, $p= 0.000$) as expected, while there was no significant difference in the ERP projects ($\beta=0.231$, $t=0.585$, $p=0.559$). In other words, H1a regarding the difference between problem- and opportunity-triggered tasks and H2a regarding the difference between tactical and strategic tasks for the number of participants seemed to hold true only for the IM projects, but not for the ERP projects.

6. Discussion

The primary goal of this study was to investigate the impacts of two different task characteristics (i.e., trigger type and task topic) on participation behavior in community-based FLOSS development tasks. Using choose tasks as a particularly important type of software development tasks, we observed that consistent with our hypotheses, problem-triggered vs. opportunity-triggered tasks and tactical vs. strategic tasks did have different impacts on the number of participants in the tasks. However, we did not find significant direct impacts of these task characteristics on the amount of communication exchanged in completing the tasks when

controlling for the effect of participants. Rather, the number of participants fully mediated the relationships between task characteristics and the amount of communication.

The results highlight that task characteristics serve as *direct* antecedents of the number of participants involved in a choose task, while as *indirect* antecedents of the amount of communication exchanged in the task. It is the number of participants, the human resource of a task that drives the amount of communication directly. It seems no matter what triggers the task or what the task is about, the messages exchanged in a task will not increase unless the task attracts more people to participate. In general, the results imply that in participation behavior, task characteristics only influence how many people would like to participate in a task; after that, other factors such as duration and project characteristics would take a leading role in affecting the contribution levels of the participants. Therefore, our study distinguishes between two important aspects of voluntary participation behavior (i.e., the number of participants and their efforts in terms of the amount of communication).

Prior research in voluntary participation mainly focuses on one aspect of participation, or treat these two aspects independently. For example, in the context of open contest, Chen *et al.* (2014) found contest characteristics such as complexity and type significantly impact the number of participants in contests. Licorish and MacDonell (2017) found that in Jazz project, significant differences exist among different types of software tasks in number of participants and number of messages exchanged; however, the authors did not control the impact of the number of participants when investigated the variance in the number of messages exchanged in different tasks. A few studies have emphasized the importance of project characteristics on developers' contribution behaviors. For example, a recent empirical study found that a match between project-level characteristics (e.g., license type, project size, etc.) and developers' motivation determines in

FLOSS developers' code contribution behavior (Belenzon and Schankerman, 2015). The differential impacts of task characteristics on participants and communication in participation behavior deserve further investigation.

Another interesting finding from our research is that, despite that our data generally supported the hypotheses that opportunity-triggered/strategic tasks involve more participants than problem-triggered/tactical tasks do (H1a and H2a), a post-hoc examination indicated between-project difference might exist. Although we realize that the small sample size (5 projects in two categories) does not allow us draw valid statistical inference from the post-hoc analysis, the interesting results have led us rethink about the drivers for participation in FLOSS development tasks, and pointed us to a boundary condition restricting the conclusions of H1a and H2a, which is project type. We speculate that participation in the choose tasks, and software development tasks in general, is driven not just by the *demand* of the task (e.g., urgency, information cues, or technical skills need to finish the task) but also by the *supply* of different actors interested in the project.

For problem-triggered tasks and tactical tasks, we initially suggested that the demand of these tasks would drive individuals' participation. These tasks seem to have only attracted participants with technical skills and abilities to contribute to the code and solve problems. As a result, we see similar numbers of participation in problem-triggered and tactical tasks regardless of the project type. However, for opportunity-triggered and strategic tasks, time constraints and technical contribution barriers are not major issues. Prior work has suggested the type of software developed by a project or software application domain as an important factor that influences user interests, defines target population types and size, and impacts development activities (Stewart *et al.*, 2006; Santos *et al.*, 2013; Comino *et al.*, 2007). In line with these studies, we posit that in our research

context, the two different types of software developed by IM and ERP projects help define different sets of actors interested in the projects, that is, the supply.

Specifically, we suggest that in the IM projects, the majority of participants understand the general workings of the whole software and therefore may be able to make some contribution to development-related tasks. As well, because IM software is designed for individual use, we expect there to be more users overall. We further expect that most of the affiliated developers use the IM software personally, hence their interests in contributing to the project. As opportunities represent areas where new features may be added that affect all users of the software, developers would naturally have an opinion on tasks that may end up changing their software use experience, and so be motivated to contribute.

In contrast, in the ERP projects, we suggest that the type of developers and users and the structure of the software limit the capability and motivation of individuals to get involved in software development tasks. First, we note that compared to IM systems, ERP systems require more specialized domain knowledge to be able to contribute. Further, these systems exhibit a modular system design (Paulish, 2002) with modules for different kinds of functions. Based on findings from prior research (Liang *et al.*, 2010), we postulate that developers specialize their development efforts in related modules of ERP systems based on their knowledge of the domain for those modules. A second difference is that a typical ERP developer is unlikely to use the software personally, but rather develops and/or implements one or more modules of the software for others (e.g., company employees or a consulting customer). Furthermore, it is possible that companies may choose to implement only a subset of the modules of the given ERP system, further limiting how many developers are interested in a development task.

For these two reasons, we expect that only a subset of developers will have the specific external knowledge and interests needed to contribute to each choose task in ERP projects. For example, a person who specializes in production-planning modules may not be interested in or knowledgeable about the accounting and tax rules that are important to financial accounting and control modules. Therefore, that developer may not be able to contribute even to opportunity-triggered or strategic tasks in those areas. Contrariwise, if the ERP project has only a few experts in the area of production planning, they would be the only ones to respond to all types of tasks involving production planning, whether the task is triggered by a problem or an opportunity, and whether it is a tactical or a strategic one. We suggest that this limit on the supply of developers is why we see about the same number of developers responding to tasks in the ERP projects, regardless of the task triggers or task topics. As another example of the effects of expertise on the supply of developers, consider the Heartbleed security bug in the OpenSSL library, which was attributed to the project having too few developers to properly audit the code (only four core developers) due in part to the complexity of the implementation making it difficult to understand the code (Williams, 2014).

In summary, in contrast to conventional software development organizations where the number of developers is a managerial decision, FLOSS projects are driven by voluntary participation. As a result, the number of participants in different software development tasks reflects the participants' interests and abilities as much as the task characteristics.

7. Implications and Conclusions

We conclude by acknowledging some limitations in our study before turning to the theoretical and practical implications of our results.

7.1 Limitations and Future Research Directions

As with all research, our results have limitations that affect their generalizability and suggest directions for future research. First, we selected only choose tasks to test the hypotheses. We argue that this task type is characteristic of FLOSS development tasks, but the choice does limit the generalizability of the results. For example, we did not study negotiation tasks that involve conflicts in our research. Although negotiation tasks are less common in FLOSS development than choose tasks, they have attracted researchers' interests in recent years (Filippova and Cho, 2016; Gamalielsson and Lundell, 2014). Future research should apply the framework of this research to other types of tasks in McGrath's task circumplex.

Second, the task processes (i.e., choice process episodes in this research) were extracted from messages sent to the developers' email fora. Thus, it is conceivable that the record of the episode does not capture all the communications related to a certain task. A specific limitation of this study is that we did not include messages from synchronous discussion fora (e.g., Internet Relay Chat, Instant Messaging or phone calls) into our analysis. While we found no evidence that these channels were used for the episodes we studied, future research should examine more systematically how people participate in these synchronous communication channels and what roles these media play in development practices.

Third, our hypothesis testing was only based on data from 269 choice process episodes in five FLOSS development projects. We purposefully selected two project categories (i.e., ERP and IM) and randomly selected 5 projects from these two categories after applying the project selection criteria. This project selection strategy might bring concerns of sample selection bias. Another concern is about the small sample size. At the time of our study, we had to rely on intensive manual coding to identify choice process episodes, different task triggers, and task topics. Having a

tractable sample size enabled us to manage the coding as well as to conduct the required statistical analysis. However, it might be beneficial in future research to sample choose tasks across a larger size of projects to assess the generalizability of our findings. To do so, some automated coding techniques are necessary to reduce the effort of manual coding in this research to a manageable level. The features implemented in current development platforms such as GitHub (e.g., issues, pull request, etc.) might make some of the coding straightforward.

Finally, the current study does not examine the content of the tasks or the task processes in detail. Understanding in more detail the process by which the participants finish tasks would complement our findings on participation.

7.2 Research Implications

Despite these limitations, this research contributes to the literature and practice in several ways.

First, this research explores participation based on task characteristics and investigates how different task characteristics influence participation in terms of the number of participants and the amount of communication in FLOSS development tasks. In contrast, most prior research has focused on motivational factors and project factors that influence participation at individual or project levels. Our findings provide empirical support to the important effects of different task characteristics on individual participation behaviors at a task level. Thus, our research contributes to the FLOSS literature by uncovering this important yet understudied relationship between task characteristics and individual behaviors.

Second, our research highlights the central role of the number of participants. In the analysis of the predictors of communication, we found that the impacts of task characteristics on communication were fully mediated by the number of participants. In other words, the tasks differ in how many participants they attracted, not how much the participants contributed to the task,

though tasks that took longer to resolve did seem to provide the opportunity for participants to contribute more.

Third, our research contributes to FLOSS literature by providing useful insights into the relationship between task characteristics and participation in different projects. By closely examining two types of projects (IM vs. ERP), we speculate that the software application domain defines the supply of different resources, which interacts with the different task characteristics to influence participation in FLOSS development tasks. Therefore, project application domain might serve as a boundary condition for the impact of task characteristics on participation behavior in FLOSS development tasks. Prior research has examined its direct impact on project outcomes such as project attractiveness and project success (Santos *et al.*, 2013; Crowston and Scozzi, 2002). However, limited research has investigated the impact of project application domain on project development activities. This research suggests a line of future research that could examine how the application domain, as a project-level characteristic, impacts FLOSS development activities directly as well as indirectly by working with other variables of interests (e.g., task characteristics in our research).

7.3 Practical Implications

The results of this research have several important practical implications for FLOSS participants and leaders as well. Many previous studies have emphasized the importance of attracting and keeping voluntary participation levels to ensure the continuity of FLOSS communities. Our study provides an understanding of the relationship between task characteristics and participation, which can enable the FLOSS administrators to manipulate task types posted to the email lists (e.g., encourage more opportunity-triggered or strategic tasks) to attract voluntary participation.

Second, our findings suggest that different task characteristics involve different levels of participation, which in turn, influence communication levels. This finding is useful for FLOSS participants to select which development tasks to participate in. For example, if a newcomer wants to gain recognition in the short term, attending opportunity-triggered and/or strategic tasks might be helpful since these two types of tasks involve a higher number of participants and generate more discussions. Participation in such a task may give a newcomer higher visibility with the other developers who are involved in the same discussion.

References

Annabi, H., Crowston, K. and Heckman, R. (2008), "Depicting what really matters: using episodes to study latent phenomenon", *in Proceedings of International Conference on Information Systems (ICIS) 2008*, December 14-17,, Paris, France. pp. Paper 183.

Bagozzi, R. P. and Dholakia, U. M. (2006), "Open source software user communities: a study of participation in Linux user groups", *Management Science,* Vol. 52 No. 7, pp. 1099-1115.

Barcellini, F., Détienne, F. and Burkhardt, J.-M. (2014), "A situated approach of roles and participation in open source software communities", *Human-Computer Interaction,* Vol. 29 No. 3, pp. 205-255.

Barlow, J. B. and Dennis, A. R. (2016), "Not as smart as we think: a study of collective intelligence in virtual groups", *Journal of Management Information Systems,* Vol. 33 No. 3, pp. 684-712.

Belenzon, S. and Schankerman, M. (2015), "Motivation and sorting of human capital in open innovation", *Strategic Management Journal,* Vol. 36 No. 6, pp. 795-820.

Bolici, F., Howison, J. and Crowston, K. (2016), "Stigmergic coordination in FLOSS development teams: integrating explicit and implicit mechanisms", *Cognitive Systems Research,* Vol. 38 No. pp. 14-22.

Campbell, D. J. (1988), "Task complexity: a review and analysis", *Academy of Management Review,* Vol. 13 No. 1, pp. 40-52.

Chattopadhyay, P., Glick, W. H. and Huber, G. P. (2001), "Organizational actions in response to threats and opportunities", *Academy of Management Journal,* Vol. 44 No. 5, pp. 937-955.

Chen, P.-Y., Pavlou, P. A. and Yang, Y. (2014), "Determinants of open contest participation in online labor markets", working paper No. 15-074, Fox School of Business, Temple University.

Clarke, P. and O'connor, R. V. (2012), "The situational factors that affect the software development process: towards a comprehensive reference framework", *Information and Software Technology,* Vol. 54 No. 5, pp. 433-447.

Cohen, M. D., March, J. G. and Olson, J. P. (1972), "A garbage can model of organizational choice", *Administrative Science Quarterly,* Vol. 17 No. 1, pp. 1-25.

Colazo, J. A. and Fang, Y. (2010), "Following the sun: temporal dispersion and performance in open source software project teams", *Journal of the Association for Information Systems,* Vol. 11 No. 11/12, pp. 684-707.

Comino, S., Manenti, F. M. and Parisi, M. L. (2007), "From planning to mature: on the success of open source projects", *Research Policy,* Vol. 36 No. 10, pp. 1575-1586.

Crowston, K. (2008), "The bug fixing process in proprietary and free/libre open source software: a coordination theory analysis", Grover, V. & Markus, M. L. (eds.) *Business Process Transformation*. M.E. Sharpe, Armonk, NY, pp. 69-100.

Crowston, K. and Howison, J. (2005), "The social structure of free and open source software development", *First Monday,* Vol. 10 No. 2, pp. https://doi.org/10.5210/fm.v10i2.1207.

Crowston, K. and Scozzi, B. (2002), "Open source software projects as virtual organisations: competency rallying for software development", *IEE Proceedings-Software,* Vol. 149 No. 1, pp. 3-17.

Crowston, K., Wei, K., Howison, J. and Wiggins, A. (2012), "Free/Libre open-source software development: what we know and what we do not know", *ACM Computing Surveys (CSUR),* Vol. 44 No. 2, pp. Article 7.

De Reuver, M., Sørensen, C. and Basole, R. C. (2018), "The digital platform: a research agenda", *Journal of Information Technology,* Vol. 33 No. 2, pp. 124-135.

Deng, X. N. and Joshi, K. D. (2016), "Why individuals participate in micro-task crowdsourcing work environment: revealing crowdworkers' perceptions", *Journal of the Association for Information Systems,* Vol. 17 No. 10, pp. 648-673.

Dennis, A. R., Fuller, R. M. and Valacich, J. S. (2008), "Media, tasks, and communication processes: a theory of media synchronicity", *MIS Quarterly,* Vol. 32 No. 3, pp. 575-600.

Dix, A., Ramduny-Ellis, D. and Wilkinson, J. (2004), "Trigger analysis: understanding broken tasks", Diaper, D. & Stanton, N. (eds.) *The handbook of task analysis for human-computer interaction*. Lawrence Erlbaum Associates, Inc, Mahwah, NJ, pp. 381-400.

Drury, M., Conboy, K. and Power, K. (2012), "Obstacles to decision making in Agile software development teams", *Journal of Systems and Software,* Vol. 85 No. 6, pp. 1239-1254.

Dutton, J. E. and Jackson, S. E. (1987), "Categorizing strategic issues: links to organizational action", *Academy of Management Review,* Vol. 12 No. 1, pp. 76-90.

Economides, N. and Katsamakas, E. (2006), "Two-sided competition of proprietary vs. open source technology platforms and the implications for the software industry", *Management Science,* Vol. 52 No. 7, pp. 1057-1071.

Ehls, D. and Herstatt, C. (2015), "Community joining, progressing and leaving: developing an open source participation lifecycle model", Herstatt, C. & Ehls, D. (eds.) *Open source*

innovation: the phenomenon, participant's behaviour, business implications. Routledge, New York, NY, pp. 115-136.

Eseryel, U. Y., Wei, K. and 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 Association for Information Systems,* Vol. 46 No. 1, pp. Artile 20.

Fang, X., Chan, S., Brzezinski, J. and Xu, S. (2005-6), "Moderating effects of task type on wireless technology acceptance", *Journal of Management Information Systems,* Vol. 22 No. 3, pp. 123-157.

Fang, Y. and Neufeld, D. (2009), "Understanding sustained participation in open source software projects", *Journal of Management Information Systems,* Vol. 25 No. 4, pp. 9-50.

Faraj, S., Jarvenpaa, S. L. and Majchrzak, A. (2011), "Knowledge collaboration in online communities", *Organization Science,* Vol. 22 No. 5, pp. 1224-1239.

Filippova, A. and Cho, H. (2016), "The effects and antecedents of conflict in free and open source software development", *in Proceedings of the 19th ACM Conference on Computer-Supported Cooperative Work & Social Computing*. ACM, pp. 705-716.

Fredrickson, J. W. (1985), "Effects of decision motive and organizational performance level on strategic decision processes", *Academy of Management Journal,* Vol. 28 No. 4, pp. 821-843.

Gamalielsson, J. and Lundell, B. (2014), "Sustainability of open source software communities beyond a fork: how and why has the LibreOffice project evolved?", *Journal of Systems and Software,* Vol. 89 No. 1, pp. 128-145.

German, D. M. (2003), "The GNOME project: A case study of open source, global software development", *Software Process: Improvement and Practice,* Vol. 8 No. 4, pp. 201–215.

Griffin, R. W., Welsh, A. and Moorhead, G. (1981), "Perceived task characteristics and employee performance: a literature review", *Academy of Management Review,* Vol. 6 No. 4, pp. 655-664.

Guzzi, A., Bacchelli, A., Lanza, M., Pinzger, M. and Van Deursen, A. (2013), "Communication in open source software development mailing lists", *in 2013 10th Working Conference on Mining Software Repositories (MSR)*. IEEE, pp. 277-286.

Hackman, J. R. (1968), "Effects of task characteristics on group products", *Journal of Experimental Social Psychology,* Vol. 4 No. 2, pp. 162-187.

Hayes, A. F. (2013), *Introduction to Mediation, Moderation, and Conditional Process Analysis: a Regression-Based Approach,* The Guilford Press, New York, NY.

Hertel, G., Niedner, S. and Herrmann, S. (2003), "Motivation of software developers in Open Source projects: an Internet-based survey of contributors to the Linux kernel", *Research Policy,* Vol. 32 No. 7, pp. 1159-1177.

Howison, J. and Crowston, K. (2014), "Collaboration through open superposition: a theory of the open source way", *MIS Quarterly,* Vol. 38 No. 1, pp. 29-50.

Krishnamurthy, S. (2002), "Cave or community? an empirical examination of 100 mature open source projects", *First Monday,* Vol. 7 No. 6, pp. Available at SSRN: https://ssrn.com/abstract=667402.

Kuk, G. (2006), "Strategic interaction and knowledge sharing in the KDE developer mailing list", *Management science,* Vol. 52 No. 7, pp. 1031-1042.

Lakhani, K. R. and Von Hippel, E. (2003), "How open source software works:" free" user-to-user assistance", *Research Policy,* Vol. 32 No. 6, pp. 923-943.

Liang, T. P., Jiang, J., Klein, G. S. and Liu, J. Y. C. (2010), "Software quality as influenced by informational diversity, task conflict and learning in project teams", *IEEE Transactions on Engineering Management,* Vol. 57 No. 3, pp. 477-487.

Licorish, S. A. and Macdonell, S. G. (2017), "Exploring software developers' work practices: task differences, participation, engagement, and speed of task resolution", *Information & Management,* Vol. 54 No. 3, pp. 364-382.

Luciano, M. M., Bartels, A. L., D'innocenzo, L., Maynard, M. T. and Mathieu, J. E. (2018), "Shared team experiences and team effectiveness: Unpacking the contingent effects of entrained rhythms and task characteristics", *Academy of Management Journal,* Vol. 61 No. 4, pp. 1403-1430.

Mcgrath, J. E. (1984), *Groups: Interaction and Performance,* Prentice-Hall Englewood Cliffs, NJ.

Medappa, P. K. and Srivastava, S. C. (2019), "Does superposition influence the success of FLOSS projects? an examination of open-source software development by organizations and individuals", *Information Systems Research,* Vol. 30 No. 3, pp. 764-786.

Michlmayr, M., Hunt, F. and Probert, D. (2007), "Release management in free software projects: practices and problems", *in* Feller, J., Fitzgerald, B., Scacchi, W. & A, S., eds, *IFIP International Conference on Open Source Systems: Open Source Development, Adoption and Innovation*. Springer, pp. 295-300.

Mintzberg, H. (1973), *The Nature of Managerial Work,* Harper & Row, New York.

Mintzberg, H., Raisinghani, D. and Theoret, A. (1976), "The structure of "unstructured" decision process", *Adminstrative Science Quarterly,* Vol. 21 No. 2, pp. 246-275.

Moe, N. B., Aurum, A. and Dybå, T. (2012), "Challenges of shared decision-making: a multiple case study of agile software development", *Information and Software Technology,* Vol. 54 No. 8, pp. 853-865.

Neuendorf, K. A. (2002), *The Content Analysis Guidebook,* Sage Publications, Thousand Oaks, CA.

Nouri, R., Erez, M., Rockstuhl, T., Ang, S., Leshem‐Calif, L. and Rafaeli, A. (2013), "Taking the bite out of culture: the impact of task structure and task type on overcoming impediments to cross‐cultural team performance", *Journal of Organizational Behavior,* Vol. 34 No. 6, pp. 739-763.

Nutt, P. C. (1984), "Types of organizational decision processes", *Administrative Science Quarterly,* Vol. 29 No. 3, pp. 414-450.

Oh, W. and Jeon, S. (2007), "Membership herding and network stability in the open source community: the Ising perspective", *Management Science,* Vol. 53 No. 7, pp. 1086-1101.

Orme, J. G. and Combs-Orme, T. (2009), *Multiple Regression with Discrete Dependent Variables,* Oxford University Press, New York, NY.

Paulish, D. J. (2002), *Architecture-Centric Software Project Management: A Practical Guide,* Addison-Wesley Professional, Boston, MA.

Poole, M. S. and Roth, J. (1989), "Decision development in small group IV: a typology of group decision paths", *Human Communication Research,* Vol. 15 No. 3, pp. 323-356.

Ransbotham, S. and Kane, G. C. (2011), "Membership turnover and collaboration success in online communities: explaining rises and falls from grace in Wikipedia", *MIS Quarterly,* Vol. 35 No. 3, pp. 613-627.

Roberts, J. A., Hann, I.-H. and Slaughter, S. A. (2006), "Understanding the motivations, participation, and performance of open source software developers: a longitudinal study of the Apache projects", *Management Science,* Vol. 52 No. 7, pp. 984-999.

Santos, C., Kuk, G., Kon, F. and Pearson, J. (2013), "The attraction of contributors in free and open source software projects", *The Journal of Strategic Information Systems,* Vol. 22 No. 1, pp. 26-45.

Scacchi, W. (2002), "Understanding the requirements for developing Open Source Software systems", *IEE Proceedings Software,* Vol. 149 No. 1, pp. 24--39.

Shaikh, M. and Vaast, E. (2016), "Folding and unfolding: balancing openness and transparency in open source communities", *Information Systems Research,* Vol. 27 No. 4, pp. 813-833.

Smart, C. and Vertinsky, I. (1977), "Designs for cirisis decision units", *Administrative Science Quarterly,* Vol. 22 No. 1, pp. 640-657.

Speier, C., Vessey, I. and Valacich, J. S. (2003), "The effects of interruptions, task complexity, and information presentation on computer-supported decision-making performance", *Decision Sciences,* Vol. 34 No. 4, pp. 771-796.

Stanko, M. A. (2016), "Toward a theory of remixing in online innovation communities", *Information Systems Research,* Vol. 27 No. 4, pp. 773-791.

Steinmacher, I., Conte, T. U. and Gerosa, M. A. (2015), "Understanding and supporting the choice of an appropriate task to start with in open source software communities", *in proceedings of the 48th Hawaii International Conference on System Sciences*, Jan. 5-8, Kauai, HI, USA. IEEE, pp. 5299-5308.

Stewart, G. L. and Barrick, M. R. (2000), "Team structure and performance: assessing the mediating role of intrateam process and the moderating role of task type", *Academy of Management Journal,* Vol. 43 No. 2, pp. 135-148.

Stewart, K. J., Ammeter, A. P. and Maruping, L. M. (2006), "Impacts of license choice and organizational sponsorship on user interest and development activity in open source software projects", *Information Systems Research,* Vol. 17 No. 2, pp. 126-144.

Storey, M.-A., Zagalsky, A., Figueira Filho, F., Singer, L. and German, D. M. (2016), "How social and communication channels shape and challenge a participatory culture in software development", *IEEE Transactions on Software Engineering,* Vol. 43 No. 2, pp. 185-204.

Straus, S. G. (1999), "Testing a typology of tasks: an empirical validation of McGrath's (1984) group task circumplex", *Small Group Research,* Vol. 30 No. 2, pp. 166-187.

Venkatesh, V., Thong, J. Y. and Xu, X. (2016), "Unified theory of acceptance and use of technology: a synthesis and the road ahead", *Journal of the Association for Information Systems,* Vol. 17 No. 5, pp. 328-376.

Von Krogh, G., Spaeth, S. and Lakhani, K. R. (2003), "Community, joining, and specialization in open source software innovation: a case study", *Research Policy,* Vol. 32 No. 7, pp. 1217-1241.

Wayner, P. (2000), *Free For All,* HarperCollins, New York.

Wei, K., Crowson, K., Eseryel, U. Y. and Heckman, R. (2017), "Roles and politeness behavior in community-based free/libre open source software development", *Information & Management,* Vol. 54 No. 5, pp. 573-582.

Williams, C. (2014). "OpenSSL Heartbleed: bloody nose for open-source bleeding hearts. The Register ", available at https://www.theregister.com/2014/04/11/openssl_heartbleed_robin_seggelmann/ (accessed January 27, 2021).

Wood, R. E. (1986), "Task complexity: definition of the construct", *Organizational Behavior and Human Decision Processes,* Vol. 37 No. 1, pp. 60-82.

Xu, B. and Jones, D. R. (2010), "Volunteers' participation in open source software development: a study from the social-relational perspective", *ACM SIGMIS Database,* Vol. 41 No. 3, pp. 69-84.

Xu, B., Jones, D. R. and Shao, B. (2009), "Volunteers' involvement in online community based software development", *Information & Management,* Vol. 46 No. 3, pp. 151-158.

Xu, J. D. (2016), "Retaining customers by utilizing technology-facilitated chat: mitigating website anxiety and task complexity", *Information & Management,* Vol. 53 No. 5, pp. 554-569.

Zhang, C., Hahn, J. and De, P. (2013), "Continued participation in online innovation communities: does community response matter equally for everyone?", *Information Systems Research,* Vol. 24 No. 4, pp. 1112-1130.

Zigurs, I. and Buckland, B. K. (1998), "A theory of task/technology fit and group support systems effectiveness", *MIS quarterly,* Vol. 22 No. 3, pp. 313-334.

Zmud, R. W. (1980), "Management of large software development efforts", *MIS quarterly,* Vol. 4 No. 2, pp. 45-55.