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

Kangning Wei
Shandong University

Kevin Crowston
Syracuse University

U. Yeliz Eseryel
East Carolina University

Abstract

Prior research on participation in FLOSS development has focused mainly on factors at the individual and/or project levels. In this research, we focus on task characteristics and explore their impacts on participation in FLOSS development tasks. Analyzing tasks from five projects in two categories, we find differences in participation related to different task triggers and task topics. Further, our results suggest the mediating role of number of participants in the relationship between task characteristics and the number of messages and the moderating role of project type in the relationships between task characteristics and the number of participants.

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

1. Introduction

Community-based Free/Libre Open Source Software (FLOSS) development (referred to simply as FLOSS throughout this paper) has attracted great interest among researchers who seek to understand this novel model of openness. Developers' and users' voluntary participation is an essential part of FLOSS development [1]. Accordingly, individual participation or involvement has been studied extensively in FLOSS research [e.g., 2, 3-6]. Extant research has primarily focused on identifying individual-related 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 [7, 8], cognitive and affective trust [3],

initial access level of developers [9], ideology [5], software licensing [10, 11], and leadership effectiveness [1]. Beyond initial participation, 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 [2, 9, 12]. Some researchers examined member roles in participating different behaviors. Wei, et al. [13] found that both the core and peripheral members of FLOSS use politeness strategies that create respect and intimacy. By examining knowledge creation behavior, Eseryel [14] found that leaders and core members participate more in knowledge creation in general and during critical events, and the small number of knowledge created by peripheral participants adds up and contributes notably to the knowledge created by the community as a whole. Taken together, this body of literature has contributed to the understanding of individual participation in FLOSS development by focusing on the characteristics of the participants and the projects.

However, we note that given the volunteer nature of FLOSS projects, just deciding to participate or not participate in a project has little impact: what matters is what work or tasks a volunteer actually does for the project. FLOSS development is task-oriented [15] and task characteristics are major configuration factors in FLOSS development. Despite the importance of this fine-grained participation for success of projects, few studies have considered factors related to participation at the level of individual tasks in the development process. At most, studies have documented the volume of participation, e.g., a recurrent finding from FLOSS studies is that the level of participation is heavily skewed, with only a few participants contributing at a high level [14, 16].

Decisions to participate or not in a particular task in FLOSS development will be influenced by the specific characteristics of the tasks in addition to individual or project-level

factors [17]. 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 [18-20]. To understand in detail individuals' participation behavior in FLOSS projects, we investigate the impact of task characteristics on individuals' participation in FLOSS development tasks. We specifically ask the following research question "What type of tasks involve higher participation?"

2. Tasks, Task Characteristics and Task Types

Task has long been a key consideration in group research. Zigurs and Buckland [21] 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 [22].

Based on different research contexts, a range of possibly relevant task characteristics has been identified and substantial studies have demonstrated that individuals' behaviors vary according to these task characteristics. For example, Deng and Joshi [23] 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 and colleagues [24] investigated the moderating role of task complexity on the relationships between interruptions and computer-supported decision-making performance and found that interruptions facilitated performance on simpler tasks while inhibiting performance on more complex tasks. In a longitudinal study of the Jazz project, Licorish and MacDonell [20] found that software practitioners engaged most intensively (i.e., exchanged more messages) in enhancement tasks, followed by defect-fixing tasks and support tasks.

Based on this prior work, we argue that to fully understand individual behaviors it is important to consider the characteristics of the tasks the individuals select to perform. This research is an attempt to assess the relationships between selected task characteristics and individuals' participation behavior in tasks in FLOSS development. We start with one task characteristic that has been studied extensively in group research, namely task type [e.g., 20, 21, 25-27]. Numerous classifications of task types have been proposed to describe differences in the tasks performed by teams [28]. Most of the major classifications were developed between the 1950s and 1980s (Zigurs and Buckland [21] offer a brief summary of these classifications). Among these, McGrath's [29] task circumplex framework is one of the most cited. This framework identifies four categories of task that reflect basic task processes: generate, choose, negotiate and execute. These differ along two dimensions: cognitive vs. behavioral performance requirements and cooperative vs. conflictual [29].

While such theoretical classifications have been used as conceptual foundations for organizational research, especially in the area of technology-mediated communication [e.g., 21, 30, 31], little research has considered how these models may be applied in software development research [20]. Instead, within the limited research that considers different types of software development tasks when investigating individuals' or team behaviors, most of them classify task types not on a theoretical framework but rather based on what concrete work the developers do, such as bug-fixing tasks [e.g., 32, 33] or defect, enhancement and support tasks [e.g., 20].

Dennis et al. [34] argues that task "is best thought of in terms of the fundamental communication processes that must be performed" (p.579). In this way, finishing a task requires participants not only to share information, but also to conduct a cognitive process to assess the information. In other words, 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 refer to this process view of task and are particularly interested in the following two specific characteristics of tasks that classify tasks into two broad categories: trigger type and task topic.

Trigger type. In a voluntary and self-managing environment such as the FLOSS development, tasks are usually not prescribed. Instead, they emerge from the interaction between participants. That is, tasks start with some stimulus that evokes them [35, 36], which we label as a *task trigger*. As we discuss later in section 3.1, task triggers can be classified along a continuum of the degree of pressure, reflecting the urgency dimension of tasks.

Task topic. Second, after the need for working on a task is identified, a task 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 (our two outcome variables which will be illustrated in section 3). Here we broadly group the topics of the tasks into two categories: tactical tasks and strategic tasks, which will be discussed in detail in section 3.2.

3. Research Model and Hypotheses Development

Our study examines task characteristics that predict participation behavior in FLOSS development. We consider participation as having two aspects. First, it is well established in FLOSS research that FLOSS teams cannot survive without sufficient voluntary participation from individuals [9]. Similarly, it is difficult to complete a task without sufficient number of participants, especially given the voluntary nature of FLOSS participation. Thus, we first

examine the quantity of participation in terms of the *number of participants* attracted to a particular task.

Second, participation is inherently an exchange process and interaction is the essence of participation. Researchers have emphasized the importance of information-exchange process of conducting tasks [e.g., 34, 37, 38]. 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 [39, 40]. 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. Therefore, the second aspect we consider for participation in FLOSS tasks is the amount of communication, specifically volume of communication (i.e., *the number of messages)* the participants exchange in carrying out the task.

The number of participants and the volume of communication indicate the needs for knowledge diversity and idea/information generation, and more generally reflect members' engagement in different task activities [20]. In conclusion, we focus specifically on the impact of task type (i.e., tasks based on trigger types and task topics) on the number of participants and number of messages involved in a FLOSS software development task. We also consider a non-task factor, namely the nature of the project or project type, though we argue that this factor also has some impact on the tasks. In the remainder of this section, we address these different components to develop specific hypotheses for our study.

3.1. Trigger type

A trigger is an event that prompts activities to happen at a particular time [41]. Different triggers can be expected to provoke different task processes. Triggers that evoke a decision task

are classified into three types along a continuum of the degree of pressure to reach a decision: opportunity and crisis triggers form the two ends of the continuum, with problem triggers in between [42]. Opportunity triggers are typically ideas that are initiated on a voluntary basis to improve an already secure situation [42]. At the other extreme, crisis triggers have high time pressure and resource demands that require immediate attention [42, 43]. Problem-triggered decisions face milder pressure than crises and can have multiple stimuli [42].

Similar classification can be found for other tasks, such as learning activities. For example, Cope [44] argues that the entrepreneurial learning activities emerge from entrepreneurs' responses to opportunities, problems or even crisis. Within the FLOSS literature, Eseryel [14] investigated the knowledge creation behaviors of participants based on crisis triggers (called 'crucial or critical events', and described as problems which may not allow a major release, or may block all releases) and found that users could participate in a limited way, and the most knowledge creation was done by developers during those periods, compared to regular periods, where the trigger is not a problem.

Given the voluntary and distributed nature of community-based FLOSS development, human resources and time pressure are usually not constraints in FLOSS development [46]. Unlike commercial software development, FLOSS development teams usually do not set up strict deadlines [47], instead having loosely defined timelines that are adjusted frequently [48]. 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 project communications confirmed this argument. Of the 300 tasks collected from 5 projects (described below in section 4.2), we found that only one task might be classified as crisis-triggered, which was about a lawsuit between Fire and AOL regarding the trademark and logo infringement.

Therefore, in this paper, we only focus on two types of triggers: problem and opportunity. 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 examples of problems that might trigger a task [45].

According to McGrath [29]'s task circumplex, tasks differ in the extent to which they require behavioral or cognitive performances. Although in general software development has been seen as complex and conceptual in nature [49, 50], not all tasks in software development are same. In specific, we argue that problem-triggered tasks contain more behavioral requirements while opportunity-triggered tasks require more cognitive or conceptual requirements, for the following reasons.

Problems usually require actions in a timelier manner than opportunities do. Prior research has suggested problem-solving tasks are action-oriented [51]. 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, the ends of the tasks are clear and immediate actions might be needed to solve these problems. In contrast, opportunities are ambiguous in nature [52]. People have less clarity about what actions are appropriate, and thus spend a large proportion of time on cognitive work such as discussing the pros and cons of the opportunities. For example, Licorish and MacDonell [20] argue that tasks related to new features enhancement involve extensive intellectual and cognitive processes.

Problem tasks usually seek correct answers. Once the correct answer is found by one or more team members, there is usually no need to debate over the solution [53]. Therefore, the need to coordinate participants' activities may be limited [53]. In the FLOSS development context, Howison and Crowston [17] observed that when a problem 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. Thus, communication is limited. Further, problem-triggered tasks are constrained by the domain-specific knowledge and skills required to solve the problems, which is especially true for software development [14]. In other words, fewer people might be qualified to participate in problem-triggered tasks.

In contrast, opportunity-triggered tasks, which are more conceptually-oriented, 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 [28]. For example, a feature-enhancement task 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. Further, compared with problem tasks, some parts of opportunity-triggered tasks do not need specific technical knowledge. For instance, people do not need to understand specific programming language to provide inputs on the desirability of a new feature.

In conclusion, we expect opportunity-triggered tasks will involve more discussion and more participants. Hence, we argue,

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

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

3.2. Task Topic

After a trigger, a task-resolution process is initiated by deploying different resources and actions to finish the task. To identify the characteristics that distinguish different tasks in FLOSS development, we apply Wood [54]'s theoretical model of tasks. Wood [54] argued that all tasks contain three essential components that also reflect the complexity of the tasks: products (entities produced by behaviors that can be observed independently of the behaviors that produce them), required acts (behavior(s) required to create a 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 [54]. In this research, we apply Wood's framework to determine different classes of tasks along the task complexity dimension by looking for differences in the information cues, required acts and products of the decisions.

We consider two levels of 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 [55]. Tactical tasks are related to routine tasks about 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, social, organizational and legal issues, or alliances and partnerships.

As noted above, the differences between tactical tasks and strategic tasks can be analyzed by using Wood [54]'s framework. Tactical tasks and strategic tasks differ in all three components in terms of products, required acts and information cues. Considering participation, we note that tactical tasks require technical cues, whereas strategic tasks require social or

process-related cues. Strategic tasks thus face greater uncertainty and require participation and discussions from a broader team of participants [56]. 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 [57]. Therefore, we argue that tactical tasks attract fewer participants than strategic decisions do.

A similar argument can be made by considering the different knowledge required to complete tactical and strategic tasks. Software development requires participants possess domain-specific knowledge about not only the functionalities but also the inner workings of the software [2, 12], which might create barriers for some participants to contribute to tactical tasks. In summary, we hypothesize:

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

Considering the number of messages, we note the impact of all three task elements. First, while tactical tasks result in a tangible technical product of software code, strategic tasks result in non-technical and non-tangible product of consensus on an aspect of the team or its process.

Second, the required acts (from Wood's framework) for tactical tasks include development-related acts such as identification of potential technical solutions, evaluating different solutions and identifying 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, prior research on FLOSS development has found that much FLOSS software development work does not require much explicit coordination [17, 58]. 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. As a result, we hypothesize:

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

3.3. Project Type

While our main focus is on task characteristics, we also generalize our findings by considering decisions made for different project types, specifically the type of the software a FLOSS project tries to develop. The structural complexity of the software being developed is an important issue in the field of software development, which captures the characteristics or attributes of the code being developed or modified [46]. Therefore, the complexity of the software being developed to some extent reflects the task complexity in a project, an important task characteristic studied in team research [e.g., 18, 38, 59]. Specifically, we expect participation in different tasks to be affected by the complexity of the software being developed.

Research has found that task complexity affects team interaction and software development processes in general [37, 60, 61]. In software development, efforts to finish a task vary greatly depending on the characteristics of the developed software itself, such as the size and structure of the software [62]. Complex tasks require more acts and contain more information cues than simple tasks, and these cues and acts can be highly interdependent [54]. Further, researchers have argued that, with the high complexity of an IT project, it is more critical to have developers and users involved in the development process [63].

In addition to internal complexity of the software, Espinosa, et al. [62] suggested that coordination challenges imposed by external factors increase the complexity of software development, which in turn affects efforts needed in completing tasks. For example, to develop accounting modules in enterprise resource planning systems, system developers must consider legal requirements (e.g., the tax calculation rules for each country and region) and general business process requirements (e.g., generally accepted accounting principles). These imposed external factors increase the dependency of the task performers on informational cues about such external factors.

These internal and external factors contribute to an increased level of acts and information cues needed to complete a task in FLOSS projects that develop complex software. Therefore, more participants with knowledge in different domains are needed in order to complete tasks in complex projects, and more discussion will be needed. Prior research indicated that project type implies technical sophistication and argued that sophisticated projects tend to stimulate more contributions from FLOSS developers [64]. Following the argument above, we hypothesize:

H3a: Tasks in FLOSS projects that develop simpler products will involve fewer participants than those in the projects that develop more complex products.

H3b: Tasks in FLOSS projects that develop simpler products will involve less communication than those in the projects that develop more complex products.

4. Research Method

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

In this research, we decided to study tasks from the "choose" quadrant of the McGrath's task circumplex [29] to represent software development tasks. We label them as decision-making tasks. Although decision-making tasks from the "choose" quadrant are usually seen as conceptual in nature [29], not all such tasks are same. Due to differences in factors like task clarity and task complexity, people might arrange different amount of time along the cognitive-behavioral continuum for different tasks in a same category [28].

We selected choose tasks for testing our hypotheses for two reasons. First, we argue that among the four task categories in McGrath's task circumplex (i.e., generate, choose, negotiate and execute), choose tasks are particularly common in FLOSS development. Software development tasks in general have been seen as complex and conceptual in nature [49, 50]. Therefore, execute task that require physical movement, coordination or dexterity [53] are unlikely to be seen in FLOSS development. As well, prior research has found that community-based FLOSS development usually do not engage in a formal and clear process of planning activities [16]. Thus, generate tasks that focus on brainstorming and planning tasks [53] will not be typical of FLOSS development. Given the voluntary and self-organizing nature of FLOSS development, we also suspect that negotiating tasks that focus on resolving conflict viewpoints [53] will not take an important role in FLOSS tasks. Overall, we believe decision tasks from choose category will be representative FLOSS development tasks more generally.

Second, decision tasks are well suited for our study. It requires considerable participation and coordination in order to achieve agreement and finish the tasks, which enables us to observe the number of participants and the number of messages devoted to finish the tasks. From a practical perspective, decision-making tasks are easy to identify from a process view of tasks.

Prior research in decision-making has provided several ways to identify decision-making process [e.g., 42, 65-67], which enable us to identify tasks in a consistent way.

In this section, we described our project selection criteria, data and level of analysis, and the measurements of the variables interested.

4.1. Project Selection

We sought projects that would provide a meaningful basis for comparison across the three factors we investigated. 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 in potential market size, which were all at production/stable development stage. Projects at this stage have relatively well-developed membership and sufficient team history to have established decision-making processes, yet the software code still has room for improvement, which enables us to observe rich team interaction processes. Acknowledging that the development tools used might structure the decision-making processes, we selected projects that were all hosted on SourceForge (www.sourceforge.net), a FLOSS development site popular at the time of data collection that provides a consistent information and communication technology infrastructure to developers.

Testing the hypotheses required comparisons of tasks with different trigger types (i.e., problems vs. opportunities), different task topics (i.e., tactical vs. strategic tasks), and projects that differ in the complexity of software they develop. For the latter, we planned a theoretical comparison of projects developing low- and high-complexity software. Specifically, we selected projects that developed Instant Messenger (IM) clients and Enterprise Resource Planning (ERP) systems.

ERP software development is known to be multifaceted and intricate [68]. Tasks in ERP projects are expected to be more complex than those in IM projects since ERP systems have high software code interdependencies and many external constraints such as accounting rules and legal reporting requirements that differ by country. ERP software developers also need to consider how the software can be engineered to fit the needs of diverse companies. In contrast, IM clients can be more generic, and thus simple, in serving the needs of many. These factors are expected to make task efforts in ERP projects different from those in IM projects. As a result, we selected 3 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]).

4.2. Data and Unit of Analysis

We studied choose tasks that took place in the email discourse on the developers' email fora, which are the primary communication venues for the members of FLOSS development team members. 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), FLOSS development used the email fora as the main communication tool for collaborating and communicating among developers and users [2]. This practice means that any discussions that took place outside of the email fora would be invisible not only to us researchers, but also to numerous developers as well. Further, our analysis of the mailing list interactions did not reveal references to such off-line

[1] We had initially also selected Compiere 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.

discussions, suggesting that the data source we used provided a complete view of the decision-making process, at least for the decisions made here. Nor did we observe project-level decisions being made via IRC when we followed the channel for a few days (though we did not do a systematic analysis over an extended period); rather IRC seemed to be a place to seek quick feedback or help with a problem.

We selected the decision episode as our primary unit of coding and analysis, defined as a sequence of messages that begins with a decision trigger that presents an opportunity or a problem that needs to be decided, includes the required acts of issue discussion and possibly ends with a decision announcement [45]. To give an example, a decision trigger may be a feature request or a report of a software bug. A decision announcement may be either a statement of the intention to do something or notice of an actual implementation of a fix. Note that some decision processes did not result in a decision that was announced to the community, while others had multiple announcements as the decision was revised. The messages in an episode capture the interactions among team members that constitute the process of making that decision and finishing the task from start to finish.

Decision 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 decision trigger or an announcement. Once we found a trigger or an announcement, we identified the sequence of messages that embodied the team process for finishing that task. We observed that teams generally organize discussions in a thread, occasionally initiating new threads with the same or similar subject line. Therefore, we developed a decision episode by combining one or more discussion 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, and in more extreme cases within 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 reached 85% and 80% respectively on decision triggers and announcements. All differences between the coders were reconciled through discussion to obtain the sample of episodes for analysis.

Sampling of decision episodes was stratified by time: we chose 20 episodes from the beginning, middle and end periods of each project[3] based on a concern that the decision-making 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 decision episodes, each a collection of messages with a trigger and a decision announcement if any. Since the subject of this research is the participation and amount of communication in a software development task, we only consider

[3] For each project, the beginning and the ending periods were the first and last 20 decision 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 decisions for a FLOSS project.

tasks that were completed. In our sample, all the tasks that did not make final decisions (i.e., did not have decision announcements) were removed from further analysis. As a result, 31 decision episodes were removed and 269 were kept (163 IM decision episodes and 106 ERP ones). Table 1 describes the distribution of the episodes across the five projects.

Table 1 Distribution and Descriptive Statistics of Decision Episodes for the Five Projects

Project Name	Project Type	No. of Decision Episodes	Number of Participants				Amount of Communication			
			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
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
Total		269	1	14	3.65	3	2	49	6.92	4

4.3. Measurements

Dependent variables. As we discussed above, we capture two aspects of participation in a decision task: *the number of participants* and *the amount of communication*. In this research, the number of participants refers to the number of unique participants involved in a task. Therefore, the number of participants was measured by the number of unique participants involved in the decision episode. The amount of communication was measured by the total number of messages posted in the decision episode. Table 1 lists the descriptive information of the number of participants and the amount of communication across projects.

Independent variables. *Task trigger* is a binary variable capturing whether a task was triggered by a problem or an opportunity. For each decision episode, the three authors coded the trigger. When there are problems to deal with, e.g., that the software code does not run correctly for the developers or the users, when software bugs were identified, or 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), these issues were identified by coders as "problems"

(coded as 0). On the other hand, a clear identification of a desired functionality or change in the code that provides new or changed functionality and 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 decision tasks and 106 were coded as opportunity-triggered decision tasks.

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 content of the task. Tactical decisions were identified as the team discussing and making decisions on one of the following questions, identified inductively from the analysis of messages in the decision episode: 1) bug reports, 2) feature requests, 3) problem reports, 4) patch submissions and 5) to-do lists. Decision announcements for tactical decisions reflected either acceptance/rejection of a need for software code modification or acceptance/rejection of a submitted code modification.

Strategic decisions were identified as discussing and making decisions on one of the following questions: 1) system design, 2) infrastructure/process, 3) business function, 4) release management and 5) other issues. Strategic decision announcements reflected either acceptance or 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 decisions and 62 were coded as strategic decisions. During the coding process, any disagreements were discussed among the researchers until they were addressed (that is, all data were multiply coded).

Project type was coded as a binary variable in which the value of 0 indicated decision tasks from the IM projects and 1 indicated those from the ERP projects.

Control variables. Several control variables were used in our analyses. Our sampling strategy involved collecting decision 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 period and 2=end period) to control for potential time effects. Our second control variable, *duration*, which was also time-related, captures task completion time by measuring the number of days the messages spanned in a decision episode. It is controlled for the possibility that decisions that took a longer time to reach a conclusion would attract more participants and produce more messages.

5. Results

In this section we present the results of our analyses.

5.1. Descriptive statistics

We first present descriptive statistics for our data. Table 2 lists the descriptive statistics and Spearman correlations among the variables (we report Spearman correlation because the data are not normally distributed). Both outcome variables, the number of participants and the amount of communication, are count variables. Both are over-dispersed, i.e., their variances are bigger than their means. We used a Lagrange Multiplier test that fits a negative binomial model with ancillary parameter equal to zero (0) [69] to test the severity of over-dispersion. The results indicated that over-dispersion should not be a problem for participation dataset ($p=0.915$ for ancillary parameter > 0). However, the results indicated a statistically-significant over-dispersion for the amount of communication variable ($p=0.002$ for ancillary parameter > 0). Therefore, to test our hypotheses, we conducted Poisson regression on H1a–H3a regarding the number of

participants, and negative binomial regression on H1b–H3b regarding the amount of communication, since negative binomial method is more suitable to over-dispersed count data [70]. We present each regression separately below[4].

Table 2 Descriptive Statistics and Spearman Correlations

	Mean	SD	Min.	Max.	1	2	3	4	5	6	7
Participants	3.65	2.15	1	14	1						
Messages	6.92	6.43	2	49	0.794**	1					
Task topic	0.23	0.42	0	1	0.189**	.144*	1				
Trigger type	0.39	0.49	0	1	0.221**	.208**	-.008	1			
Project type	0.39	0.49	0	1	-.123**	-.057	.101	.097	1		
Period	0.96	0.81	0	2	.272**	.254**	-.042	-.012	.018	1	
Duration	3.33	4.26	1	28	.333**	.417**	.072	.031	-.099	.086	1

**p<0.01, *p<0.05

5.2. Results of Hypothesis Testing

We used Poisson regression on two models to test hypotheses H1a-H3a regarding the impacts of task characteristics and project type on the number of participants in decision tasks. Variables were added to the regression models in a stepwise way. Model 1 included only control variables, namely, duration and the period from which the episodes were taken (the end period was used as default), while model 2 represented a full test of the proposed factors predicting participation (Model 3 is discussed later in Section 6). The results are shown in Table 3 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.

Table 3 Poisson Regression Model Estimation Results Using Number of Participants as Dependent Variable

	Model 1	Model 2	Model 3
Constant	4.438*** (0.058)	5.917*** (0.088)	4.277*** (0.131)
Control Variables			
Period-beginning	0.657*** (0.078)	0.645*** (0.078)	0.654*** (0.079)
Period-middle	0.733*** (0.077)	0.792** (0.078)	0.821* (0.079)
Duration	1.012 (0.007)	1.012 (0.007)	1.015* (0.007)

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

Direct Effects			
H1: Trigger type-Problem		0.760*** (0.065)	0.890 (0.109)
H2: Task topic-Tactical		0.663*** (0.071)	0.920 (0.119)
H3: Project type-IM		1.270*** (0.067)	2.034*** (0.146)
Interaction Effects			
Project type (IM) X trigger type (problem)			0.799* (0.136)
Project type (IM) X task topic (tactical)			0.598*** (0.148)
Log Likehood	-529.8	-501.87	-494.67
LR Chi-squire	35.95***	91.89***	106.29***

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

The findings related to the number of participants are as follows. Regarding task triggers, problem-triggered tasks involved significantly fewer participants than opportunity-triggered tasks ($p < 0.001$), supporting H1a. Tactical tasks involved significantly fewer participants than strategic tasks ($p < 0.001$), thus supporting H2a. However, our results indicated tasks in IM projects involved significantly more participants than tasks in ERP projects ($p < 0.001$), which is contrary to our hypothesis. Therefore, H3a was not supported.

Regarding our control variables, the results showed that periods during which the decision 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. The duration of the decision tasks was found to have no impact on the number of participants.

We used negative binomial regressions to test H1b–H3b. Variables were added in a step-wise way as for the previous test. The results are shown in Table 4. Model 1 included only periods (the end period was used as default) and duration as control variables. In model 2, we added the direct impacts of trigger type, task topic, and project type to test H1b-H3b.

Table 4 Negative Binomial Model Estimation Results Using Number of Messages as Dependent Variable

	Model 1	Model 2
Constant	8.804*** (0.085)	12.873*** (0.119)
Control Variables		
Period-beginning	0.555*** (0.107)	0.566*** (0.100)
Period-middle	0.649*** (0.106)	0.755** (0.101)
Duration	1.023* (0.011)	1.024* (0.103)
Direct Effects		
H1b: Trigger type-Problem		0.675*** (0.084)
H2b: Task topic-Tactical		0.649*** (0.094)
H3b: Project type-IM		1.159* (0.085)
Log Likehood	-758.41	-737.00
LR Chi-squire	36.66***	79.49***

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

The findings regarding the amount of communication are as follows. We find that problem-triggered tasks had significantly less communication than opportunity-triggered tasks, thus supporting H1b. Similarly, tactical tasks resulted in significantly less communication than strategic tasks, thus supporting H2b. However, following the same pattern as H3a, tasks in IM projects were found to have significantly more communication than those in ERP projects. So H3b was not supported.

5.3. Post hoc Assessment of Mediating Effect of Number of Participants

In the previous section, we discussed the impact of task characteristics on the amount of communication. However, it is reasonable to expect that more participants will lead to more communication. Therefore, we examined whether the task characteristics result in more communication even when controlling for the number of participants. 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 [71]. We examined the total and direct effects of each task characteristic and project type on the number of messages and the indirect effects through the number of participants. This method allows for testing each IV in a separate model. In each

model, we selected one task characteristic as the main IV to be tested and treated the others (together with the two control variables) as covariates to both the DV and the mediator. Table 5 summarizes the mediation test results.

Table 5. Mediating Effect of Number of Participants

	Total Effects		**Direct Effects**		**Indirect Effects**		
IV	Coefficient	T-value	Coefficient	T-value	Point Estimate	Bias-corrected bootstrap 95% Confidence Intervals	
						Lower	Upper
Trigger Type-Opportunity	2.938	4.053***	0.170	0.393	2.768	1.500	4.430
Task Topic-Strategic	4.183	4.971***	-.051	-.099	4.234	2.297	6.628
Project type-ERP	-1.621	-2.224*	0.702	1.630	-2.324	-3.819	-1.128

* $p<0.05$, *** $p<0.001$

From the results we can see that trigger type, task topic and project type all had significant total effects on the number of messages ($p < 0.001$, $p < 0.001$ and $p < 0.05$ respectively). 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 all the three task characteristics were significant, with the point estimates of 2.768, 4.234 and (-2.324) respectively, and 95 percent bias-correct bootstrap confidence intervals of 1.500–4.430, 2.297–6.628 and (-3.819)–(-1.128) respectively. Therefore, we can conclude that the number of participants fully mediated the impacts of the two task characteristics and project type on the amount of communication.

6. Discussion

The primary goal of this study was to investigate the impacts of different task characteristics on participation behavior in community-based FLOSS development tasks. Using

decision-making tasks as a particularly important category 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 participation. As we expected, problem-triggered tasks and tactical tasks both involved fewer participants and discussions than opportunity-triggered tasks and strategic tasks respectively.

On the other hand, we expected that projects that develop simpler software would involve fewer participants (H3a) and fewer messages (H3b) than those that develop more complex software. However, our findings show that tasks in simple projects included more participants and discussions, counter to our hypotheses. Since the mediation test showed that the number of participants fully mediates the relationship between the task characteristics and the number of messages, we only focus on the number of participants in the following analysis. To further explore the unexpected findings for H3a/b, we ran a *post hoc* analysis looking at the interaction effects between project type and task topics, and between project type and trigger type on the number of participants. The results are summarized in the third column in Table 3. To present the interaction effects more clearly, we plotted them in Figure 1 and Figure 2 separately.

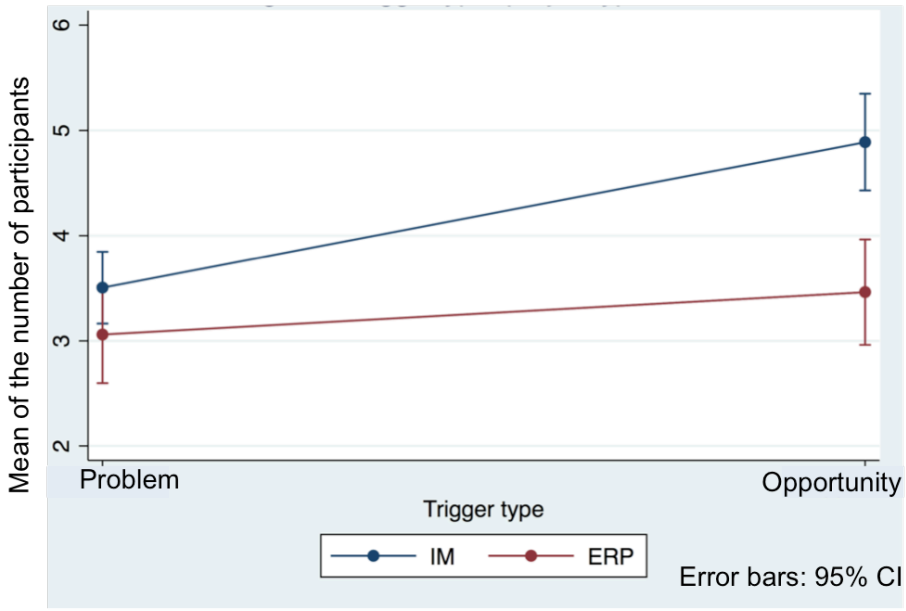

Figure 1. Interaction Effects between Task Triggers and Project Types

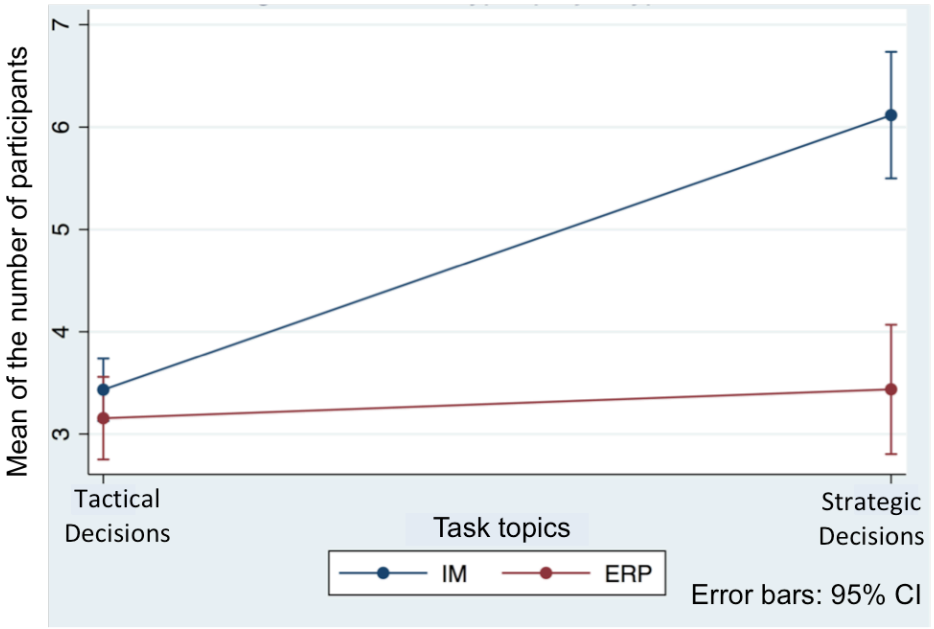

Figure 2. Interaction Effects between Task Topics and Project Types

The results show that opportunity-triggered decisions in the IM projects involved more participants than problem-triggered decisions ($p < 0.05$) as expected, while there was no significant difference in the ERP projects. Similarly, strategic decisions in the IM projects

involved more participants than tactical decisions ($p < 0.001$) as expected, while there was no significant difference in the ERP projects. In other words, H1a regarding the difference between problem- and opportunity-triggered tasks and H2a regarding the difference between tactical and strategic tasks seem to hold true only for the IM projects, but not for the ERP projects. These differences explain why IM projects had more participants than ERP, counter to H3a.

These contrary results led to a revision in our thinking about the drivers for participation in FLOSS development tasks. Our initial hypotheses were based on a consideration of the demand facing the project for increased knowledge from more participants as an input to finish a task. We suggest that for problem-triggered tasks and tactical tasks, the needs of the tasks (e.g., urgency or technical skills need to finish the task) drive individuals' participation in these tasks. Regardless of the project type, problem-triggered and tactical 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 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. To offer a possible explanation for the interaction effect of project type, we speculate that participation in these decision tasks and software development tasks in general is driven not just by the demand of the task but also by the supply of different actors interested in the project.

Prior work has suggested the type of software developed by a project or software application domain as an important factor that influence user interests, define target population types and size, and impact development activities [10, 11, 64]. In line with these studies, we

posit that the two different types of software developed by IM and ERP projects help define different actors interested in the projects.

Specifically, we suggest that in the IM projects, the majority of participants understand the general workings of the whole software (it being simpler) 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 use the IM software personally, hence their interest in contributing to the project. As opportunities represent areas where new features may be added, which 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 have larger scale and complexity (indeed, this complexity was the reason for selecting these projects). Further, these systems exhibit a modular system design [72] with modules for different kinds of functions. Based on findings from prior research [73], we postulate that developers specialize their development efforts in just a few related modules of ERP systems based on their functional and technical interests. 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 skills and interests needed to contribute to each task. 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. And 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 the decision task is a tactical one 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 project complexity 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 [74].

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, participation in different software development tasks reflects the participants' interests and abilities as much as the characteristics of the tasks.

7. Implications and Conclusions

This research contributes to the literature and practice in several ways.

7.1. Research Implications

First, this research explores participation based on task characteristics and investigates how different task characteristics and project type influence participation in terms of the number of participants and the number of messages 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 behaviors, i.e., participation in software development tasks. 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 our *post hoc* analysis of the predictors of communication, we found that the impacts of task characteristics on communication were fully mediated by the number of participants.

Third, our research contributes to FLOSS literature by providing useful insights into the moderating role of project types in the relationship between task types and participation in these tasks. We treated project types as complex projects versus simple projects and hypothesized that decisions in complex projects will involve more participants than those in simple projects. However, our contradictory results using ERP (i.e., complex) and IM (i.e., simple) projects suggested another possible project-type explanation for the projects we selected. That is, the software application domain defines the supply of different resources, which interacts with the different task characteristics to influence participation in FLOSS development tasks. Prior research has examined its direct impact on project outcomes such as project attractiveness and project success [11, 75]. However, little 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).

7.2. Practical Implications

The results of this research have several important practical implications for FLOSS participants and leaders as well. First, our findings suggest that different task types 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 many others who are involved in the same discussion.

7.3. Limitations and Future Work

Although we have discussed a number of theoretical and practical insights generated from this research, we acknowledge some limitations that might affect the generalizability of the research results.

First, we selected only decision-making tasks to test the hypotheses. We argue that this task type is representative of other FLOSS development tasks, but the choice does limit the generalizability of the results. For example, we did not study negotiation tasks that involve conflict in our research. Although negotiation tasks are less common in FLOSS development than decision tasks, they have attracted researchers' interests in recent years [76, 77]. Future research should apply the framework of this research to other types of tasks in McGrath [29]'s task circumplex.

Second, the task processes (i.e., decision 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 the non-inclusion of synchronous discussion fora, such as Internet Relay Chat, Instant Messaging or phone calls. Future research should examine more systematically how people participate in these synchronous communication channels and what roles they play in development practices.

Another limitation of this research is the small sample size (i.e., five projects in two categories and 269 decision episodes). Having a tractable sample size enabled us to conduct very intensive and detailed manual coding for identifying decision-making episodes and different task triggers and task topics, and increased our understanding of the task types from different perspectives. However, it limited the types of statistical analysis we could run with our data. For example, we only used two types of FLOSS projects (i.e., IM projects vs. ERP projects) to test H3, which limited our analysis of the differences of participation between projects that develop simpler products and those develop more complex ones (or alternately, between more and less attractive projects), thus limiting the generalizability of the result. Future research should apply the framework of this research to a larger and more representative sample of FLOSS projects.

Fourth, our H3 proposes tasks in more complex projects involve more participants and more messages than those in simpler projects. We purposefully chose two different types of projects and use project types as the proxy to project complexity. Although we believe that our argument that ERP projects are more complex than IM projects is reasonable, it would be better to use some more objective measures to retest the hypothesis regarding project complexity and participation. For example, Colazo and Fang [46] used software structural complexity to indicate

project complexity and measured it by calculating the project's average McCabe's cyclomatic complexity factor.

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.

References

[1] B. Xu, D.R. Jones, and B. Shao, *Volunteers' involvement in online community based software development*. Information & Management, 2009. **46**(3): p. 151 – 158.

[2] C. Zhang, J. Hahn, and P. De, *Research note—Continued participation in online innovation communities: does community response matter equally for everyone?* Information Systems Research, 2013. **24**(4): p. 1112-1130.

[3] B. Xu and D.R. Jones, *Volunteers' participation in open source software development: a study from the social-relational perspective*. ACM SIGMIS Database, 2010. **41**(3): p. 69-84.

[4] G. Robles and J.M.a.M.M. Gonzalez-Barahona. *Evolution of volunteer participation in libre software projects: Evidence from Debian*. in *Proceedings of the First International Conference on Open Source Systems*. 2005. Genova, Italy.

[5] R.P. Bagozzi and U.M. Dholakia, *Open source software user communities: A study of participation in Linux user groups*. Management Science, 2006. **52**(7): p. 1099–1115.

[6] F. Barcellini, F. Détienne, and J.-M. Burkhardt, *A situated approach of roles and participation in Open Source Software Communities*. Human–Computer Interaction, 2014. **29**(3): p. 205-255.

[7] J. Roberts, I.-H. Hann, and S.A. Slaughter, *Understanding the Motivations, Participation, and Performance of Open Source Software Developers: A Longitudinal Study of the Apache Projects*. Management Science, 2006. **52**(7): p. 984-999.

[8] G. Hertel, S. Niedner, and S. Herrmann, *Motivation of software developers in Open Source projects: An Internet-based survey of contributors to the Linux kernel*. Research Policy, 2003. **32**: p. 1159–1177.

[9] Y. Fang and D. Neufeld, *Understanding sustained participation in open source software projects*. Journal of Management Information Systems, 2009. **25**(4): p. 9-50.

[10] K.J. Stewart, A.P. Ammeter, and L.M. Maruping, *Impacts of License Choice and Organizational Sponsorship on User Interest and Development Activities in Open Source Software Projects*. Information Systems Research, 2006. **17**(2): p. 126-144.

[11] C. Santos, G. Kuk, F. Kon, and J. Pearson, *The attraction of contributors in free and open source software projects*. The Journal of Strategic Information Systems, 2013. **22**(1): p. 26-45.

[12] G. von Krogh, S. Spaeth, and K.R. Lakhani, *Community, joining, and specialization in Open Source Software innovation: A case study*. Research Policy, 2003. **32**(7): p. 1217–1241.

[13] K. Wei, K. Crowson, U.Y. Eseryel, and R. Heckman, *Roles and politeness behavior in community-based free/libre open source software development*. Information & Management, 2017. **54**(5): p. 573-582.

[14] U.Y. Eseryel, *IT-enabled knowledge creation for open innovation*. Journal of the Association for Information Systems, 2014. **15**(11): p. 355-432.

[15] U.Y. Eseryel and D. Eseryel, *Action-embedded transformational leadership in self-managing global information technology teams*. The Journal of Strategic Information Systems, 2013. **22**(2): p. 103-120.

[16] K. Crowston, K. Wei, J. Howison, and A. Wiggins, *Free/libre Open Source Software development: What we know and what we do not know*. ACM Computing Surveys, 2012. **44**(2): p. 7:1–7:35.

[17] J. Howison and K. Crowston, *Collaboration through open superposition: A theory of the open source way*. MIS Quarterly, 2014. **38**(1): p. 29–50.

[18] D.J. Campbell, *Task Complexity: A Review and Analysis*. Academy of Management Review, 1988. **13**(1): p. 40-52.

[19] J.D. McKeen, T. Guimaraes, and J.C. Wetherbe, *The relationship between user participation and user satisfaction: an investigation of four contingency factors*. MIS quarterly, 1994: p. 427-451.

[20] S.A. Licorish and S.G. MacDonell, *Exploring software developers' work practices: Task differences, participation, engagement, and speed of task resolution*. Information & Management, 2017. **54**(3): p. 364-382.

[21] I. Zigurs and B.K. Buckland, *A theory of task/technology fit and group support systems effectiveness*. MIS quarterly, 1998. **22**(3).

[22] R.W. Griffin, A. Welsh, and G. Moorhead, *Perceived task characteristics and employee performance: A literature review*. Academy of management Review, 1981. **6**(4): p. 655-664.

[23] X.N. Deng and K.D. Joshi, *Why individuals participate in micro-task crowdsourcing work environment: Revealing crowdworkers' perceptions*. Journal of the Association for Information Systems, 2016. **17**(10): p. 648.

[24] C. Speier, I. Vessey, and J.S. Valacich, *The Effects of Interruptions, Task Complexity, and Information Presentation on Computer-Supported Decision-Making Performance*. Decision Sciences, 2003. **34**(4): p. 771-796.

[25] R.T. Nakatsu, E.B. Grossman, and C.L. Iacovou, *A taxonomy of crowdsourcing based on task complexity*. Journal of Information Science, 2014. **40**(6): p. 823-834.

[26] X. Fang, S. Chan, J. Brzezinski, and S. Xu, *Moderating Effects of Task Type on Wireless Technology Acceptance*. Journal of Management Information Systems, 2005-6. **22**(3): p. 123-157.

[27] U.S. Murthy and D.S. Kerr, *Decision making performance of interacting groups: an experimental investigation of the effects of task type and communication mode*. Information & Management, 2003. **40**(5): p. 351-360.

[28] G.L. Stewart and M.R. Barrick, *Team Structure and Performance: Assessing the Mediating Role of Intrateam Process and the Moderating Role of Task Type*. Academy of Management Journal, 2000. **43**(2): p. 135-148.

[29] J.E. McGrath, *Groups: Interaction and performance*. Vol. 14. 1984: Prentice-Hall Englewood Cliffs, NJ.

[30] V. Peñarroja, V. Orengo, A. Zornoza, J. Sánchez, and P. Ripoll, *How team feedback and team trust influence information processing and learning in virtual teams: A moderated mediation model*. Computers in Human Behavior, 2015. **48**: p. 9-16.

[31] J.B. Barlow and A.R. Dennis, *Not as smart as we think: A study of collective intelligence in virtual groups*. Journal of Management Information Systems, 2016. **33**(3): p. 684-712.

[32] K. Crowston, *The bug fixing process in proprietary and free/libre open source software: A coordination theory analysis*, in *Business Process Transformation*. 2015, Routledge. p. 85-116.

[33] K. Crowston and B. Scozzi, *Bug fixing practices within free/libre open source software development teams*. Journal of Database Management (JDM), 2008. **19**(2): p. 1-30.

[34] A.R. Dennis, R.M. Fuller, and J.S. Valacich, *Media, tasks, and communication processes: A theory of media synchronicity*. MIS quarterly, 2008. **32**(3): p. 575-600.

[35] H. Mintzberg, *The Nature of Managerial Work*. 1973, New York: Harper & Row.

[36] C. Smart and I. Vertinsky, *Designs for cirisis decision units*. Administrative Science Quarterly, 1977. **22**(1): p. 640-657.

[37] P. Clarke and R.V. O'Connor, *The situational factors that affect the software development process: Towards a comprehensive reference framework*. Information and Software Technology, 2012. **54**(5): p. 433-447.

[38] J.D. Xu, *Retaining customers by utilizing technology-facilitated chat: Mitigating website anxiety and task complexity*. Information & Management, 2016. **53**(5): p. 554-569.

[39] E.S. Raymond, *The cathedral and the bazaar*. First Monday, 1998. **3**(3).

[40] P. Wayner, *Free For All*. 2000, New York: HarperCollins.

[41] A. Dix, D. Ramduny-Ellis, and J. Wilkinson, *Trigger analysis: Understanding broken tasks*. The handbook of task analysis for human-computer interaction, 2004: p. 381-400.

[42] H. Mintzberg, D. Raisinghani, and A. Theoret, *The Structure of "Unstructured" Decision Process*. Adminstrative Science Quarterly, 1976. **21**(2): p. 246-275.

[43] P.C. Nutt, *Types of organizational decision processes*. Administrative Science Quarterly, 1984. **29**(3): p. 414-450.

[44] J. Cope, *Entrepreneurial learning and critical reflection: Discontinuous events as triggers for 'higher-level' learning*. Management learning, 2003. **34**(4): p. 429-450.

[45] H. Annabi, K. Crowston, and R. Heckman. *Depicting what really matters: Using episodes to study latent phenomenon*. in *International Conference on Information Systems (ICIS)*. 2008. Paris, France.

[46] J.A. Colazo and Y. Fang, *Following the sun: Temporal dispersion and performance in open source software project teams*. Journal of the Association for Information Systems, 2010. **11**(11): p. 684.

[47] W. Scacchi, *Understanding the requirements for developing Open Source Software systems*. IEE Proceedings Software, 2002. **149**(1): p. 24--39.

[48] M. Michlmayr, F. Hunt, and D. Probert. *Release management in free software projects: Practices and problems*. in *IFIP International Conference on Open Source Systems*. 2007. Springer.

[49] R.W. Zmud, *Management of large software development efforts*. MIS quarterly, 1980: p. 45-55.

[50] M. de Reuver, C. Sørensen, and R.C. Basole, *The digital platform: a research agenda*. Journal of Information Technology, 2018. **33**(2): p. 124-135.

[51] J.R. Hackman, *Effects of task characteristics on group products*. Journal of Experimental Social Psychology, 1968. **4**(2): p. 162-187.

[52] M.D. Cohen, J.G. March, and J.P. Olson, *A Garbage Can Model of Organizational Choice*. Administrative Science Quarterly, 1972. **17**(1): p. 1-25.

[53] S.G. Straus, *Testing a typology of tasks: An empirical validation of McGrath's (1984) group task circumplex*. Small Group Research, 1999. **30**(2): p. 166–187.

[54] R.E. Wood, *Task Complexity: Definition of the Construct*. Organizational Behavior and Human Decision Processes, 1986. **37**: p. 60-82.

[55] M. Drury, K. Conboy, and K. Power, *Obstacles to decision making in Agile software development teams*. Journal of Systems and Software, 2012. **85**(6): p. 1239-1254.

[56] N.B. Moe, A. Aurum, and T. Dybå, *Challenges of shared decision-making: A multiple case study of agile software development*. Information and Software Technology, 2012. **54**(8): p. 853-865.

[57] D.M. German, *The GNOME project: A case study of open source, global software development*. Software Process: Improvement and Practice, 2003. **8**(4): p. 201–215.

[58] S. Krishnamurthy, *Cave or Community? An Empirical Examination of 100 Mature Open Source Projects*. First Monday, 2002. **7**(6).

[59] P. Liu and Z. Li, *Task complexity: A review and conceptualization framework*. International Journal of Industrial Ergonomics, 2012. **42**(6): p. 553-568.

[60] B.E. Mennecke, J.S. Valacich, and B.C. Wheeler, *The Effects of Media and Task on User Performance: A Test of the Task-Media Fit Hypothesis*. Group Decision and Negotiation, 2000. **9**: p. 507-529.

[61] S.G. Straus and J.E. McGrath, *Does the medium matter? The interaction of task type and technology on group performance and member reactions*. Journal of Applied Psychology, 1994. **79**: p. 87–97.

[62] J.A. Espinosa, S.A. Slaughter, R. Kraut, and J.D. Herbsleb, *Team knowledge and coordination in geographically distributed software development*. Journal of Management Information Systems, 2007. **24**(1): p. 135–169.

[63] J.-G. Park and J. Lee, *Knowledge sharing in information systems development projects: Explicating the role of dependence and trust*. International Journal of Project Management, 2014. **32**(1): p. 153-165.

[64] S. Comino, F.M. Manenti, and M.L. Parisi, *From Planning to Mature: On the Success of Open Source Projects*. Research Policy, 2007. **36**(10): p. 1575-1586.

[65] M.S. Poole and C.L. Baldwin, *Developmental Processes in Group Decision Making*, in *Communication and Group Decision Making*, R.Y. Hirokawa and M.S. Poole, Editors. 1996, SAGE: Thousands Oaks, CA. p. 215–241.

[66] M.S. Poole, *Decision development in small groups II: A study of multiple sequences in decision making*. Communication Monographs, 1983. **50**(3): p. 206-232.

[67] M.S. Poole and J. Roth, *Decision Development in Small Group IV: A typology of Group Decision Paths*. Human Communication Research, 1989. **15**(3): p. 323-356.

[68] H.W. Ryan, *Managing development in the era of large complex systems*. Information Systems Management, 1999. **16**(89-91).

[69] J.G. Orme and T. Combs-Orme, *Multiple regression with discrete dependent variables*. 2009: Oxford University Press.

[70] M.A. Stanko, *Toward a theory of remixing in online innovation communities*. Information Systems Research, 2016. **27**(4): p. 773-791.

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

[72] D. Paulish, *Architecture-centric software project management*. 2002, Boston, MA: Addison-Wesley.

[73] T.P. Liang, J. Jiang, G.S. Klein, and J.Y.C. Liu, *Software quality as influenced by informational diversity, task conflict and learning in project teams*. IEEE Transactions on Engineering Management, 2010. **57**(3): p. 477-487.

[74] C. Williams, *OpenSSL Heartbleed: Bloody nose for open-source bleeding hearts*, in *The Register*2014.

[75] K. Crowston and B. Scozzi, *Open source software projects as virtual organizations: Competency rallying for software development*. IEE Proceedings Software, 2002. **149**(1): p. 3--17.

[76] A. Filippova and H. Cho. *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*. 2016. ACM.

[77] J. Gamalielsson and B. Lundell, *Sustainability of Open Source software communities beyond a fork: How and why has the LibreOffice project evolved?* Journal of Systems and Software, 2014. **89**: p. 128-145.