Decision-Making Processes and Team Performance in Self-Organizing Virtual Teams: The Case of Free/Libre Open Source Software Development Teams

Kangning Wei
School of Management
Shandong University
Jinan, Shandong, China
kwei@sdu.edu.cn

Kevin Crowston
School of Information Studies
Syracuse University
Syracuse, NY 13244, U.S.A
crowston@syr.edu

Robert Heckman
School of Information Studies
Syracuse University
Syracuse, NY 13244, U.S.A
rheckman@syr.edu

Qing Li
School of Information Studies
Syracuse University
Syracuse, NY 13244, U.S.A
qli03@syr.edu

Acknowledgement: This work is partially funded by US National Science Foundation Human and Social Dynamics Program Grant 05-27457.

Decision-Making Processes and Team Performance in Self-Organizing Virtual Teams: The Case of Free/Libre Open Source Software Development Teams

Abstract

Effective decision making is critical to team effectiveness. We examine decision making in the setting of self-organizing virtual teams, a setting that we expect to pose particular problems for effective decision making. A self-organizing team must develop effective practices in the absence of a formal organizational structure that guides the practices. A virtual team's reliance on technological support to span temporal and organizational discontinuities makes an effective decision-making process more difficult to achieve. We examine decision-making processes and their relation to team performance in Free/Libre Open Source Software (FLOSS) teams, chosen as extreme examples of self-organizing virtual teams. We identified six paths that describe the decision-making process. Further analysis revealed that diversity in decision-making paths appeared to be related to differences in task types and group performance. The study provides insight into decision making in FLOSS teams and has implications for decision making in self-organizing virtual teams more generally.

Keywords: Decision making process, self-organizing; virtual team; team performance

1. Introduction

Decision making is an important component of team behavior (Guzzo and Salas 1995), and as such is an important and extensively-studied topic in small group research (Kerr and Tindale 2004; Rousseau et al. 2006). We examine decision-making processes within self-organizing virtual teams. By process, we mean an interrelated sequence of events that occur over time leading to an organizational outcome of interest (Robey et al. 2000), in this case, to a group decision. We define a group decision as one that binds the team as a whole to a future course of action and on which the team reaches (perhaps implicit) consensus (Kerr and Tindale 2004), in contrast to decisions that affect only the individual making the decision. Much of the past work on small-group decision making has tended to focus on linear, antecedent-consequence-type relations, which simplifies a complex set of processes (Kerr and Tindale 2004). However, researchers have argued that it is not possible to simply relate inputs to outputs while ignoring the process by which a group arrives at a decision (Guzzo and Salas 1995; Poole and Baldwin 1996), motivating our focus on the process of decision making.

By virtual teams, we mean those whose members interact primarily or even exclusively via information and communication technologies (ICT). Virtual teams are valuable to organizations because of their ability to bridge discontinuities of time, geography, organization and culture, enabling access to and transfer of knowledge across these boundaries (Duarte and Snyder 2001). They have become increasingly common within and among organizations (Martins et al. 2004; Schiller and Mandviwalla 2007) with the growth of inter-organizational alliances and advances in ICT. However, virtual teams face many challenges to effectiveness, as the extensive use of ICTs changes the way

members interact and so how they make decisions (Kiesler and Sproull 1992). The lack of shared context and numerous discontinuities faced by virtual team members can hamper decision making (Watson-Manheim et al. 2002). For example, a lack of common language or agreement about role structure may hinder member's understandings of tasks and relationships, thus bringing conflicts among group members and making it difficult to reach consensus (Armstrong and Cole 2002).

In this paper, we examine decision-making process in a particular type of virtual team, namely self-organizing virtual teams. Self-organizing teams are teams that have discretion in deciding how to carry out and allocate tasks within the team (Langfred 2004). We are interested in self-organizing teams for two reasons. On the one hand, self-organization seems to be a common response to the setting of virtual work. Taking advantage of the ability of the technology to span boundaries, members of virtual teams may come from different geographically-dispersed units within a single organization, or from entirely different organizations, and so have no leaders with formal authority over all members. Even if there are appointed leaders, they may lack influence over team members due to the organizational or physical separation. As a result, responsibility and decision making are often delegated downward. For example, upper-level managers may set general strategic directions, but leave day-to-day decisions about tasks and tactics to the team members. Indeed, some researchers argue that to make virtual teams successful, it is important to provide team members with as much autonomy as possible (Bourgault et al. 2008), which increases team's flexibility to adapt to changing situations (Manz and Sims 1987; Druskat and Wheeler 2003; Douglas et al. 2006). On the other hand, self-organizing teams often face challenges brought by the absence of formal role structures that grant authority. Previous studies link decision making in particular with leadership and authority (e.g. Eisenhardt 1989; Srivastava et al. 2006; Boone and Hendriks 2009)). However, in self-organizing teams, the role of managers moves towards facilitation instead of decision making, complicating development of effective decision-making processes.

Despite the growing importance of self-organizing and virtual teams and the likely differences in their decision-making practices, our brief review of the extensive research on group decision making reveals relatively little research that has empirically investigated the processes of group decision making in this setting. To address the limitations of prior research, we present a study of decision-making processes in an extreme setting, a setting deliberately chosen to bring out the impacts of virtual self-organizing on decision making, namely Free/Libre Open Source Software (FLOSS) development teams. We addressed the following research questions in this research:

1. How are decision-making processes structured in FLOSS development teams?
2. How do contingency factors such as task type and task complexity affect the decision-making processes adopted in FLOSS development teams?
3. How are decision-making processes related to team performance in FLOSS development teams?

2. Literature Review

In this paper, we chose to examine decision making through an investigation of decision-making processes. We made the choice of a process theory for two reasons. First, as noted above, the phenomenon of decision making is complex, making it difficult to draw conclusions by looking only at inputs and outputs. It is rather necessary to "open up" the black box and examine the process by which the decision was made. Furthermore, a significant problem in studying group decision making is that it is a multi-level phenomenon involving not only individual participants but a group outcome. A process theory provides a link because individuals carry out the various steps in the process, while the process as a whole can have a group-level outcome (Crowston 2000).

Group decision making in particular has been analyzed using a process approach, viewing the decision as structured interactively through communication (Poole and Baldwin 1996). As Hirokawa (1992) pointed out, to understand decision-making dynamics, it is very important to examine interaction patterns in group decision making: interaction is the essence of group decision making (Poole and Baldwin 1996). Poole and Roth (1989) argued that a decision is not a discrete, independent event but rather involves a series of activities and choices. In order to examine the nature of decision development, it is necessary to generate descriptive data to understand how the group interaction creates, elaborates and finalizes the decision over time.

2.1 Theories of Decision-Making Processes

A number of frameworks have been proposed to describe the phases of decision-making processes. A phase is defined as "a period of coherent activity that serves some decision-related function, such as problem definition, orientation, solution development, or socio-emotional expression" (Poole and Baldwin 1996, p. 216). Early studies proposed normative models to describe how decisions are made in a unitary sequence of decision phases (Poole and Roth 1989). Many of these models stemmed from Simon's rational model of individual decision making (Simon 1965). According to Simon (1965), after a decision opportunity is identified, three phases are involved in a decision: "intelligence" (all relevant information is collected), "design" (all the possible alternatives are generated) and "choice" (the final decision is made). Fisher (1970) proposed a model with four phases of decision making: orientation, conflict, emergence and reinforcement. Though the two models propose different phases, both assume that groups follow a systematic logic to reach decisions (Miller 2008). However, in a study of decision development in small groups, Poole and Roth (1989) pointed out that the normative models are not adequate to capture the nature of decision-making sequences. Simon (1976) himself noted that decision makers do not always follow rational models.

An alternative to a normative sequential model is a non-phasic model, such as the garbage can model, proposed by Cohen and his colleagues in a study of organizational anarchies (Cohen et al. 1972). This model proposes that decision making is a process wherein problems, solutions, participants and choices dump together in a relatively independent fashion, and a decision is made

when elements from these four streams coincide under certain organizational structures. In this way, the decision reached is not the result of logical search but merely a coincidence of problem and solution with sustained attention (Miller 2008). Other researchers suggest that decision process are best analyzed with continuous models (Seeger 1983; Cissna 1984). Although non-phasic models do capture the fact that decisions are not always orderly, Poole and Roth (1989) argued that it is an over-reaction to assume that there are no coherent periods of group work and the continuous approach lacks compelling models to support it.

Acknowledging the fact that there may be more than one sequence of decision development and the dynamic nature of decision making, Poole and his colleagues proposed another class of phase models, multiple sequence models (Poole 1981; Poole 1983; Poole and Roth 1989; Poole and Roth 1989). These models argue that groups may follow different developmental sequences, depending on contingency factors of the decision situation. Besides the sequence mentioned in the unitary sequence models, groups may also follow "more complicated paths in which phases repeat themselves and groups cycle back to previously completed activities as they discover problems or encounter problems. Also, possible are shorter, degenerate sequences containing only part of the complement of unitary sequence phases" (Poole and Baldwin 1996, p. 217). Based on a study of 47 group decisions, Poole and Roth (1989) discovered 11 different decision paths that fell into three main groups: unitary sequences, complex sequences and solution-centered sequences. Similarly, by studying 25 strategic decision processes, Mintzberg, et al (1976) suggested 7 different types of paths under a general three-phase decision-making process. In general, comparing these three classes of decision-making processes, multiple sequence models are advantageous because they not only capture the complexity of the decision-making process, but also provide a systematic model to study the complexity (e.g. Mintzberg et al. 1976; Poole and Roth 1989). Further, multiple sequence models provide normative models additional strength to help practitioners adapt to changing demands (Poole 1983; Poole and Baldwin 1996), which can be useful as it provides a framework for structuring analyses of decision processes by providing terminology and a basis for comparison between diverse processes (Poole and Baldwin 1996). We therefore adopted this approach in this paper.

Another important aspect to consider when investigating decision-making processes is the impact of context on the process. As noted above, under different circumstances, groups might follow different paths as they make decisions. Poole and Roth (1989) proposed a contingency model to describe how team members structure their decision-making process subject to the contingency factors of the decision situation. Two sets of contingency factors were found to be effective in predicting decision paths and the complexity of the paths: task characteristics and group internal structures that define work relationships in the group (i.e., cohesiveness, power concentration, conflict history and size of the decision making group). For example, task complexity has been studied extensively in decision-making literature (Payne 1976; Speier et al. 2003; Kamis et al. 2008), where it has been shown as an important predicator of decision processes. According to Wood (Wood 1986),

task has three essential components: products, required acts and information cues. Based on the relationships between required acts and information cues, task complexity has multiple dimensions in terms of component complexity (i.e., the number of distinct acts and information cues required in the performance of the task), coordinative complexity (i.e., the nature of relationships between acts, information cues and products) and dynamic complexity (i.e., changes in the states of the world which influences the coordinative complexity) (Wood 1986). Complex tasks require more acts and contain more information cues than simple tasks, with these cues and acts are highly interdependent (Wood 1986). Therefore when processing complex tasks, decision makers need to employ different problem-solving strategies to reach decisions (Payne 1976).

2.2 Technology-Use and Decision-Making Processes

A further factor affecting decision processes in our context is the use of ICT. There is considerable evidence that ICT affects the group communication pattern, which reflects changes in the group internal structures mentioned above. For example, Baltes et al. (2002) found that computer-mediated group decision making achieved more equal participation since group members felt freer to express their opinions. The use of ICT has provided both advantages and disadvantages for information sharing in technology-supported groups. In particular, virtual teams can facilitate information sharing by increasing the accessibility to diverse team members and their information (Sole and Applegate 2000; Majchrzak et al. 2005). On the other hand, a virtual team's geographic, temporal, organizational and cultural discontinuities (Watson-Manheim et al. 2002) may create problems that can hinder information sharing among team members that are hard to overcome with ICT.

Most studies of technology and decision making have focused on the synchronous, text-based decision-support system, such as GDSS (Sambamurthy and Chin 1994; Barkhi and Pirkul 2004). Even though email has been used extensively for many years and remains one of the most frequently used communication tool in organizations, less research has examined the impact of such asynchronous technologies. By examining 25 years of the business communication and management literature, Berry (2006) concluded that computer-mediated asynchronous communication systems such as email enhance organizational communication and decision-making effectiveness because it increases team member participation, offers flexibility over time and distance, creates additional time for reflection and archives all discussions. However, these studies have primarily examined the advantages of using asynchronous technologies instead of face-to-face communication, and their impact on decision making outcomes, rather than examining effects on the decision-making process itself (Schmidt et al. 2001; Benbunan-Fich et al. 2002). Baltes et al. (2002) pointed out that few studies have examined how email actually supports and influences group-decision processes (Olikowski and Yates (1994) being one exception).

2.3 Decision-Making Process and Team Performance

Finally, we consider the link from team decision-making processes to team-performance outcomes. The dominant paradigm in recent group decision making research is information processing (Eisenhardt 1989; Kerr and Tindale 2004). This body of literature argues that team members are less-than-optimal users of information, for example ignoring information that is not widely shared (Kerr and Tindale 2004). The extent to which decision-making procedures involve adequate gathering and analysis of relevant information (called procedural rationality) has been found to be associated with decision effectiveness (Simon 1976). For example, Dean and Sharfman (1996) suggested a close link between decision-making processes and decision effectiveness in their studies examining the influence of procedural rationality and political behavior on decision success. They found that decisions made after information collection and analysis are more effective than the others. Hirokawa (1985) observed clear differences between successful and unsuccessful teams in terms of decision-making phases. The successful teams tried to analyze a problem in depth before moving into a solution-searching period, while the unsuccessful teams tended to skip this analysis phase. Time in the solution-development phase tended to peak in the middle period of the successful team, unlike in the unsuccessful team. Poole and Holmes (1989; 1995) examined the relationship between decision path and three outcome variables, consensus change, perceived decision quality and decision scheme satisfaction, and showed that processes that most resembled logical normative sequences generate better outcomes than the others.

In summary, the literature has proposed several sequence models for group decision making, which provide a basis for comparing observed decision-making processes. A few studies examined the detailed communications in groups and used these as a basis for identifying phases in the decision-making process. These descriptions provide a starting point for our analyses. However, there are few studies that examine the details of decision-making processes in self-organizing virtual contexts. The current study is designed to address these gaps in the literature.

3. Setting: Free/Libre Open Source Software (FLOSS) Development Teams

To bring out the impacts of virtual self-organizing on decision-making processes, we chose to study decision-making processes in an extreme setting, namely Free/Libre Open Source Software (FLOSS[1]) development teams. FLOSS development teams are a good setting for our study for several reasons.

First, the projects are teams, as members have a shared goal of developing a product and a user base to satisfy. Indeed, FLOSS development projects have attracted great interest among management researchers because this mode of work provides a new model of innovation (von Krogh and von Hippel 2006). Furthermore, teams have a shared social identity. Team members are interdependent in

[1] FLOSS is a broad term used to embrace software developed and released under a license allowing inspection, modification and redistribution of the software's source code. While there are important differences between free software and open source software (Kelty 2008), their software development practices are largely similar, hence our use of an umbrella term.

their tasks and core members know and acknowledge each other's contributions. Second, the teams are generally virtual, as developers contribute from around the world, meet face-to-face infrequently if at all, and coordinate their activity primarily by means of ICT (Raymond 1998; Wayner 2000). Discontinuities among team members make such emergence and indeed any kind of consistent decision process seemingly harder to attain, yet effective teams seem to have developed productive ways of making decisions. As well, in contrast to Group Decision-Supported Systems (GDSS), FLOSS teams make most of decisions in an asynchronous fashion using email or message fora. Third, many FLOSS teams are self-organizing, without formally appointed leaders or explicit indications of rank or role, raising the question of how decision-making (and other) practices and authority emerge. Previous research has found that self-assignment is the most common task assignment mechanism in FLOSS teams (Crowston et al. 2007).

These features make FLOSS teams extreme examples of self-organizing virtual teams, which is appropriate for our goal of understanding the implications of this mode of organization. But even though they are extreme cases, they are not inconsistent with what many organizations are facing in recruiting and motivating professionals and in developing distributed teams. As Peter Drucker put it, "increasingly employees are going to be volunteers, because a knowledge worker has mobility and can go pretty much every place, and knows it… Businesses will have to learn to treat knowledge workers as volunteers" (Collins and Drucker 1999). Other researchers have treated FLOSS teams as a template from which tomorrow's organizational leader can learn. As Markus et al. (2000) argue "there is a relatively high degree of correspondence between the open-source movement and popular depictions of the organization of the future and the virtual networked organization. Therefore, the open-source movement provides some suggestions about how management must change in the years ahead" (p. 25).

3.1 Contextual Factors Affecting FLOSS Decision-Making Processes

Given the important roles of contextual factors in the decision-making process, we next discuss FLOSS team technology use, group internal structures and task characteristics (Poole and Roth 1989). FLOSS development is usually organized in virtual teams supported by ICT. A variety of asynchronous communications tools are used by participants to support their activities, including decision making, such as project email lists, discussion fora, project websites, bug trackers and source code repositories (Scacchi 2007). While they are available, many projects discourage the use of synchronous media, such as chat or telephone, since such interactions do not provide useful records for other participants to review. So-called "forges", websites such as SourceForge, provide free access to bundles of tools, facilitating project development. An important characteristic of FLOSS development is its high modularity in software development which enables parallel development (Rossi 2004). The use of source code repositories such as Concurrent Version System (CVS),

Subversion or Git facilitate simultaneous parallel development, providing a flexible working environment that minimizes the level of interdependence and needed coordination among developers.

We consider the structure of FLOSS teams. In most teams, a small core group oversees the overall design and contributes the bulk of the code. Other developers play an important role by contributing bug fixes, new features or documentation, by providing support for new users and filling other roles in the teams. Core group membership can bestow some rights, including the right to have input on group decisions, such as what features should be integrated in the release of the software, when and how to empower other code maintainers, or to "pass the baton" to the next volunteer (Raymond 1999). However, in comparison to traditional organizations, more people can share power and be involved in group activities. In most projects, anyone with enough interests and skills can access the code, contribute patches, make suggestions to the group and participate in important decisions.

Previous research has found that task type affects team interaction (Straus and McGrath 1994; Mennecke et al. 2000; Fang et al. 2005-6). For our research, we classified the tasks in FLOSS development into two broad categories: software-modification (SM) and non-software (NS) tasks. SM decisions involved decisions about the primary work of the team, that is, software development, including decisions such as bug fixes, additions of new features or product enhancements through a change in software. These are defined as group decisions because the source code is a shared product of the group and changes to the code thus affect all developers. NS decisions involved other group decisions, such as strategic direction for the project, organizational and legal issues, or alliances and partnerships. The SM task process is relatively well-structured as it is based on specific routines of software development procedures, such as design and testing (Hauptman 1986). High modularity of FLOSS source code minimizes the required coordination efforts among developers. In contrast, NS tasks face more uncertainty and so need more discussion and participation from the developers. Hence, we expected the decision patterns to be different between processing SM decision tasks and NS tasks.

Finally, we also examined task complexity through our choice of projects. In software development, task complexity varies greatly depending on not only the characteristics of the software task itself such as size and internal structure, but also on the coordination challenges imposed by external factors such team structure (Espinosa et al. 2007). We expected that decisions about software and projects with more external constraints, such as legal requirements, would be more complex because development requires understanding of the constraints.

3.2 Decision-Making in FLOSS Teams

Though there are some studies that examine decision-making practices adopted by FLOSS teams, very few of them have actually looked at the decision processes in detail. One common topic in studies of FLOSS teams' decision making is decision style, which depends on the hierarchy of the developers. As Gacek and Arief (2004) pointed out, a stricter hierarchy differentiates between levels of developers and generates a more centralized power structure, while a looser hierarchy treats

developers on a similar level and implies a decentralized decision-making process. So, different projects might have different decision making styles. For example, in Linux, Linus Torvalds is reported to have originally made most of the key decisions (Moon and Sproull 2000). Such a decision style has been characterized as a "benevolent dictatorship" (Raymond 1998). At the other extreme are teams with a decentralized communications structure and more consultative decision-making style. Some teams, such as the Apache httpd web-server project, are reported to try to reach "rough" consensus in decisions (Fielding 1999). German (2003) described decentralized decision making in the GNOME project. Committees and task forces composed of volunteers are created to complete important tasks, flattening the organizational structure of the project and allowing broader participation in the process. In addition, participation in decision making might change over the life of the project. Fitzgerald (2006) suggested that a small group will control decision making early in the life of a project, but as the project grows, more developers will get involved.

4. Data and Methods

Given the current state of the literature on decision-making processes in self-organizing virtual teams, we adopted an exploratory approach to the research (Yin 2003). We studied six FLOSS project teams to understand the decision-making process in FLOSS development. The following section describes our project selection strategy, data collection and analysis method.

4.1 Project Selection

We sought to pick projects for our study that would provide a meaningful basis for comparison across contextual features and that would illuminate the relationship between decision making and outcomes. FLOSS projects are quite diverse, ranging from large widely-used systems such as the GNU/Linux operating system and user applications such as Firefox to more specifically focused applications such as system development or management tools. To control unwanted systematic variance, we chose projects that were roughly similar in age and in potential market size, and that were all at production/stable development status. Projects at this stage have relatively 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 decision making, we selected projects that were all hosted on SourceForge, a popular FLOSS development site that provides a consistent mix of tools.

From our literature review, we concluded that a satisfactory explanation of the decision-making processes of self-organizing virtual teams must take into account task characteristics and team structural elements. To manipulate task complexity, we selected projects that developed two different kinds of software applications: three projects that developed Enterprise Resource Planning (ERP) systems (Compiere, WebERP and Apache OFBiz) and three that developed Instant Messenger (IM) clients (Gaim, currently known as Pidgin, aMSN and Fire). Tasks in ERP projects were expected to be

more complex than those in IM projects since they have high software code interdependencies, and many external constraints such as accounting rules and legal reporting requirements. ERP software and IM clients also have different kinds of users, which again affects the nature of team interactions. ERP software targets companies. Because these companies depend on the software for crucial company operations, they are likely to dedicate developers or pay for consultants to work on the project or provide resources in other ways, such as by purchasing a service contract. On the other hand, the users of IM clients are individuals, whose involvement in the project is often less extensive and motivated by personal interest.

The final factor is project performance. To address this factor, we picked projects with different levels of performance. Project effectiveness is a multi-dimensional construct, including the project's outputs, team member satisfaction and continued project activities (Hackman and Morris 1978). We therefore applied the multivariate approach to effectiveness in the FLOSS context suggested by Crowston et al. (2006) aiming to discover a rank order of the projects within the IM and ERP categories. Project outputs were measured by downloads and page views. Developer satisfaction was measured through developer numbers and participation on the developer mailing lists. Since projects within a single topic category are potential competitors, comparisons of outcomes such as downloads between these projects are valid. For example, a user who needs an ERP system could potentially choose any of the three, so it can be argued that the one with more users has been more effective in meeting user demands. On the other hand comparing downloads of an ERP system to downloads of an IM client says little about either's effectiveness, since they have such different potential users.

The performances of the six projects were assessed using data collected by the FLOSSmole project (Howison et al. 2006) from the project establishment in SourceForge until around March 2006. In general, on three out of the four measures (downloads, page views and mail participants), Gaim performed obviously better than aMSN, which in turn was slightly better than Fire. The three IM projects had no big difference on the number of developers. As to ERP projects, Compiere also performed obviously better than OFBiz and WebERP on three out of the four measures (developers, downloads and page views). OFBiz performed slightly better than WebERP on these three measures. There was no big difference in mailing list participants among these three projects. Since the small number of cases ruled out statistical analyses at the project level, having a rough ordering of projects in terms of effectiveness was felt to be sufficient. As a result, we identified Gaim and Compiere as the more successful projects, while aMSN and Fire, OFBiz and WebERP were less successful projects. It should be noted though that all of these projects were successful during the period studied, just at varying levels. However, since the time of our study, Fire ceased development, while the others has continued, though Gaim is now under a new name, Pidgin. In summary then, cases were selected to control for topic, ICT tools, age and project status and to allow for comparison on task complexity and level of success.

4.2 Data

To find evidence of group decision-making processes, we analyzed the email discourse on the developers' email lists or fora, which are the primary communication venue for the members of these virtual teams. Data were obtained in May 2006 from the FLOSSmole project (http://flossmole.org/). Though we cannot rule out the possibility of off-list discussions occurring through other channels (e.g., IRC, IM, phone or face-to-face meeting), the projects used the email lists as the main communication tool for development, meaning that such discussions would be invisible to numerous developers as well as to us as researchers. Furthermore, our analysis of the mailing list interactions did not reveal references to such off-line discussions, suggesting that this data source provided a complete view of the decision-making process, at least for the decisions made in this venue.

4.3 Unit of Analysis: Decision Episodes

We selected the decision episode as our primary unit of coding and analysis, defined as a sequence of messages that begins with a triggering message (a "decision trigger") presenting an opportunity for choice (such as a feature request or a report of a software bug) and that includes discussion related to the issue and possibly an announcement of the final decision (a "decision announcement", either a statement of the intention to do something or an actual implementation). These messages capture the interactions among group members that constituted the process of making a particular decision.

Decision episodes were identified from the continuous stream of available messages through an initial coding process (Annabi et al. 2008). We first read through the messages until we identified one containing a decision trigger or announcement. Once we had a trigger or announcement message for a decision, we identified the sequence of messages that embodied the group decision-making process around the decision. Teams generally organize their discussions in a discussion thread, occasionally initiating new threads with the same or similar subject line. Therefore, we developed an episode by combining one or more discussion threads that included more than a single message, that used the same or a similar subject line as in the initial message and that discussed the same main issue. The process of identifying messages to include in an episode proceeded iteratively, as two researchers collected messages, shared the process they used with the research team, and revised their process as a result of feedback from the team. Since we were analyzing the messages retrospectively, we could collect all of the messages related to the decision over time.

We then examined the collected messages to identify the message that included the final decision announcement. In some cases, a decision announcement message was intermediate in an episode (i.e., more discussions might follow the decision announcement, possibly resulting in a revised decision announcement). In those cases, we used the final decision announcement message for further analysis. In other cases, we found no decision announcements in the messages responding to a trigger, indicating that the group apparently did not reach a decision on the issue.

Coding for decision triggers, announcements and related messages was inductive, with three independent analysts reading the messages to identify the codes. The inter-coder reliability reached 85% and 80% respectively on decision triggers and announcements. All differences between coders in the process of identifying messages representing decision episodes were reconciled to obtain the sample of episodes for analysis. The result of this initial coding process was a collection of episodes (i.e., sets of related messages) from each project, each episode representing the group discussion that addressed a particular decision.

4.4 Episode Sampling Procedure

To develop a comprehensive understanding of the projects' decision-making processes, we sampled 20 decision episodes from three time periods in each project's life. For each project, the beginning and the ending periods are the first and last 20 decision episodes found as of May 2006 (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 group decisions for a FLOSS project. Figure 1 shows the specific time periods sampled for each project. Note that because projects changed their pace of development during the study period, 20 decision episodes span different temporal durations in different projects and in different periods. In particular, the Fire project ceased development shortly after our study and the pace of decision making slowed towards the end of project, meaning that the 20 episodes sampled covered a much longer time period for this project than for the others. In this paper, the 60 episodes for each project are analyzed together.

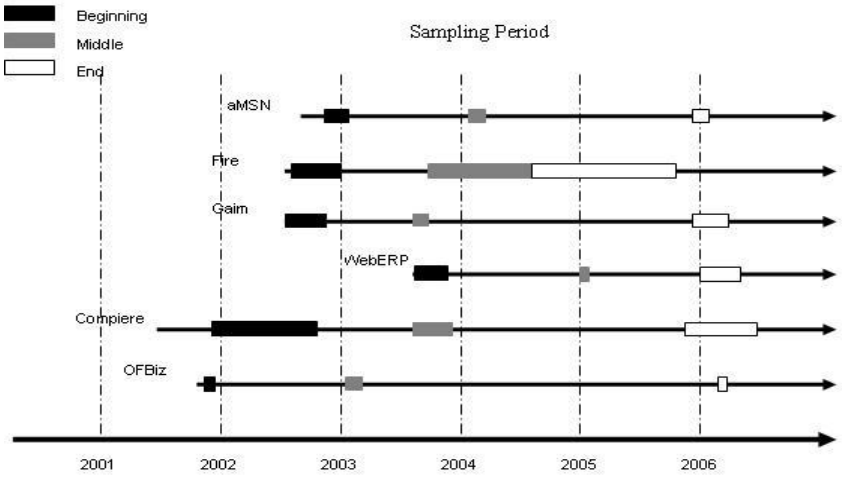


Figure 1. Sampling Periods for Decision Episodes by Projects

Table 1 presents descriptive statistics for the episodes: the number of days spanned by the messages in an episode (duration), number of messages included (length) and numbers of participants in an episode (participants). In our sample, a decision episode took an average of four days to conclude, included an average of six messages and attracted on average three members. However, as

suggested by the minimums and maximums, the distribution of duration, length and participation counts are highly skewed. For example, while the average episode was completed in 4 to 5 days, some took more than a month to reach the final decision. There were 45 (12.5%) decision episodes for which we did not find a decision announcement.

Table 1. Descriptive Characteristics of Decision Episodes (N=360)

	Minimum	Maximum	Mean	Std. Deviation
Duration (Number of days per episode)	1	48	4.42	7.050
Length (Number of message per episode)	1	49	6.34	6.027
Participation (Number of person per episode)	1	14	3.36	2.057

4.5 Data Analysis Procedure

In this section, we describe how we analyzed the decision process in the selected decision episodes. We began analysis of decision episodes by identifying which were software-modification (SM) and non-software (NS) decisions. Three independent analysts (members of the research team) first read through the episodes to identify different types of decisions represented. Decision episodes about software modification were identified as having the following triggers: 1) bug reports, 2) feature requests, 3) problem reports, 4) patch submissions and 5) project to-do lists. Decision announcements for software-modification related decision reflected either acceptance/rejection of the need for software code modification or acceptance/rejection of a submitted code modification. Non-software decision episodes were identified through the following triggers: 1) system design, 2) infrastructure/process, 3) business function, 4) release management and 5) other issues. Non-software decision announcements reflected either acceptance or rejection of a proposed long-term strategic plan for system design, infrastructure change and process improvement or resource allocation including task assignment and time schedule. Among the 360 decision-making episodes, we found 258 (71.7%) software-modification decision episodes and 102 (28.3%) non-software decision episodes. The ERP projects had more non-software decision episodes (63, 35.0%) than did the IM projects (39, 21.7%), which might reflect the greater complexity of these projects.

The main focus of our analysis was on the process of decision making, which we examined by coding the steps in the decision-making process, adopting the "functional move" as the primary unit of coding. A functional move is "the function or purpose served by a particular segment of the conversational discourse" (Wittenbaum et al. 2004). The functional move has been used extensively to understand the nature of interaction in both face-to-face and computer-mediated environments (Poole et al. 1985; Poole and Holmes 1995; Herring 1996; Macoubrie 2003). However, few studies to date have used functional move to analyze complex, asynchronous, text-based environments such as email, electronic bulletin board or threaded discussion fora.

A decision-function coding scheme was developed both deductively and inductively. Deductively, our coding scheme was based on the Decision Functions Coding System (DFCS) developed by Poole

and Roth (1989) and phases of decision making proposed by Mintzberg and his colleagues (1976). DFCS segments the decision activities into four categories: 1) problem activity, including problem analysis and problem critique; 2) executive activities, including orientation and process reflection; 3) solution activity, including solution analysis, design, elaboration, evaluation and confirmation; and 4) others, including tangents, simple agreement and simple disagreement. Mintzberg, et al.'s (1976) proposed decision-making process included three central phases, identification, development and selection with each phase containing one or more functional moves.

Inductively, we sought in the initial stages of content analysis to identify functional moves from previous studies that were not used in the FLOSS context and so could be dropped, or novel moves that had to be added. The coding system was revised through discussion among the authors. As a result, we made the following changes to the coding scheme. First, we categorized functional moves in FLOSS development into a normative decision making process that includes four phases: identification, development, evaluation and announcement. Second, several frequently-used moves in traditional decision-making contexts rarely occurred in our context, specifically "screen" and "authorization". In the FLOSS context, with distributed leadership, there was not a specific person in charge of decision-making process who might screen issues as needing or not needing discussion. Instead, discussions usually started immediately after an alternative was proposed. Similarly, a decision generally did not need to be authorized by a certain person or institution. In the very few cases, e.g., where the discussed issue needed to be handled by the administrator or the project leader, the authorization move might have been activated, but due to its very low occurrences, we decided not to include it in our coding scheme. Third, we divided the move "Solution evaluation" into two types: "Solution evaluation-opinion" and "Solution evaluation-action". Unlike in synchronous interaction, the asynchronicity of communication in FLOSS teams meant that group members could take some time to test a proposed solution and post the results of their actions rather than simply posting opinions. The final coding scheme is presented in Appendix 1. Three coders coded the moves in the collected decision-making episodes by projects separately, then compared their results. Discrepancies were discussed until the three coders agreed. The coding process took about one month.

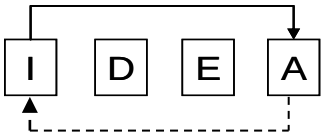
Finally, we clustered episodes along two dimensions based on the sequences of moves represented in the episodes. The first dimension is the level of procedural rationality reflected in the process, that is, whether a group conducts information gathering and analysis activities in a way that matches the normative models of decision making (what we called the sequences of decision phases). Specifically, in our analysis, we examined how many of the theoretically-identified phases of decision making were actually covered in the process of group decision-making in the episode. Second, we considered whether the decision episodes progressed linearly through the phases (what we labeled as Type I decision-making paths) or rather looped through phases repeatedly, as suggested by researchers such as Mintzberg et al. (1976) (what we labeled as Type II decision-making paths).

5. Findings

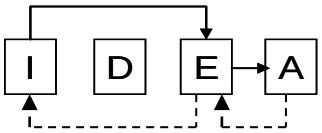
We first describe the results of coding the decision moves in the episodes. We then examine the relation between the decision patterns exhibited and the task and project performance.

5.1 Observed Decision-Making Paths

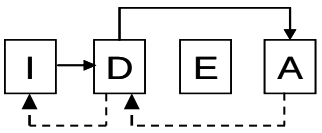
Following the procedure described in section 4.5, we identified six decision-making paths. The dashed lines in the figures indicate points at which there might be one or more loops, leading to Type II decision paths. The loop from decision announcement to problem identification indicates that an intermediate decision was made before the decision was finalized.



Short-Cut. This path represents the simplest pattern, in which a decision is made right after opportunity recognition, with no explicit diagnosis or evaluation. Examples of this kind are often observed in the bug report or problem solving discussions. For example, in one decision episode in the aMSN project, a user reported a bug (Decision Recognition), which was quickly followed by the response of an administrator that "I just fixed it" (Decision Announcement), with no further discussion or evaluation. While there is an absence of group input, these cases do represent a group decision, as the decision made affects the shared group output. Furthermore, given the use of a shared discussion forum, the absence of comments before or after the decision announcement can be read as implicit agreement from the other team members rather than simple non-participation.

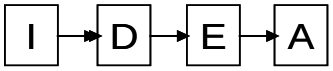


Implicit-Development (Implicit-D). In this path, the solution development phase is skipped, which does not necessarily mean the non-existence of development phase, but rather the invisibility of development phase in the online discussions. In these episodes, the person who brings up an issue also provides a solution. The subsequent discussions concentrate on evaluating its feasibility or the pros and cons of implementation, rather than looking for more alternatives. For example, in the Compiere project, a user initiated a discussion by posting two alternatives to address an identified problem: "One layout suggestion would be to place the Menu Tree on the left of a main Backing Window. Then windows invoked from the Menu Tree would appear on the right of the Backing Window. Or as an alternative, convert the Menu Tree into a second level menu bar, or even merge it with the main menu bar" (Decision recognition, Solution design). Subsequent discussion focused on the evaluations of these alternatives (Solution evaluation-opinion) and the final decision was made based on a revision of original proposal (Decision announcement).

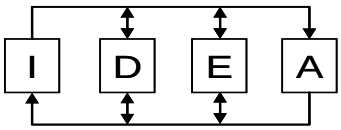


Implicit-Evaluation (Implicit-E). The third type of decision-making path we called "Implicit-Evaluation", indicating a lack of online evidence of evaluation discussion. In these episodes, a decision is announced directly after the alternatives are generated. For example, in aMSN, an administrator brought up a technical issue and proposed three solutions (Decision recognition, Solution design). The

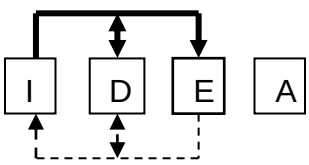
next message posted by another administrator did not continue exploring solutions, but asked, "Remind me a bit what the problem is" (Diagnosis). Most of the subsequent messages concentrated on whether the problem was one for the aMSN project or just a problem from its supporting software such as a KDE problem (Diagnosis). After some discussion and testing, members confirmed it was not a KDE problem, but an aMSN tray icon problem (Diagnosis). Then the group attention returned to solution generation (Solution design) and the problem was fixed quickly after a little revision of the existing solution alternatives (Decision announcement).



Normative. The fourth category, "Normative decision-making path", are those that adhere most closely to the rational approach described in earlier studies. In these episodes, the group goes through all phases of decision-making in a linear sequence. For example, in the Fire project, a user reported a build failure (Decision identification). The administrator pointed out the problem immediately (Diagnosis) and provided a solution (Solution design). The user did some testing and confirmed the usability (Solution evaluation-action). Then the administrator promised to commit the code into CVS soon (Decision announcement).



Dynamic. The "Dynamic Decision-making path" represents the most complex decision-making process we observed. Many of these decision episodes resemble the Garbage Can Model described in the literature. People may report a number of problems in a single message. Multiple issues, regardless of relevance, are discussed in parallel. New problems can be triggered by the discussion of existing problems, and the discussion may leave the original problems unsolved and unattended by participants. Discussions may loop back to any previous phase at any time, even after a decision is announced. As with the normative path, the dynamic decision-making path goes through all four phases, but with loops back at almost every phase. For example, in the Gaim project, a user reported a crash (Decision recognition). Several users showed the same concern with possible solutions (Diagnosis, Solution design). The discussion went on until an administrator announced the decision: "I've checked in the fix. Equivalent code is already in the later version of libyahoo2 (0.60) which we haven't yet upgraded to." However, this announcement prompted the user to raise a strategic decision opportunity about when to release a new build: "Ok, I think a new build should go out soon, as this is happening to a lot of my fire friends now (I sent them my build)" (New Trigger which leads to a new decision episode). After three days, the administrator responded by posting test build and asking for extensive testing: "It seems this is a pretty serious problem, and I think the tree is in pretty good shape right now (I'm living on TOT pretty much and haven't had problems.) I've posted a test build of Fire 0.31.e on my mac.com account if you want to take a look at it. My plan is to post this on SourceForge later tonight or tomorrow" (Diagnosis, Decision announcement).



Interrupted. The final category we called "Interrupted decision-making path" since no decision was actually made at the point that we observed the

decision process. Interruptions may occur in any phase of discussion and various reasons may account for the failure to reach a decision. An interruption in the identification phase may be a disagreement on whether there is a real problem or if there is a need to fix it. An interruption in the development phase may result from differences among various technical approaches and concerns. An interruption in the evaluation phase can be brought by the existence of multiple parties pursuing individual interests. For example, in the Gaim project, an administrator suggested adding audio functionality to the product (Decision recognition). Several core members challenged the availability of this functionality (Diagnosis). The discussions revealed two different preferred solutions—releasing a stable version with minor changes or releasing an unstable version with a major innovation (Solution analysis). Both sides extensively examined the current solutions, took relevant consequences into account and provided feasible suggestions (Solution design, Solution evaluation-opinion). However, after 11 days no final decision was reached.

Table 2 and Figure 2 show the distribution of the six decision-making paths across the 360 decision episodes. From the table we can see that only 32% of decisions episodes analyzed went through all four phases (the normative and dynamic paths) while 39% of the discussions reached a decision while skipping one or two phases (no decision was reached in the remaining 13% of cases). In 29% of the decision episodes, the group made a decision right after the problem was recognized (short-cut path). While 24% of decisions were made without the evaluation phase (implicit evaluation path), only 2% of the decisions were made without a visible development phase (implicit development path). We also found that 45% of decisions were made in a linear sequence (type I decision path), while the other 55% included one or more loops (type II decision path).

Table 2. Count of Observed Decision Paths for All Episodes

	Short-cut	**Implicit-D**	**Implicit-E**	**Normative**	**Dynamic**	**Interrupted**	**Total**
Type I	87 *(24%)*	4 *(1%)*	35 *(10%)*	11 *(3%)*	0 *(0%)*	25 *(7%)*	163 *(45%)*
Type II	19 *(5%)*	3 *(1%)*	53 *(14%)*	0 *(0%)*	103 *(29%)*	20 *(6%)*	197 *(55%)*
Total	106 *(29%)*	7 *(2%)*	88 *(24%)*	11 *(3%)*	103 *(29%)*	45 *(13%)*	360 *(100%)*

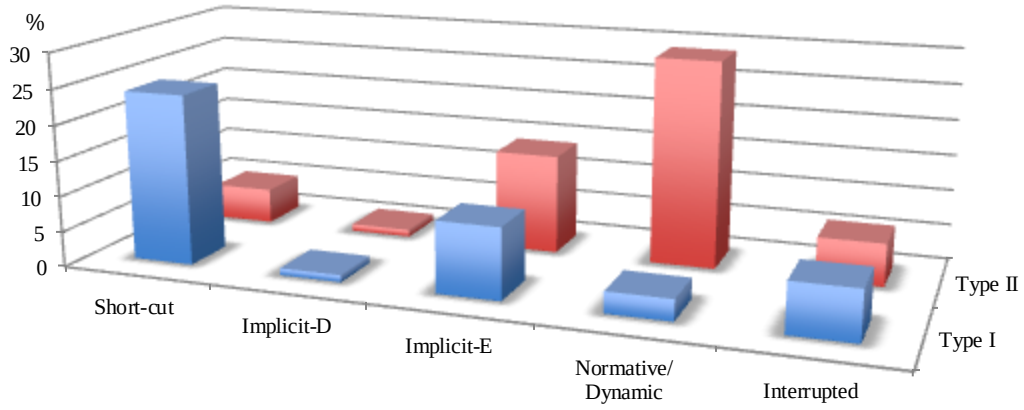


Figure 2. Distribution of Observed Decision Paths for All Episodes

5.2 Relations between Task Characteristics and Decision Paths

As noted above, task characteristics have been observed to influence the way people structure their decision-making processes. We therefore analyzed the episodes to look for significant difference on decision-making paths between software modification and non-software decision episodes, and between IM and ERP projects.

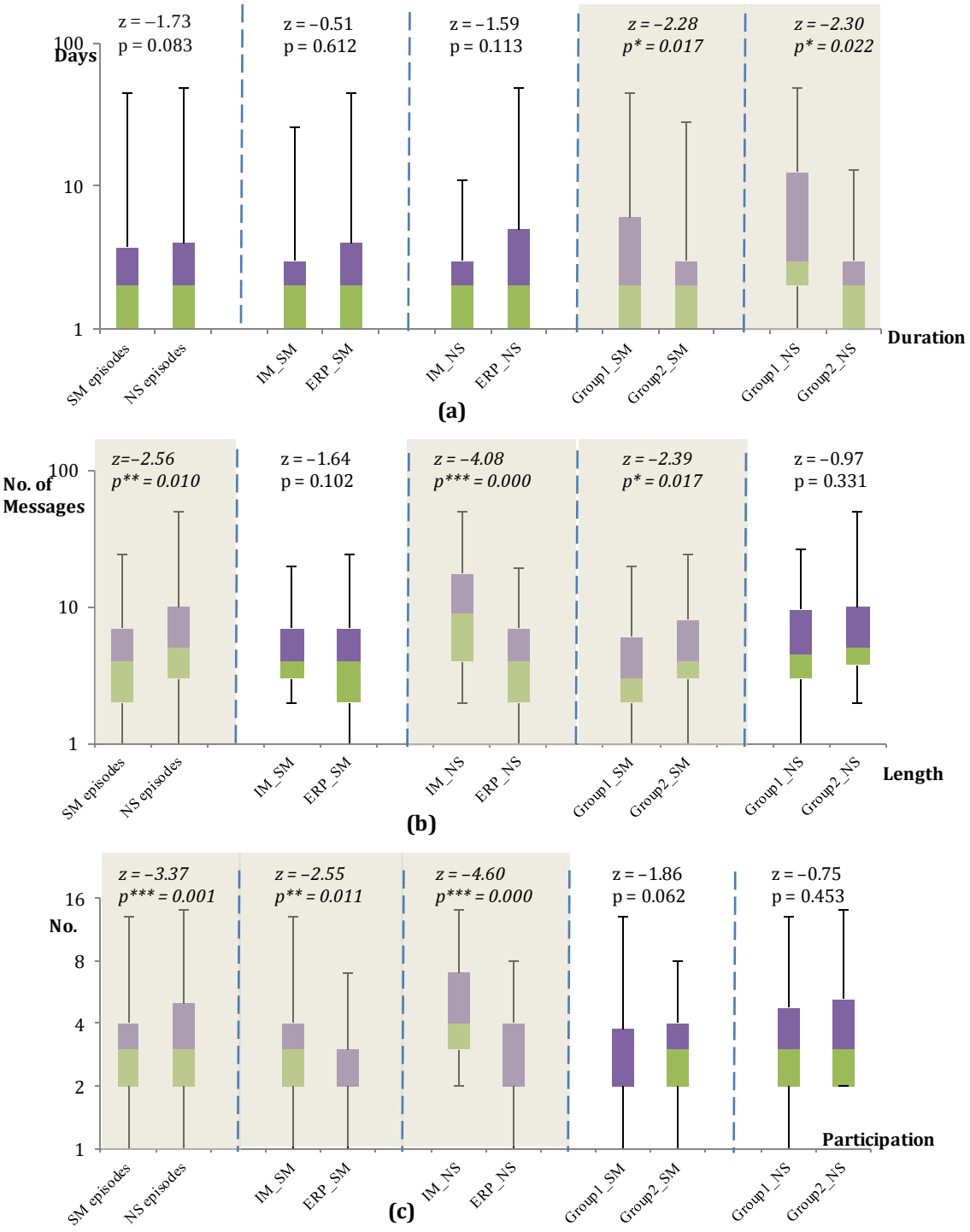


Figure 3. Differences between Decision Episodes on Duration (a), Length (b) and Participation (c) *(Shaded background for boxplots indicate that the differences are statistically significant)*

5.2.1 Software vs. non-software decisions

Because the data were not normally distributed, we used Mann-Whitney U tests to test the differences between software modification and non-software decision episodes in duration, length, and the number of participants per episode. The results, shown in the first column of Figure 3, indicated significant differences between software (SM) and non-software (NS) decision episodes on length ($z = -2.56$; $p = 0.010$) and the number of participants per episode ($z = -3.37$; $p = 0.001$) but not on duration. The non-software episodes had more messages and attract more participants than software-modification episodes. A χ^2 test of the relation between decision-episode types (software vs. non-software) and decision paths indicate that there are also significant differences in the decision paths adopted for these different decision tasks (as shown in Table 3 and Figure 4). More non-software decision episodes failed to reach a decision (21%) than did software-modification episodes (9%). Meanwhile, more software-modification episodes (31%) had implicit evaluation activities than did non-software modification episodes (9%). Finally, we found more cyclic decision paths in software modification episodes (58%) than in non-software ones (42%). Because the decision episodes and decision paths in the software modification episodes differ from those in non-software episodes, we decided that we should not analyze them together when considering the relation of other factors to decision-making paths.

Table 3. Distribution of Decision-Making Paths between SM and NS Decision Episodes

	Sequence of Decision making Paths						Linear or Cyclic decision paths		Total
	Short-cut	Implicit-D	Implicit-E	Normative	Dynamic	Interrupted	Type I	Type II	
SM	72 *(28%)*	2 *(1%)*	79 *(31%)*	6 *(2%)*	75 *(29%)*	24 *(9%)*	108 *(42%)*	150 *(58%)*	258 *(100%)*
NS	34 *(33%)*	5 *(5%)*	9 *(9%)*	5 *(5%)*	28 *(28%)*	21*(21%)*	55 *(54%)*	47 *(46%)*	102 *(100%)*
	106 *(29%)*	7 *(2%)*	88 *(24%)*	11*(3%)*	103 *(29%)*	45 *(13%)*	163 *(45%)*	197 *(55%)*	360 *(100%)*
	$\chi^2 = 30.444$, df = 5, p*** = 0.00						$\chi^2 = 4.292$, df = 1, p* = 0.038		

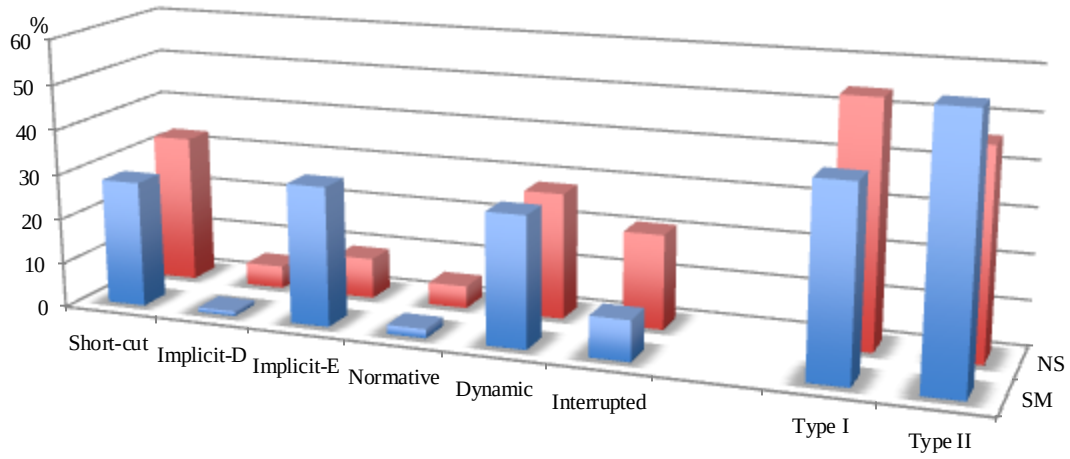


Figure 4. Distribution of Decision Paths across SM and NS Episodes

5.2.2 Relation between task complexity and decision paths

As we discussed earlier, we expected decisions for ERP projects to be more complex than those for IM projects since we expected ERP projects to have higher software code interdependencies and many external constraints such as accounting rules and legal reporting requirements. As a result, we expected to see different patterns of decision making in the IM and ERP projects.

Software-modification episodes.

We first analyzed software modification episodes. As shown in the second column of Figure 3, SM episodes in IM projects and ERP projects differed only on participation ($z = -2.55$, $p = 0.011$): IM projects attracted somewhat greater participation in SM decision making than did ERP projects. However, contrary to our expectation, IM projects and ERP projects displayed similar patterns for both sequences of decision paths and Type I/II decision paths: χ^2 tests of the relationship between type of project and decision paths (Table 4 and Figure 5) shows no relation ($\chi^2 = 2.53$, $p = 0.772$ for phases of decision paths and $\chi^2 = 0.263$, $p = 0.608$ for Type I/II).

Table 4. Distribution of Decision-Making Paths between IM and ERP Projects for SM episodes

	Sequence of Decision making Paths						**Linear or Cyclic decision paths**		**Total**
	Short-cut	**Implicit-D**	**Implicit-E**	**Normative**	**Dynamic**	**Interrupted**	**Type I**	**Type II**	
IM	37 *(26%)*	1 *(1%)*	44 *(31%)*	5 *(4%)*	41 *(29%)*	13 *(9%)*	57 *(40%)*	84 *(60%)*	141*(100%)*
ERP	36 *(31%)*	1 *(1%)*	35 *(30%)*	1 *(1%)*	33 *(28%)*	11 *(9%)*	51 *(44%)*	66 *(56%)*	117*(100%)*
	73 *(28%)*	2 *(1%)*	79 *(31%)*	6 *(2%)*	74 *(29%)*	24 *(9%)*	108 *(42%)*	150 *(58%)*	258*(100%)*
	$\chi^2 = 2.53$, df = 5, p = 0.772						$\chi^2 = 0.263$, df = 1, p = 0.608		

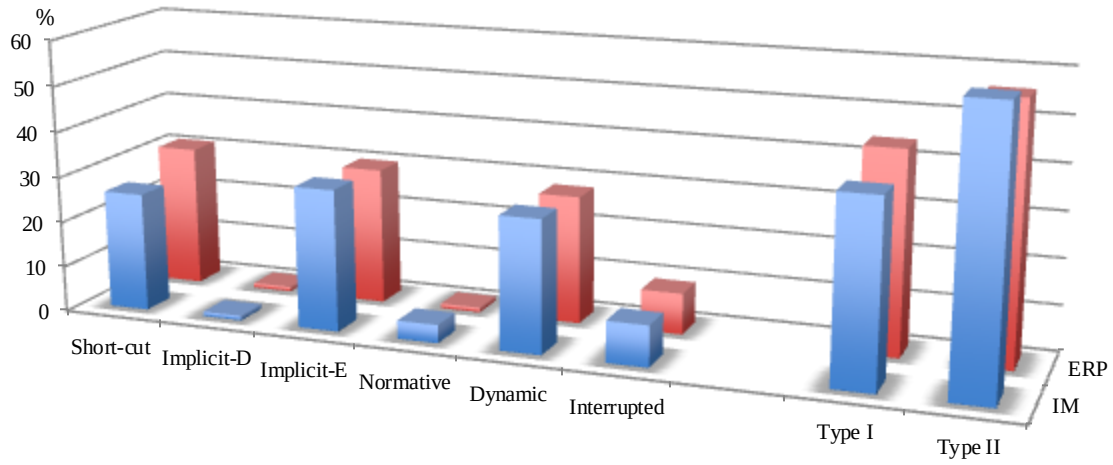


Figure 5. Distribution of Decision Paths on SM Episodes between IM and ERP Projects

Non-software episodes.

For non-software episodes, IM projects had more messages per episode ($z = -4.08$, $p^{***} = 0.000$) and more participants than ERP projects ($z = -4.60$, $p^{***} = 0.000$) (Figure 3, column 3). Consistent with our expectation, the distribution of decision paths in non-software decisions in IM projects was

quite different from in ERP projects (see Table 5 and Figure 6). However, the differences were not what might be expected. Decision episodes for IM projects had more dynamic paths (44% vs. 19%), while ERP projects were more likely to use short-cut paths (38% vs. 23%). ERP projects had more interrupted decision paths (27% vs. 10%). IM projects had more loops in non-software episodes than did ERP projects (62% Type II vs. 37%) (see Table 5 and Figure 6).

Table 5. Distribution of Decision-Making Paths between IM and ERP Projects for NS Episodes

	Sequence of Decision making Paths						**Linear or Cyclic decision paths**		**Total**
	Short-cut	**Implicit-D**	**Implicit-E**	**Normative**	**Dynamic**	**Interrupted**	**Type I**	**Type II**	
IM	9 *(23%)*	4 *(10%)*	2 *(5%)*	3 *(8%)*	17 *(44%)*	4 *(10%)*	15 *(38%)*	24 *(62%)*	39 *(100%)*
ERP	24 *(38%)*	1 *(2%)*	7 *(11%)*	2 *(3%)*	12 *(19%)*	17 *(27%)*	40 *(63%)*	23 *(37%)*	63 *(100%)*
	33 *(32%)*	5 *(5%)*	9 *(9%)*	5 *(5%)*	29 *(28%)*	21 *(21%)*	55 *(54%)*	47 *(46%)*	102 *(100%)*
	$\chi^2 = 15.7$, df = 5, p** = 0.008						$\chi^2 = 6.07$, df = 1, p* = 0.014		

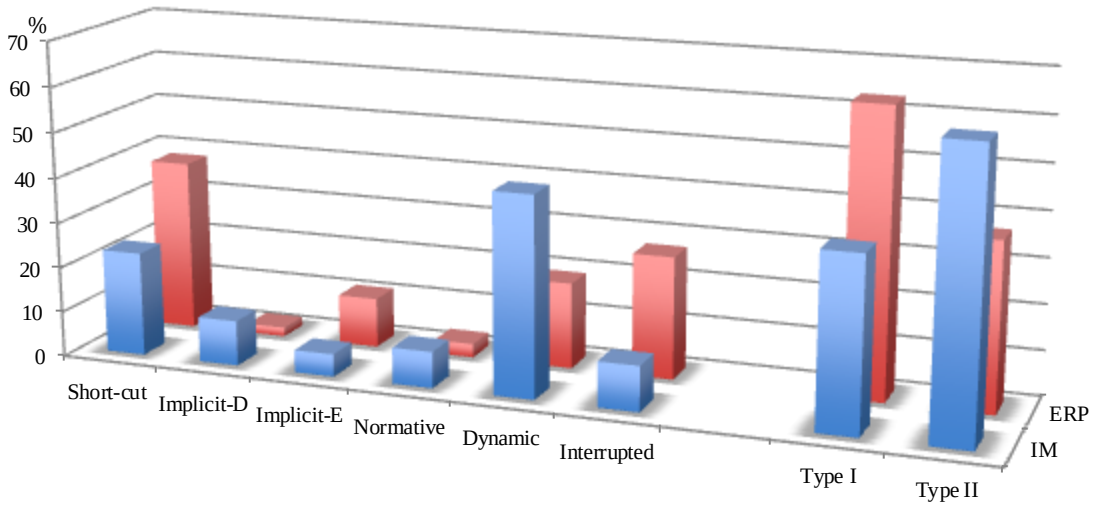


Figure 6. Distribution of Decision Paths on NS Episodes between IM and ERP Projects

5.2.3 Relation between decision paths and project performance

Finally, we examined the relationship between the decision-making paths and project performance to assess whether certain kinds of decision processes were associated with more effective projects. As noted above in the sampling section, although our measure of effectiveness is quite limited, the six FLOSS projects we studied can be roughly grouped based on overall performance: first, Gaim and Compiere, and second, aMSN, OFbiz, Fire and WebERP.

Software modification episodes.

Overall, decision episodes from the more successful teams had longer duration (z = –2.28, p* = 0.017), but shorter length (fewer messages) (z = –2.39, p* = 0.017), i.e., in more successful teams, members attend to the decision making process over a longer time period, but do not initiate more

messages around the issues (column 4 of Figure 3). Table 6 and Figure 7 present the data pertaining to project performance and decision paths in software modification episodes. For the relationship between team performance and phases of decision-making paths, projects in the most successful group have more short-cut (32%) and interrupted decision paths (17%) than the other group (27% and 6% respectively), but they have fewer four-phase decision paths (23% vs. 34%). The less successful projects have more implicit-evaluation decision paths (32%) than the more successful projects (28%). A χ^2 test also indicates a significant difference between these two groups ($\chi^2 = 13.6$, $p^* = 0.019$) (see Table 6). However, considering Type I vs. Type II decision paths, a χ^2 test indicated that the difference is not statistically significant ($\chi^2 = 3.27$, $p = 0.070$).

Table 6. Distribution of Decision-Making Paths across Performance Groups for SM Episodes

	Sequence of Decision making Paths						Linear or Cyclic decision paths		Total
	Short-cut	Implicit-D	Implicit-E	Normative	Dynamic	Interrupted	Type I	Type II	
1	26 *(32%)*	0 *(0%)*	23 *(28%)*	0 *(0%)*	19 *(23%)*	14 *(17%)*	41 *(50%)*	41 *(50%)*	82 *(100%)*
2	47 *(27%)*	2 *(1%)*	56 *(32%)*	6 *(3%)*	55 *(31%)*	10 *(6%)*	109 *(62%)*	67 *(38%)*	176 *(100%)*
	73 *(28%)*	2 *(1%)*	79 *(31%)*	6 *(2%)*	74 *(29%)*	24 *(9%)*	150 *(58%)*	108*(42%)*	258 *(100%)*
	$\chi^2 = 13.6$, df = 5, $p^* = 0.019$						$\chi^2 = 3.27$, df = 1, $p = 0.070$		

1: Group 1 includes the better performing projects (Gaim and Compiere)
2: Group 2 includes the less well performing projects (aMSN, Fire, OfBiz, and WebERP)

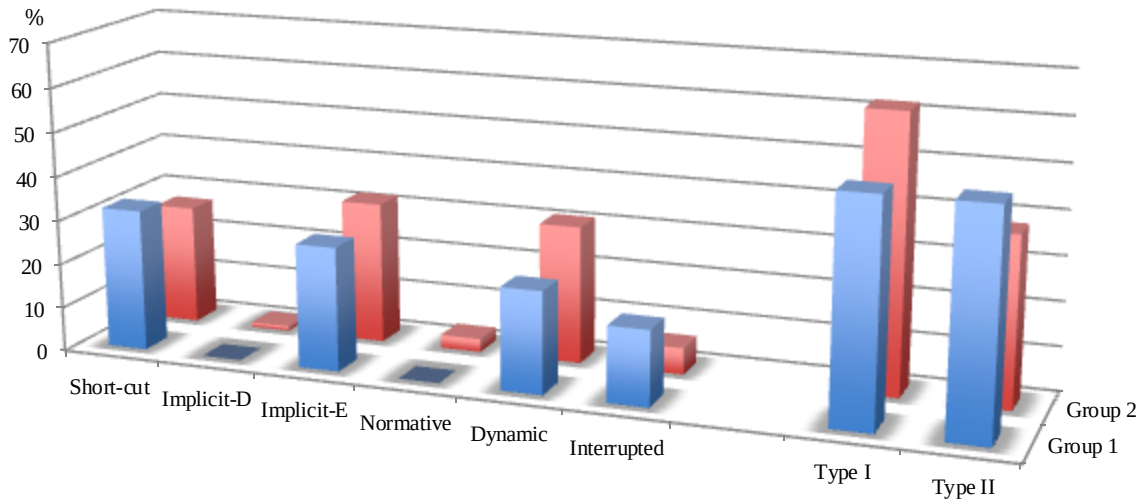


Figure 7. Distribution of Decision-Making Paths across Performance Groups for SM Episodes

Non-software episodes

For non-software decisions, we observed significant differences in duration between the two groups, similar to the differences on software episodes. Overall, decision episodes from the more successful teams had longer duration ($z = -2.30$, $p^* = 0.022$), while length and participation were not significantly different, i.e., in more successful teams, members spend more time attending the decision making process, but they do not initiate more discussions around the issues. For non-

software decisions, we observed no significant difference in the distribution of decision-making paths among the different projects, as shown in the fifth column of Figure 3.

Table 7. Distribution of Decision-Making Paths across Performance Groups for NS Episodes

	Sequence of Decision making Paths						Linear or Cyclic decision paths		Total
	Short-cut	Implicit-D	Implicit-E	Normative	Dynamic	Interrupted	Type I	Type II	
1	15 *(40%)*	1 *(3%)*	3 *(8%)*	2 *(5%)*	7 *(18%)*	10 *(26%)*	22 *(58%)*	16 *(42%)*	38 *(100%)*
2	18 *(28%)*	4 *(6%)*	6 *(10%)*	3 *(5%)*	22 *(34%)*	11 *(17%)*	33 *(52%)*	31 *(48%)*	64 *(100%)*
	33 *(32%)*	5 *(5%)*	9 *(9%)*	5 *(5%)*	29 *(28%)*	21 *(21%)*	55 *(54%)*	47 *(46%)*	102 *(100%)*
	$\chi^2 = 4.76$, df = 5, p = 0.466						$\chi^2 = 0.39$, df = 1, p = 0.535		

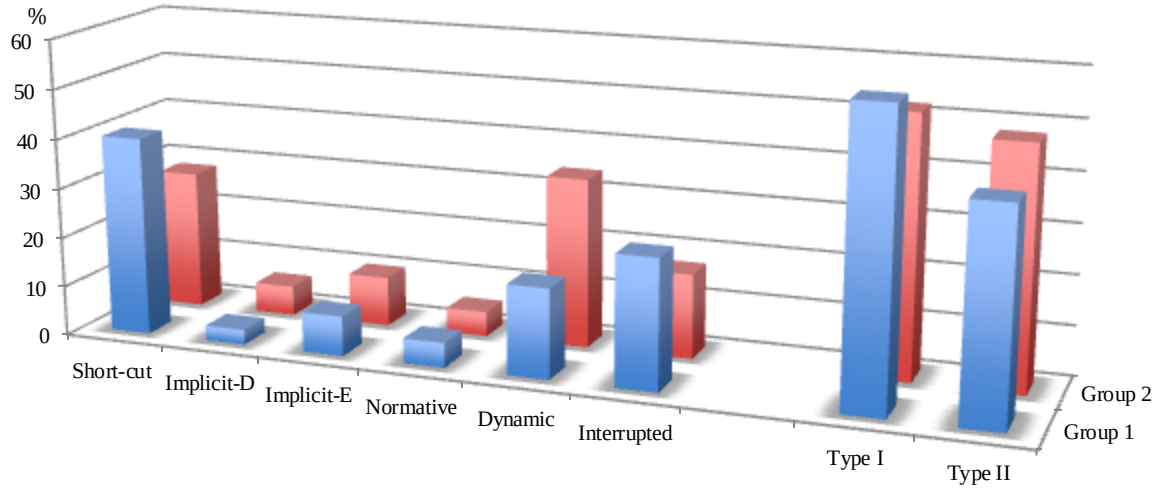


Figure 8. Distribution of Decision-Making Paths across Performance Groups for NS Episodes

6. Discussion

The goal of this study was to investigate decision-making processes in self-organizing virtual teams, FLOSS development teams in particular. We found that decision making operates differently in this setting than in settings previously studied. Specifically, previous studies (e.g., Poole and Holmes 1995) found a positive relationship between a normative decision path and group effectiveness: groups going through more information-processing phases in decision making performed better than those that did not. This relationship did not hold in our context. The more successful projects did not use normative paths more often than less successful projects, nor did they cover more phases. Instead, decision making in these teams shows novel patterns that reflect the impact of self-organization and ICT support for virtuality, as well as the nature of the tasks in FLOSS development teams.

Our findings show that decision-making patterns differ between software (SM) and non-software (NS) decision episodes. In our setting, SM decisions represent the teams' routine work that leads to the shared work product, while NS episodes are often strategic changes for project rather than simple changes to the code. We found that the NS episodes were more active and attracted more participants

than SM episodes, suggesting that NS decisions exhibit increased complexity and uncertainty, needing a higher level of group participation and more time to discuss. The decision paths in these two types of decision episodes are quite different too. More non-software episodes fail to reach decisions (21% vs. 9%) while more software-modification episodes skipped the evaluation phase (31%) than non-software episodes (9%). It appears that due to the uncertainty and ambiguity of NS decisions, it is sometimes more difficult to reach consensus. Further, software modification episodes had somewhat more loops (58%) than the non-software ones (46%). Because these differences seem to reflect different work practices, we discuss the further findings for these decisions separately.

6.1 Software Decision Making

We begin with software modification decisions (SM). The importance of SM decisions for projects is shown by our finding of significant difference between more successful and less successful projects in decision making for SM episodes. Interestingly, we found no significant difference between IM and ERP projects for the process of SM decisions, that is, our findings suggest that the FLOSS projects studied have a similar process for making SM decisions, a process heavily influenced by the self-organizing nature of the teams and their reliance on ICT to support virtuality.

As noted above, in FLOSS projects much of the software development is carried out by a small core set of developers. Self-organization empowers core individuals to pick tasks and to implement decisions on their own. In FLOSS in particular, qualified developers with commit rights can submit changes directly into the code repository, without the need for explicit permission from others. However, this freedom is tempered by the need to make decisions that will be accepted by the group. In other words, decisions (at least software modification decisions) are made by individuals (who write the code) but accepted on the group level (when code is committed to the shared repository). Developers generally announce decisions so other developers know what is happening and can comment if they see a need. However, 29% of the decisions were made without any explicit discussion of solutions (i.e., 106 of 360 decisions were Short-Cut). Though we cannot completely rule out unarchived offline discussion, it appears that in a virtual setting, explicit discussion of decision options is often unnecessary, perhaps even distracting: projects in the most successful group have somewhat higher proportion of short-cut decision paths, and fewer four-phase decision paths. We suggest that the use of ICT that provides team members with visibility into the shared work products substitutes for discussion. Using these tools, developers do not have to explain in detail what they have done to others. If other developers are curious, they can examine the code themselves; if they concur, they need say nothing. The result of this approach is that when examining the discussions leading to decisions, the process of how the final choice was reached is often invisible. This finding is consistent with other research on FLOSS development that finds that much FLOSS development work does not require much explicit discussion (Krishnamurthy 2002; Howison 2009).

The balance between individual freedom and responsibility to the group is further reflected in the other observed decision paths. FLOSS practitioners often maintain that everyone in the community has a right to contribute to the project by providing solutions and submitting code. Activity in the development phase contributes to the creativity and innovative capacity of a project. The evaluation phase then serves as the phase of "quality control," which ensures that only contributions of good quality consistent with the overall goal of the project are added to the application. However, given a set of alternatives, it appears that a skilled developer can perform the evaluation phase individually, selecting an appropriate alternative without group discussion. We found that only 2% of decision episodes (a total of 7) followed "Implicit-Development" decision-making path, while 24% (a total of 88) followed the "Implicit-Evaluation" path, i.e., a fair number of decision episodes skipped an explicit evaluation phase, but few included evaluation without a development phase. There is little evidence of dissent from these decisions, which suggests that those with commit rights and who choose to implement a solution have the expertise to do so. The lack of evaluation seems further to reflect an action orientation in FLOSS development groups' decision-making: that it is preferable to try out a solution rather than spending a lot of time in detailed evaluation of possible options. This approach is supported by the fact that is always possible to return to a previous stage of development by reversing a contribution that turns out to be unsatisfactory.

A final implication of the reliance on asynchronous communication is that this mode of communications allows developers time to work independently, to think about and test solutions rather than immediately discussing them with others. As a result, developers can propose more fully developed potential solutions for consideration by the group. This effect of the technology is seen in the fact that SM episodes in more successful groups appeared to take more time, but with fewer messages per episode, which suggests that in more successful teams, members took more time to develop and evaluate options before discussing them.

Taken together, these features of decision making are consistent with Ramesh and Dennis's argument about a new type of organization for global virtual teams. In contrast to traditional virtual team design, which strives to tightly couple team members through information rich media, the new form strives to decouple team members through the use of well-defined process and semantically-rich interaction media (Ramesh and Dennis 2002). While FLOSS development is an extreme case, we expect to see similar impacts in other virtual settings. For example, co-workers may be able to substitute examination of shared documents (e.g., with systems such as Google Documents) for extensive discussion of their contents and direct contribution to the shared work for negotiation about who will do what.

6.2 Non-Software Decisions

We next consider the implications for virtuality and self-organization for non-software decisions (NS). Interestingly, we did not find differences between more and less successful teams on the decision-

making process for NS decisions. However, it is worth noting that the project performance measures we adopted are all code-related, such as downloads and page views, which might not reflect performance on these decisions. Contrariwise, for NS decisions, the nature of the project has an impact. We expected ERP projects to be more complex than IM projects in terms of software code interdependencies and external constraints such as accounting rules and legal reporting requirements. This complexity may explain the finding that ERP projects have more NS decisions than do IM projects and more interrupted decisions. ERP and IM software also have different kinds of users, which also affects the nature of team interactions: ERP software targets companies while IM software targets individual end users. We therefore expected that more expertise would be needed to participate in ERP software development. These observations may explain why the IM projects were able to attract more people to participate and initiate more discussions in the development. Furthermore, the easier involvement of participants in decisions may lead to more proposed solutions and objections to proposals, which can explain why IM project decisions have more dynamic paths than ERP projects. On the other hand, the ERP projects had more short-cut paths (38% vs. 23%), perhaps reflecting a greater level of expertise for the developers that are involved.

The self-organizing nature of the teams seems to result in open communication around decision making, as no individual controls the discussions. The asynchronicity of communication also means that members may not be synchronized in progressing through the phases of the decision-making process. Developers may raise questions about others' actions based on their knowledge, leading back to previous phases of decision making, leading the high proportion of dynamic paths observed.

7. Conclusion

The primary goal of this study was to understand the decision-making process in FLOSS development teams, as extreme examples of self-organizing virtual teams, and its relations to project performance. A limitation of this study is the exclusion of other channels of group communication, such as Internet Relay Chat (IRC), Instant Messaging, phone calls and so forth. Though the usages of these non-archived tools are generally discouraged in FLOSS projects, they are adopted at various levels in different projects. It is possible that some of the steps in the decision-process that we infrequently observed were in fact carried out by a subgroup using such alternative channels. However, the use of such channels would not change our main conclusion, namely that many decisions that bind the project to a course of action are made without explicit involvement of the entire group in seemingly important phases of the decision process.

The study makes several important contributions to the literature. First, our research increases the understanding of the process of decision making by analyzing specific decision episodes in detail. Although studies have examined aspects of decision making such as decision making styles, we are not aware of any study that has examined decision-making process in terms of decision-making paths. Yet, it is very important to open the black box of decision-making processes in order to understand

how decisions are made in teams and how decision-making process is structured differently across projects. Such an in-depth examination of the microstructures of decision-making processes compliments existing macro-level research on decision making (e.g. Raymond 1998; German 2003).

Another distinctive contribution of this study is to adopt functional move as unit of coding in analyzing decision episodes. Few studies to date have used functional move to analyze complex, asynchronous, text-based environments such as email or threaded forums. In the present study, email transcripts were used as the data source to understand decision making. The length of a single email varies from one sentence that may make only one functional move to a thousand words that crosses all decision-related functions before reaching the collective decision. These challenges make it difficult to equate a functional move with any naturally occurring grammatical segment such as sentence or message. Thus functional moves in thematic units would be serve better as a phrase, a sentence, a paragraph, or even a complete email message. Moreover, in this self-organizing context, individuals may be involved in multiple issues at one time, posing challenges to tracing individual decisions.

A particular contribution of this study is a better understanding of the nature of decision-making processes in FLOSS development. Our findings reinforce the idea that FLOSS teams are not all alike. Our findings show that differences in contextual attributes such as software type and project type, as exemplified by the comparison of IM and ERP projects, have an influence on the emergence of work practices. Nevertheless, our results have practical implications for structuring decision-making processes in FLOSS development teams. First, our results suggest that software-modification decisions are quite different from non-software decisions. Therefore, FLOSS development teams might want to adopt different strategies when making these two types of decisions. For software-modification decisions, the modularity and specific ICT adopted by FLOSS projects greatly reduced the need for explicit coordination and communication among multiple developers, allowing members to perform tasks in an independent fashion. For non-software decisions, FLOSS development teams might want to fully use the asynchronicity of ICT to encourage participation and discussion. Second, our results also suggest that more successful teams appear to be more self-organized in terms of having more short cut paths and less dynamic paths in software modification episodes. Therefore, FLOSS teams may want to support such self-organizing behaviors, perhaps by purposeful modular design of software.

Though FLOSS development teams were deliberately selected as an extreme case, many of our findings can be applied to organizational self-organizing virtual teams more generally. Indeed, Markus et al. (2000) argues that "Although managers in industries other than software development may prefer more traditional styles of management, they should remember that the world is changing and workers are changing along with it. In a labor force of volunteers and virtual teams, the motivational and self-governing patterns of the open source movement may well become essential to business success." (p.25). The results of this study offer several practical insights that can benefit organizations in decision making in a distributed, self-organizing work environment. First, managers

should consider to establishing mechanisms that enable team members to coordinate through their work. For example, modularity of work products enables team members to work independently, while making their contributions visible through ICT tools. Second, more successful teams seemed to take longer time to make decisions. Therefore, managers might want to establish processes that allow team members to spend more time to reflect instead of responding immediately, e.g., by building asynchronous communication channels.

In summary, the variables and relationships we have identified provide the foundation for deeper exploration and potentially richer explanations of the relationships we have described. Future studies should replicate and extend this analysis to additional self-organizing virtual teams, FLOSS and otherwise, to examine the relationship between the unique features of this setting, decision-making processes and team effectiveness in more detail.

Appendix 1. Coding System for Stages in the Decision-Making Process

Phase	**Functional Move**	**Explanation**	**Examples**
Identifi-cation	Decision recognition routine	This move mainly recognizes the opportunity that may lead to a decision. In our context, we focus on:1) whether the fix is needed; 2) the patch is accepted.	problem analysis (Poole and Roth 1989); decision recognition (Mintzberg et al. 1976)
	Diagnosis	This move focuses on understanding the underlying reasons that cause problems or create opportunities for decisions. It also includes asking and providing background information, such as installation environment, computer configuration etc.	problem critique (Poole and Roth 1989); diagnosis (Mintzberg et al. 1976)
Develop-ment	Solution analysis	This move describes the activities trying to develop its solution in general terms, rather than providing specific solution, such as group rule/norm, criteria and general directions to guide the solution.	solution analysis (Poole and Roth 1989)
	Solution search	This move describes the activities trying to look for ready-made solutions based on experiences and existing resources, rather than designing solution by themselves.	search (Mintzberg et al. 1976), solution search (Poole and Roth 1989)
	Solution design	This move describes the activities designing and providing specific solutions and suggestions by themselves, or modifying the ready-made/existing ones according to the new context.	design (Mintzberg et al. 1976), solution elaboration (Poole and Roth 1989)
Evalua-tion	Solution evaluation-opinion	This move explicit or implicitly comments on potential alternatives, based on personal experiences/ preferences, rather than real testing/checking.	evaluation-choice (Mintzberg et al. 1976); solution evaluation (Poole and Roth 1989)
	Solution evaluation-action	This move explicit or implicitly comments on potential alternatives, based on actual testing/checking. It also includes describing the details how the alternatives are tested and what results come out of that.	

	Solution confirmation	This move describes the activity to ask for confirmation or initiate voting.	solution confirmation (Poole and Roth 1989)
Announ-cement	Decision announcement	This move announces the final decision on group level.	

References

Annabi, H., Crowston, K. and Heckman, R. (2008). Depicting what really matters: Using episodes to study latent phenomenon. Proceedings of the International Conference on Information Systems (ICIS 2008). Paris, France.

Armstrong, D. J. and Cole, P. (2002). Managing distance and differences in geographically distributed work groups. Distributed Work. P. Hinds and S. Kiesler. Cambridge, MA, MIT Press**:** 167–186.

Baltes, B. B., Dickson, M. W., Sherman, M. P. and Bauer, C. C. (2002). "Computer-Mediated Communication and Group Decision Making: A Meta-Analysis." Organizational Behavior and Human Decision Processes **87**(1): 156-179.

Barkhi, R. and Pirkul, H. (2004). "The Influence of Communication Mode and Incentive Structure on GDSS Process and Outcomes." Decision Support Systems **37**(2): 287-305.

Benbunan-Fich, R., Hiltz, S. R. and Turoff, M. (2002). "A comparative content analysis of face-to-face vs. asynchronous group decision making." decision Support Systems **34**: 457-469.

Berry, G. R. (2006). "Can Computer-Mediated Asynchronous Communication Improve Team Processes and Decision Making? Learning From the Management Literature." Journal of Business Communication **43**(4): 344-366.

Boone, C. and Hendriks, W. (2009). "Top Management Team Diversity and Firm Performance: Moderators of Functional-Background and Locus-of-Control Diversity." Management Science **55**(2): 165-180.

Bourgault, M., Drouin, N. and Hamel, E. (2008). "Decision Making Within Distributed Project Teams: An Exploration of Formalization and Autonomy as Determinants of Success." Project Management Journal **39**(Supplement): S97-S110.

Cissna, K. (1984). "Phases in Group Development: the Negative Evidence." Small Group Behavior **14**: 3-32.

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

Collins, J. and Drucker, P. (1999). A Conversation between Jim Collins and Peter Drucker. Drucker Foundation News. **7:** 4–5.

Crowston, K. (2000). Processes as theory in information systems research. IFIP TC8 WG8.2 International Working Conference on the Social and Organizational Perspective on Research and Practice in Information Technology, Arlborg, Denmark, Kluwer Academic.

Crowston, K., Howison, J. and Annabi, H. (2006). "Information systems success in Free and Open Source Software development: Theory and measures." Software Process—Improvement and Practice **11**(2): 123–148.

Crowston, K., Li, Q., Wei, K., Eseryel, U. Y. and Howison, J. (2007). "Self-organization of teams for free/libre open source software development." Information and Software Technology **49**: 564-575.

Dean Jr, J. W. and Sharfman, M. P. (1996). Does decision process matter? A study of strategic decision-making effectiveness, JSTOR. **39:** 368-396.

Douglas, C., Martin, J. S. and Krapels, R. H. (2006). "Communication in the Transition to Self-Directed Work Teams." Journal of Business Communication **43**(4): 295-321.

Druskat, V. U. and Wheeler, J. V. (2003). "Managing From the Boundary: The Effective Leadership of Self-Managing Work Teams." Academy of Management Journal **46**(4): 435-457.

Duarte, D. L. and Snyder, N. T. (2001). Mastering virtual teams. San Francisco, Jossey-Bass.

Eisenhardt, K. M. (1989). "Making Fast Strategic Decisions in High-Velocity Environments." Academy of Management Journal **32**(3): 543-576.

Espinosa, J. A., Slaughter, S. A., Kraut, R. E. and Herbsleb, J. D. (2007). "Familiarity, Complexity, and Team Performance in Geographically Distributed Software Development." Organization Science **18**(4): 613-630.

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 **22**(3): 123-157.

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

Fisher, B. A. (1970). "Decision Emergence: Phases in Group Decision-Making." Communication Monographs **37**(1): 53-66.

Fitzgerald, B. (2006). "The Transformation of Open Source Software." MIS Quarterly **30**(3): 587-598.

Gacek, C. and Arief, B. (2004). "The many meanings of Open Source." IEEE Software **21**(1): 34-40.

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

Guzzo, R. A. and Salas, E. (1995). Team Effectiveness and Decision Making in Organizations. San Francisco, CA, Jossey-Bass.

Hackman, J. R. and Morris, C. G. (1978). Group tasks, group interaction process, and group performance effectiveness: A review and proposed integration. Group Processes, volume 8 of Advances in Experimental Social Psychology. L. Berkowitz. New York, Academic Press**:** 45–99.

Hauptman, O. (1986). "Influence of Task Type on the Relationship between Communication and Performance: the Case of Software Development." R&D Management **16**(2): 127-139.

Herring, S. C., Ed. (1996). Computer-Mediated Communication: Linguistic, Social, and Cross-Cultural Perspectives. Philadelphia, John Benjamins.

Hirokawa, R. Y. (1985). "Discussion procedures and decision-making performance: A test of a functional perspective." Human Communication Research **12**(2): 203–224.

Hirokawa, R. Y. and Rost, K. M. (1992). "Effective Group Decision Making In Organizations: Field Test of the Vigilant Interaction Theory." Management Communication Quarterly **5**(3): 267.

Howison, J. (2009). Alone Together: A Socio-Technical Theory of Motivation, Coordination and Collaboration Technologies in Organizing for Free and Open Source Software Development. School of Information Studies. Syracuse, NY, Syracuse University. **PhD**.

Howison, J., Conklin, M. and Crowston, K. (2006). "FLOSSmole: A collaborative repository for FLOSS research data and analyses." International Journal of Information Technology and Web Engineering **1**(3): 17–26.

Kamis, A., Koufaris, M. and Stern, T. (2008). "Using an Attribute-Based Decision Support System for User-Customized Products Online: An Experimental Investigation." MIS Quarterly **32**(1): 159-177.

Kelty, C. M. (2008). Two Bits: the Cultural Significance of Free Software. Durham, NC, Duke University Press.

Kerr, N. L. and Tindale, R. S. (2004). "Group Performance and Decision Making." Annual Review of Psychology **55**: 623-655.

Kiesler, S. and Sproull, L. (1992). "Group Decision Making and Communication Technology." Organizational Behavior and Human Decision Processes **52**: 96-123.

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

Langfred, C. W. (2004). "Too Muuch of a Good Thing? Negative Effects of High Trust and Individual Autonomy in Self-Managing Teams." Academy of Management Journal **47**(3): 385-399.

Macoubrie, J. (2003). "Logical Argument Structures in Decision-making." Argumentation **17**(3): 291-313.

Majchrzak, A., Malhotra, A. and John, R. (2005). "Perceived Individual Collaboration Know-How Development Through Information Technology-Enabled Contextualization: Evidence from Distributed Teams." Information Systems Research **16**(1): 9-27.

Manz, C. C. and Sims, H. P. (1987). "Leading Workers to Lead Themselves: The External Leadership of Self-Managing Work Teams." Administrative Science Quarterly **32**(1): 106-129.

Markus, M. L., Manville, B. and Agres, E. C. (2000). "What makes a virtual organization work?" Sloan Management Review **42**(1): 13--26.

Martins, L. L., Gilson, L. L. and Maynard, M. T. (2004). "Virtual Teams: What do We Know and Where do We Go from Here?" Journal of Management **30**(6): 805-835.

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

Miller, K. (2008). Organizational Communication: Approaches and Processes. Bonston, MA, Wadsworth Cengage Learning.

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

Moon, J. Y. and Sproull, L. (2000). "Essence of distributed work: The case of the Linux kernel." First Monday **5**(11).

Orlikowski, W. J. and Yates, J. (1994). "Genre Repertoire: The Structuring of Communicative Practices in ORganizations." Administrative Science Quarterly **39**(4): 541-574.

Payne, J. W. (1976). "Task Complexity and Contingent Processing in Decision making: An Information Search and Protocol Analysis." Organizational Behavior and Human Performance **16**(2): 366-387.

Poole, M. S. (1981). "Decision development in small groups I: A comparison of two models." Communication Monographs **48**(1): 1-24.

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

Poole, M. S. and Baldwin, C. L. (1996). Developmental Processes in Group Decision Making. Communication and Group Decision Making. R. Y. Hirokawa and M. S. Poole. Thousands Oaks, CA, SAGE**:** 215-241.

Poole, M. S. and Holmes, M. E. (1995). "Decision development in computer-assisted group decision-making." Human Communication Research **22**(1): 90–127.

Poole, M. S. and Roth, J. (1989). "Decision development in small-groups 4. A typology of group decision paths." Human Communication Research **15**(3): 323–356.

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

Poole, M. S. and Roth, J. (1989). "Decision Development in Small Groups V: Test of a Contingency Model." Human Communication Research **15**(4): 549-589.

Poole, M. S., Seibold, D. R. and McPhee, R. D. (1985). "Group decision-making as a structurational process." Quarterly Journal of Speech **71**(1): 74–102.

Ramesh, V. and Dennis, A. R. (2002). The Object-Oriented Team: Lessons for Virtual Teams from Global Software Development. Proceedings of the 35th Hawaii International Conference on System Sciences, Big Island, HI, U.S.A.

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

Raymond, E. S. (1998). "Homesteading the noosphere." First Monday **3**(10).

Raymond, E. S. (1999). "Interview: Linux and open source success." IEEE Software **16**(1): 85--89.

Robey, D., Khoo, H. M. and Powers, C. (2000). "Situated-learning in cross-functional virtual teams." IEEE Transactions on Professional Communication(Feb/Mar): 51--66.

Rossi, M. A. (2004). Decoding the ``Free/Open Source (F/OSS) Software Puzzle'': A survey of theoretical and empirical contributions. Working Paper. Available at http://opensource.mit.edu/papers/rossi.pdf.

Rousseau, V., Aube, C. and Savoie, A. (2006). "Teamwork Behaviors: a Review and an Integration of Frameworks." Small Group Research **37**(5): 540-570.

Sambamurthy, V. and Chin, W. W. (1994). "The Effects of Group Attitudes Toward Alternative GDSS Design on the Decision-Making Performance of Computer-Supported Groups." Decision Sciences **25**(2): 215-241.

Scacchi, W. (2007). Free and open source software development: Recent research results and methods. Advances in Computers. M. Zelkowitz, Elsevier Press. **69:** 243-295.

Schiller, S. Z. and Mandviwalla, M. (2007). "Virtual Team Research: An Analysis of Theory Use and a Framework for Theory Appropriation." Small Group Research **38**(1): 12-59.

Schmidt, J. B., Montoya-Weiss, M. M. and Massey, A. P. (2001). "New Product Development Decision-Making Effectiveness: Comparing Individuals, Face-to-Face teams, and Virtual Teams." Decision Sciences **32**: 575-600.

Seeger, J. A. (1983). "No Innate Phases in Group Problem-solving." Academy of Management Review **8**: 683-689.

Simon, H. A. (1965). The Shape of Automation. New York, Harper and Row.

Simon, H. A. (1976). Administrative Behavior. New York, NY, Free Press.

Sole, D. and Applegate, L. (2000). Knowledge Sharing Practices and Technology Use Norms in Dispersed Development Teams. International Conference in Information Systems, Brisbane, Australia.

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 **34**(4): 771-796.

Srivastava, A., Bartol, K. M. and Locke, E. A. (2006). "Empowering Leadership in Management Teams: Effects on Knowledge sharing, Efficacy, and Performance." Academy of Management Journal **49**(6): 1239-1251.

Straus, S. G. and McGrath, J. E. (1994). "Does the Medium Matter? The Interaction of Task Types and Technology on Group Performance and Member Reactions." Journal of Applied Psychology **79**(1): 87-97.

von Krogh, G. and von Hippel, E. (2006). "The Promise of Research on Open Source Software." Management Science **52**(7): 975-983.

Watson-Manheim, M. B., Chudoba, K. M. and Crowston, K. (2002). "Discontinuities and Continuities: a new way to understand virtual work." Information, Technology & People **15**(3): 191-209.

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

Wittenbaum, G. M., Hollingshead, A. B., Paulus, P. B., Hirokawa, R. Y., Ancona, D. G., Peterson, R. S., Jehn, K. A. and Yoon, K. (2004). "The functional perspective as a lens for understanding groups." Small Group Research **35**(1): 17–43.

Wood, R. E. (1986). "Task Complexity: Definition of the Construct." Organizational Behavior and Human Decision Processes **37**: 60-82.

Yin, R. K. (2003). Case Study Research: Design and Methods. Thousand Oaks, CA, SAGE Publications.