International Conference on Information Systems (ICIS)

ICIS 2008 Proceedings

Association for Information Systems *Year* 2008

Decision Making Paths in Self-Organizing Technology-Mediated Distributed Teams

Qing Li* Robert Heckman† Kevin Crowston‡
James Howison** Eileen Allen†† U. Yeliz Eseryel‡‡

*Syracuse University, qli03@syr.edu
†Syracuse University, rheckman@syr.edu
‡Syracuse University, crowston@syr.edu
**Syracuse University, jhowison@syr.edu
††Syracuse University, eeallen@syr.edu
‡‡Syracuse University, uyeserye@syr.edu

DECISION-MAKING PATHS IN SELF-ORGANIZING TECHNOLOGY-MEDIATED DISTRIBUTED TEAMS

Les chemins de prise de décision dans les équipes à distance auto-organisées

Research-in-Progress

Qing Li
Syracuse University
Syracuse, NY U.S.A.
qli03@syr.edu

Robert Heckman
Syracuse University
Syracuse, NY U.S.A.
rheckman@syr.edu

Kevin Crowston
Syracuse University
Syracuse, NY U.S.A.
crowston@syr.edu

James Howison
Syracuse University
Syracuse, NY U.S.A.
jhowison@syr.edu

Eileen Allen
Syracuse University
Syracuse, NY U.S.A.
eeallen@syr.edu

U. Yeliz Eseryel
Syracuse University
Syracuse, NY U.S.A.
uyeserye@syr.edu

Abstract

This paper investigates decision making in self-organizing technology-mediated distributed teams. This context provides an opportunity to examine how the use of technological support to span temporal and organizational discontinuities affects decision-making processes. 258 software-modification decision episodes were collected from the public emailing lists of six Free/Libre Open Source Software (FLOSS) projects over a span of five years. Six decision-making paths were identified as 1) short-cut decision-making path; 2) implicit-development decision-making path; 3) implicit-evaluation decision-making path; 4) normative decision-making path; 5) dynamic decision-making path; and 6) interrupted/delayed decision-making path. We suggest that the nature of the tasks and the affordances of the technology used reduce the need for explicit coordination, resulting in a broader range of possible decision processes than are observed in face-to-face groups.

Keywords: Decision Making Path; Group Decision Making; Self-Organizing Technology-Mediated Distributed Team; Free/Libre Open Source Software; FLOSS

Résumé

Ce document étudie la prise de décision dans les équipes à distance auto-organisées. Six chemins de prise de décision ont été identifiés en observant 258 situations de décision relatives à des modifications logicielles. Nous suggérons que la nature des tâches et les affordances de la technologie utilisée réduisent le besoin de coordination explicite, ayant pour résultat une large variété de processus de décision possibles.

摘要

本文深入研究了自组织虚拟团队的决策模式，尤其是基于现代信息通信技术的虚拟合作模式对决策过程的影响。该研究从258个软件修改类决策案例中识别出六种主要决策模式。我们发现决策任务的性质和各类信息技术的应用极大地降低了团队成员间的显性合作需求，从而使虚拟决策模式比传统决策模式更为复杂和多变。

1 Introduction

Distributed teams are groups of geographically dispersed individuals working together towards a common goal. Although distributed work has a long history, recent advances in information communication technologies (ICT) such as the Internet, email, videoconferencing and groupware have been crucial enablers for wider adoption of this organizational form (Ahuja et al. 2003). As organizations become increasingly knowledge-based and dependent on effective coordination of specialized knowledge for competitive advantage, virtual teams have grown in importance. Among the most well-known virtual teams are Internet-enabled collaborations such as Free/Libre Open Source Software (FLOSS) development teams. In contrast to organizationally-situated teams, FLOSS teams are self-organized groups in which developers from around the world contribute virtually without belonging to a common organization. Such groups meet face-to-face (FTF) infrequently and rather coordinate their activities by means of ICTs (Raymond 1998). Due to the absence of formal external role structures that grant authority, decision-making practices in these teams must emerge from the interactions among the team members rather than from an organizational context. Although temporal and organizational discontinuities make such emergence even more difficult to attain, effective teams seem to have developed productive ways of making decisions. By examining FLOSS teams, we can describe how decisions are made under temporally- and geographically-separated circumstances to inform our understanding of the relation between technology-support and group practices. Such an understanding may also provide practical guidance to managers responsible for distributed teams. In this research, we focused particularly on group decision-making practices enabled by asynchronous communications technologies.

2 Literature review

Early studies of decision-making processes can be categorized into those presenting normative models and those presenting descriptive models. Normative models prescribe how decisions should be made to be most effective. This approach stems from Simon (1960)'s rational decision-making model that defined three main phases: "intelligence", "design" and "choice". Descriptive models attempt to describe how decisions are actually made in practice. One of the most famous of these models is Garbage Can Model proposed by Cohen et al. (1972). This model describes decision-making as comprising several randomly occurring steps, rather than following an orderly sequence of steps. Integrating these two approaches, Mintzberg et al. (1976) proposed a general model of decision-making process by introducing interruptions, delays and deviations. This model recognized the presence of phases of decision-making, but argued that decision makers could loop back and forth among different phases due to various interruptions. Consistent with the later view, Poole & Roth (1989) found that normative models were not adequate to capture the nature of all decision-making sequences. They claimed that situational contingencies such as conflicts and unexpected tasks could interrupt group decision process and initiate loop-back. They adopted a structurational perspective to examine how group interactions create, elaborate on and finalize the decision over time and developed a Decision Functions Coding System that segmented the decision activities into multiple functional moves. For example, problem activity included two moves—problem analysis and problem critique—and solution activity included four moves—solution analysis, solution design, solution elaboration and solution confirmation. The phases of decision making were clustered based on the sequences of decision functional moves. Their results provided evidence for the existence of multiple possible decision sequences rather than a unitary sequence.

Information System (IS) researchers have paid much attention to the impact of technology on group decision making. Many IS studies on group decision-making have been design-focused, offering important suggestions to improve the quality of team decisions (Baltes et al. 2002). These studies have primarily examined the influence of different variables on group outcomes, rather than examining decision process itself. One of the few exceptions is Poole and Holmes (1995), who used a phase-based approach to observe the impact of Group Decision Support Systems (GDSS) on group decision-making. In that study, decision paths were categorized and analyzed based on the number of phases and the number of loops among phases. They also found that the decision path that most resembled the normative sequence generated better outcomes. A few studies have examined the process by which the use of information technology affects the decision-making process. For example, Lemus et al. (2004) suggested that because technology-supported communication conveys less observable status characteristics (position, age and sex, etc.), technology-supported groups are more task-oriented, thus generating more alternatives and more likely finding the best solution compared to FTF teams. Baltes et al. (2002) suggested that computer-mediated group decision-making achieved more even participation since group members felt more freedom to express their opinions. However, as Baltes et al. (2002) pointed out, the majority of GDSS studies have used student groups and hypothetical tasks rather than employees working in a real organizational setting. Due to the lack of ecological validity, these zero-history laboratory samples have been challenged by many scholars (e.g., Cragan & Wright 1990; Propp & Nelson 1996). Another gap in the group decision-making literature is the lack of examination of the impact of asynchronous technologies. Though email is the most frequently used communication tool in organizations, most studies have focused on the synchronous, text-based decision-support system (Baltes et al. 2002). In this study, we examine how group decisions are made via asynchronous communication tools in a real setting.

As a successful example of technology-supported distributed team, FLOSS has received extensive attention in the past few years. The discontinuities of distributed work are a pervasive problem for virtual teams, but their presence seems particularly problematic for software developers. Software development has been recognized as non-routine task since it is highly uncertain, ambiguous and complex. It requires a high degree of knowledge integration and coordination effort among multiple developers (Kraut and Streeter 1995). Additional efforts are required for interaction when participants are distant and unfamiliar with each other's work (Ocker and Fjermestad 2000), as in the FLOSS context, where developers may come from around the world and from a variety of organizations. Moreover, the lack of formal status and authority can hinder team members in making sense of the tasks and communications from others and in establishing a shared context. These issues would be expected to pose great challenges to the achievement of effective decision-making, but some FLOSS projects have been very successful.

Several studies have examined the decision-making processes adopted by FLOSS teams. For example, in Linux, Linus Torvalds is reported to have originally made most of the key decisions (Moon & 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 web-server, try to reach consensus in decisions (Fielding 1999) , albeit with a home-grown set of practices. 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. German (2003) also identified such a transition in the case of the Gnome project. However, the relationship between decision style and team effectiveness has not been empirically studied yet. In-depth studies are needed to explore the details of decision-making processes in self-organizing, technology-supported, temporal distributed contexts such as FLOSS teams.

3 Method

To understand the decision-making practices in FLOSS teams, content analysis was conducted to analyze the process by which project decisions were made via interactions on the primary communication venues for the team's developers (usually email lists, but in one case a forum). Archives of these venues are publicly available on project websites hosted on the repositories such as SourceForge.net and have been collected and made available through the FLOSSmole project (Howison et al. 2006). The research was conducted in two stages. The first stage focused on the basic descriptive characteristics of decision-making in self-organizing distributed (SOD) teams, such as how long decisions take, how many messages are required, how many participants are involved and how these things change over time (Heckman et al. 2006, Heckman et al. 2007). The second stage coded the steps of decision-making processes and tried to understand how the nature of voluntariness and asynchronicity of communication may affect

decision-making paths. This research-in-progress paper will report only preliminary results from the second stage of the project.

Case Selection

To balance maximization of variability and control of unwanted systematic variance, six FLOSS projects were chosen according to the following dimensions. First, in order to be able to compare projects meaningfully, we wanted to control for the complexity of the product and the potential audience of the project. Accordingly we picked three Enterprise Resource Planning (ERP) projects (Compiere, WebERP and Apache OFBiz) and three Instant Messenger (IM) projects (Gaim, aMSN and Fire). Compared to the IM projects, the ERP projects are more complex since they have high software code interdependencies and many external constraints such as accounting rules and legal reporting requirements. However, within each category, we expected complexity and potential audience to be comparable. Second, the six selected cases varied in level of project effectiveness, allowing us to potentially probe the relationship between decision-making processes and effectiveness. Group effectiveness was assessed following Crowston et al. (2006)'s OSS success model which includes downloads and page views, the number of developers over time and participation on the developer mailing lists.

Unit of Analysis: Decision Episode

The decision episode was adopted as our primary unit of analysis. A group decision was defined as a decision that commits the group as a whole to a future course of action, e.g., accepting a new piece of code, changing the system architecture or accepting a new developer, as opposed to individual decisions, e.g., about which task to undertake. A decision episode was defined as a sequence of messages in the communications venue (an email list or discussion forum) that begins with a triggering message presenting an opportunity for a group choice (e.g., a feature request or a bug report), and that includes discussion related to the issue and an announcement of a decision about the issue (which need not occur at the end of the episode). Thus, to analyze a decision, we examined the group of messages that reflect as complete a picture as possible of the interactions that constitute the process of making that decision. The first level of coding was to identify decision episodes. Two coders identified the decision episodes independently with an inter-coder reliability of 92%. All disagreements in episode identification were resolved after discussion. A total of 360 decision episodes were collected from six FLOSS projects (sixty episodes from each, spanning the life of the projects over five years, 2001–2006). These episodes were classified into two types based on the nature of decision. We classed 258 episodes as software modification decisions that focused on daily technical decisions, which are the primary work of the team, while the other 102 were classified as non-software decision episodes that generally did not result in a change in software code, though they might exert an influence on the future of project development. In this research-in-progress paper, we will report the results from our analysis of the 258 software modification decision episodes. Future work will examine the non-software episodes.

Unit of Coding: Functional Move

To examine the micro-level decision-making process, we used the "functional move" as our unit of coding. A "functional move" is "literally the function or purpose served by a particular segment of the conversational discourse" (Paulus 2004). As a well-established method for discourse analysis, the "functional move" has been used extensively to understand the nature of interaction in both face-to-face and computer-mediated environments (Poole 1985; Poole & Holmes 1995). However, few studies to date have used functional moves to analyze complex, asynchronous, text-based environments such as email, electronic bulletin boards, threaded fora and so on. Previous research found that the asynchronous environment led to richer and broader discussion, since the lack of constraints on time provided individuals more time to create and elaborate their ideas (Turoff et al. 1993; Heckman and Annabi 2005). The decision function coding scheme we used in this study is based on Poole & Roth (1989)'s Decision Functions Coding System (DFCS) and Mintzberg et al. (1976)'s model of unstructured decision-making. The coding scheme is presented in Table 1.

In the present study, email transcripts were used as the data source to understand the decision development in self-organizing distributed teams. The length of a single email varied from one sentence that made only one move, to multiple-paragraph emails that covered all decision-related functions. Moreover, individuals might be involved in multiple issues at one time. These challenges made it difficult to equate a functional move with any naturally

occurring grammatical segment such as sentence or message. Thus we coded functional moves in thematic units, which can be a phrase, a sentence, a paragraph, or even a complete email message. Each episode was coded by two analysts (the second level of coding after coding for episodes). The inter-coder reliability reached 80%. After the initial coding, the two coders discussed all the disagreements and reconciled to achieve essentially complete agreement.

Table 1. Normative Decision Making Process Coding Scheme

1-Identification phase		
1a	Decision Recognition	This move recognizes the opportunity that may lead to a group decision.
1b	Diagnosis	This move focuses on understanding the underlying reasons that cause the problem. It also includes asking and providing background information.
2-Development phase		
2a	Solution Analysis	This move describes the activity trying to develop its solution in general terms such as group rule/norm, criteria and general directions to guide the solution
2b	Solution Search	This move describes the activity trying to look for ready-made solutions based on personal experiences and existing resources
2c	Solution Design	This move describes the activity designing a solution from the scratch, or modifying the ready-made/ existing solutions in a new context.
3-Evaluation phase		
3a	Solution Evaluation	This move explicitly or implicitly comments on the usability of potential alternatives 1) opinion expression; 2) usability testing
3b	Solution Confirmation	This move describes the activity explicitly asking for group confirmation or initiating voting
4-Announcement phase		
4	Decision announcement	This move announces the final decision at collective level

4 Findings and Discussion

To capture the nature of the decision-making process, we focused on two dimensions: the coverage of main decision phases and the existence of loops among different phases. Here, a loop refers to the situation when the linear sequence of decision-making process is interrupted and the discussion loops back to a previous phase. Of the 258 software modification decision episodes, only 31% went through all four phases. The other discussions made decisions while skipping one or two phases. We also found that while 42% of decisions were made in a linear sequence, the other 58% included one or more loops. According to these characteristics, we categorized the observed decision-making processes into six paths, as shown in Table 2. In the remainder of this section, we discuss each type in turn.

Table 2 A Typology of Decision-Making Path

Decision-Making Path	**Phases**				**Loop-Back**	**N (%)**
	Identification	**Development**	**Evaluation**	**Announcement**		
Short-Cut	Y	N	N	Y	N	72 (27.9%)
Implicit-Development	Y	N	Y	Y	Y/N	2 (0.8%)
Implicit-Evaluation	Y	Y	N	Y	Y/N	79 (30.6%)
Normative	Y	Y	Y	Y	N	6 (2.3%)
Dynamic	Y	Y	Y	Y	Y	75 (29.1%)
Interrupted	Y	Y/N	Y/N	N	Y/N	24 (9.3%)

Short-Cut. We found 72 software modification decision episodes that followed what we called a "Short-Cut Decision-Making Path". This path represents the simplest pattern where the decision is made right after opportunity recognition, with no 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.

Implicit-Development. "Implicit-Development" here does not mean the non-existence of development phase, but rather the invisibility of development phase in the online discussions. In the episodes of this kind, the person who brings up an issue also provides a detailed solution. The subsequent discussions mainly concentrate on evaluating its feasibility, the pros and cons of implementation etc, rather than looking for more alternatives of the original issue. For example, in the Compiere project, a user initiated discussion by posting two alternatives: "*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 (Evaluation-opinion) and the final decision was made based on the revision of original proposal (Decision Announcement).

Implicit-Evaluation. The third type of decision-making path we call "Implicit-Evaluation" due to the lack of online evaluation discussion. The decision is announced directly after the alternatives are generated in the development stage. 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 on the existing solution alternatives (Decision Announcement).

Normative. The fourth category, "Normative decision-making path", adheres most closely to the rational approach described in previous studies. In the episode of this kind, 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 (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. Due to the asynchronous nature of email communication, people may dump a number of problems into a single message. Multiple issues, regardless of relevance, are discussed in parallel. New problems can be triggered by existing problems, and may leave the original ones unsolved and unattended by participants. Discussions may loop back to any previous phase at any time, even after a decision is announced. 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). 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 is actually made at the point we observed. Interruptions may occur in any phase of discussion and different 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 whether there is a need to fix it. An interruption in the development phase may reside in the 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 were stuck between two different points of view—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, Evaluation-opinion). However, the whole episode went on for 11 days and did not reach any final decision.

Table 2 shows the distribution of six decision-making paths in 258 software-modification decision episodes. Although FLOSS projects are usually considered as the results of collaborative efforts, surprisingly we observed 27.9% of the decisions were made without any discussion (Short-Cut decision-making path). Setting aside offline activities, which are not archived, these episodes appear to be independent work without collaboration with the rest of the group. The frequency of this path can be attributed to the modularity of product and coordination mechanisms adopted by FLOSS projects. High modularity reduces the interdependence of different modules, greatly reducing the need for coordination and communication among different developers. The implementation of version control system such as CVS (Concurrent Version System) or Subversion enables the team to work in a relatively independent and parallel fashion. Such systems allow source code to be checked in and checked out simultaneously in a seamless operation. A qualified developer with commit rights can submit changes directly into the code repository without online discussion or seeking explicit permission, while others must obtain permission by reporting the problems or submitting the patches before their contributions are accepted. This specialization of labor allows individuals to handle daily technical issues in a more independent and efficient way.

Another interesting phenomenon was that only 0.8% of decision episodes followed the "Implicit-Development" path while 30.6% followed the "Implicit-Evaluation" decision-making path. This disparity actually reflects how FLOSS development works by balancing extensive idea generation and product quality control. 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 only contributions of good quality that are consistent with the overall goal of the project are added to the project's source code repository and thus the application. A skilled developer can perform the evaluation phase individually by selecting an alternative without group discussion. 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 power and authority to do so. The lack of evaluation seems to reflect an action orientation in FLOSS development groups' decision-making. This approach may also be supported by the fact that commits to repositories are 'rewindable' and therefore are always open to future revision. It is always possible to return to a previous stage of development if the current solution is not satisfactory.

We found that FLOSS teams rarely made decisions in a normative fashion: only 2.3% of software-modification episodes fell into this category. Though some of the previous studies (e.g., Poole and Holmes 1995) found a positive relationship between the use of a normative decision path and group effectiveness, this did not seem to be the case in the open source context, in that the more successful projects did not use this method more often than the less successful projects. Indeed, it may be difficult for FLOSS projects to follow this path since decisions are made in asynchronous communication channels. Members are distributed around the world in different time zones. They can choose to reply any message they are interested in at any time. This distribution over time may lead to more loops in online discussions than would be likely in face-to-face interactions.

The lack of formal leadership may be another reason for the large number of loops observed in decision-making processes. Due to the largely voluntary nature of the projects, no one person can control group discussions or keep all members working at same pace. Even if the group adopted a rational procedure to make the decision, it rarely followed a linear discussion sequence, leading the episodes to fall into the category "dynamic decision-making path". The lack of formal leadership led to another observed phenomenon in FLOSS decision-making, where decision episodes in which a voting procedure was adopted tended to fail to reach a final decision. This result shows that although open source adopts a collaborative mode, strict group consensus is very difficult to achieve. Since no one really takes charge of the poll, the voting usually ends in nothing but a bunch of votes.

We also noticed that even if many of the development and evaluation activities were implicit (i.e., not public), most of the final decisions were made in public by the participants. It may be that this publicity is an attempt to validate individual activities or to create opportunities for collaboration, even if the group does not always take them up. Publicizing individual decisions does create an atmosphere of participation and consultation, regardless of whether there is active participation by the group.

5 Conclusion

In this paper, we analyzed the nature of software modification decision-making processes in technology-supported self-organizing distributed groups. One contribution in this study was to identity six paths by which decisions were made. According to the coverage of decision-making phases and presence or absence of loops among phases, we classified episodes as: 1) short-cut; 2) implicit-development; 3) implicit-evaluation; 4) normative; 5) dynamic; and 6) interrupted/delayed decision-making path.

A distinctive feature of our study is that we conducted our research in an asynchronous, technology-supported self-organizing context. We suggest that the specific information and communications technologies adopted by FLOSS projects greatly reduced the need for explicit coordination and communication among multiple users, allowing members to perform tasks in an independent fashion. We also suggest that the asynchronicity of communication complicated group decision-making processes by leading to more loops among different phases of the decision making process. We found that only 2.3% of decision-making practices in FLOSS projects followed the normative linear decision-making path, and no evidence that the normative decision path was used more in the more successful projects, counter to prior findings.

The methodological contribution of this study was to apply the functional move as a unit of coding in a complex, asynchronous, text-based environment. This thematic use of the functional move will provide a useful approach to examine the micro-level structure of decision-making processes.

A limitation of this study is the exclusion of other channels of group communication, such as Internet Relay Chat (IRC), Instant Messaging (IM), phone calls and so forth. Though the usages of these private, non-archived tools are generally discouraged in the open source communities, they are adopted at various levels in different groups. It is possible that some of the steps in the decision-process that were 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 entire group to some course of action are made without explicit involvement of the group in seemingly important phases of the decision process.

This Research-in-Progress paper reports the preliminary results of our research. Future work will focus on 1) the differences in decision-making paths between software-modification decision episodes and non-software decision episodes; 2) the relationship between decision-making paths and team effectiveness; 3) the influences of contingency variables on decision-making paths, such as how the size and activity of the distributed team, activity and maturity of the project, and size of the project's user base influence these choices of decision paths; and 4) the differences between observed decision-making paths and traditional face-to-face decision-making processes.

References

Ahuja, M. K., Galletta, D. F. and Carley, K. “Individual Centrality and Performance in Virtual R&D groups: An Empirical Study,” *Management Science* (49:1), January 2003, pp. 21–38.

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

Cohen, M. D., March, J. G. and Olsen, J. P. “A Garbage Can Model of Organizational Choice”, *Administrative Science Quarterly* (17:1), March 1972, pp. 1–25.

Cohen, S. G. and Gibson, C. B. “In the Beginning: Introduction and Framework”, in *Virtual Teams that Work: Creating Conditions for Virtual Team Effectiveness*, Gibson, C. B. and Cohen, S. G. (Ed.), Hoboken, New Jersey, Jossey-Bass, 2003, pp. 1-13.

Cragan, J. F., and Wright, D. W. “Small Group Communication Research of the 1980s: A Synthesis and Critique”, *Communication Studies* 41, 1990, pp. 212-236.

Crowston, K., Howison, J. and Annabi, H. “Information Systems Success in Free and Open Source Software development: Theory and Measures”, *Software Process—Improvement and Practice* (11:2), March/April 2006, pp. 123–148.

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

Fitzgerald, B. “The transformation of Open Source Software”, *MIS Quarterly*, (30:3), September 2006, pp. 587-598.

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

Heckman, R. and Annabi, H. "A content analytic comparison of learning processes in online and face-to-face case study discussions". *Journal of Computer-Mediated Communication* (10:2), January 2005, available: http://jcmc. indiana. edu/vol10/issue2/heckman.Html.

Heckman, R., Crowston, K., Eseryel, U. Y., Howison, J., Allen, E. and Li, Q. "Emergent Decision-Making Practices in Free/Libre Open Source Software (Floss) Development Teams", *IFIP International Federation for Information Processing* 234, 2007, pp. 71-84.

Heckman, R., Crowston, K., Li, Q., Allen, E., Eseryel, U. Y., Howison, J. and Kangning, W. "Emergent Decision-Making Practices in Technology-Supported Self-Organizing Distributed Teams", in *Proceedings of the Twenty-Seventh International Conference on Information Systems*, Milwaukee, Wisconsin, December 2006.

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

Kraut, R. E. and Streeter, L. A. "Coordination in Software Development", *Communications of the ACM* (38: 3), March 1995, pp. 69-81.

Lemus, D. R., Seibold, D. R., Flanagin, A. J. and Metzger, M. J. "Argument and Decision Making in Computer-Mediated Groups", *Journal of Communication* (54: 2), June 2004, pp. 302–320.

Mintzberg, H., Raisinghani, D. and Theoret, A. "The Structure of 'Unstructured' Decision Processes". *Administrative Science Quarterly* (21:2), June 1976, pp. 246-275.

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

Ocker, R.J. and Fjermestad, J. "High Versus Low Performing Virtual Design Teams: A Preliminary Analysis of Communication", In *Proceedings of the 33rd Hawaii International Conference on System Sciences*, Maui, Hawaii, January 2000

Paulus, T. M. "Collaboration or cooperation? Small group interactions in a synchronous educational environment", In *Computer-Supp. orted Collaborative Learning in Higher Education*, Roberts, T. S. (Ed.), Hershey, PA: Idea Group, Inc, 2004, pp. 100-124.

Poole, M. S. "Task and interaction sequences: A theory of coherence in group decision-making", In *Sequence and Patten in communication behavior*, Street, R. and Cappella, J. N. (Eds.), London, Edward Arnold, 1985, pp. 206-224.

Poole, M. S. and Holmes, M. E. "Decision Development in Computer-Assisted Group Decision Making," *Human Communication Research* (22:1), 1995, pp. 90–127.

Poole, M. S. and Roth, J. "Decision development in small groups V: Test of a contingency model". *Human Communication research* (15:4), June 1989, pp. 549-589.

Propp, K. M., and Nelson, D. "Problem-solving performance in naturalistic groups: The ecological validity of the functional perspective", *Communication Studies* 47, 1996, pp. 127-139.

Raymond, E. S. "The cathedral and the bazaar", *First Monday* (3:3), February 1998, available: http://www. firstmonday. dk/issues/issue3_3/raymond/.

Simon, H. A. The New Science of Management Decision Making, New York-Evanston, 1960.

Turoff, M., Hiltz, S. R., Bahgat, A. N. F. and Rana, A. R. "Distributed group support systems", *MIS Quarterly* (17:4), December 1993, pp. 399–417.